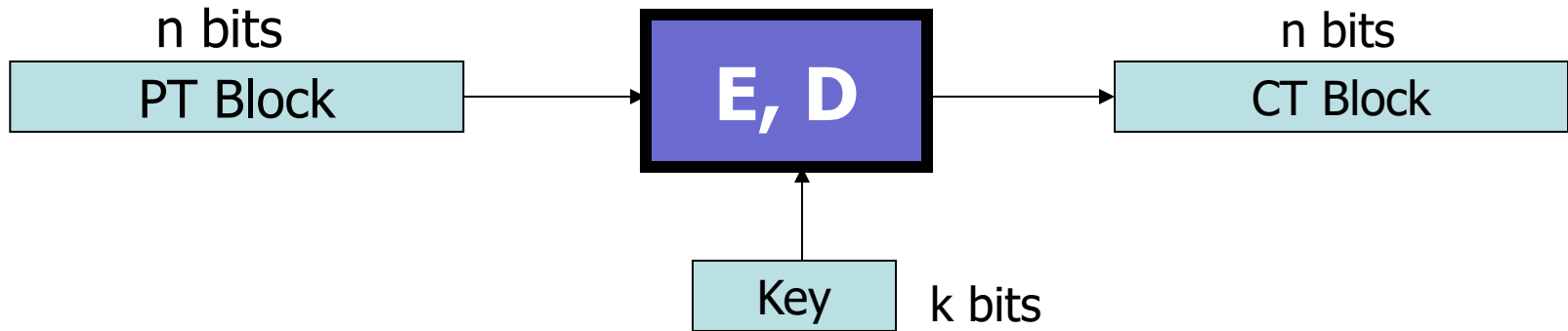


CS255: Winter 2026

CPA Security: How to use a key multiple times

Quick Recap

A **block cipher** is a pair of efficient algs. (E, D):



Canonical examples:

- **AES:** $n=128$ bits, $k = 128, 192, 256$ bits
(hardware support for many blocks in parallel)
- **3DES:** $n= 64$ bits, $k = 168$ bits (historical)

Abstract block ciphers: PRFs and PRPs

PRF: an efficiently computable $F: K \times X \rightarrow Y$

PRP: (a.k.a block cipher) $E: K \times X \rightarrow X$

is a PRF, such that

- for all $k \in K$: the function $E(k, \cdot)$ is one-to-one,
- there is an “efficient” inversion algorithm $D(k, x)$.

Secure PRF (resp. PRP):

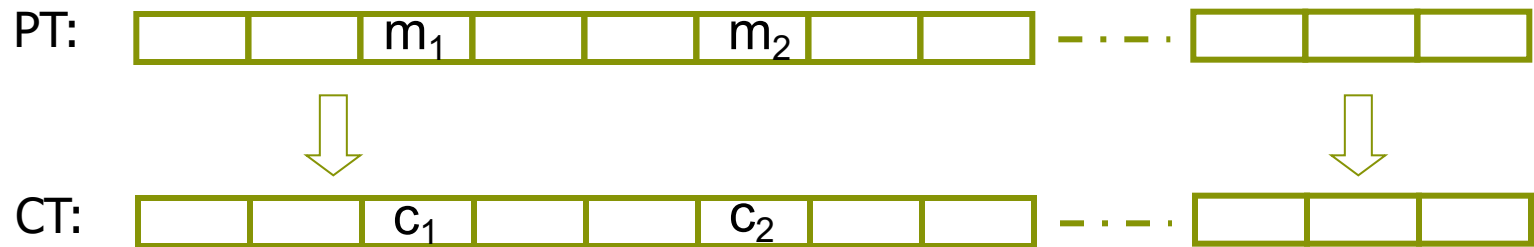
the uniform distribution on $S_F := \{ F(k, \cdot) : k \in K \}$

is **indistinguishable by queries** from

the uniform distribution on $\text{Funs}[X, Y]$ (resp. $\text{Perms}[X]$).

ECB: Incorrect use of a PRP

Electronic Code Book (ECB):



Problem:

– if $m_1 = m_2$ then $c_1 = c_2$

How to use a block cipher?

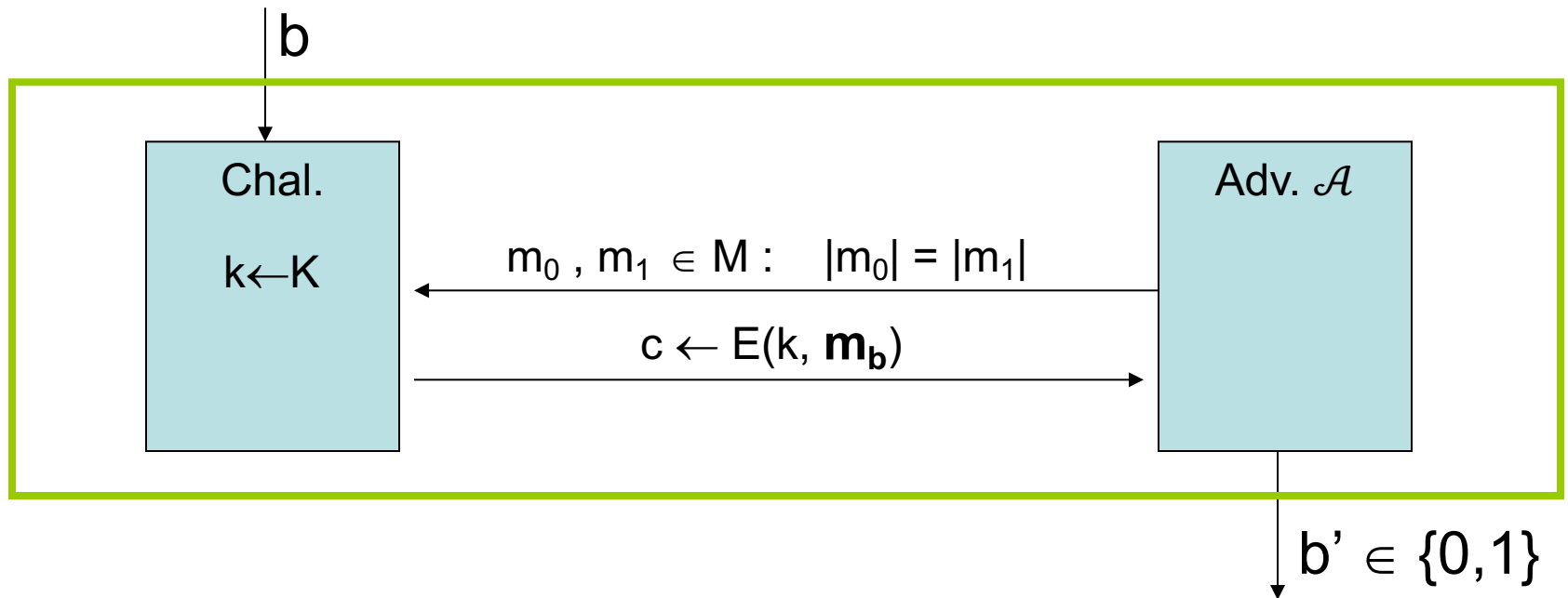
Modes of Operation for One-time Use Key

Example application:

Encrypted email. New key for every message.

Semantic Security for a one-time key

- $\mathbb{E} = (E, D)$ a cipher defined over (K, M, C)
- For $b=0,1$ define $\text{EXP}(b)$ as:



- Def: \mathbb{E} is sem. sec. for one-time key if for all “efficient” \mathcal{A} :

$$\text{Adv}_{\text{ss}}[\mathcal{A}, \mathbb{E}] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right|$$

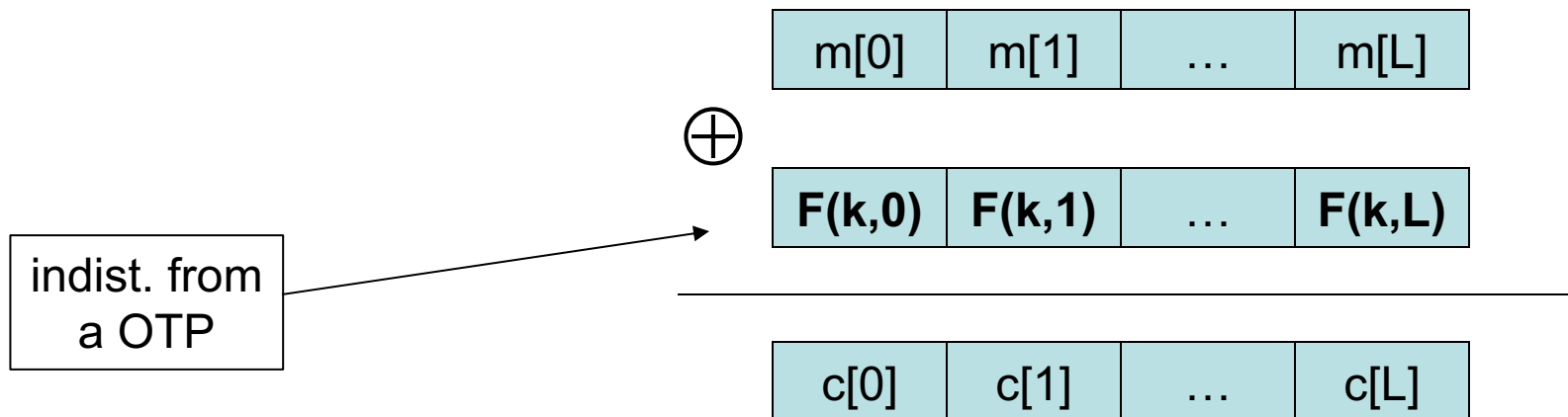
is “negligible.”

A Semantically Secure Scheme

Deterministic counter mode from a PRF

$$F: K \times \{0,1, \dots, L\} \rightarrow \{0,1\}^n$$

$$E_{\text{DETCTR}}(k,m) =$$



\Rightarrow Stream cipher built from PRF (e.g. AES)

How to use a block cipher?

Modes of Operation for Many-time Key

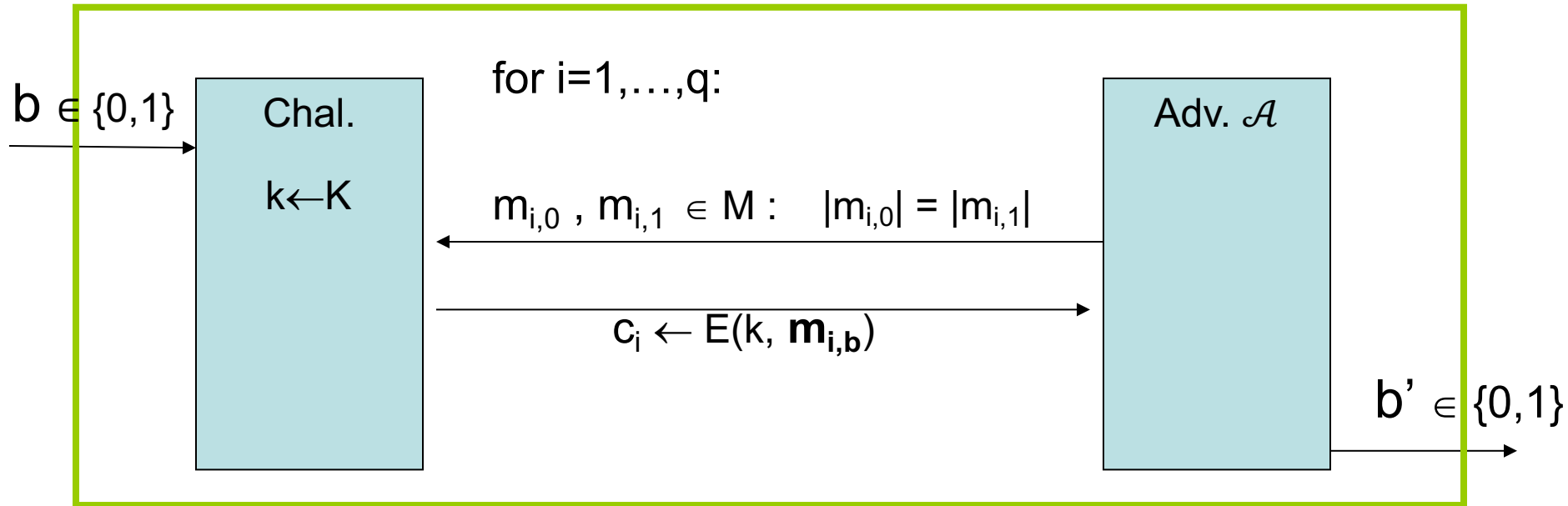
Example applications:

1. File systems: Same AES key used to encrypt many files.
2. IPsec: Same AES key used to encrypt many packets.

Semantic Security for many-time key (CPA security)

Cipher $\mathbb{E} = (E, D)$ defined over (K, M, C) .

For $b=0,1$ define $\text{EXP}(b)$ as:



if adv. wants $c = E(k, m)$ it queries with $m_{j,0} = m_{j,1} = m$

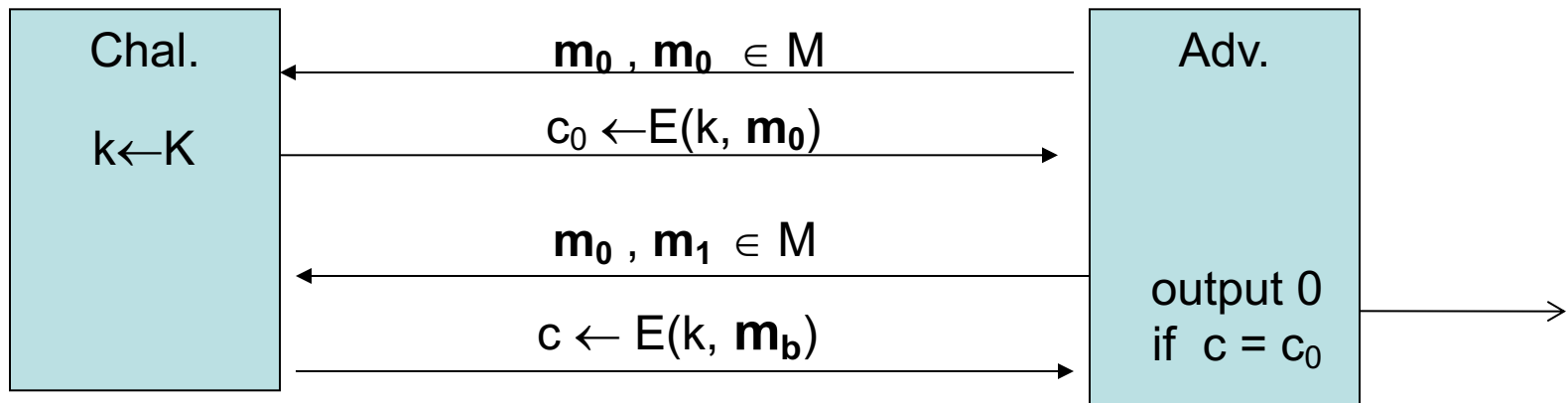
Def: \mathbb{E} is sem. sec. under CPA if for all “efficient” \mathcal{A} :

$\text{Adv}_{\text{CPA}}[\mathcal{A}, \mathbb{E}] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right|$
is “negligible.”

Security for many-time key

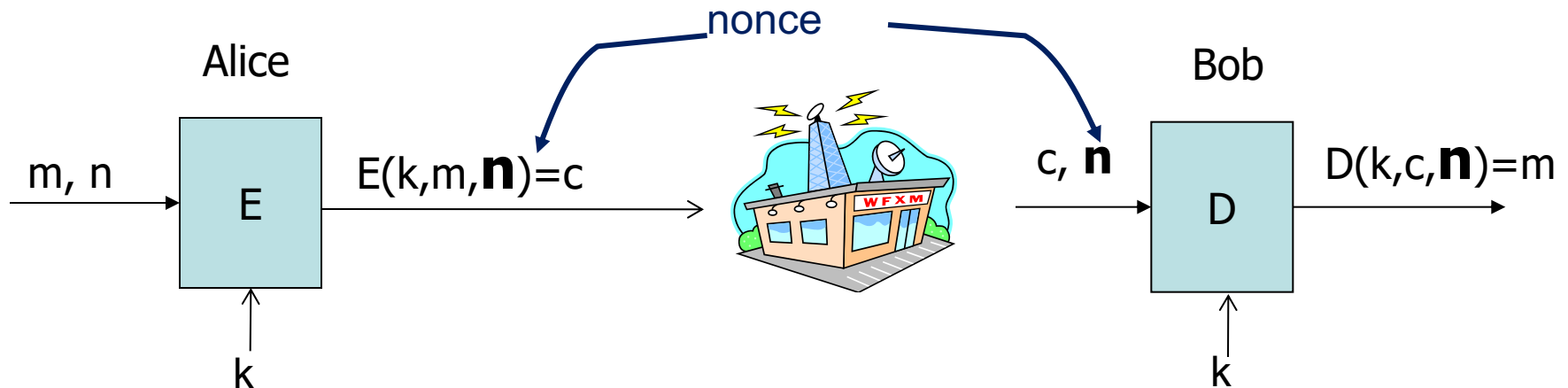
Fact: stream ciphers are insecure under CPA.

- More generally: if $E(k, m)$ always produces same ciphertext, then cipher is insecure under CPA.



If secret key is to be used multiple times \Rightarrow
given the same plaintext message twice,
the encryption alg. must produce different outputs.

Nonce-based Encryption

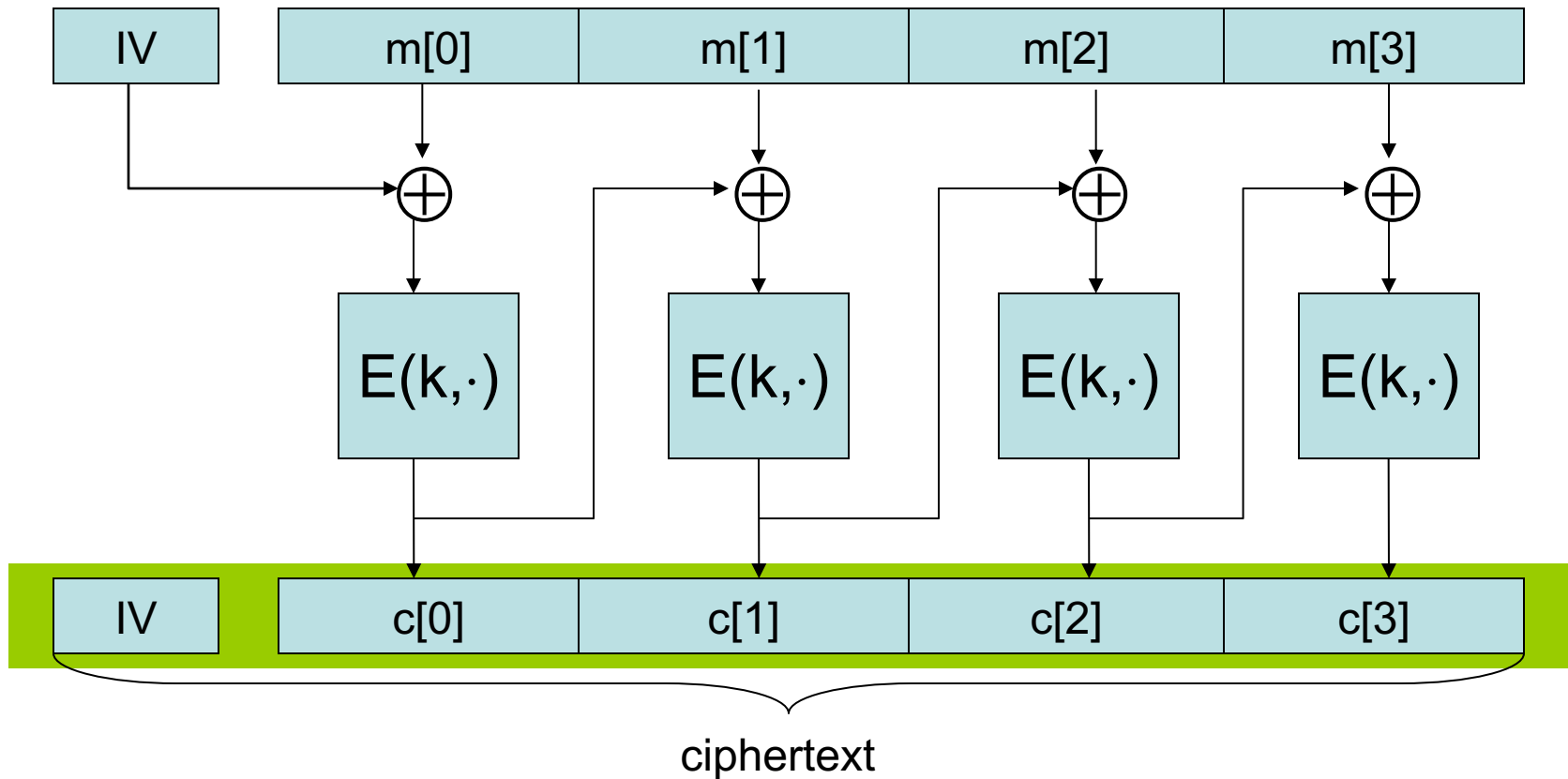


nonce n : a value that changes from msg to msg
(k, n) pair never used more than once

- method 1: encryptor chooses a random nonce, $n \leftarrow \mathcal{N}$
- method 2: nonce is a counter (e.g. packet counter)
 - used when encryptor keeps state from msg to msg
 - if decryptor has same state, need not send nonce with CT

Construction 1: CBC with random nonce

Cipher block chaining with a random IV (IV = nonce)



note: CBC where attacker can predict the IV is not CPA-secure. HW.

CBC: CPA Analysis

CBC Theorem: For any $L > 0$,

If E is a secure PRP over (K, X) then

E_{CBC} is a sem. sec. under CPA over (K, X^L, X^{L+1}) .

In particular, for a q -query adversary A attacking E_{CBC} there exists a PRP adversary B s.t.:

$$\text{Adv}_{\text{CPA}}[A, E_{\text{CBC}}] \leq 2 \cdot \text{Adv}_{\text{PRP}}[B, E] + 2 q^2 L^2 / |X|$$

Note: CBC is only secure as long as $q^2 \cdot L^2 \ll |X|$

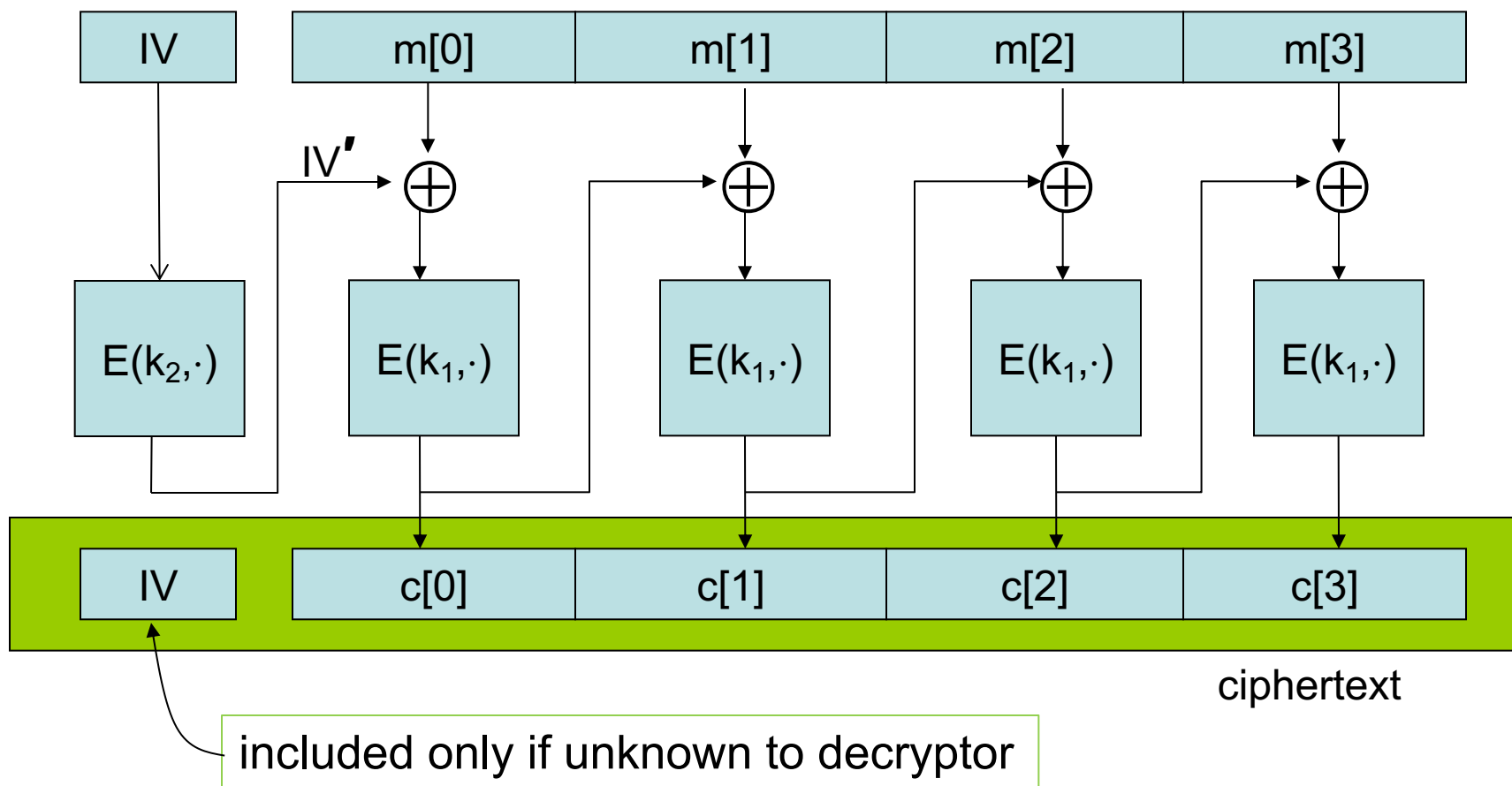
messages enc. with key

max msg length

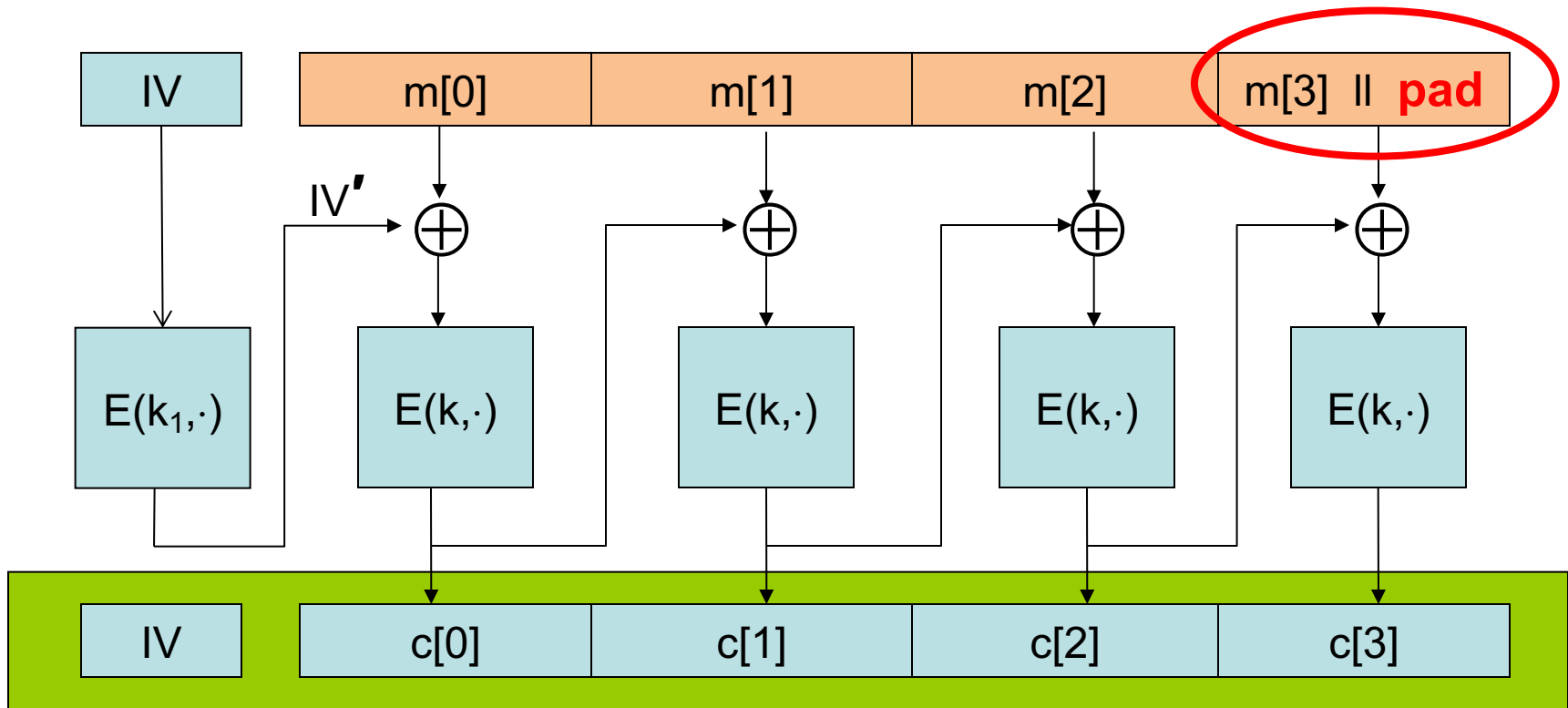
Construction 1': CBC with **unique** nonce

Cipher block chaining with unique IV (IV = nonce)

unique IV means: (key, IV) pair is used for only one message



A CBC technicality: padding



TLS 1.0: if need n -byte pad, $n > 0$, use:

$n-1 \quad n-1 \quad \dots \quad n-1$

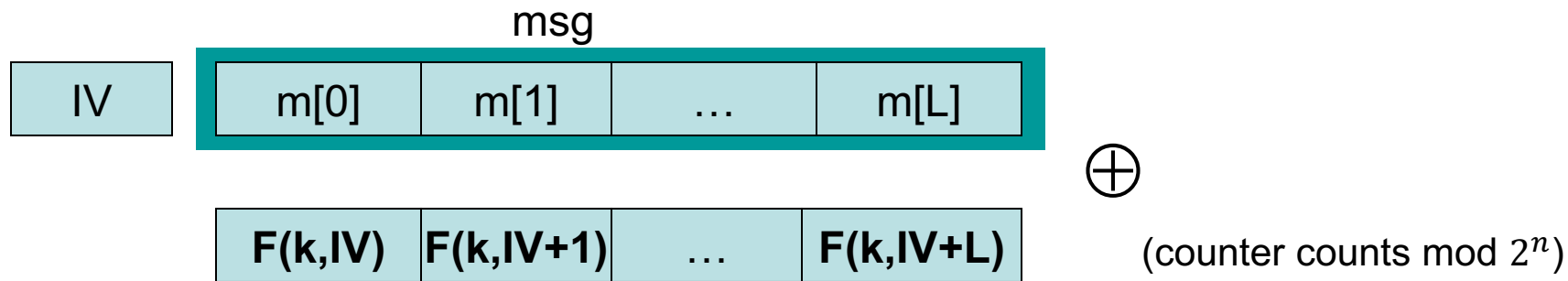
if no pad needed, add a dummy block

pad is
removed
during
decryption

Construction 2: rand ctr-mode

F: PRF defined over (K, X, Y) where $X = \{0, 1, \dots, 2^n - 1\}$ and $Y = \{0, 1\}^n$

(e.g., $n=128$)

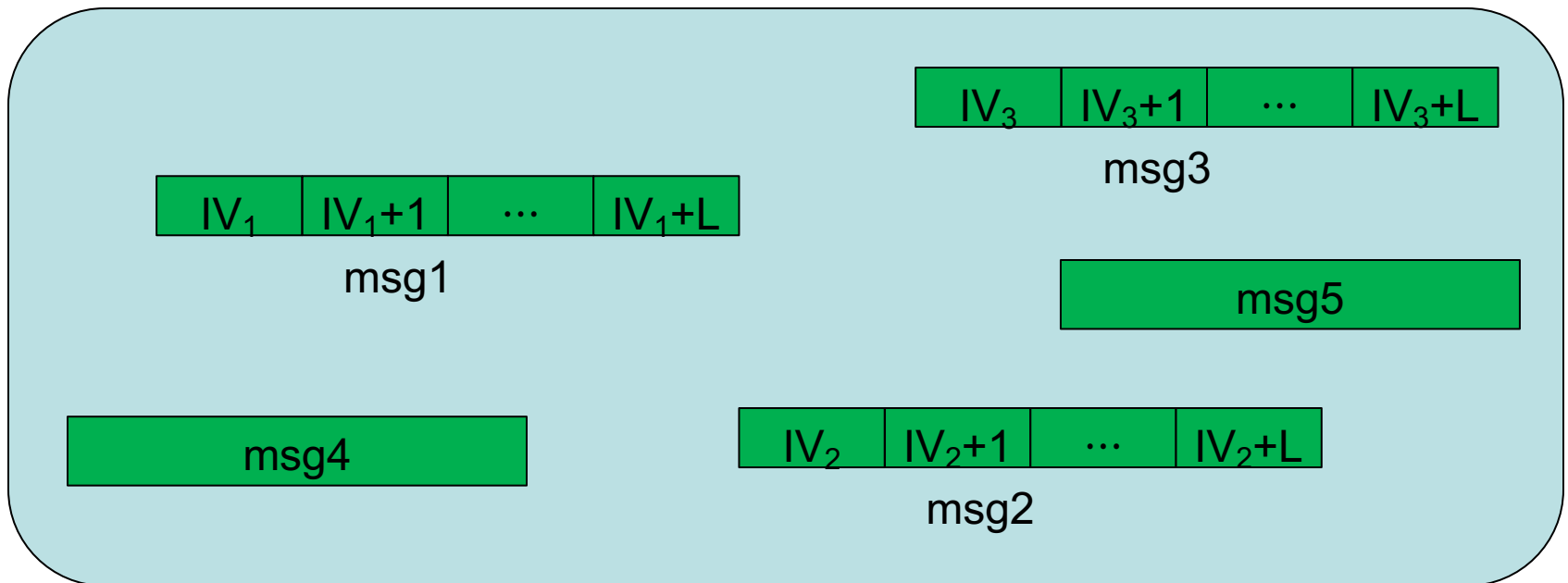


IV - chosen at random for every message

note: parallelizable (unlike CBC)

Why is this CPA secure?

the set X: domain of PRF



CPA security holds as long as intervals do not intersect

- q msgs, L blocks each $\Rightarrow \Pr[\text{intersection}] \leq \underbrace{2 q^2 L / |X|}_{\text{needs to be negligible}}$

rand ctr-mode: CPA analysis

Randomized counter mode: random IV.

Counter-mode Theorem: For any $L > 0$,

If F is a secure PRF over (K, X, X) then

E_{CTR} is a sem. sec. under CPA over (K, X^L, X^{L+1}) .

In particular, for a q -query adversary A attacking E_{CTR} there exists a PRF adversary B s.t.:

$$\text{Adv}_{CPA}[A, E_{CTR}] \leq 2 \cdot \text{Adv}_{PRF}[B, F] + 2 q^2 L / |X|$$

Note: ctr-mode only secure as long as $q^2 \cdot L \ll |X|$

Better than CBC !

An example

$$\text{Adv}_{\text{CPA}}[A, E_{\text{CTR}}] \leq 2 \cdot \text{Adv}_{\text{PRF}}[B, E] + 2 q^2 L / |X|$$

q = # messages encrypted with k , L = length of max msg

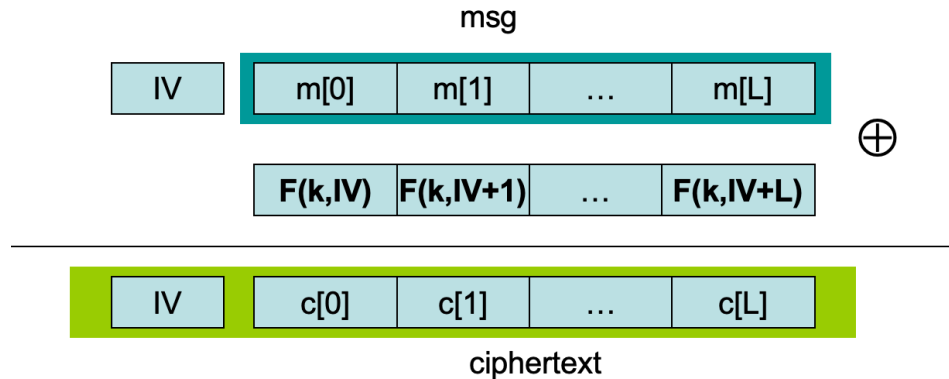
Suppose we want $\text{Adv}_{\text{CPA}}[A, E_{\text{CTR}}] \leq 1 / 2^{31}$

- Then need: $q^2 L / |X| \leq 1 / 2^{32}$

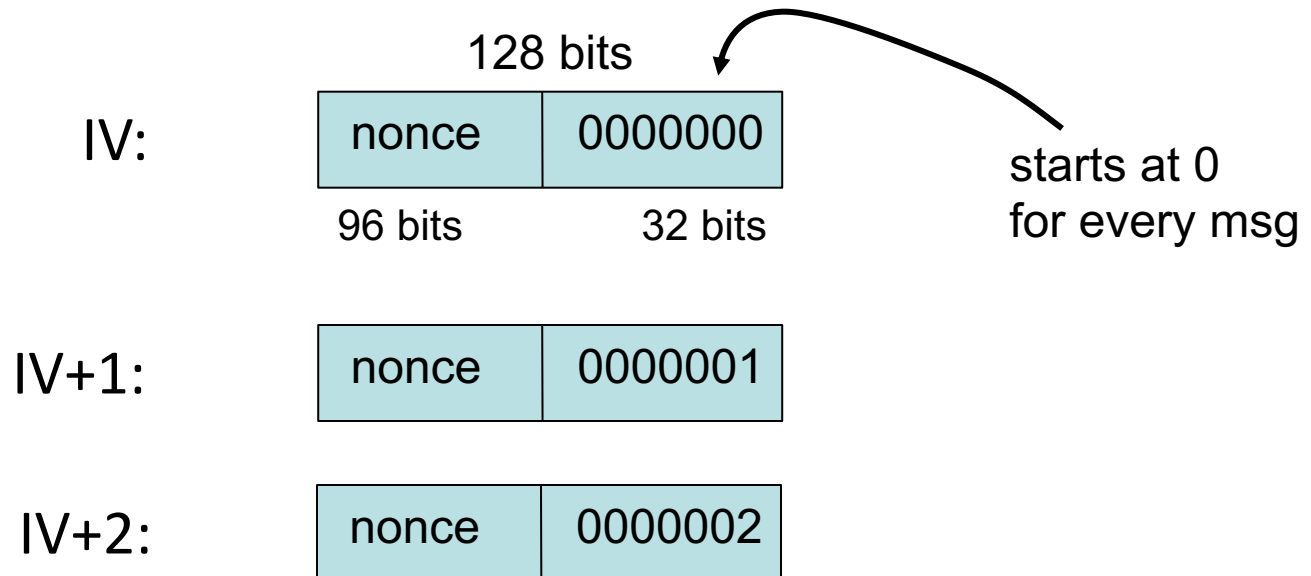
- AES: $|X| = 2^{128} \Rightarrow q L^{1/2} < 2^{48}$

So, after 2^{32} CTs each of len 2^{32} , must change key
(total of 2^{64} AES blocks)

Construction 2': nonce ctr-mode



To ensure $F(k,x)$ is never used more than once, choose IV as:



Comparison: ctr vs. CBC

	CBC	ctr mode
required primitive	PRP	PRF
parallel processing	No	Yes
security	$q^2 L^2 \ll X $	$q^2 L \ll X $
dummy padding block	Yes*	No
1 byte msgs (nonce-based)	16x expansion	no expansion

* for CBC, dummy padding block can be avoided using *ciphertext stealing*

Summary

PRPs and PRFs: a useful abstraction of block ciphers.

We examined two security notions:

1. Semantic security against one-time.
2. Semantic security against many-time CPA.

Note: neither mode ensures data integrity.

Stated security results summarized in the following table:

<div>Power</div> <div>Goal</div>	one-time key	Many-time key (CPA)	CPA and CT integrity
Sem. Sec.	stream-ciphers det. ctr-mode	rand CBC rand ctr-mode	later

Attacks on block ciphers

Goal: distinguish block cipher from a random permutation

- if this can be done efficiently then block cipher is broken

Harder goal:

find key k given many $c_i = E(k, m_i)$ for random m_i

(1) Linear and differential attacks

[BS'89,M'93]

Given *many* (m_i, c_i) pairs, can recover key much faster than exhaustive search

Linear cryptanalysis (overview) : let $c = \text{DES}(k, m)$

Suppose for random k, m :

$$\Pr[m[i_1] \oplus \cdots \oplus m[i_r] \oplus c[j_1] \oplus \cdots \oplus c[j_v] = k[l_1] \oplus \cdots \oplus k[l_u]] = \frac{1}{2} + \varepsilon$$

For some ε .

For DES, this exists with $\varepsilon = 1/2^{21} \approx 0.00000000477$!!

Linear attacks

$$\Pr \left[m[i_1] \oplus \dots \oplus m[i_r] \oplus c[j_1] \oplus \dots \oplus c[j_v] = k[l_1] \oplus \dots \oplus k[l_u] \right] = \frac{1}{2} + \varepsilon$$

Thm: given $1/\varepsilon^2$ random pairs $(m, c = \text{DES}(k, m))$ then

$$k[l_1] \oplus \dots \oplus k[l_u] = \text{MAJ} \left[m[i_1] \oplus \dots \oplus m[i_r] \oplus c[j_1] \oplus \dots \oplus c[j_v] \right]$$

with prob. $\geq 97.7\%$

\Rightarrow with $1/\varepsilon^2$ inp/out pairs can find $k[l_1] \oplus \dots \oplus k[l_u]$ in time $\approx 1/\varepsilon^2$

Linear attacks

For DES, $\varepsilon = 1/2^{21} \Rightarrow$

with 2^{42} inp/out pairs can find $k[l_1] \oplus \dots \oplus k[l_u]$ in time 2^{42}

Roughly speaking: can find 14 key “bits” this way in time 2^{42}

Brute force remaining $56-14=42$ bits in time 2^{42}

Attack time: $\approx 2^{43}$ ($\ll 2^{56}$) with 2^{42} random inp/out pairs

Lesson

A tiny bit of linearity leads to a 2^{42} time attack.

⇒ don't design ciphers yourself !!

(2) Side channel attacks on software AES

Attacker measures the **time** to compute AES128(k,m) for many random blocks m.

- Suppose that the 256-byte S table is not in L1 cache at the start of each invocation
 - ⇒ time to encrypt reveals the order in which S entries are accessed
 - ⇒ leaks info. that can compromise entire key

Lesson: don't implement AES yourself !

Mitigation: AES-NI or use vetted software (e.g., BoringSSL)

(3) Quantum attacks

Generic search problem:

Let $f: X \rightarrow \{0,1\}$ be a function.

Goal: find $x \in X$ s.t. $f(x) = 1$.

Classical computer: best generic algorithm time = $O(|X|)$

Quantum computer [Grover '96]: time = $O(|X|^{1/2})$

(requires a long running quantum computation)

Quantum exhaustive search

Given m , $c=E(k,m)$ define

$$f(k) = \begin{cases} 1 & \text{if } E(k,m) = c \\ 0 & \text{otherwise} \end{cases}$$

Grover \Rightarrow quantum computer can find k in time $O(|K|^{1/2})$

AES128: quantum key recovery time $\approx 2^{64}$

Adversary has access to a quantum computer \Rightarrow

encrypt data using a cipher with 256-bit keys (AES256)

THE END