Berkeley CS276 & MIT 6.875

Pseudorandom Permutations and Symmetric Key Encryption

Lecturer: Raluca Ada Popa

Sept 15, 2020

Announcements

- Starting to record
- Psets grading policy:
  - We count your best 5 out of 6 psets
  - Total of 10 days late, but at most 5 days late for every pset so that we can post solutions in a timely way
  - 5% participation grade, 95% psets
    - If extenuating circumstances prevent participation (e.g. due to timezone), solve a problem of the 6^{th} pset and tell us which one you want graded when you submit the pset

Overview

Last time: PRFs

Today

- PRPs/ Block ciphers
  - Theoretical constructions
  - Practical constructions: AES
- Symmetric key encryption schemes
  - Definitions
  - Practical constructions from block ciphers

Pseudorandom permutations (PRPs) or block ciphers - intuition

A family of functions $f: \{0,1\}^{|k|} \times \{0,1\}^n \rightarrow \{0,1\}^n$ indexed by the "key" k.

Correctness: f_{k} is a permutation (bijective function)

Efficiency: Can sample k, compute $f_k(x)$ and invert it with k

Pseudorandomness: For a random k, f_k "behaves" like a random permutation from the perspective of a PPT distinguisher

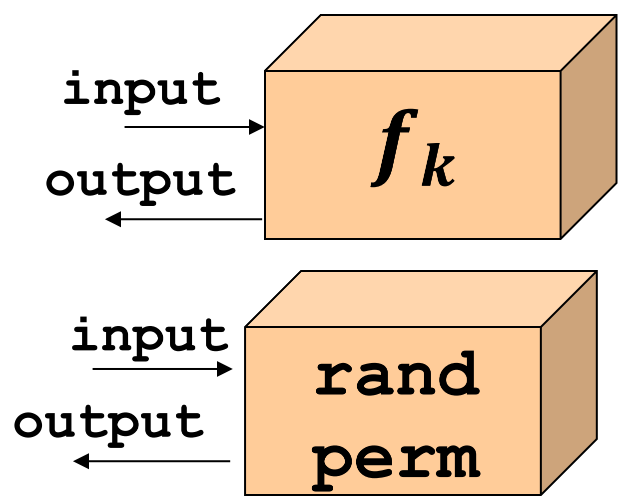
Block cipher: security game

Attacker is given two boxes, one for f_k and one for a random permutation (also called "oracles")

Attacker can give inputs to each oracle, look at the output, repeat as many times as he/she desires

Attacker wins if it guesses which is f_k

??? which is f_k???



PRP

Let $H_n = \{ f: \{0,1\}^n \to \{0,1\}^n\}$ be all permutations from n bits to n bits.

Definition: A sequence of random variables $F = \{F_n\}_{\mathrm{n}}$ with F_n a distribution over H_n is a **pseudorandom permutation ensemble** iff there

Efficiently computable and invertible

1. exists PPT alg $Gen(1^n) \to k$ s.t. $f_k \in F_n$ $\{k \leftarrow Gen(1^n); f_k\}$ is equal to F_n (efficient sampling)
2. exists PPT alg E such that $E(k, x) = f_k(x)$ (efficient eval)
3. exists PPT alg I such that $I(k, x) = f_k^{-1}(x)$ (efficient inversion)

4. for all PPT oracle distinguishers D, for all sufficiently large n,

$$\left|\Pr\left[Gen(1^n) \to k; D^{\{f_k\}}(1^n) = 1\right] - \Pr[R \leftarrow H_n;\ D^R(1^n) = 1]\right| = negl(n)$$

(pseudorandom)

Exercises

Let $H_n = \{ f: \{0,1\}^n \to \{0,1\}^n\}$ be all permutations from n bits to n bits.

[…]

for all PPT oracle distinguishers D, for all sufficiently large n,

$$\left|\Pr\left[Gen(1^n) \to k; D^{\{f_k\}}(1^n) = 1\right] - \Pr[R \leftarrow H_n;\ D^R(1^n) = 1]\right| = negl(n)$$

(pseudorandom)

Q: Let $\{U_n\}_n \subseteq H_n$ where U_n is the uniform distribution over all permutations from n to n bits. Is U_n pseudorandom?

A: yes

Q: Let $\{U_n^*\}_n \subseteq H_n$ where $U^*{}_n$ is the uniform distribution over all permutations from n to n bits except for the identity distributions. Is it pseudorandom?

A: yes, still statistically close to random

How can we construct PRPs?

The theory way:

Luby-Rackoff'86: PRF $\Rightarrow$ PRP

The practical way:

Rijmen and Daemen'03: AES proposal to NIST

The theory way - warmup

Let $f: \{0,1\}^n \to \{0,1\}^n$ be any function. Let's build a permutation $g: \{0,1\}^{2n} \to \{0,1\}^{2n}$ from f.

Let $g(x, y) = (y, f(x))$. Is it a permutation?

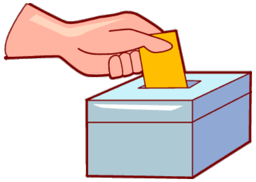
No. Let $f(x) = c$. Then $g(1, 10) = g(2,10)$

The theory way

Let $f\colon\{0,1\}^n \to \{0,1\}^n$ be any function. Let's build a permutation $g\colon\{0,1\}^{2n} \to \{0,1\}^{2n}$ from f.

Let $g(x, y) = (y, f(y) \oplus x)$.

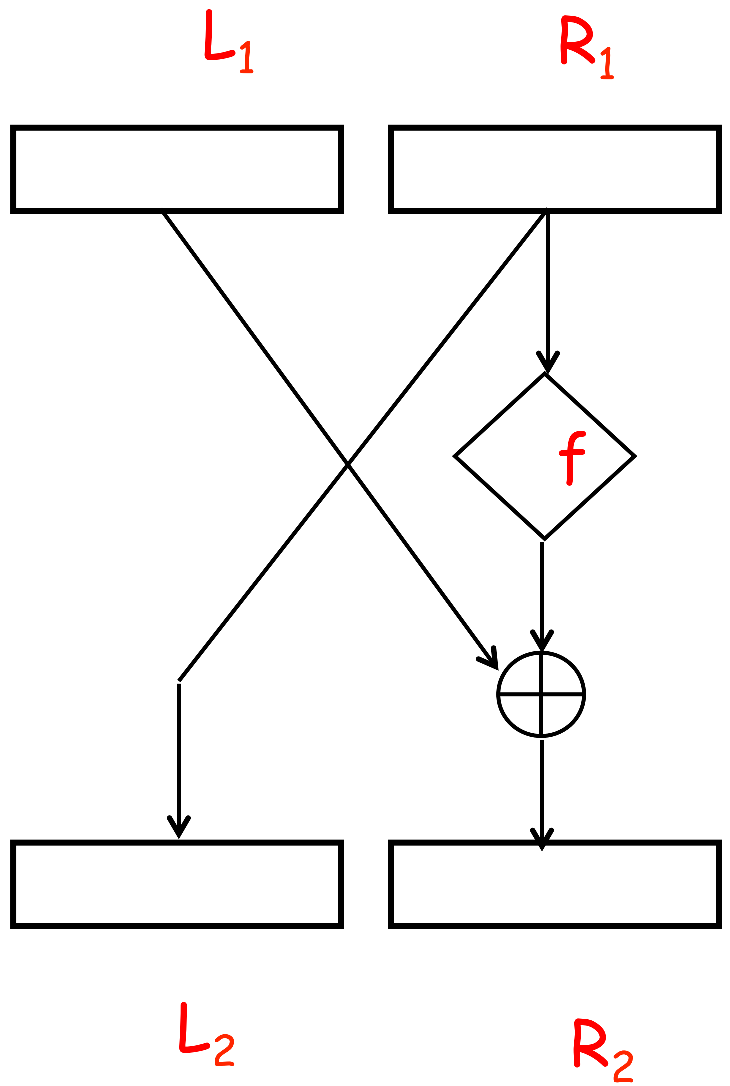
Is it a permutation?



Yes. $g^{-1}(y, \alpha) = (\alpha \oplus f(y), y)$

Feistel permutations

Feistel permutation: a permutation from any $f:\{0,1\}^n \to \{0,1\}^n$



Luby-Rackoff '86

Informal theorem: Let $\{F_n\}_n$ be a pseudorandom function family. Let

$$p_{\{k_1,k_2,k_3,k_4\}}(x) = g_{k_4}(g_{k_3}(g_{k_2}\left(g_{k_1}(x)\right)))$$

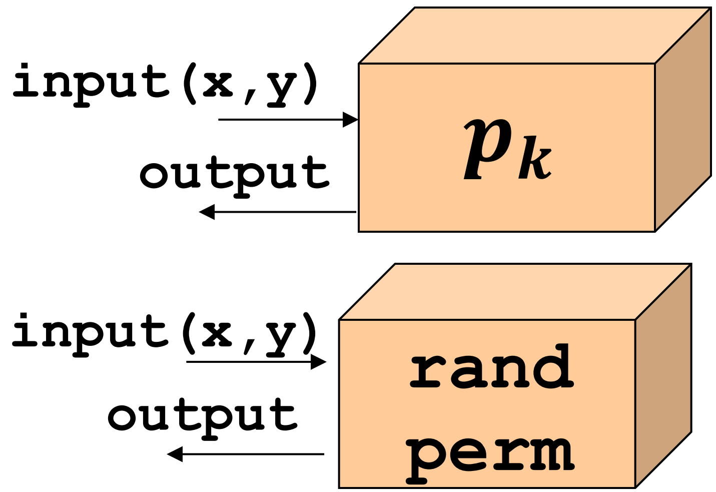
with g_k being the Feistel permutation from f_k.

Then $\{P_{2n}\}_{2n}$ is a pseudorandom permutation family.

Proof (optional): see assigned reading

Luby-Rackoff ’86 intuition

??? which is p_k???



How can the attacker distinguish?

$g_{k_1}(x,y) = (y, f_{k_1}(y) \oplus x)$ Sees y in the output.

$g_{k_2}(g_{k_1}(x,y)) = (f_{k_1}(y) \oplus x, f_{k_2}(f_{k_1}(y) \oplus x) \oplus x)$

Two inputs of same y can distinguish lefts.

How can we construct PRPs?

The theory way:

Luby-Rackoff'86: PRF $\Rightarrow$ PRP

The practical way:

Rijmen and Daemen'03: AES proposal to NIST

Advanced Encryption Standard (AES)

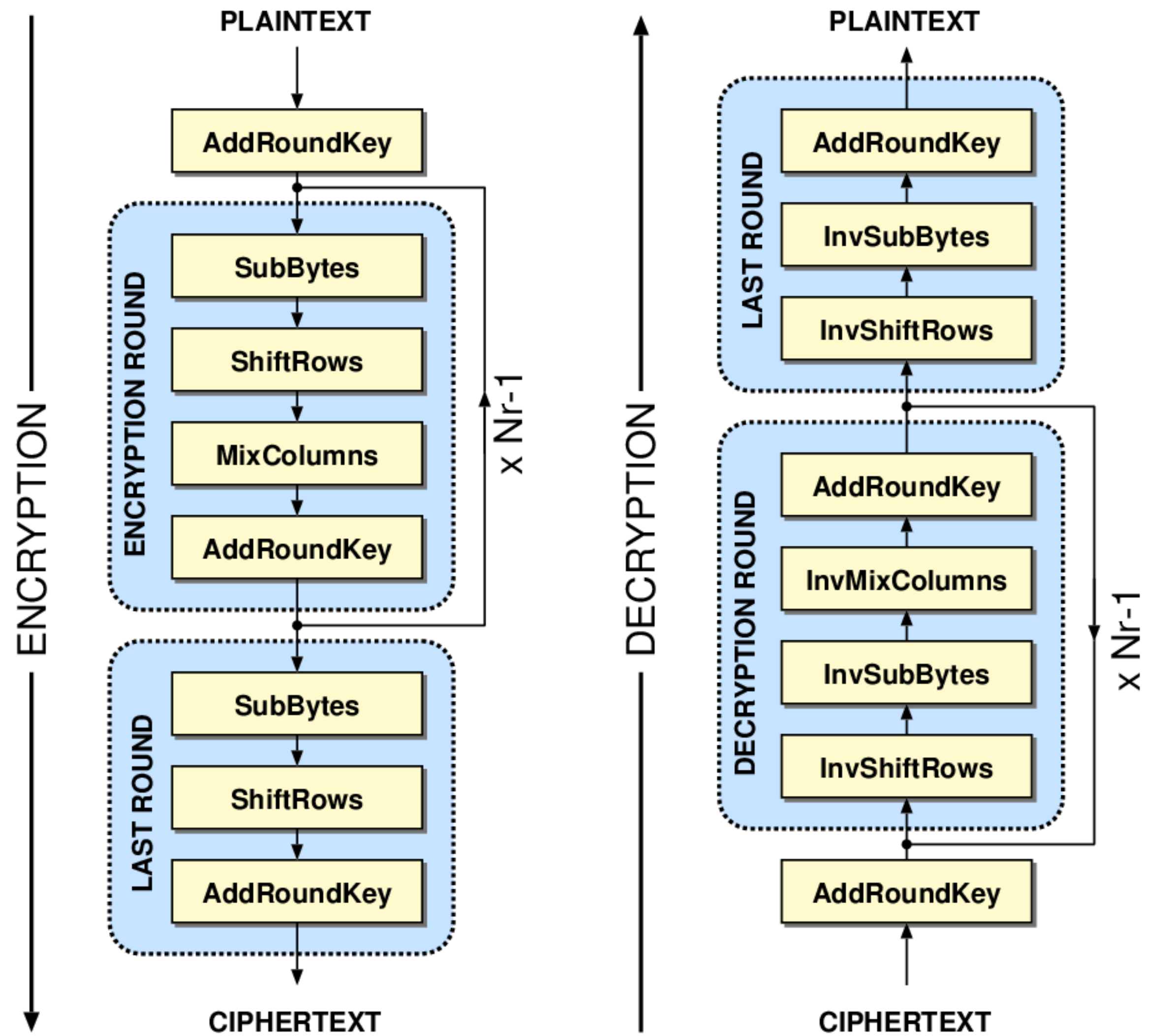
- Block cipher developed in 1998 by Joan Daemen and Vincent Rijmen
- Submitted as a proposal to NIST (US National Institute for Standard and Technology) during the AES selection process
- It won, so it was recommended by NIST
- It was adopted by the US government and then worldwide
- Block length n is 128bits, key length k is 256bits

Cryptanalysis

Not provably secure but an educated assumption that it is

- It stood the test of time and of much cryptanalysis (field studying attacks on crypto schemes)
    - [Bogdanov et al.'11]: $2^{126.2}$ operations to recover an AES-128 key.
    - Snowden documents attempts by the NSA to break it
- So far, no efficient algorithm comes close to breaking it.

AES ALGORITHM

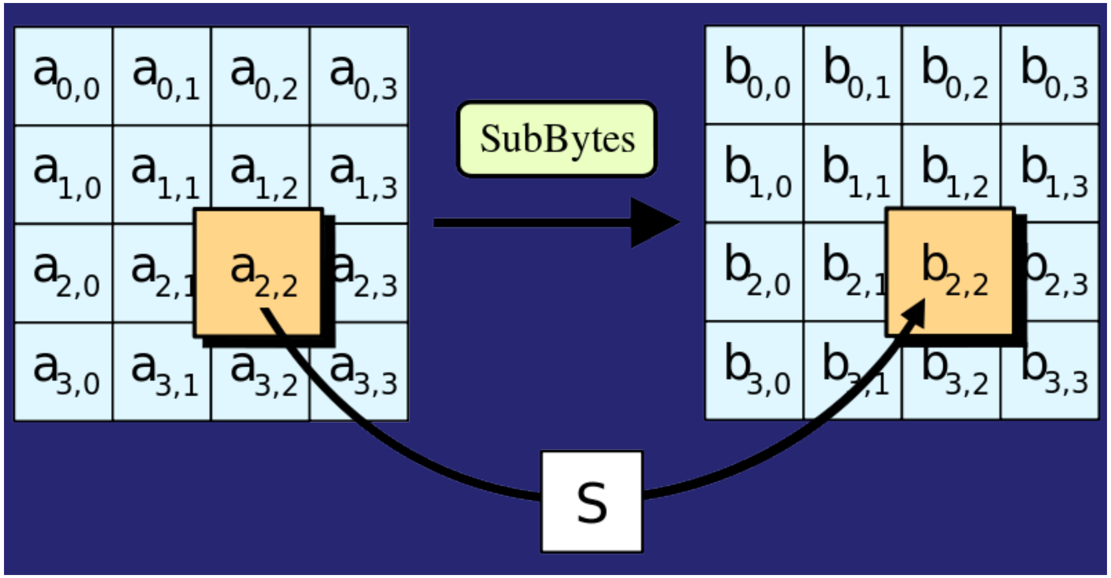


- 14 cycles of repetition for 256-bit keys.

You don't need to understand why AES is this way, just get a sense of its inner workings

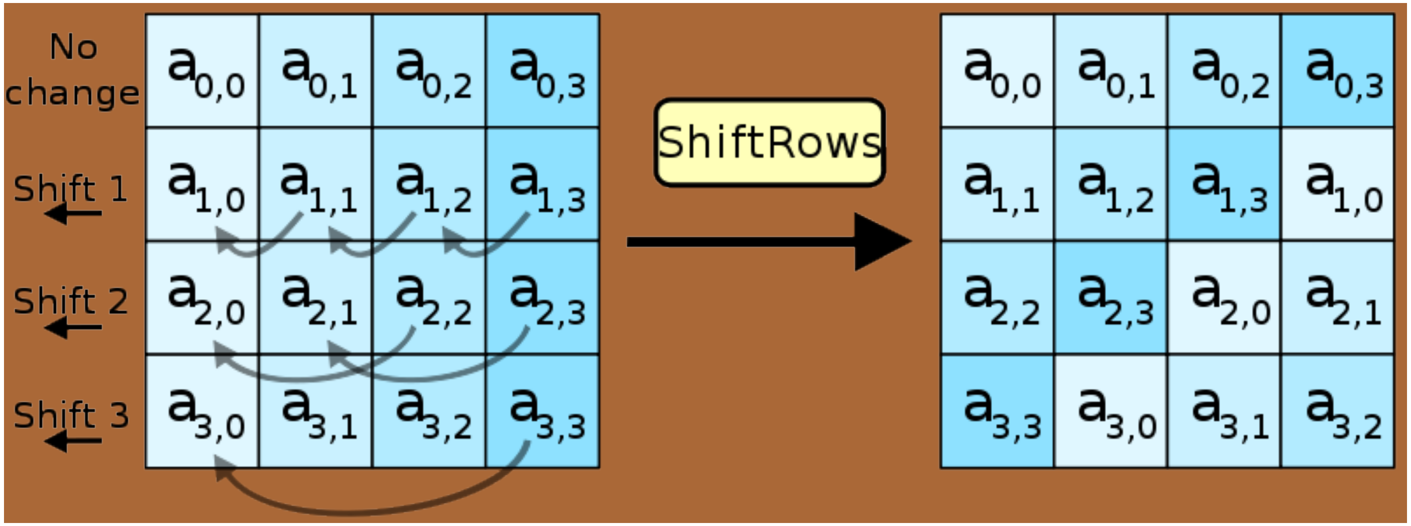
Algorithm Steps - Sub bytes

- each byte in the *state* matrix is replaced with a SubByte using an 8-bit substitution box
- $b_{ij} = S(a_{ij})$

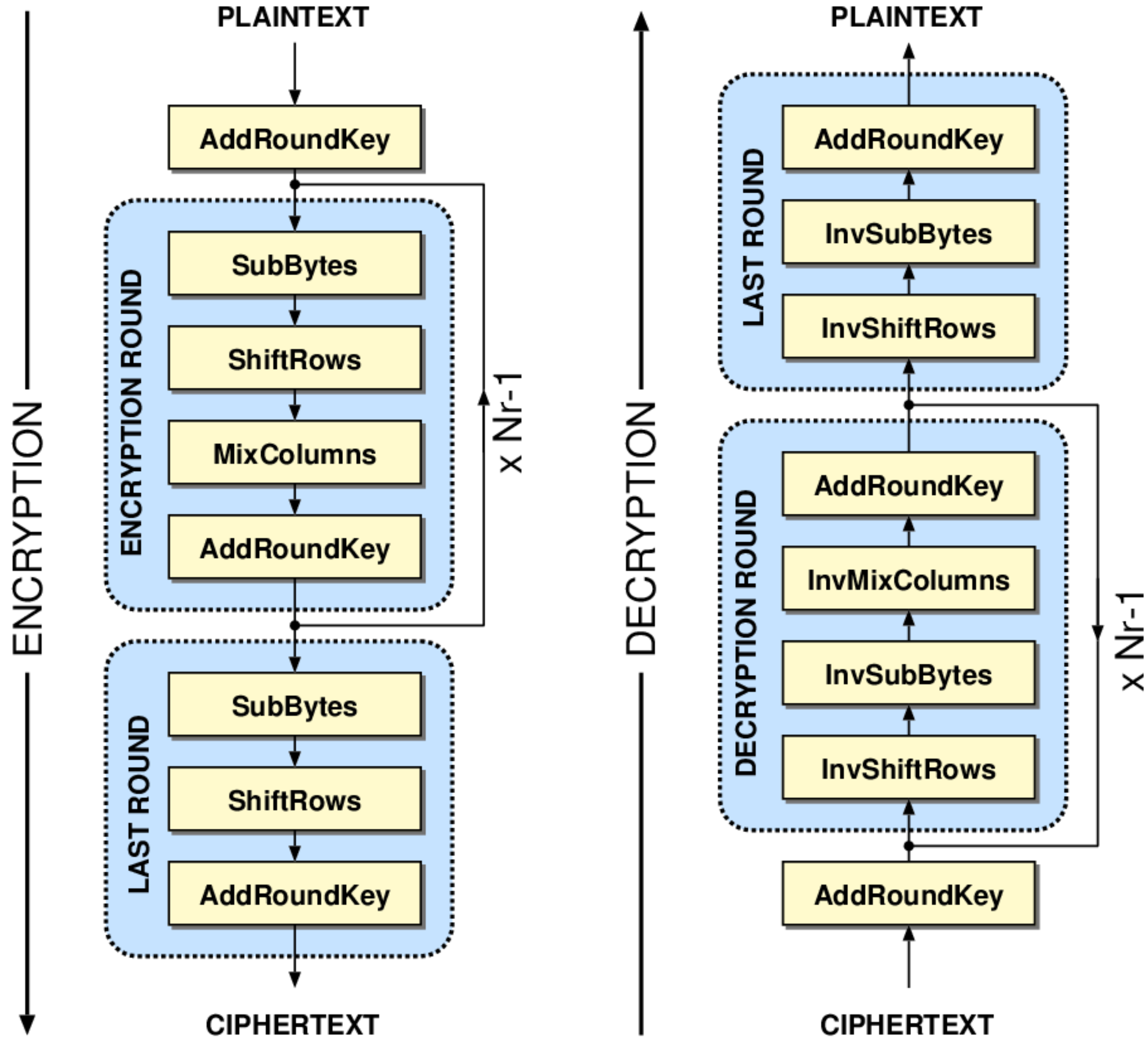


Shift Rows

- Cyclically shifts the bytes in each row by a certain offset
- The number of places each byte is shifted differs for each row



AES ALGORITHM



- The key gets converted into round keys via a different procedure
- 14 cycles of repetition for 256-bit keys.

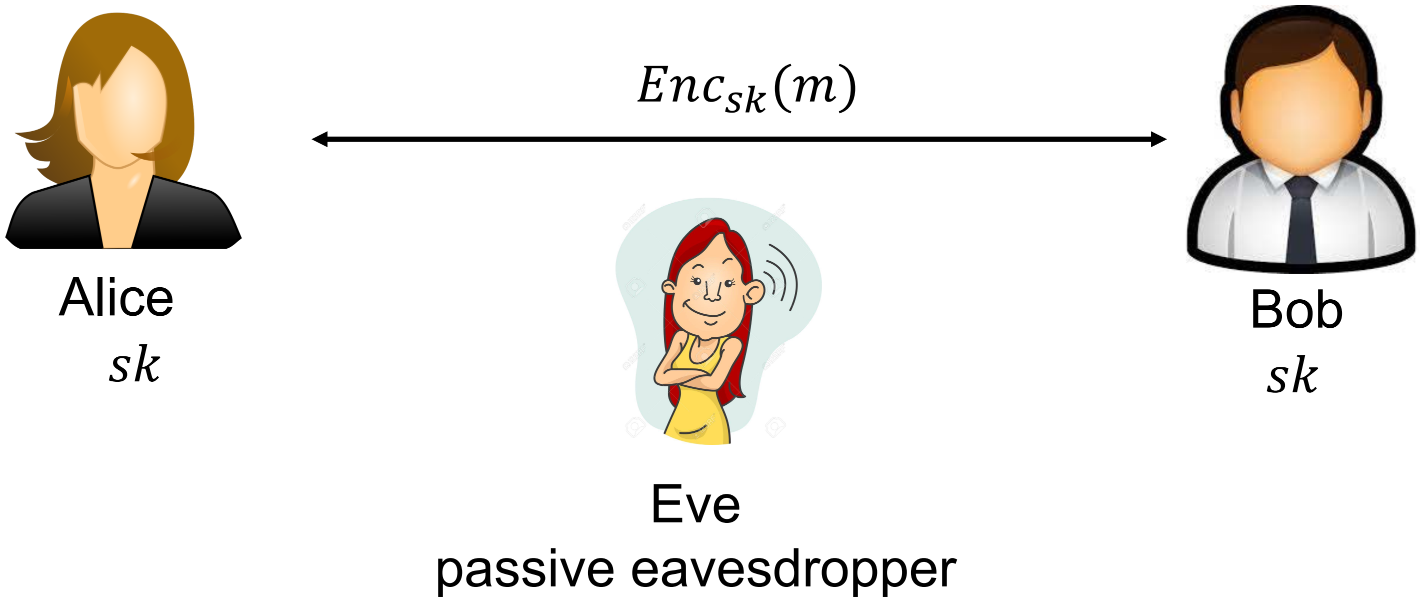
You don't need to understand why AES is this way, just get a sense of its inner workings

Widely used

- Government Standard
  - AES is standardized as Federal Information Processing Standard 197 (FIPS 197) by NIST
  - To protect classified information
- Industry
  - SSL / TLS
  - SSH
  - WinZip
  - BitLocker
  - Mozilla Thunderbird
  - Skype

Used as part of symmetric-key encryption or other crypto tools

Symmetric-key encryption scheme



Alice can send a message m to Bob encrypted using sk and Bob can decrypt it using sk, but Eve cannot learn what the message is other than its <u>length</u>

Symmetric-key encryption scheme

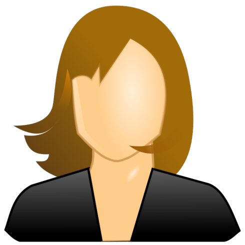
An encryption scheme (Gen, Enc, Dec) is a triple of PPT algs, where

- Key generation $Gen(1^n)$ outputs a secret key sk (n is security parameter)
- Encryption $Enc(sk, m) \to c$ a ciphertext
- Decryption $Dec(sk, c) \to m$

Correctness: For all $n, m,\ sk \leftarrow Gen(1^n)$,

$$Dec\big(sk, Enc(sk, m)\big) = m$$

Security intuition



Alice

sk

$Enc_{sk}(m)$

Eve

Bob

sk

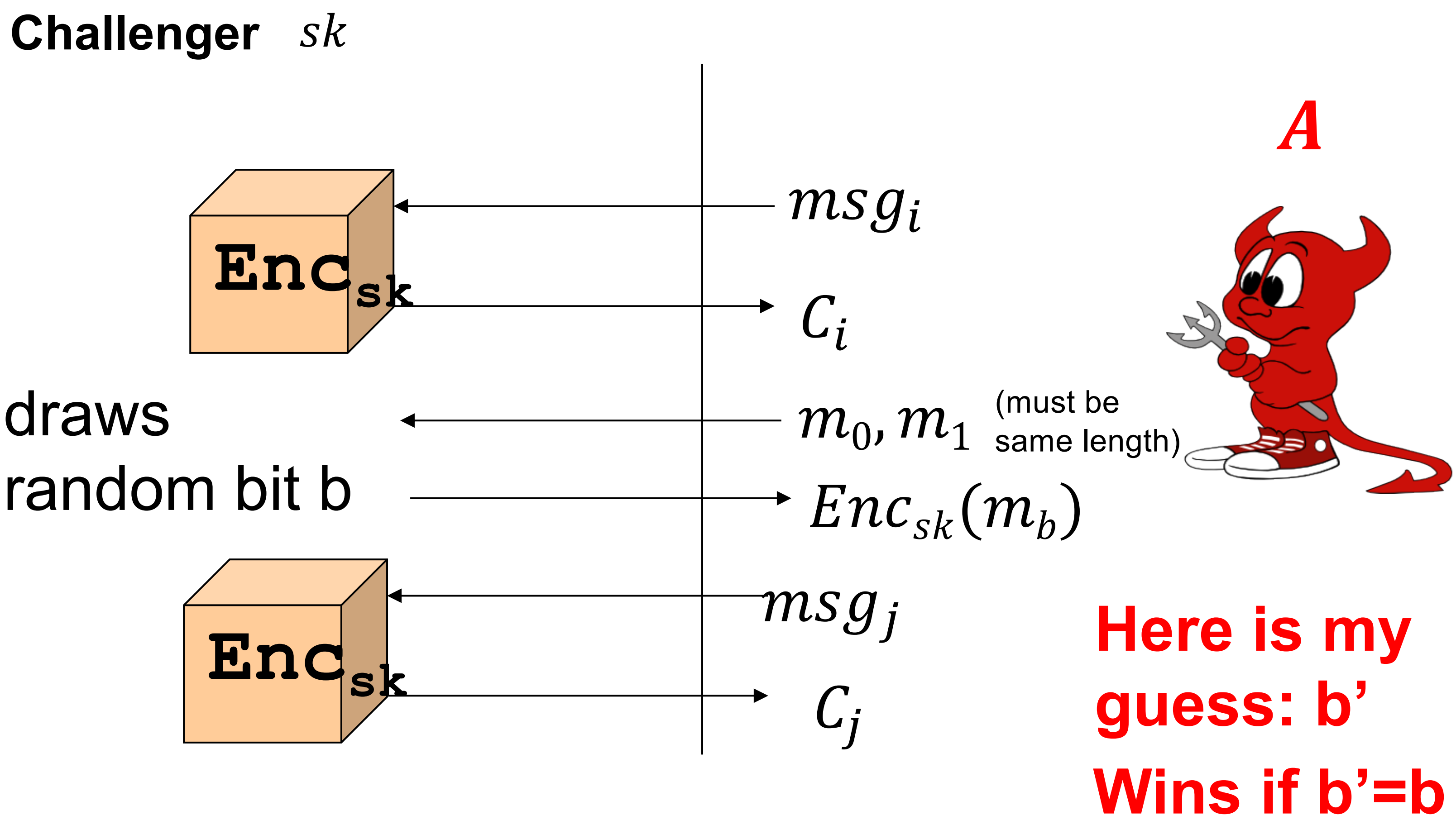
$\boldsymbol{A}$

Eve should learn nothing about the message other than its length,

even if she sees other encryptions of messages she chose

IND-CPA = indistinguishability under chosen plaintext attack

IND-CPA game



Attacker must not win much more than random guessing

IND-CPA

Definition. An encryption scheme (Gen, Enc, Dec) is IND-CPA secure if for every PPT adversary A,

$$\Pr\left[\begin{array}{c} sk \leftarrow Gen(1^n); A^{\{Enc(\mathrm{sk},*)\}}(1^n) = (m_0, m_1), \\ with\ |m_0| = |m_1| \\ b \leftarrow \{0,1\}; A^{Enc(sk,*)}\big(Enc(sk, m_b)\big) = b' : \\ b' = b \end{array}\right] < \frac{1}{2} + \ negl(n)$$

Let's construct an IND-CPA symmetric key encryption scheme
using a block cipher (e.g. AES)
the way people do in practice

Attempt: use a block cipher directly

Let $Enc(sk, m) = f_{sk}(m)$, for f a block cipher.

What problem(s) do we run into?

Problem 1: message might have a different size than the block size of the block cipher

Q: Is $Enc(sk, m) = f_{sk}(m)$ IND-CPA?

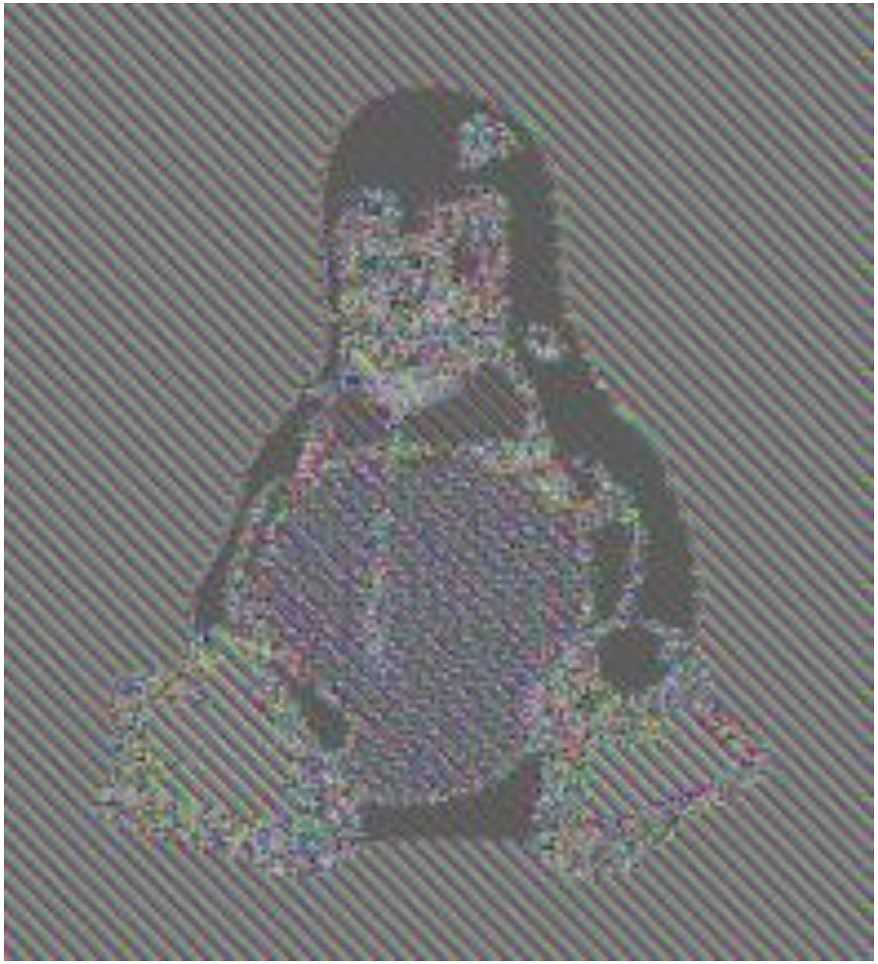
Problem 2: No, because it is deterministic

Here is an attacker that wins the IND-CPA game:

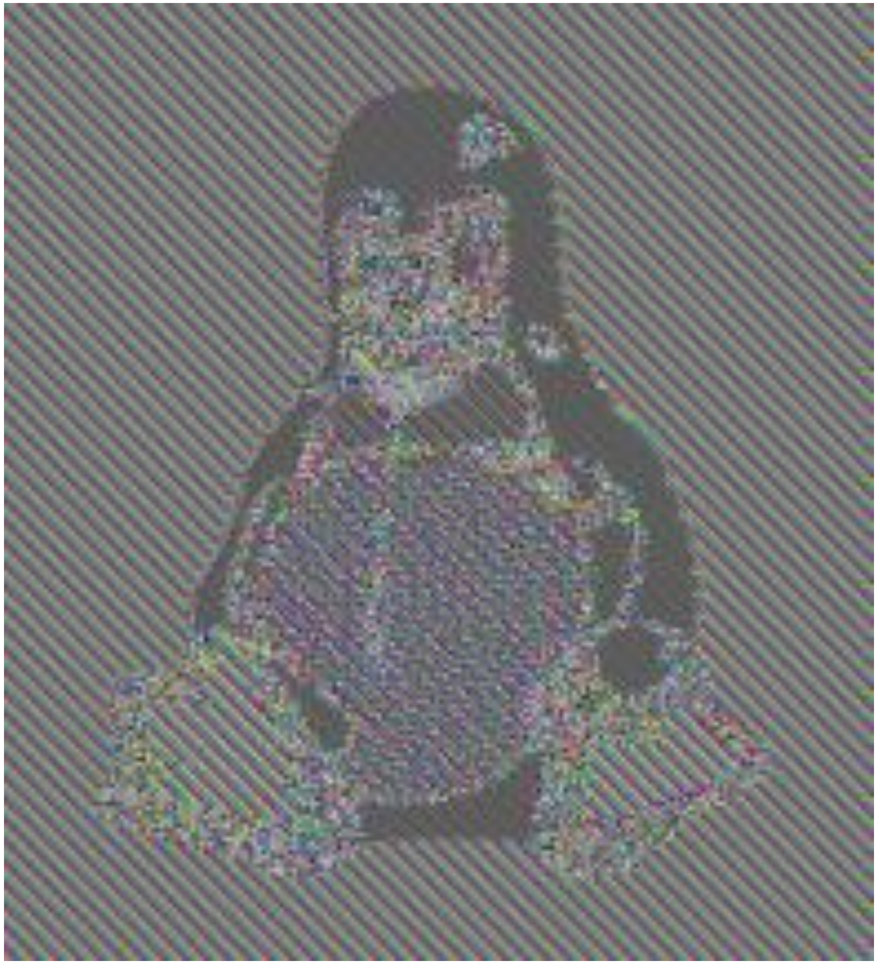
- A asks for encryption of “bread”, receives C_{br}
- Then, A provides (m_0 = bread, m_1 = honey)
- A receives C
- If $C=C_{br}$, Adv says bit was 0 (for “bread”), else A says says bit was 1 (for “honey”)
- Chance of winning is 1

IND-CPA $\Rightarrow$ randomized encryption

Original image



Eack block encrypted with a block cipher



Later (identical) message again encrypted

Goals

1. IND-CPA security even when reusing the same key to encrypt many messages (unlike OTP)
2. Can encrypt messages of any length

use a block cipher in
certain **modes of operation**

Modes of operation

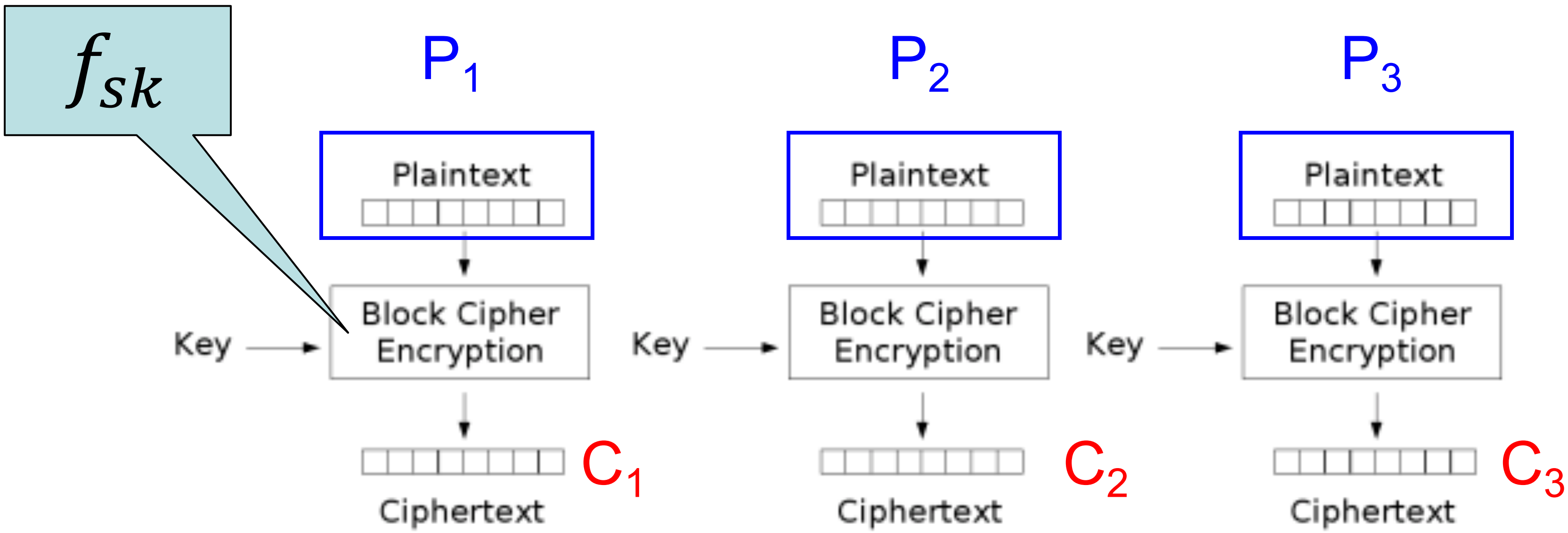
Split the plaintext message in blocks based on the block size of the block cipher

Invoke the block cipher for each block

Need randomness: **nonce or initialization vector IV**

ECB: Encryption

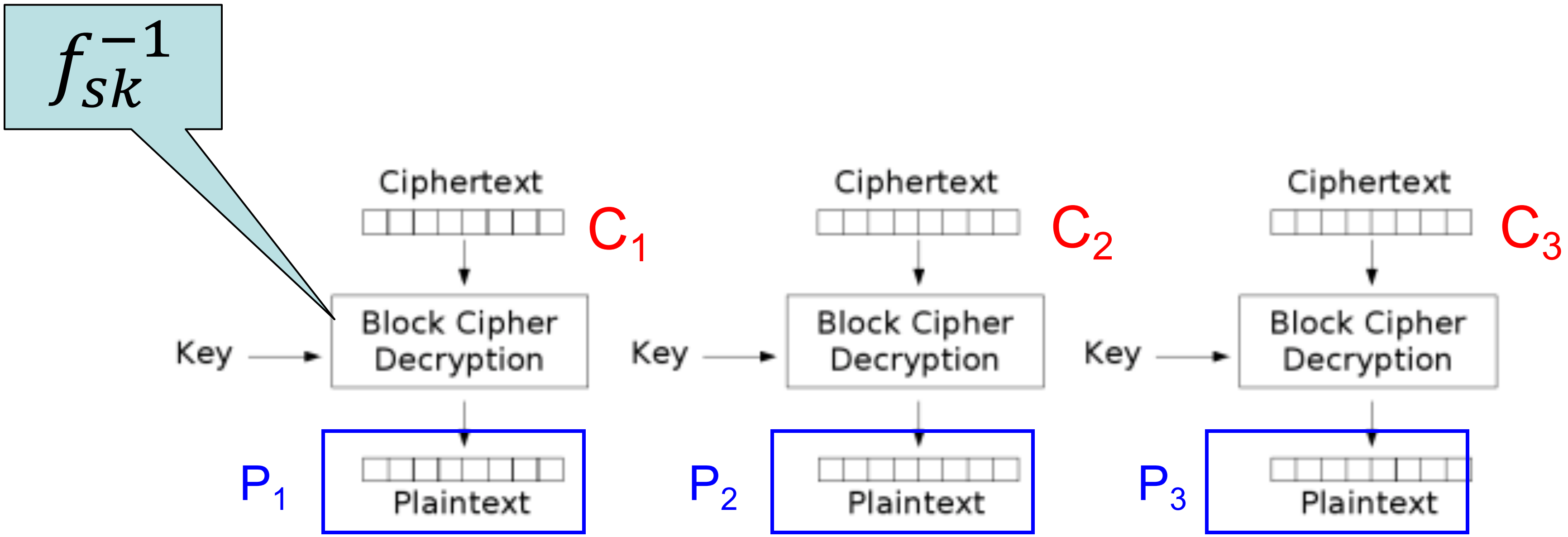
break message m into $P_1|P_2|\ldots|P_m$ each of n bits = block size of block cipher



Electronic Codebook (ECB) mode encryption

$$Enc(sk, P_1|P_2|..|P_m) = (C_1, C_2, \ldots, C_m)$$

ECB: Decryption



Electronic Codebook (ECB) mode decryption

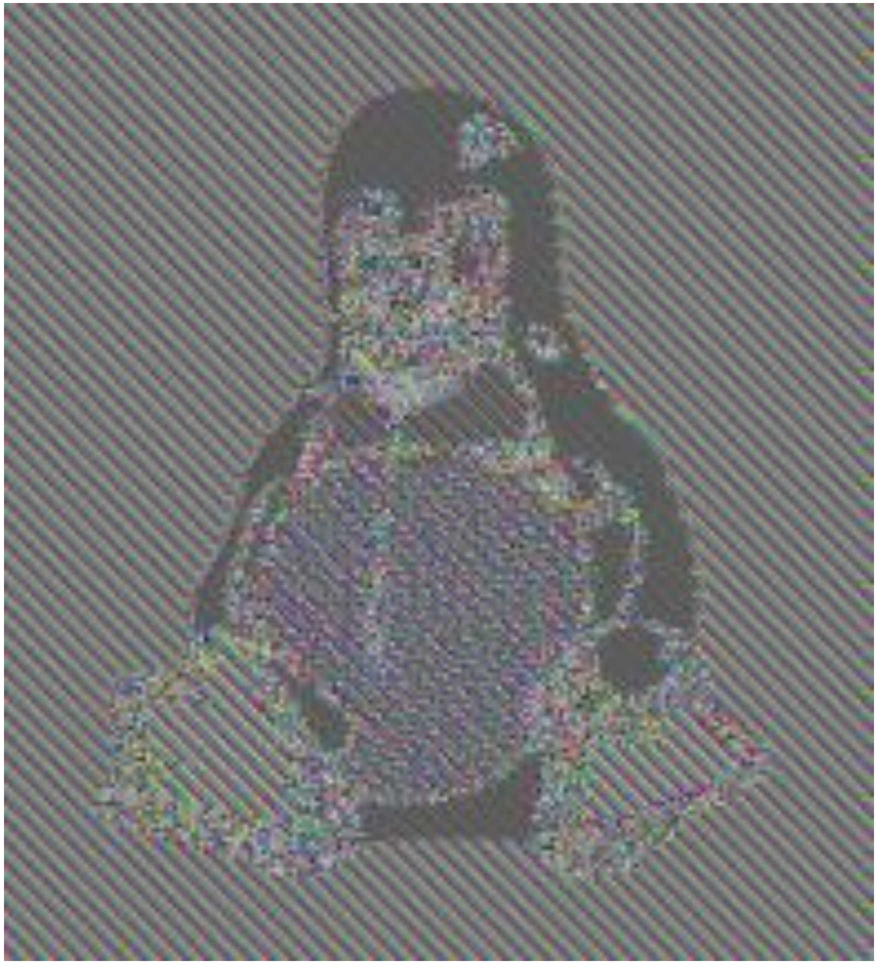
$$Dec(sk, (C_1, C_2, .., C_n)) = (P_1, P_2, .., P_m)$$

What is the problem with ECB?

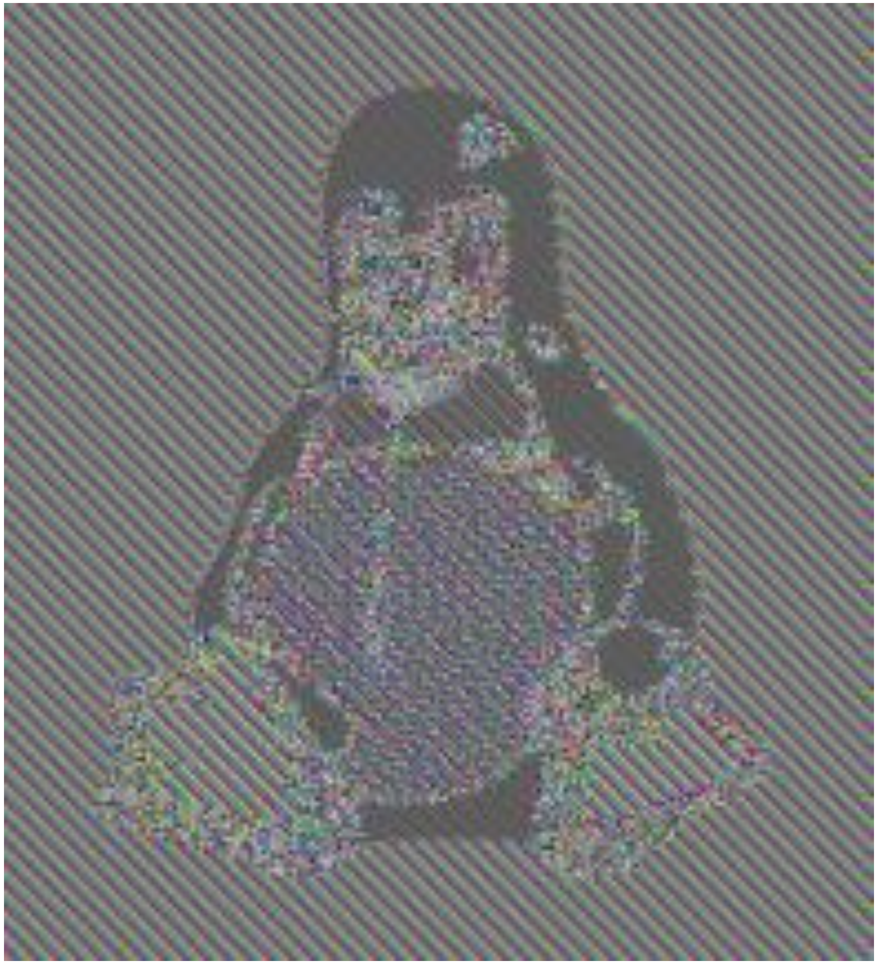
Q: Does this achieve IND-CPA?

A: No, attacker can tell if $P_i = P_j$

Original image



Encrypted with ECB



Later (identical) message again encrypted with ECB

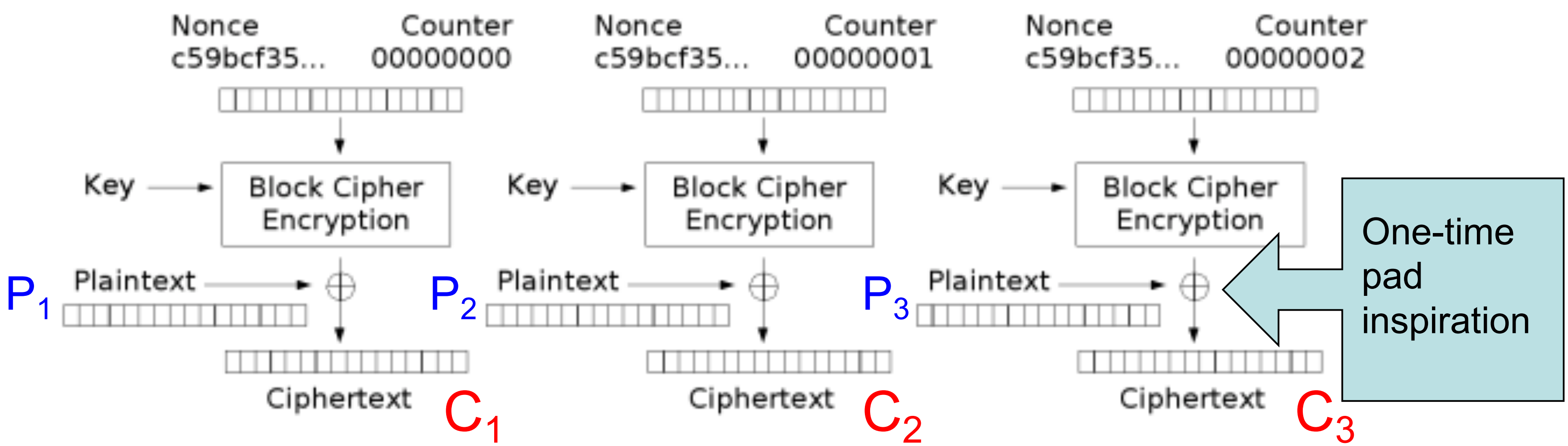
Counter mode (CTR)

CTR: Encryption

$Enc(sk, m)$:

- Split the message m in blocks of size n: $P_1, P_2, P_3, ..$
- Choose a random nonce
- Compute:

Important that nonce does not repeat across different encryptions (choose it at random from large space)



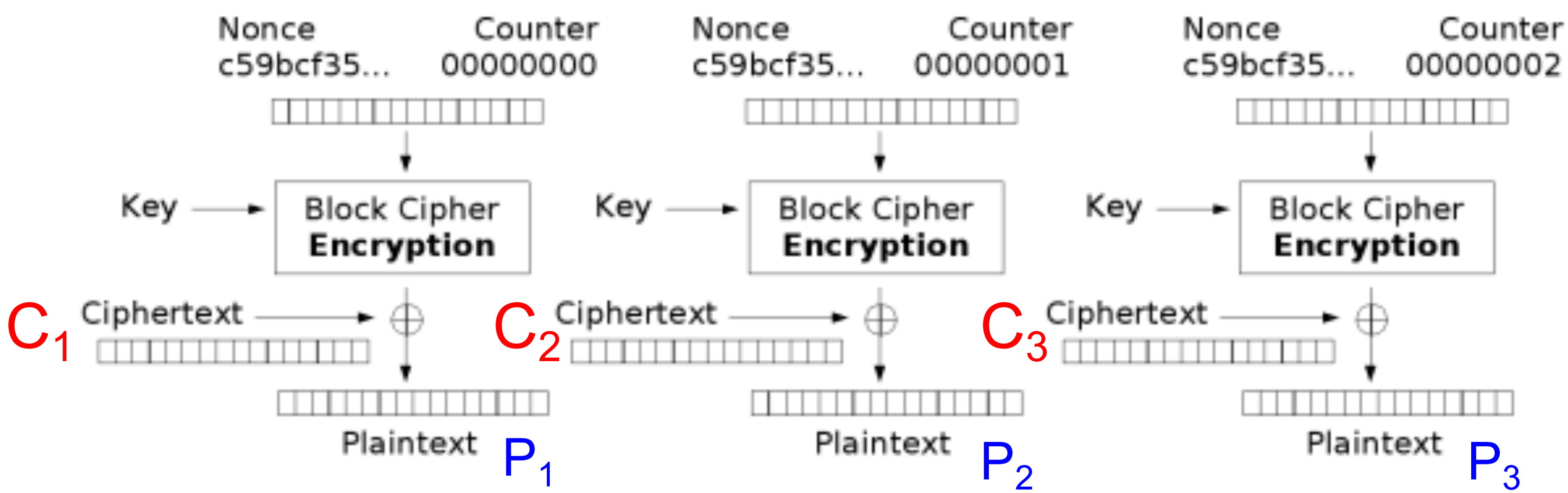
Counter (CTR) mode encryption

$$Enc(sk, m) = (nonce, C_1, C_2, \ldots,)$$

CTR: Decryption

$Dec(sk,$ ciphertext= $[nonce, C_1, C_2, C_3, \ldots].)$:

- Take nonce out of the ciphertext
- Split the ciphertext in blocks of size n: $C_1, C_2, C_3, ..$
- Now compute this:

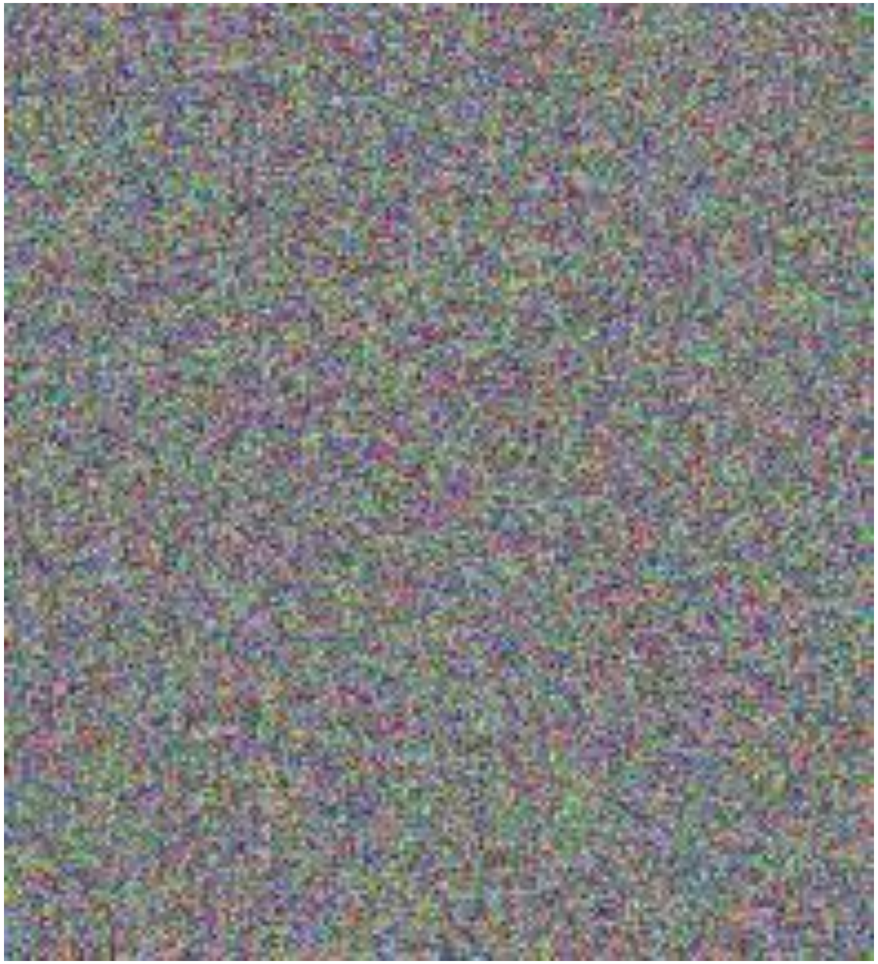


Counter (CTR) mode decryption

- Output the plaintext m as the concatenation of $P_1, P_2, P_3, \ldots$

Note, CTR decryption uses block cipher's *encryption*, not decryption

Original image



Encrypted with CBC

PRP $\Rightarrow$ IND-CPA enc

Claim. If F is a pseudorandom permutation ensemble, using F in CTR mode results in an IND-CPA symmetric-key encryption scheme.

Informal proof. By contradiction. Assume A breaks IND-CPA and construct B that breaks PRP property. B runs A using the PRP oracles.

Summary

PRPs and how to construct them

- The theory way:

  Luby-Rackoff'86: PRF $\Rightarrow$ PRP

- The practical way:

  Rijmen and Daemen'03: AES proposal to NIST

Symmetric-key encryption and IND-CPA

- Construct using block cipher in cipher chaining modes