# Symmetric-Key Encryption

***CS 161: Computer Security***

**Prof. Raluca Ada Popa**

**Feb 7, 2019**

# Announcements

- Midterm 1 is Wednesday February 19, 8:00-9:30pm
- Midterm 2 is Monday April 6, 8:30-10:00pm
- Homework 1 is due today
- Project 1 is out. I encourage you to get started early. We had to update the VM on Thursday – if you downloaded it before then, please delete and re-download.

# Block cipher

A function $E : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. Once we fix the key K, we get

$E_K : \{0,1\}^n \rightarrow \{0,1\}^n$ defined by $E_K(M) = E(K,M)$.

Three properties:

- Correctness:
  - $E_K(M)$ is a permutation (bijective/ one-to-one function)
- Efficiency
- Security

# Block cipher security

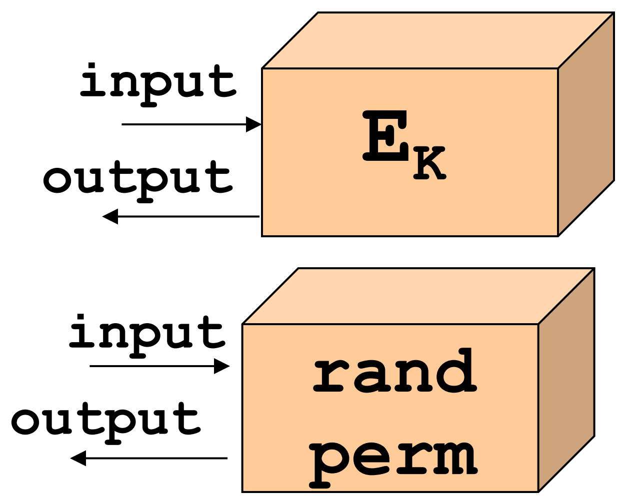
For an unknown key K, $E_K$ “behaves” like a random permutation

For all polynomial-time attackers, for a randomly chosen key K, the attacker **cannot distinguish** $E_K$ from a random permutation

# Block cipher: security game

- Attacker is given two boxes, one for $E_K$ and one for a random permutation (also called “oracles”)
- Attacker does not know which is which (they were shuffled randomly)
- Attacker can give inputs to each box, look at the output, as many times as he/she desires
- Attacker must guess which is $E_K$

**??? Which is $E_K$???**



# Security game

For all polynomial-time attackers,

Pr[attacker wins game] <= ½+negl

# Security

For an unknown key K, $E_K$ "behaves" like a random permutation

Q: If the attacker receives $E_K(x)$ and nothing else about x, can he determine x?

A: No. If he could, he could distinguish the block cipher from a random permutation

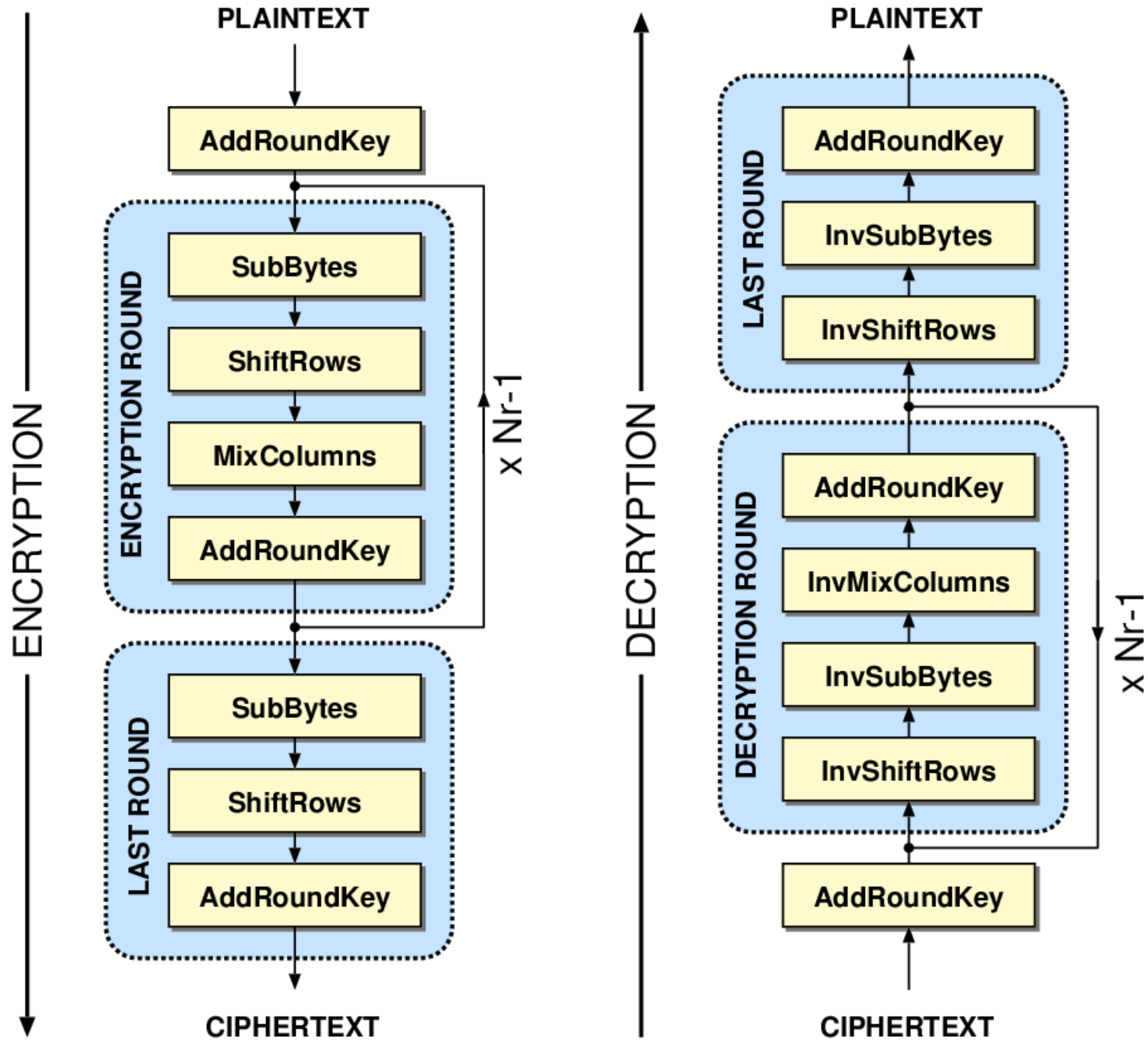
Similarly, if the attacker receives only $E_K(x_1)$, $E_K(x_2)$, …, $E_K(x_n)$. The only information he sees is if any $x_i = x_j$ but not their values

**So block ciphers provide some confidentiality, but not enough for IND-CPA (because they have this deterministic leakage)**

# Advanced Encryption Standard (AES)

- Block cipher developed in 1998 by Joan Daemen and Vincent Rijmen
- Recommended by US National Institute for Standard and Technology (NIST)
- Block length n = 128bits, key length k = 256bits

# AES ALGORITHM

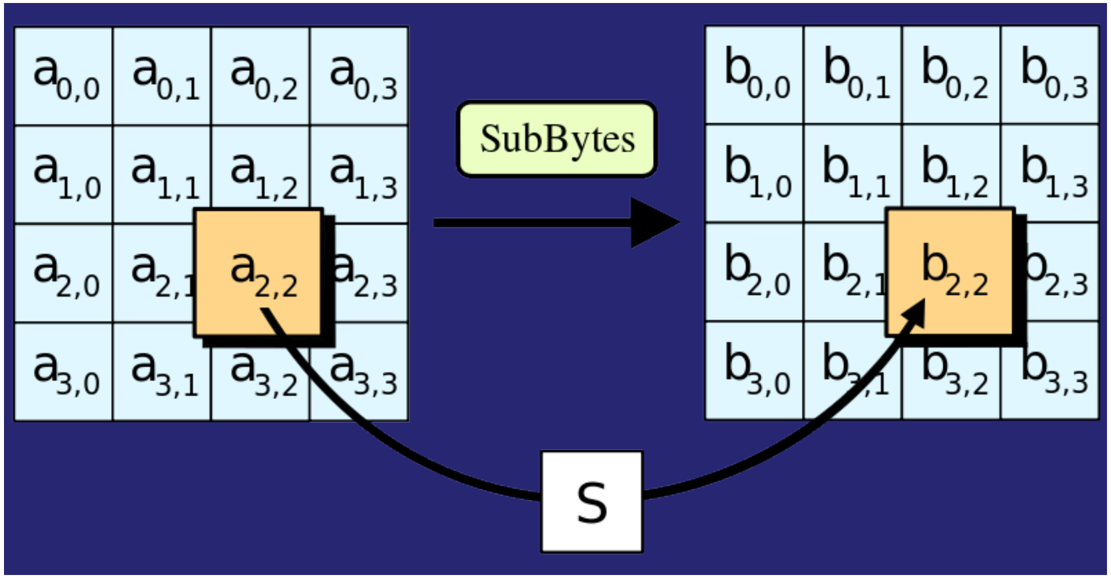



- 14 cycles of repetition for 256-bit keys.

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

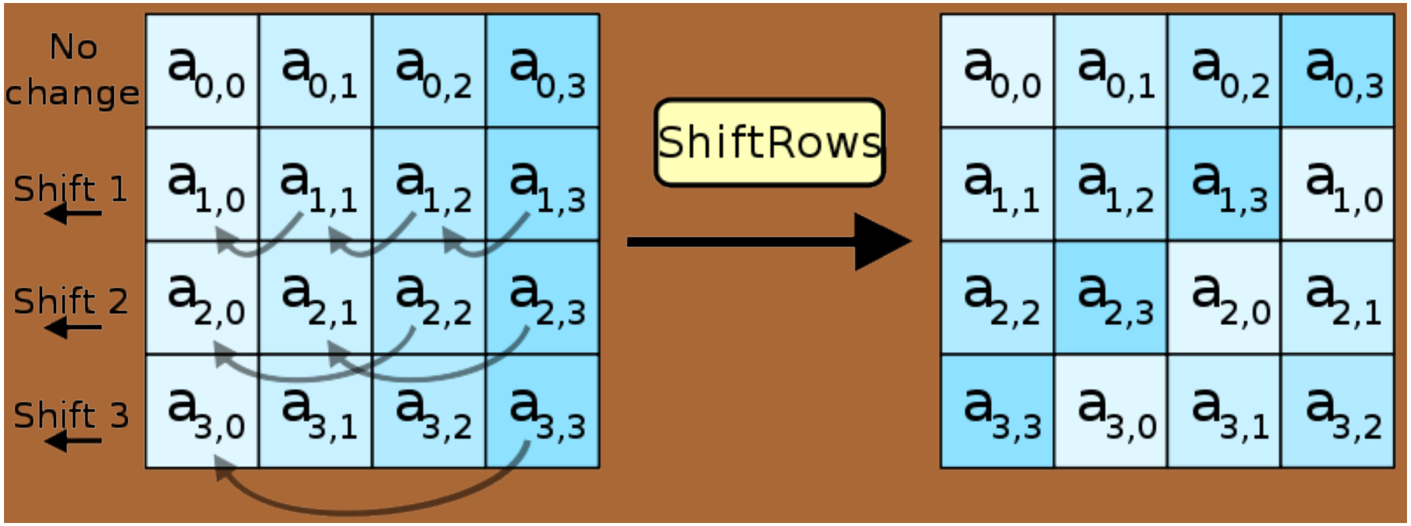
# Algorithm Steps - Sub bytes

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

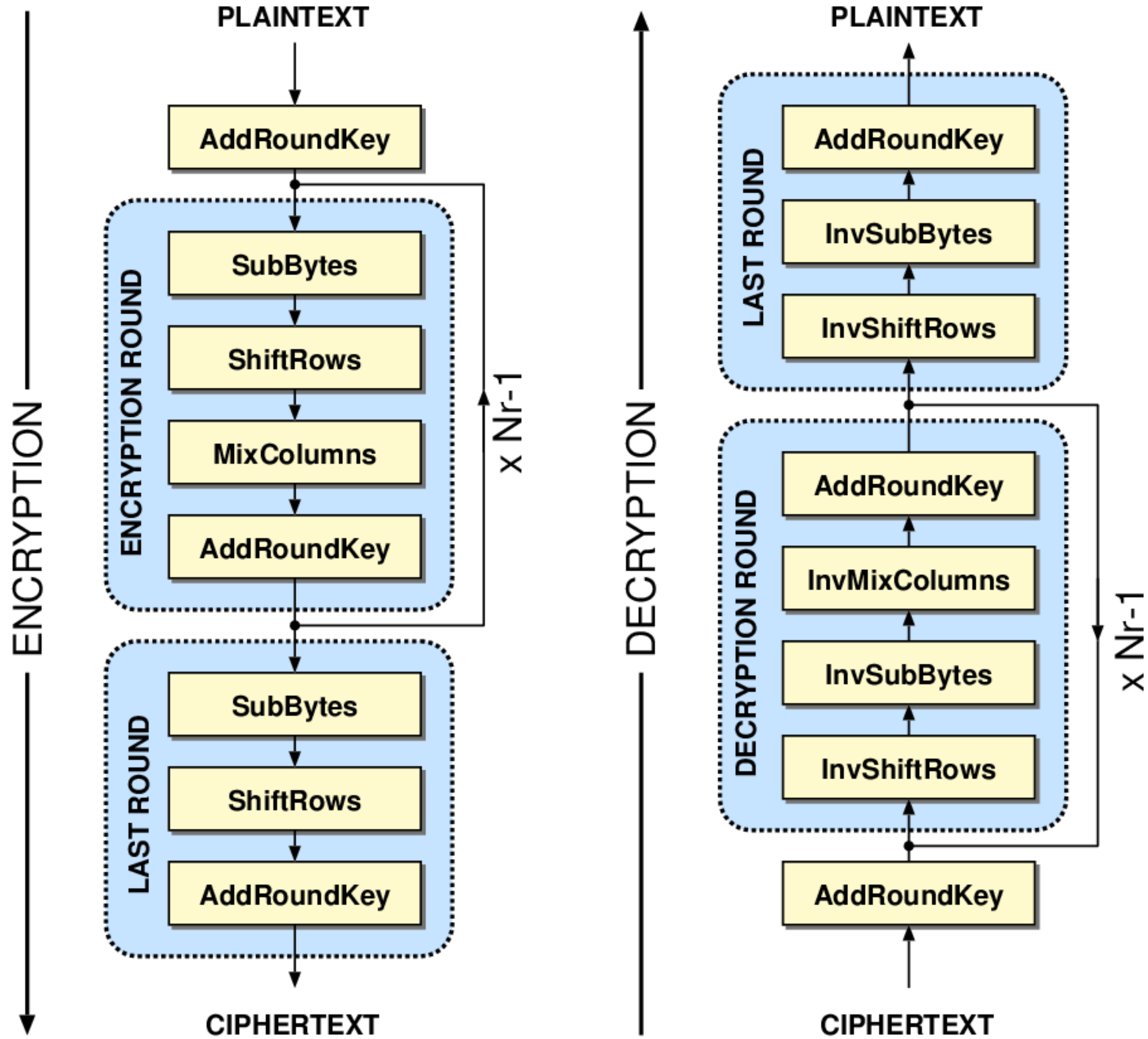


# Shift Rows

- 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**

# Why secure?

- Not provably secure but we assume it is
- By “educated” belief/assumption: it stood the test of time and of much cryptanalysis (field studying attacks on encryption schemes)
- Various techniques to boost confidence in its security
- If we were to have something provably secure, P is not NP

# Uses

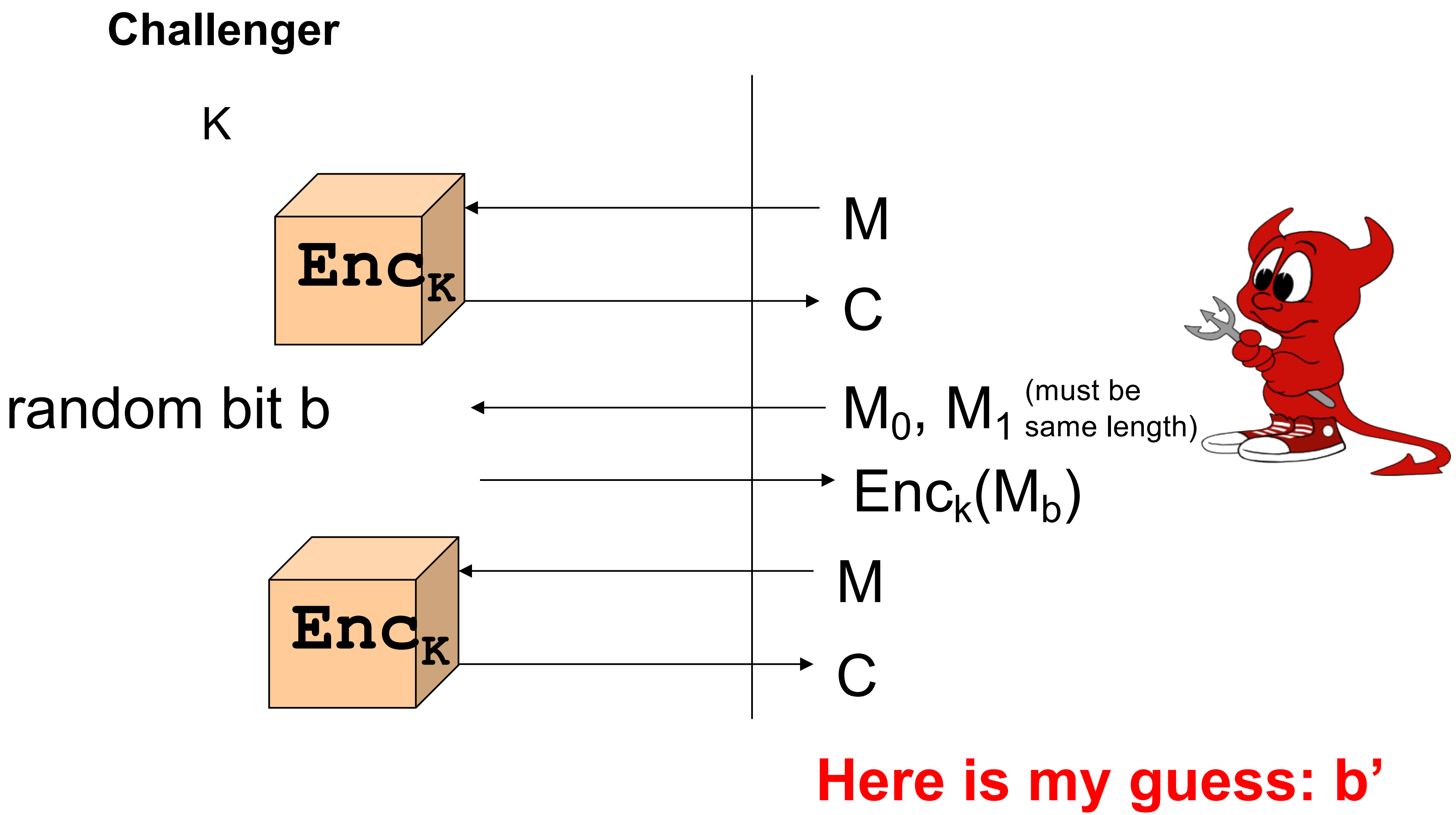
- Government Standard
  - AES is standardized as Federal Information Processing Standard 197 (FIPS 197) by NIST
  - To protect classified information
- Industry
  - SSL / TLS
  - SSH
  - WinZip
  - BitLocker
  - Mozilla Thunderbird
  - Skype

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

# Desired security: Indistinguishability under chosen plaintext attack (IND-CPA)

- Strong security definition
- Nothing leaks about the encrypted value other than its length

# IND-CPA (Indistinguishability under chosen plaintext attack)



# IND-CPA

An encryption scheme is IND-CPA if
for all polynomial-time adversaries

$$\Pr[\text{Adv wins game}] <= \frac{1}{2} + \text{negligible}$$

Note that IND-CPA requires that the encryption scheme is randomized

(An encryption scheme is deterministic if it outputs the same ciphertext when encrypting the same plaintext; a randomized scheme does not have this property)

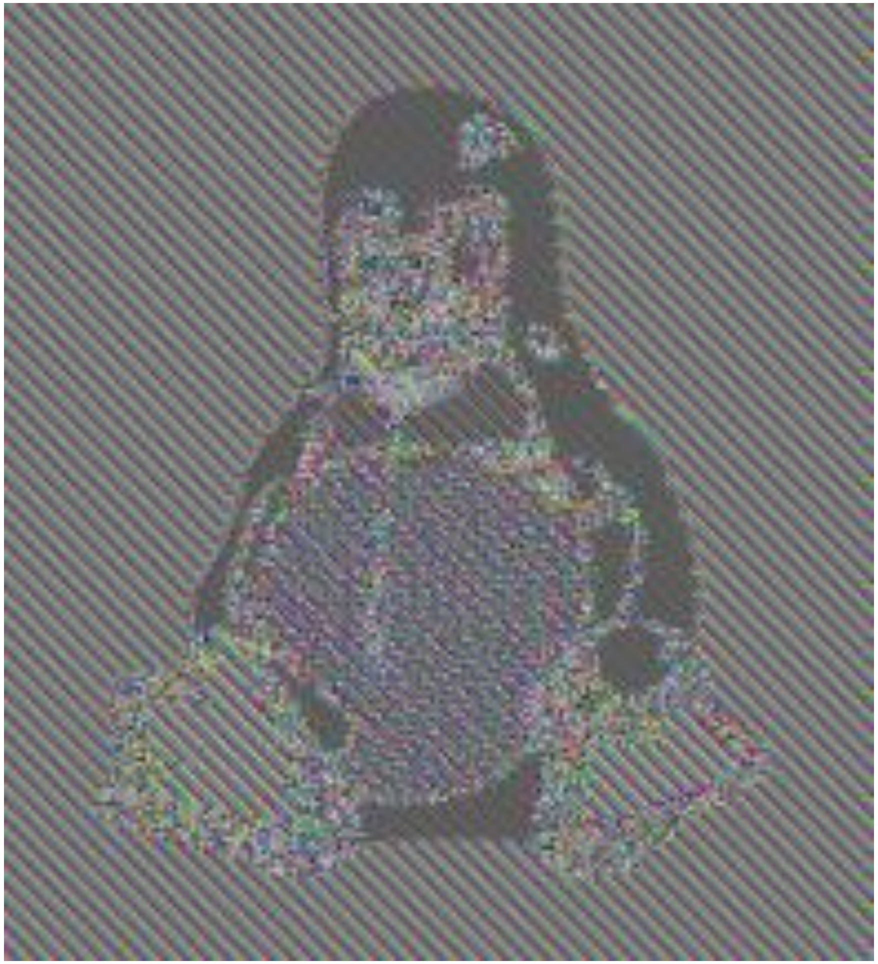
# Are block ciphers IND-CPA?

Recall: $E_K : \{0,1\}^n \rightarrow \{0,1\}^n$ is a permutation (bijective)

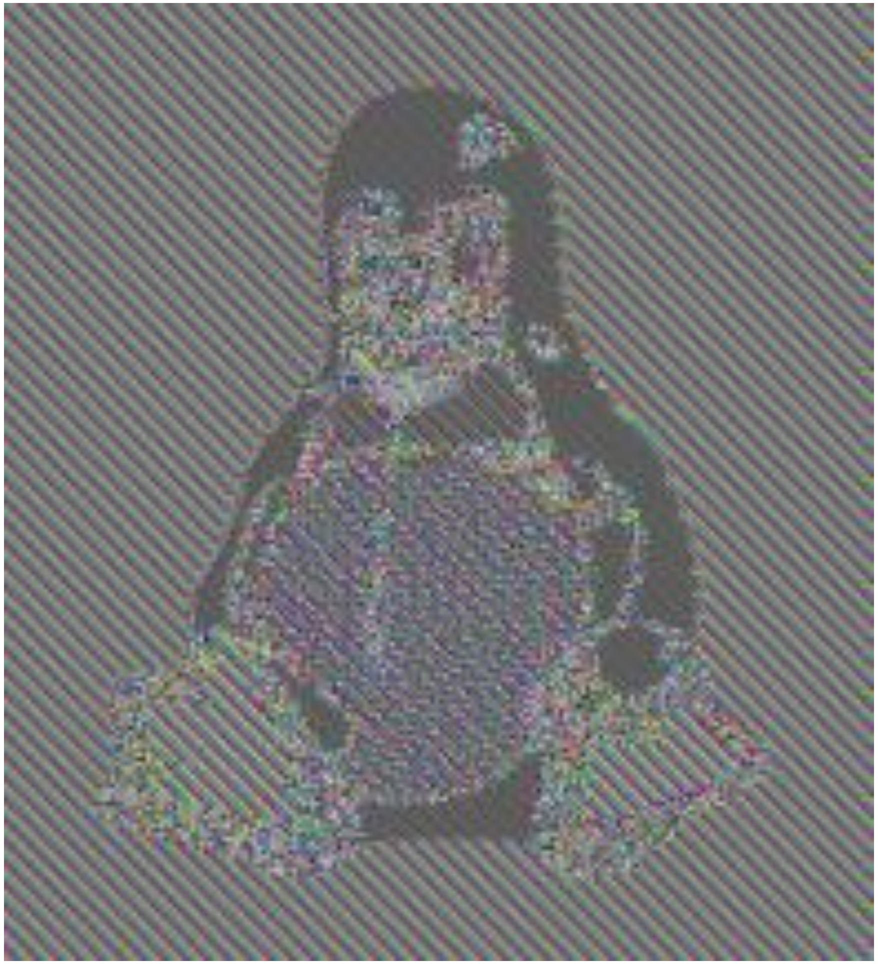
# Q: Are block ciphers IND-CPA?

- A: No, because they are deterministic
- Here is an attacker that wins the IND-CPA game:
  - Adv asks for encryptions of “bread”, receives $C_{br}$
  - Then, Adv provides ($M_0$ = bread, $M_1$ = honey)
  - Adv receives C
  - If $C=C_{br}$, Adv says bit was 0 (for “bread”), else Adv says says bit was 1 (for “honey”)
  - Chance of winning is 1

Original image



Each block encrypted with a block cipher



Later (identical) message again encrypted

# Why block ciphers not enough for encryption by themselves?

- Can only encipher messages of a certain size
- Not IND-CPA (If message is encrypted twice, attacker knows it is the same message)

# Use block ciphers to construct symmetric-key encryption

- Want two properties:
  - IND-CPA security even when reusing the same key to encrypt many messages (unlike OTP)
  - Can encrypt messages of any length

- Build symmetric key encryption on block ciphers:
  - Can be used to encrypt long messages
  - Wants to hide that same block is encrypted twice
- Uses block ciphers in certain modes of operation
- There are many block ciphers besides AES

# Modes of operation

Chain block ciphers **in certain modes of operation**

– Invoke block cipher multiple times on inputs related to other blocks

Need some **initial randomness IV** **(initialization vector)**

Q: Why?

A: To prevent the encryption scheme from being deterministic

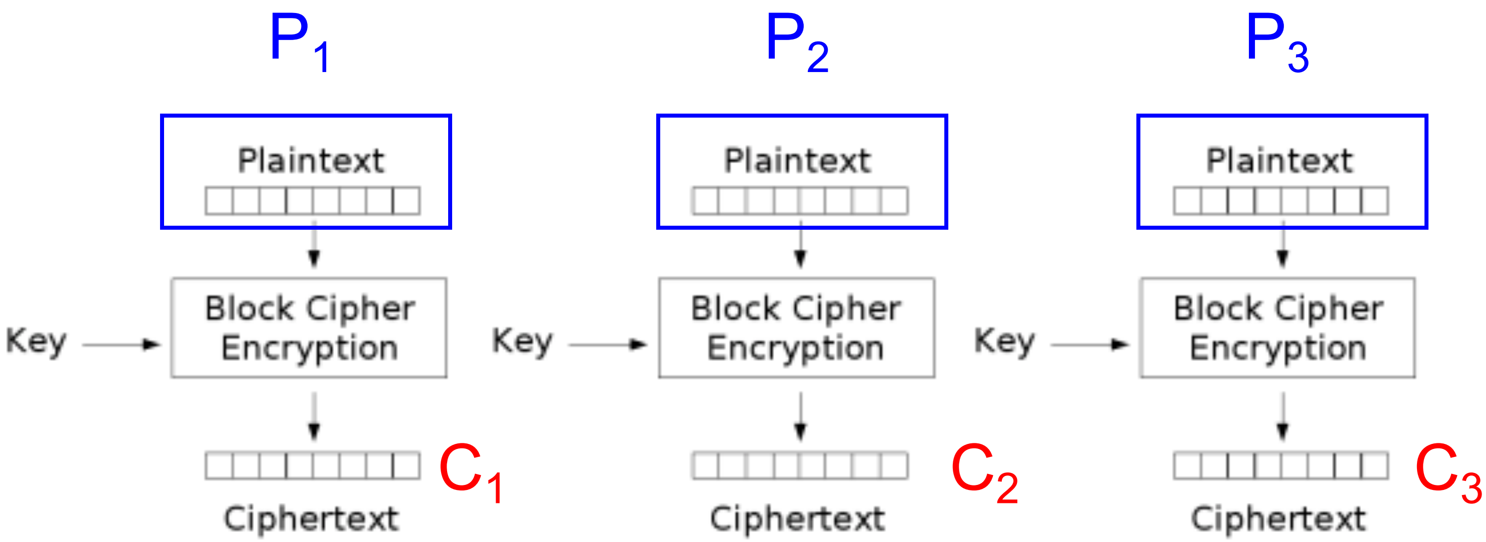
# Electronic Code Book (ECB)

- Split message M in blocks $P_1$, $P_2$, … where each plaintext block is as large as n, the block cipher input size
  - For now assume that M is a multiple of n, but we will see how to pad if that is not the case
- Each block is a value which is substituted, like a codebook
- Each block is encoded independently of the other blocks

$$C_i = E_K(Pi)$$

# ECB: Encryption

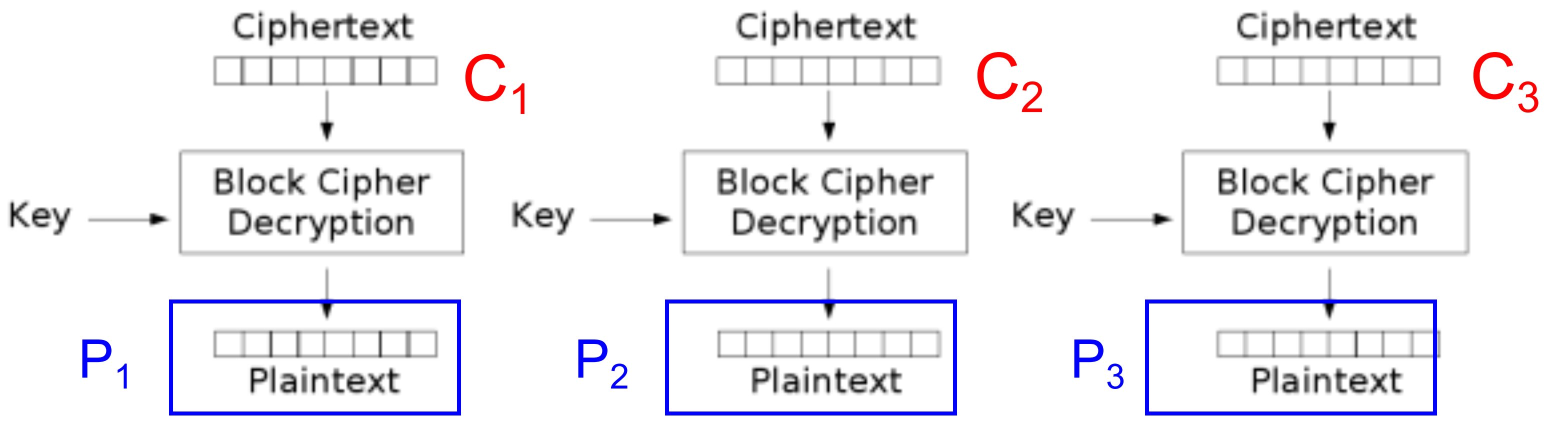
break message M into $P_1|P_2|\ldots|P_m$ each of n bits (block cipher input size)




Electronic Codebook (ECB) mode encryption

$$\mathbf{Enc(K,\ P_1|P_2|..|P_m) = (C_1,\ C_2,...,\ C_m)}$$

# ECB: Decryption




Electronic Codebook (ECB) mode decryption

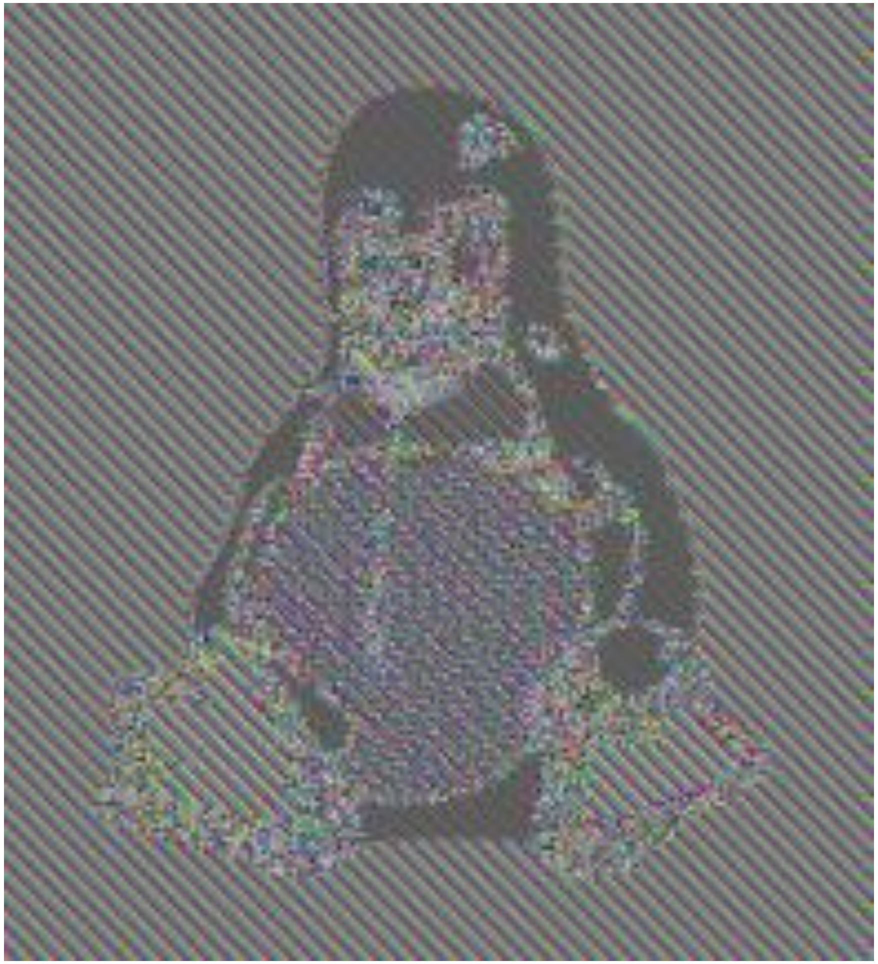
$$\mathbf{Dec}(K, (C_1, C_2, .., C_n)) = (P_1, P_2, .., P_m)$$

What is the problem with ECB?

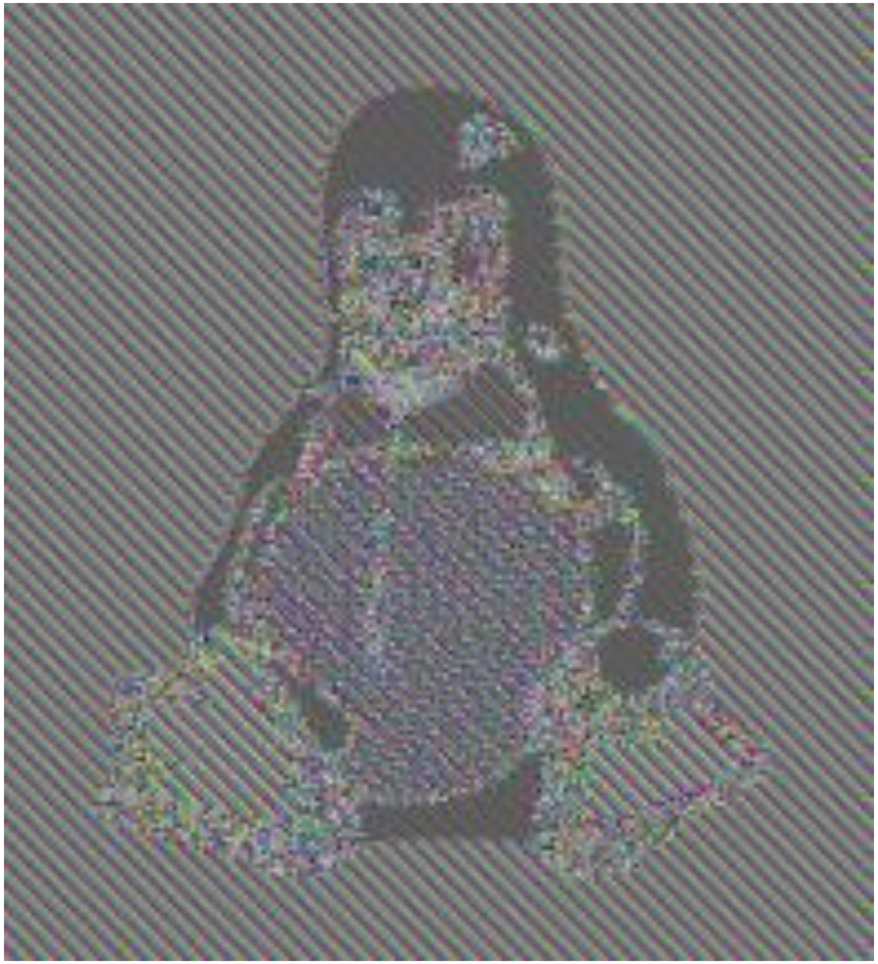
# Q: Does this achieve IND-CPA?

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

Original image



Encrypted with ECB

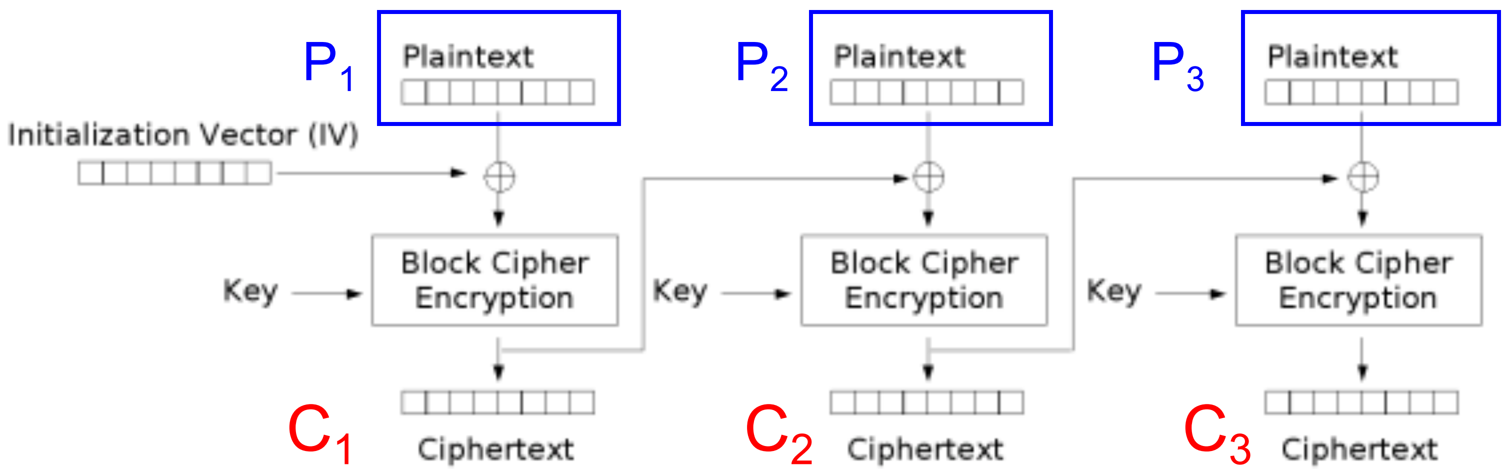


Later (identical) message again encrypted with ECB

# CBC: Encryption

Break message M into $P_1|P_2|\ldots|P_m$

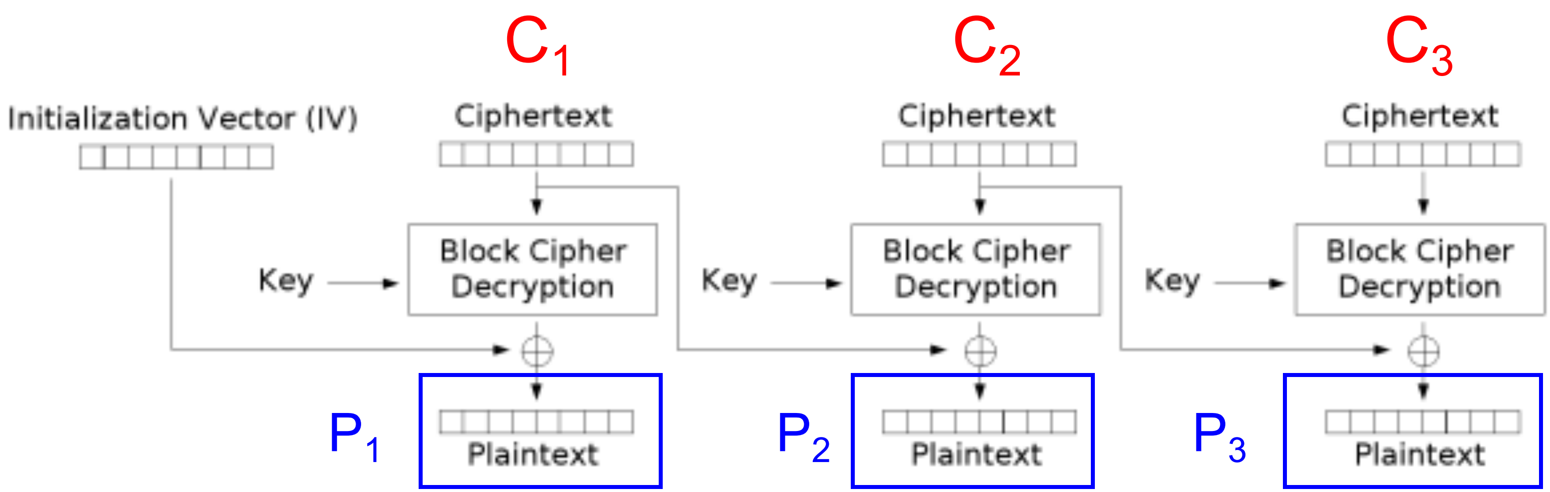
Choose a random IV (it may not repeat for messages with same $P_1$ , it is not secret and is included in the ciphertext)




Cipher Block Chaining (CBC) mode encryption

$$\mathbf{Enc}(K,\ P_1|P_2|..|P_m) = (IV,\ C_1,\ C_2,\ldots,\ C_m)$$

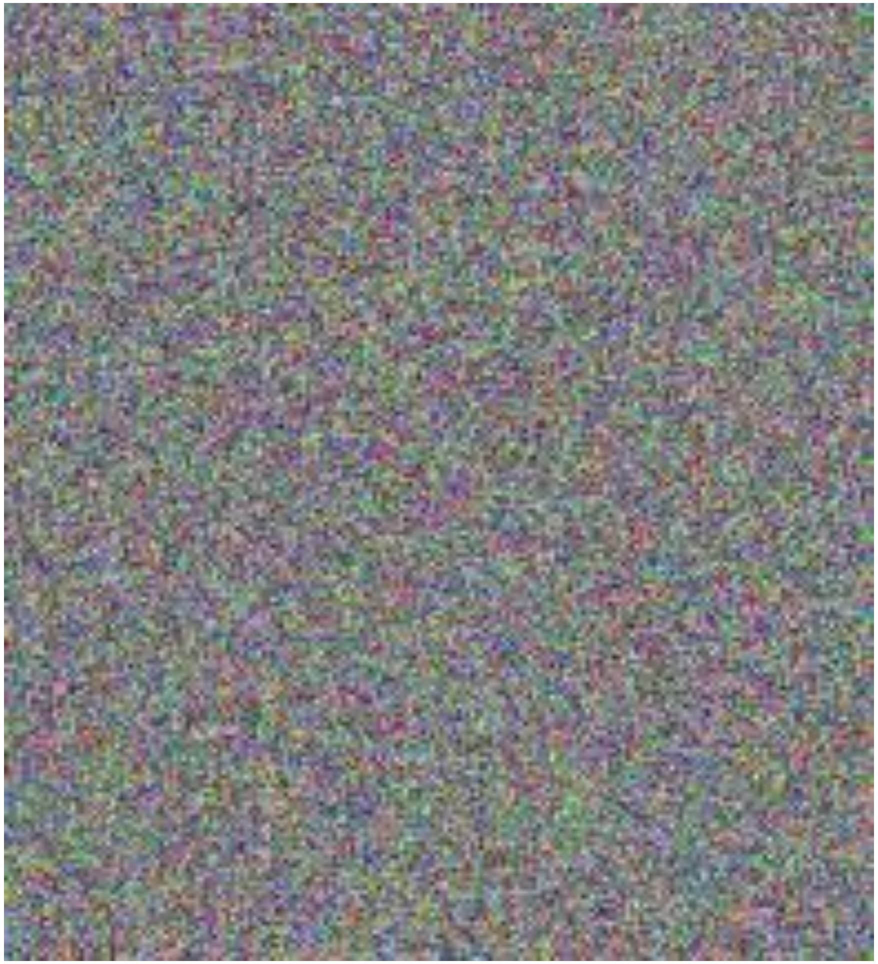
# CBC: Decryption




Cipher Block Chaining (CBC) mode decryption

$$\mathtt{Dec(K,\ (IV,C_1,C_2,..,C_m))\ =\ (P_1,\ P_2,\ ..,\ P_m)}$$

Original image



Encrypted with CBC

# CBC

Popular, still widely used
Achieves IND-CPA

Slight caveat: sequential encryption, hard to parallelize

CTR mode gaining popularity

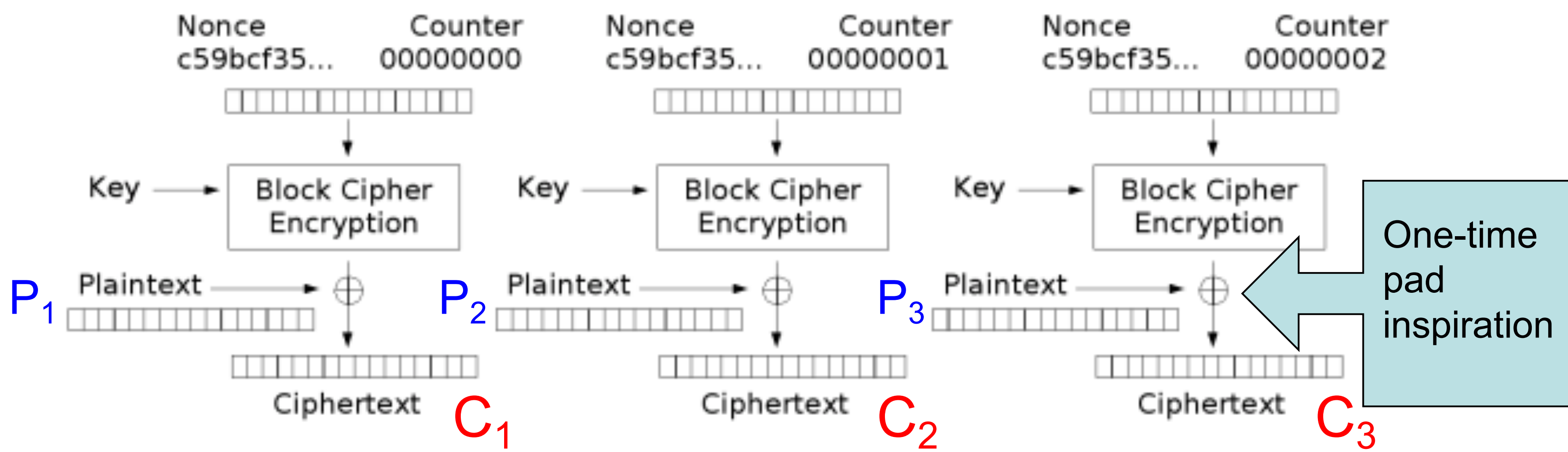
# Counter mode (CTR)

# CTR: Encryption

Enc(K, plaintext):

- If n is the block size of the block cipher, split the plaintext in blocks of size n: $P_1, P_2, P_3, ..$
- Choose a random nonce (Nonce = Same as IV)
- Now compute:

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



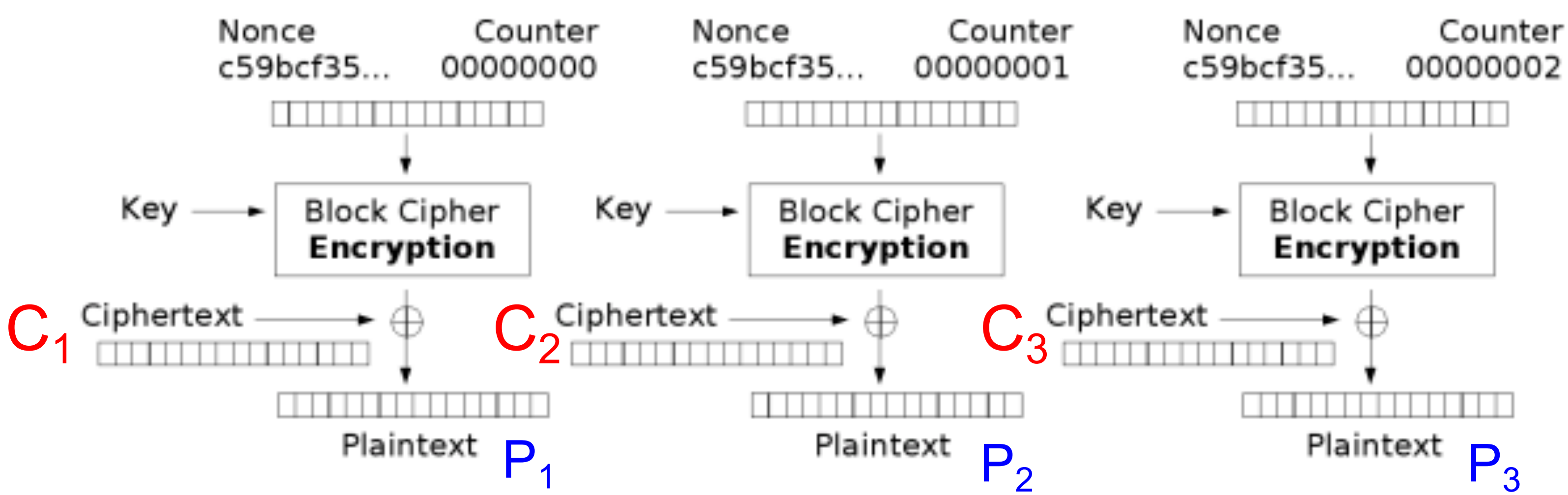

Counter (CTR) mode encryption

- The final ciphertext is (nonce, $C_1$, $C_2$, $C_3$)

# CTR: Decryption

Dec(K, ciphertext=[nonce,$C_1$, $C_2$, $C_3$,…].):

- Take nonce out of the ciphertext
- If n is the block size of the block cipher, split the ciphertext in blocks of size n: $C_1$, $C_2$, $C_3$,..
- Now compute this:



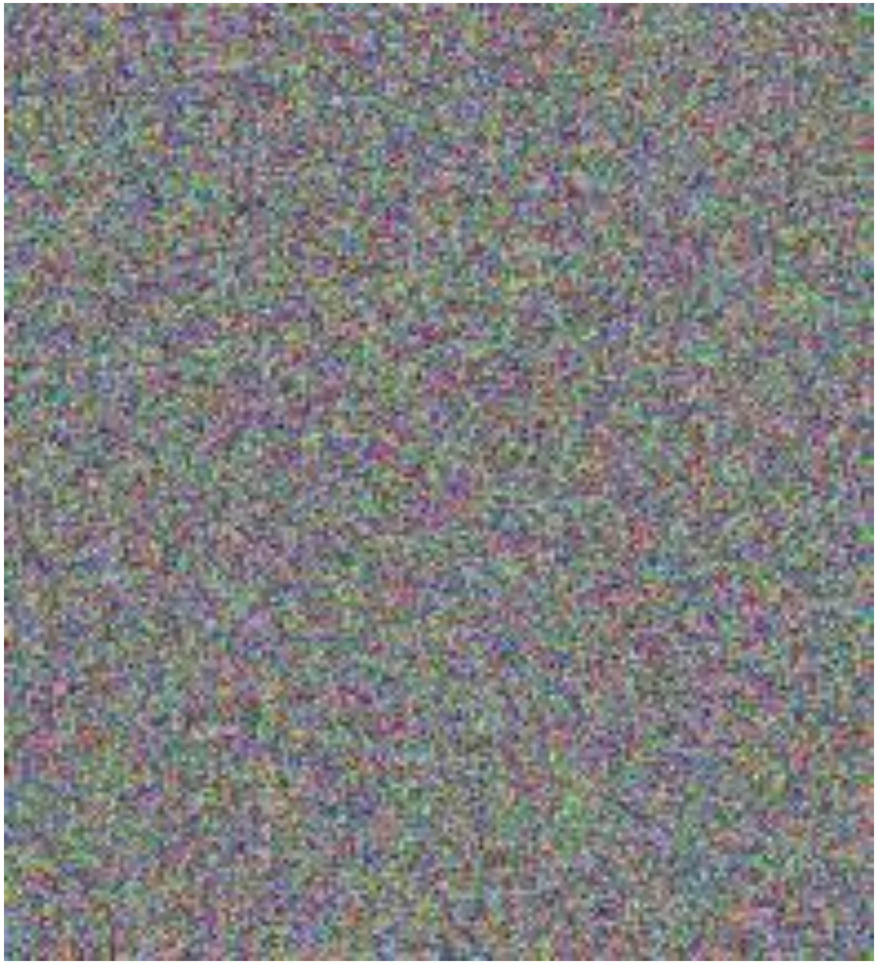

Counter (CTR) mode decryption

- Output the plaintext as the concatenation of $P_1$, $P_2$, $P_3$, ...

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

Would you like me to explain CTR one more time?

Original image



Encrypted with CBC

# CBC vs CTR

**Security**: If no reuse of nonce/IV, **both are IND-CPA.**

**Speed:** Both modes require the same amount of computation, but CTR is parallelizable for encryption as well (CBC was parallelizable for decryption but not for encryption)

# Padding

If messages might not be multiple of n, the block cipher length, we pad the message before encryption and unpad after decryption.

Bad padding:

`message 00000000000`

*n bits*

Q: Why bad?

A: When unpadding, it is not clear which 0s belong to the padding vs the message

Good padding:

`message 10000000000`

*n bits*

If the message is exactly n bits long, still pad by adding another n bits.

# Pseudorandom generator (PRG)

# Pseudorandom Generator (PRG)

- Given a seed, it outputs a sequence of random bits

  PRG(seed) -> random bits

- It can output arbitrarily many random bits

# PRG security

- Can PRG(K) be truly random?

No. Consider key length |K|=k. Have $2^k$ possible initial states of PRG. Deterministic from then on. There are more random states.

- A secure PRG suffices to "look" random ("pseudo") to an attacker (no attacker can distinguish it from a random sequence)

# Example of PRG: using block cipher in CTR mode

If you want m random bits, and a block cipher with $E_k$ has n bits, apply the block cipher m/n times and concatenate the result:

$PRG(K \mid IV) = E_k(IV|1) \mid E_k(IV| 2) \mid E_k(IV|3) \ldots E_k(IV| ceil(m/n))$, where | is concatenation

# Application of PRG: Stream ciphers

- Another way to construct encryption schemes
- Similar in spirit to one-time pad: it XORs the plaintext with some random bits
- But random bits are not the key (as in one-time pad) but are output of a pseudorandom generator PRG

# Application of PRG: Stream cipher

Enc(K, M):

- Choose a random value IV
- C = PRG(K | IV) XOR M
- Output (IV, C)

Q: How decrypt?

A: Compute PRG(K | IV) and XOR with ciphertext C

Q: What is advantage over OTP?

A: Can encrypt any message length because PRG can produce any number of random bits, and multiple times because IV is chosen at random in Enc

# Summary

- Desirable security: IND-CPA
- Block ciphers have weaker security than IND-CPA
- Block ciphers can be used to build IND-CPA secure encryption schemes by chaining in careful ways
- Stream ciphers provide another way to encrypt, inspired from one-time pads