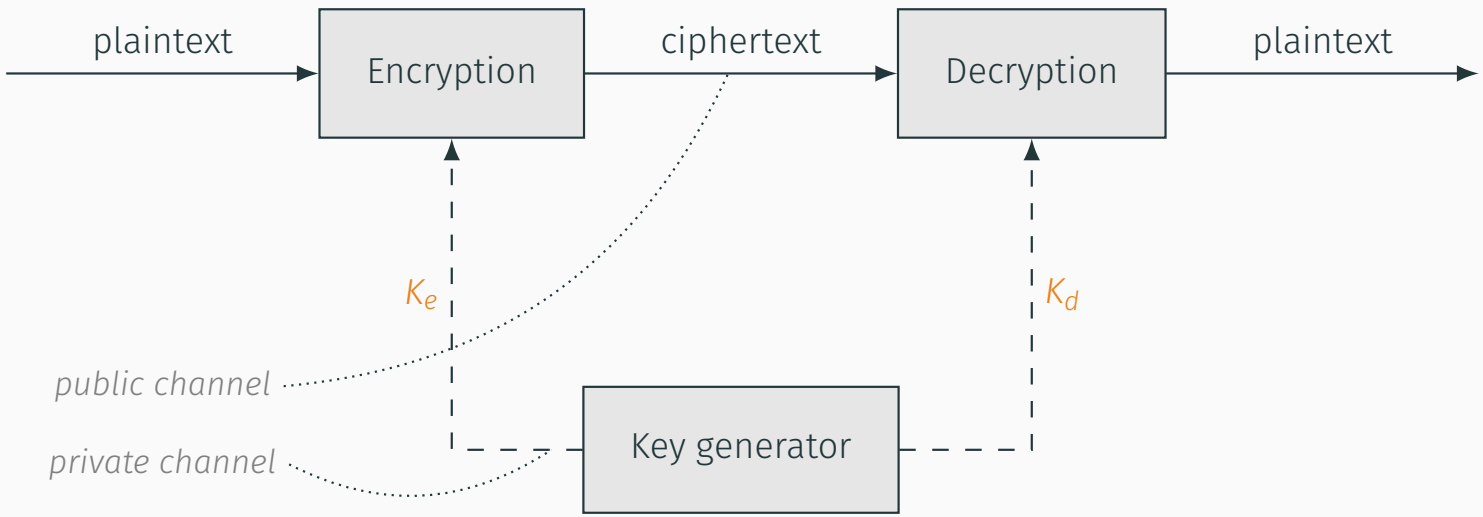
Introduction to Cryptography

Mario Cagalj

University of Split

Symmetric Cryptography

Symmetric vs Asymmetric Cryptography System



- Symmetric cryptography implies $K_e = K_d$
- Asymmetric cryptography implies $K_e \neq K_d$

Cryptographic System

- $\mathcal{P}$ - the set of plaintext messages (messages in clear)
- $\mathcal{C}$ - the set of ciphertext messages (encrypted messages)
- $\mathcal{K}_e$ and $\mathcal{K}_d$ - the set of encryption and decryption keys
- $KeyGen : \mathcal{N} \to \mathcal{K}_e \times \mathcal{K}_d$ - key generation algorithm
- $E : \mathcal{K}_e \times \mathcal{P} \to \mathcal{C}$ - encryption algorithm
- $D : \mathcal{K}_d \times \mathcal{C} \to \mathcal{P}$ - decryption algorithm

For each plaintext $p \in \mathcal{P}$ and all keys $k_e \in \mathcal{K}_e$ there is the corresponding key $k_d \in \mathcal{K}_d$ such that $D_{k_d}(E_{k_e}(p)) = p$.

Kerckhoffs' Principle

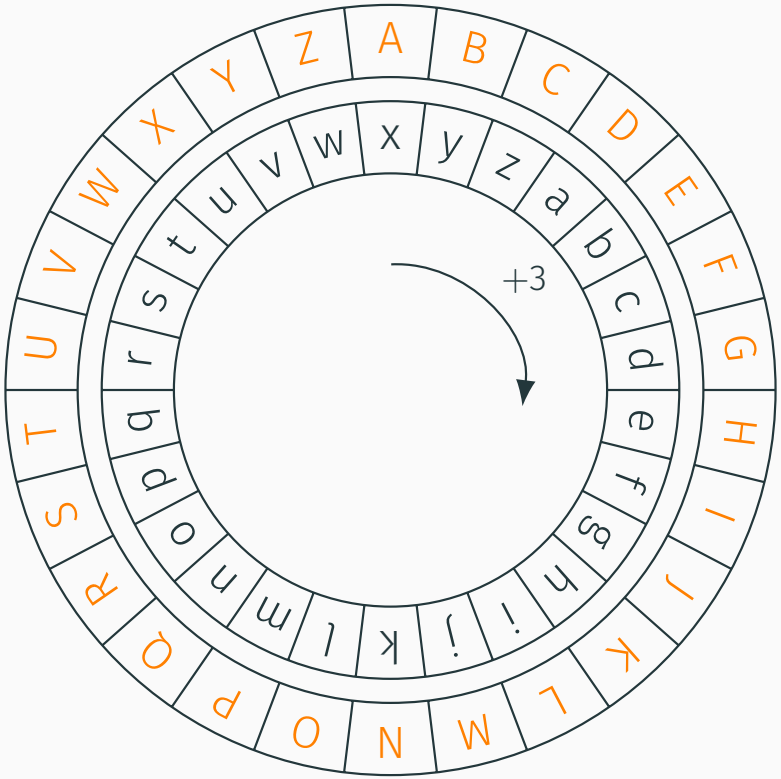
Definition

The security of a cryptographic system should rely solely on the secrecy of the key, rather than on the secrecy of the algorithm or system design.

- Put forward by Auguste Kerckhoffs in 19th century: *Cryptographic systems should be secure even if all details, except the key, are publicly known.*
- Emphasizes transparency and open algorithms for independent analysis. Security comes from strong keys and proven algorithms, not secrecy of implementation.
- Security through obscurity not a good practice.

Classical Encryption Systems

Caesar (Shift) Cipher



Example

Example plaintext = cryptography and network security

ciphertext = FUBSWRJUDSKB DQG QHWZRUN VHFXULWB

Caesar Cipher

- Let us encode English alphabet as follows: $a = 0, b = 1, \ldots, z = 25$.
- Let p, c, and k denote *plaintext*, *ciphertext* and *the secret key*, respectively; note, $p, c, k \in \mathbb{Z}_{26} = \{0, 1, \ldots, 25\}$
- Encryption: $c = p + k \mod 26$
- Decryption: $p = c + (-k) \mod 26 = c + (26 - k) \mod 26$
- Monoalphabetic supstitution cipher

Q. Perform cryptanalysis of Caesar cipher.

Caesar Cipher

- There are only 26 possible keys in Caesar cipher
- It is easy to exhaust/test all keys (a brute-force attack)
- However, we can easily increase the number of keys by allowing an arbitrary permutation of the input alphabet

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
K	Q	V	F	C	M	A	U	L	B	S	E	W	O	Z	H	Y	G	I	D	P	X	J	N	T	R

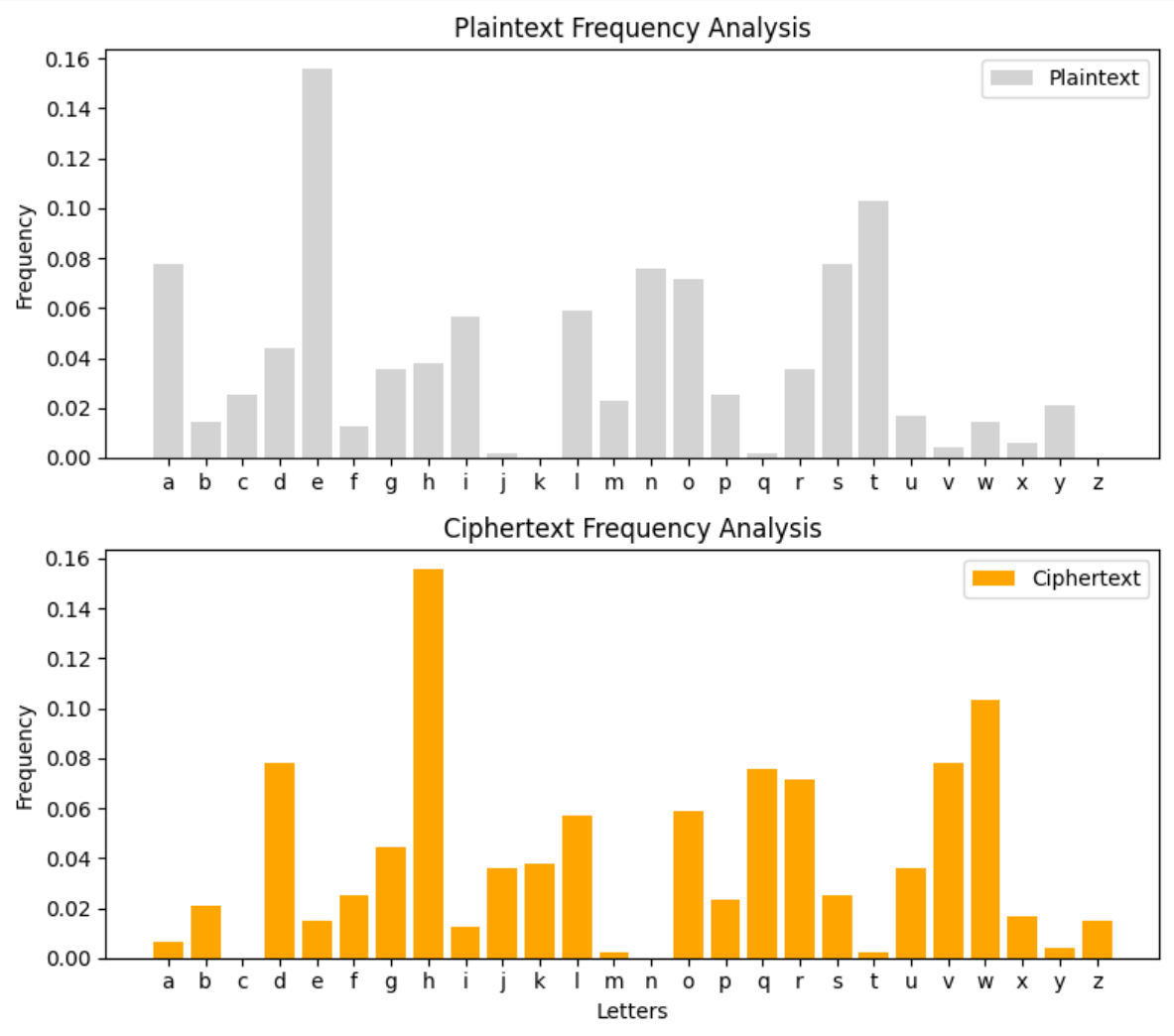
- The number of keys increased to $26! > 2^{80}$
- Brute-force attack thwarted
- Note: not a shift cipher anymore

Q. Perform cryptanalysis of ”imporved” Caesar cipher.

Cryptanalysis of Caesar Cipher

Letter frequency in English text:

Letter	Frequency
e	15.58%
t	10.32%
a	7.79%
⋮	⋮
q	0.21%
j	0.21%
x	0.63%



Vernam Cipher

- Plaintext space: $\mathcal{P} = \{0,1\}^n$, plaintext $P \in \mathcal{P}$
- Key space: $\mathcal{K} = \{0,1\}^n$, key $K \in \mathcal{K}$
- Ciphertext space: $\mathcal{C} = \{0,1\}^n$, ciphertext $C \in \mathcal{C}$
- Encryption algorithm: $C = P \oplus K$
- Decryption algorithm: $P = C \oplus K$

Note, P, K, C are all *random variables*. Also, the decryption function is an *inverse* function of the encryption function:

$$P = C \oplus K = P \oplus K \oplus K = P$$

Vernam Cipher - example

```
      Plaintext P: Test

 Plaintext P (hex): 54657374
       Key K (hex): 00010203
Ciphertext C (hex): 54647177

 Plaintext P (bin): 01010100 01100101 01110011 01110100
       Key K (bin): 00000000 00000001 00000010 00000011
Ciphertext C (bin): 01010100 01100100 01110001 01110111
```

Vernam Cipher - Key Reuse

By reusing the same encryption key over and over renders the Vernam cipher completely insecure. Indeed:

$$C_1 = P_1 \oplus K$$

$$C_2 = P_2 \oplus K$$

Then, by *xoring* two **public ciphertexts** C_1 and C_2 we have:

$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

Observe:

- $C_1 \oplus C_2 = \mathbf{0}$ implies $P_1 = P_2$
- $P_2 = C_1 \oplus C_2 \oplus P_1$, where P_1 might be easily guessable
- Finally, $H(P_2) = H(P_1)$ (equal entropies)

Vernam Cipher - Key Reuse

```
                    Key K: abf1021df4

          Plaintext P1: Hello
   Plaintext P1 (hex): 48656c6c6f
  Ciphertext C1 (hex): e3946e719b

          Plaintext P2: world
   Plaintext P2 (hex): 776f726c64
  Ciphertext C2 (hex): dc9e707190

              P1 xor P2: 3f0a1e000b
              C1 xor C2: 3f0a1e000b

P1 xor C1 xor C2 (hex): 776f726c64
P1 xor C1 xor C2 (utf): world
```

Perfect Secrecy

We show how to convert Vernam cipher into an *ideal cipher*.

Definition (Perfect secrecy)

A cipher is said to have a *perfect secrecy* property if:

$$Pr(P = p|C = c) = Pr(P = p), \quad \forall p \in \mathcal{P}, c \in \mathcal{C}.$$

Here, $Pr(P|C)$ denotes conditional *posterior probability*, and $Pr(P)$ *prior probability* of a plaintext.

Intuitively, a given cipher perfectly protects message confidentiality if resulting ciphertexts, once captured by the attacker, do not help him/her to gain additional insights into encrypted plaintexts; no matter how powerful the attacker is.

Perfect Secrecy

Let us restate the previous definition using the language of information entropy[1].

Definition (Perfect secrecy)

A cipher is said to have a *perfect secrecy* property if:

$$H(P|C) = H(P), \quad P \in \mathcal{P}, C \in \mathcal{C}.$$

Here, $H(P|C)$ is conditional entropy. Intuitively, a given cipher perfectly protects message confidentiality if resulting ciphertexts, once captured by the attacker, do not decrease attacker's *apriori* uncertainty about encrypted plaintexts.

[1] Please check accompanying slides "Write-up on information entropy".

One-Time Pad

Definition (One-time pad)

One-time-pad is identical to Vernam cipher with an important difference that the key K is selected *uniformly at random* for each new plaintext to be encrypted.

Theorem

One-time pad has a perfect secrecy property.

Note that it still can happen that two plaintexts are by chance encrypted using the same key, but this is of no help to the attacker as he/she does not know which ones.

One-Time Pad - example

Let $P, K, C \in \{0, 1\}$ and consider two plaintext messages: $P_1 = 1$ and $P_2 = 1$. Using *one-time-pad*, we encrypt them as follows:

- Generate randomly K_1, then calculate $C_1 = P_1 \oplus K_1$
- Generate randomly K_2, then calculate $C_2 = P_2 \oplus K_2$

After *xor*-ing two public ciphertexts C_1 and C_2 and rearranging:

$$P_2 = C_1 \oplus C_2 \oplus P_1 \oplus K_1 \oplus K_2$$

Now, even if the attacker knows P_1, he still cannot calculate P_2 since he cannot predict $K_1 \oplus K_2$ better than randomly guessing. Hence, C_1 and C_2 are useless to the attacker, i.e., $H(P|C) = H(P)$.

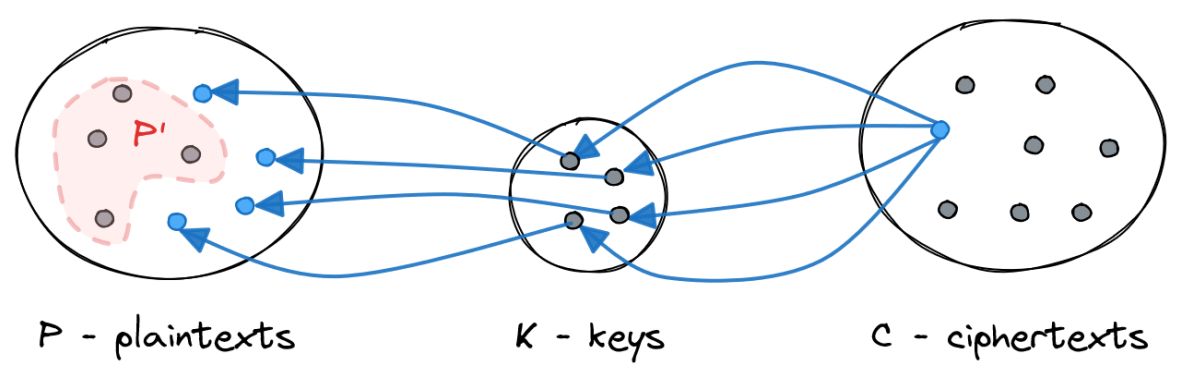
One-Time Pad - Bad News

Key space size

While being perfect/ideal, this cipher is not practical as it implies that the number of keys is greater or equal to the number of plaintext messages, i.e., $|\mathcal{K}| \geq |\mathcal{P}|$.

Ideal Cipher - Bad News

Necessary condition for a cipher to be ideal: $|\mathcal{K}| \geq |\mathcal{P}|$



- Assume plaintexts P are selected randomly from $\mathcal{P}$, where $|\mathcal{P}| = 8$
- A priori, the attacker's uncertainty is $Pr(P = p) = \frac{1}{8}$, $\forall p \in \mathcal{P}$
- A posteriori, $Pr(P = p|C = c) = 0$, $\forall p \in \mathcal{P}'$
- Moreover, $Pr(P = p|C = c) = \frac{1}{4}$, $\forall p \in \mathcal{P} \setminus \mathcal{P}'$
- Hence, $Pr(P = p|C = c) \neq Pr(P = p)$, violating perfect secrecy condition.

How to deal with the bad news?

We know that ideal cipher implies $|\mathcal{K}| \geq |\mathcal{P}|$.

Yet, modern practical *secure* ciphers satisfy $|\mathcal{K}| \ll |\mathcal{P}|$[2].

Relax assumption about an adversary

Perfect security definition sets unrealistic assumption of an *almighty adversary* who has *unlimited processing power*.

In practice, though, adversaries are limited in the number of operations (and amount of memory) they can perform (afford) in a given period of time, leading to the notion of *computational security*.

[2] E.g., AES can *securely* encrypt 1GB using only one 128 bit key.

Computational Security

1. Consider only realistic *computationally-bounded* adversaries who can perform a limited (albeit still large) number of computations (e.g. $q = 2^{60}$) in a fixed period of time.
2. Embrace the risk that a *computationally-bounded* adversary can succeed in breaking your system, but only with an acceptably small (negligible) probability ε.

For example, consider a cipher that uses a randomly selected 128-bit key to encrypt multiple plaintext messages. In this scenario, an attacker capable of performing $q = 2^{60}$ operations (testable guesses) can succeed in guessing the correct key with a probability of at most $\varepsilon = \frac{q}{2^{128}} = 2^{-68}$.

Brute-Force Attack

Algorithm 1: Generic brute-force attack

Data: *(plaintext, ciphertext)* pairs (p_i, c_i)
Result: Found key

```
currentKey ← Initial Key;
while true do
    found ← true;
    forall pairs (p_i, c_i) do
        if DecryptWithKey(c_i, currentKey) ≠ p_i then
            found ← false;
            break;
    if found then
        print("Found key:", currentKey);
        break;
    currentKey ← GenerateNextKey(currentKey);
```

Brute-Force Attack

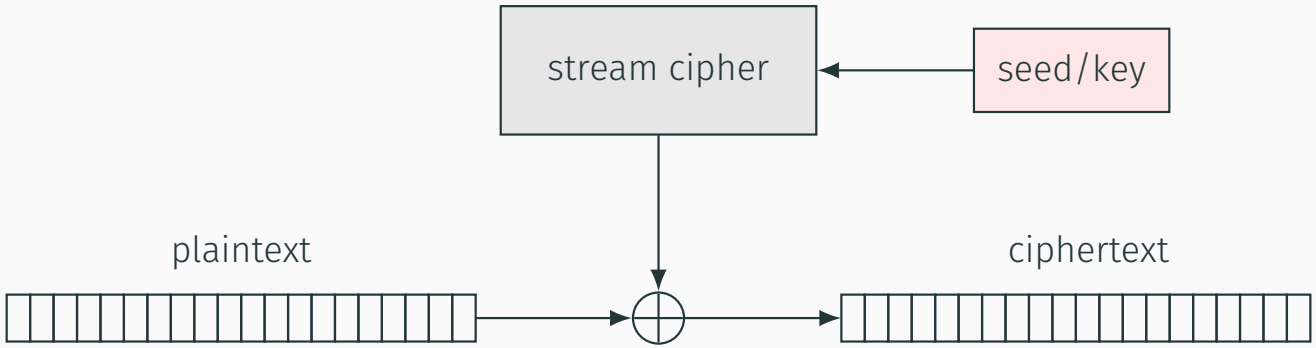
Reflect:

- Brute-force attack always possible against practical ciphers that assume *computationally-bounded* adversaries.
- However, this attack is *irrelevant for perfectly secure/ideal ciphers*. Why?

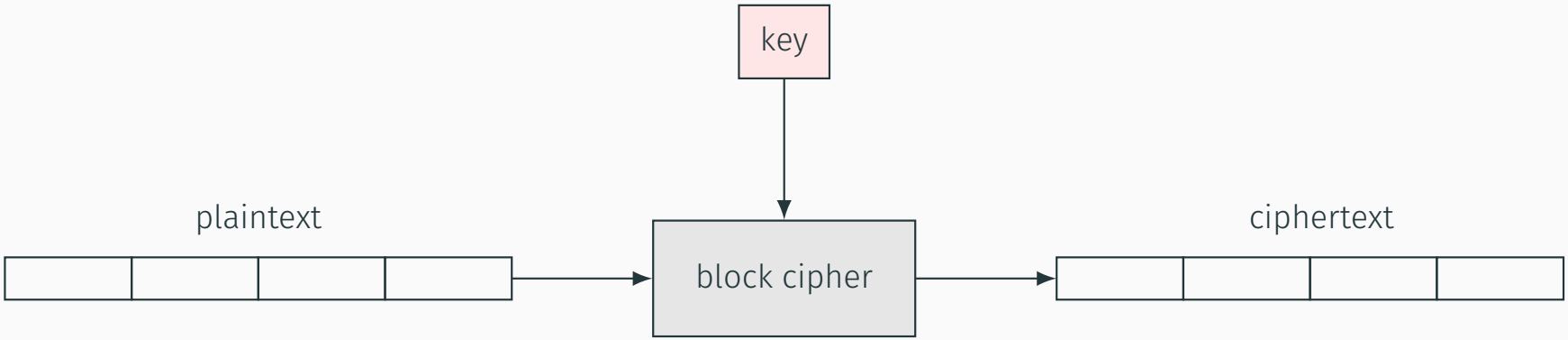
Modern Encryption Systems

Stream Cipher vs Block Cipher

- Stream cipher processes messages bit/byte by bit/byte



- Block cipher processes messages in blocks



Modern Block and Stream Ciphers

Block ciphers (not an exhaustive list):

- AES (Advanced Encryption Standard)

Stream ciphers (not an exhaustive list):

- AES-CTR, AES-GCM (AES in *counter mode* of operation)
- ChaCha20 (based on Salsa20)

Advanced Encryption Standard (AES)

- In 1997. the US National Institute of Standards and Technology (NIST) announced a competition for a new encryption standard to replace the existing standard DES. The new algorithm would be named AES.
- In 1998. NIST selected a group of 15 algorithms as candidates for AES.
- In 2001. NIST announced that Rijndael (pronounced *[reinda:l]*) was chosen for AES. Rijndael was designed by two Belgian cryptographers, Daemen and Rijmen.

Advanced Encryption Standard (AES)

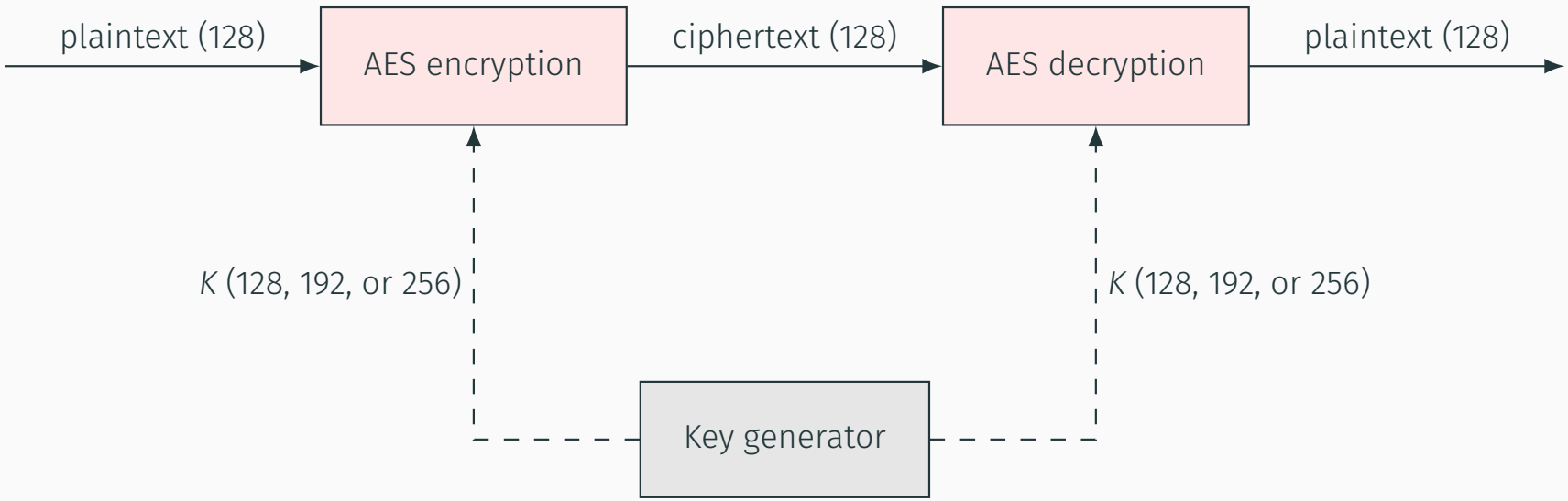
- AES is a symmetric-key block cipher that operates with 128-bit plaintext and ciphertext blocks.
- It supports encryption keys of 128, 192, and 256 bits[3].

More formally:

- $\mathcal{P} = \mathcal{C} = \{0,1\}^{128}$
- $\mathcal{K} = \{0,1\}^{128,\ 192 \text{ or } 256}$
- $E : \mathcal{K} \times \mathcal{P} \to \mathcal{C}$ - AES encryption
- $D : \mathcal{K} \times \mathcal{C} \to \mathcal{P}$ - AES decryption

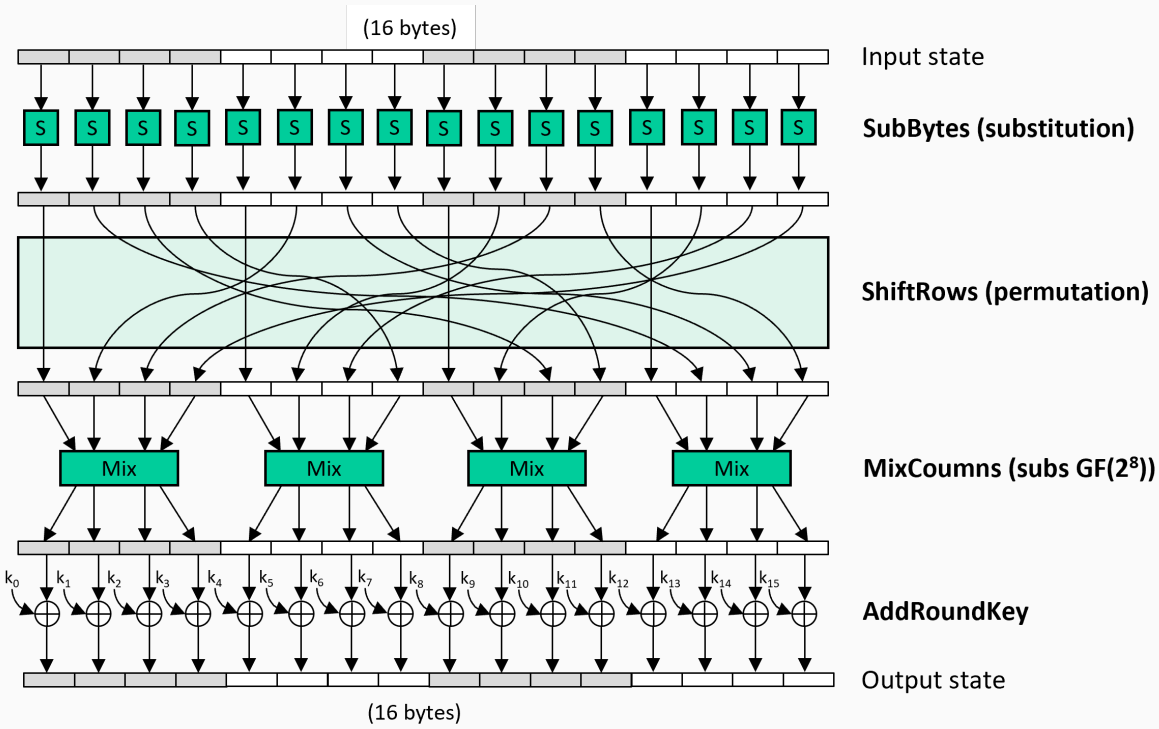
[3] The number of internal rounds in AES depends on the key size: 10 rounds for 128 bits, 12 for 192 bits, and 14 rounds for 256-bit keys. This can incur additional performance cost.

Advanced Encryption Standard (AES)



Glance on Internals of AES

- AES processes messages in multiple rounds (10, 12, or 14 depending on the selected key size)
- Performs substitutions, permutations and mixes in the key with messages in each round[4]

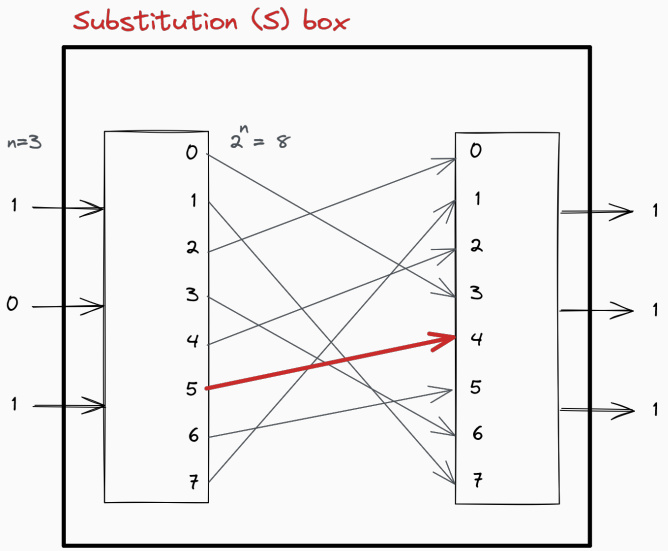


[4] Internals of AES are rather complex and outside of scope of this course.

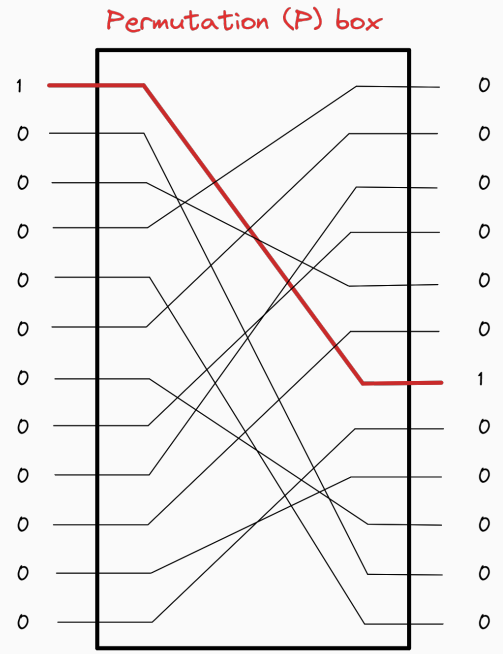
Product Ciphers

Combine multiple rounds of simple transformations such as substitution (S-box), permutation (P-box), and modular arithmetic. Introduced by Claude Shannon (also the father of information entropy) - Wikipedia.

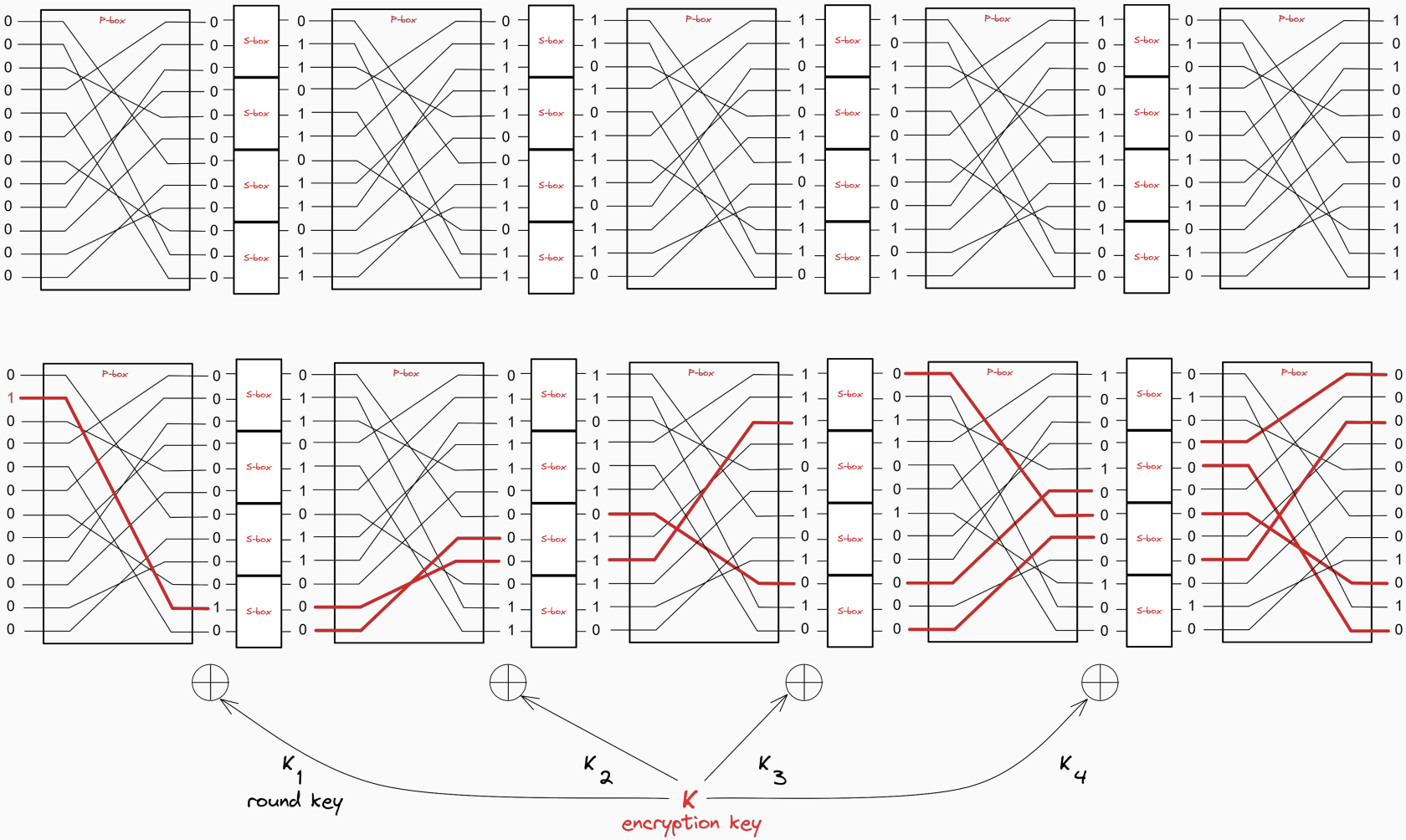
AES is an example of a product cipher.



input	output
000	011
001	111
010	000
011	110
100	010
101	100
110	101
111	001



Product Cipher System



Product cipher *avalanche effect*

Block Cipher Model - Keyed Family of Functions

Block cipher (e.g., AES) is a keyed function. More formally, let us define a family of functions:

$$F : \mathcal{K} \times \mathcal{P} \to \mathcal{C}.$$

For $k \in \mathcal{K}$, $F_k : \mathcal{P} \to \mathcal{C}$:

- F_k is a permutation: $\forall p \in \mathcal{P},\ F_k^{-1}(F_k(p)) = p$.
- F_k is easy to evaluate (both, F_k and F_k^{-1}).

Block Cipher Model - Keyed Family of Functions

Encryption functions

$C = E(K, P) = F_K(P)$

P	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
000	011	001	010	010	001	110	111	101
001	100	100	111	100	011	101	000	110
010	001	011	001	011	000	010	001	100
011	111	000	011	111	110	001	101	001
100	010	010	101	001	101	011	010	010
101	101	110	110	110	010	100	011	000
110	110	111	100	000	100	111	100	011
111	000	101	000	101	111	000	110	111

Decryption functions

$P = D(K, C) = F_K^{-1}(C)$

C	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
000	111	011	111	110	010	111	001	101
001	010	000	010	100	000	011	010	011
010	100	100	000	000	101	010	100	100
011	000	010	011	010	001	100	101	110
100	001	001	110	001	110	101	110	010
101	101	111	100	111	100	001	011	000
110	110	101	101	101	011	000	111	001
111	011	110	001	011	111	110	000	111

Block Cipher Security

AES, as a block cipher, *tries* to mimic a random function.

Algorithm 2: Random Function Algorithm

Data: $T = \{\}$, associative array (dictionary)

Function $Lookup(p \in \{0,1\}^{\ell})$**:**

Data: Plaintext p of size ℓ bits

Result: Ciphertext c of size ℓ bits

if $T[p]$ == *undefined* **then**

select c randomly from $\{0,1\}^{\ell}$;

set $T[p] = c$;

return $T[p]$

Block Cipher Security

AES *tries* to mimic a random function. How to formalize this?

Indistinguishability (analogy to Turing test of intelligence)

We consider two scenarios: one where computationally bounded adversary $\mathcal{A}$ interacts with a **real block cipher** and another where $\mathcal{A}$ interacts with an **ideal block cipher** (represented by the random function algorithm). $\mathcal{A}$ can requests the encryption of a finite number of plaintexts.

The real block cipher is deemed **secure** if $\mathcal{A}$ cannot distinguish between interactions with the real and ideal ciphers, except with negligible probability.

Block Cipher Security

Pseudo-Random Function/Permutation

A block cipher that satisfy indistinguishability definition is said to be pseudo-random function or more precisely pseudo-random permutation (as block cipher has to be a permutation).

It is believed that AES behaves as a pseudo-radnom permutation (but we do not have a proof).

AES as Pseudo-Random Function/Permutation

Some implications[5]:

- AES output looks practically random.
- Two plaintext messages that differ in only a single bit result in highly different ciphertexts (avalanche property).
- Decrypting a ciphertext that has not been encrypted should result in a random looking plaintext.
- Encrypting the same message using a different key results (likely) in a different ciphertext.
- Having multiple *(plaintext, ciphertext)* pairs $(p_i, c_i),\ i = 1, 2, \ldots, n$, does not reveal the encryption key.

[5] Demo (CNS 2020/21)

Block Cipher Modes of Operation (Encryption Modes)

Motivation

While modern ciphers like AES showcase resilience against computationally-bounded adversaries, practical scenarios prompt two crucial questions:

- How to ensure secure encryption of the same message more than one time?
- How to securely encrypt a message exceeding the block size of the used block cipher?

In the sequel, we will explore how to accomplish above using fundamental block cipher modes of operation.

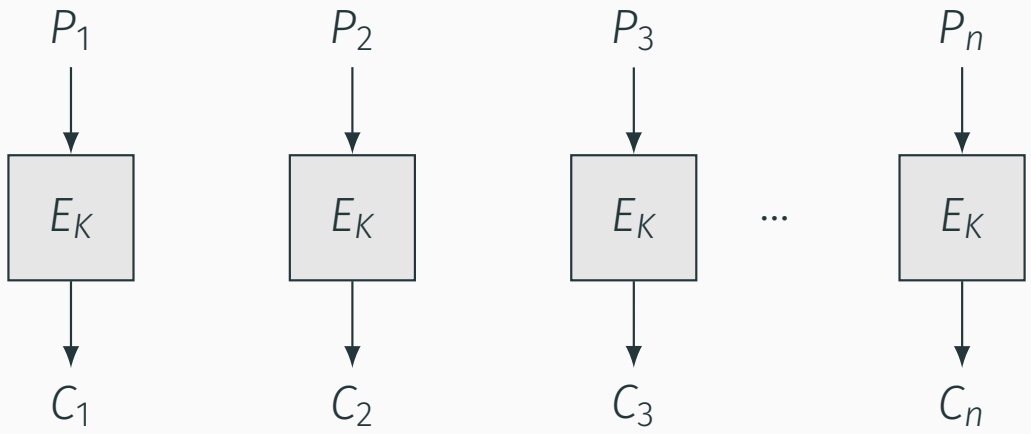
Basic Block Cipher Encryption Modes

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Counter (CTR)

Not an exhaustive list.

Electronic Codebook (ECB) Mode - encryption

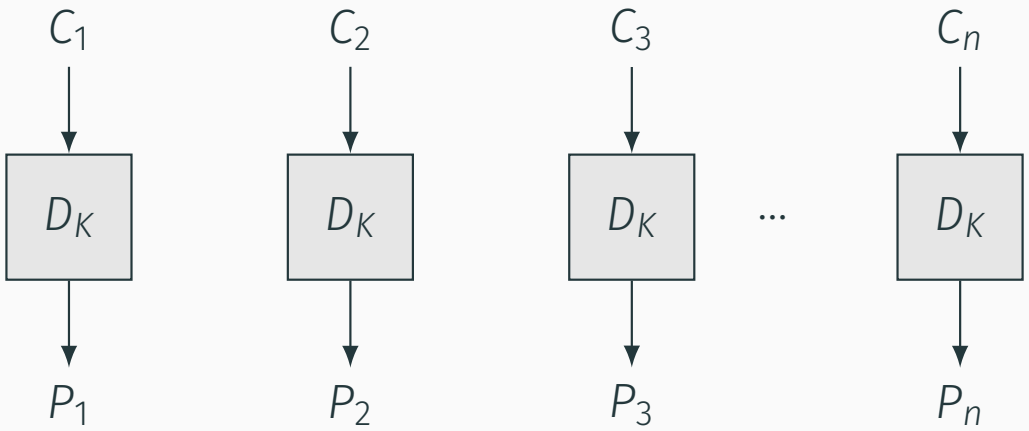
- E_K - block cipher using key K
- $P = P_1 P_2 \ldots P_n$ - plaintext splitted into blocks P_i
- $C = C_1 C_2 \ldots C_n$ - ciphertext



- $C_i = E_K(P_i),\ i = 1, 2, \ldots, n$
- You should never encrypt your messages in this way

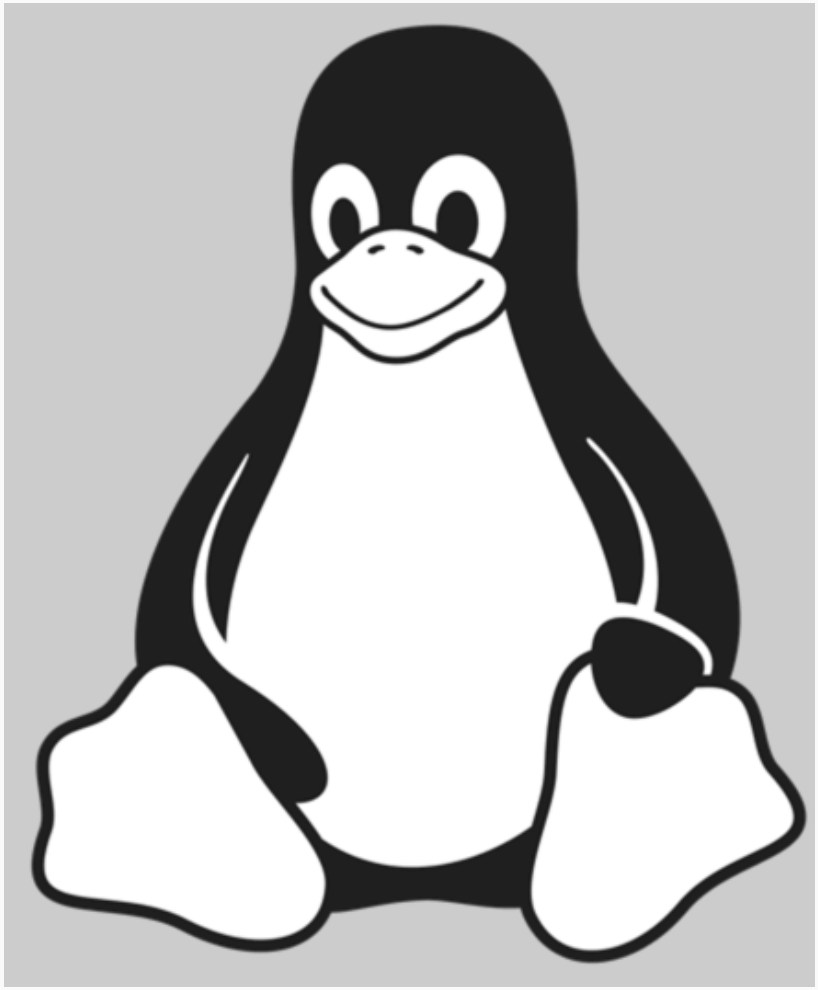
Electronic Codebook (ECB) Mode - decryption

- D_K - block cipher using key K (decryption direction)
- $P = P_1 P_2 \ldots P_n$ - plaintext splitted into blocks P_i
- $C = C_1 C_2 \ldots C_n$ - ciphertext

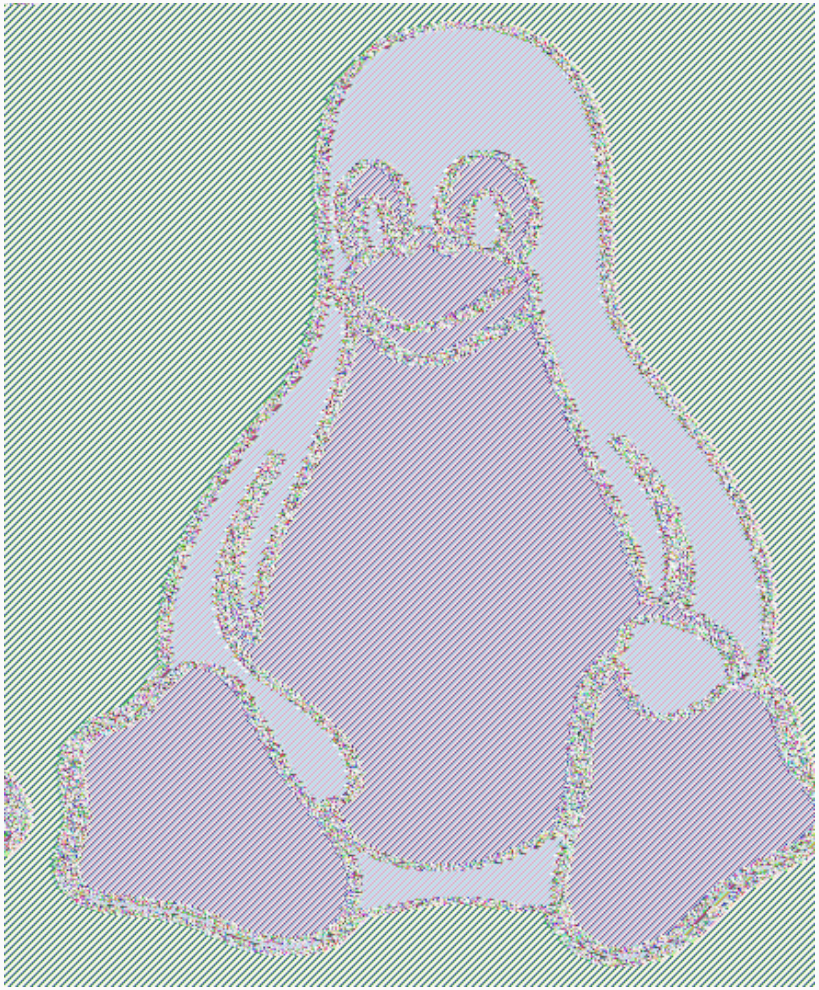


- $P_i = D_K(C_i),\ i = 1, 2, \ldots, n$
- You should never use ECB mode to encrypt your messages

Electronic Codebook (ECB) Mode - warning

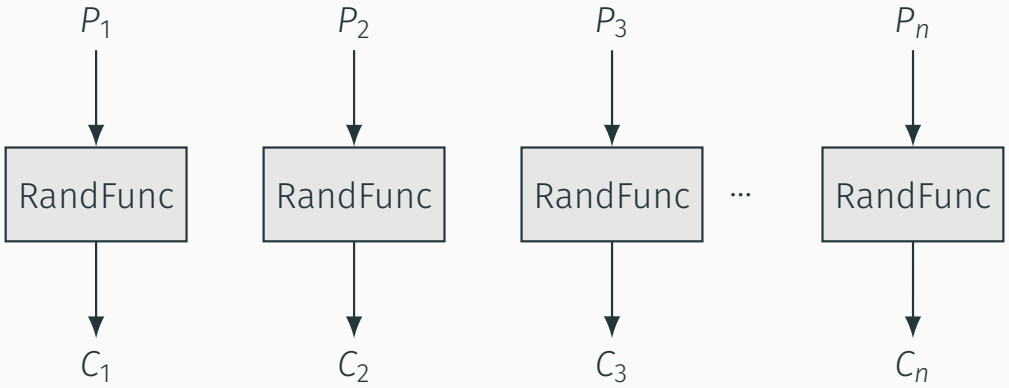


Original Image



Encrypted in ECB mode

Chosen-Plaintext Attack (CPA) - Security



ECB mode is insecure even when using ideal block cipher.

Instead of relying on vague arguments or listing potential attacks, let's formalize security for encrypting messages larger than the block size of the cipher or encrypting the same message multiple times, as done previously with *perfect secrecy* and *ideal block cipher* definitions.

Let's play a game ...

Chosen-Plaintext Attack (CPA) - Security

Game 1: CPA-security game

Input: Encryption scheme $Enc(K, M)$

Output: Single bit 0 or 1

Adversary ($\mathcal{A}$) selects messages $m_0 \neq m_1$ and $\|m_0\| = \|m_1\|$;
Challenger selects a random bit $b \in \{0, 1\}$;
Challenger selects a random key k;
if $b = 0$ **then**
    Challenger encrypts m_0 to obtain $c \leftarrow Enc(k, m_0)$;
else
    Challenger encrypts m_1 to obtain $c \leftarrow Enc(k, m_1)$;
Adversary ($\mathcal{A}$) receives c;
Adversary ($\mathcal{A}$) outputs $b' \in \{0, 1\}$;

return 1 *if* $b = b'$, 0 *otherwise*

Chosen-Plaintext Attack (CPA) - Security

CPA-Security

An encryption system $Enc(K, M)$ is considered CPA-secure if and only if the advantage Adv of the adversary $\mathcal{A}$ in distinguishing which message was encrypted in the previous game is negligibly small. More formally, for security parameter n (e.g., the key size):

$$Adv^{CPA}(\mathcal{A}) = \left| Pr[\text{Game} \rightarrow 1] - \frac{1}{2} \right| = \left| Pr[b' = b] - \frac{1}{2} \right| \leq \mathcal{O}(2^{-n}).$$

Note: This definition is provided for completeness but isn't required knowledge for the exam.

Electronic Codebook (ECB) Mode - Security

Theorem

ECB encryption mode is not CPA-secure.

Consider AES block cipher used in ECB encryption mode. Convince yourself that adversary $\mathcal{A}$ can win against this encryption system, in the previous game, by selecting messages m_0 and m_1 as follows:

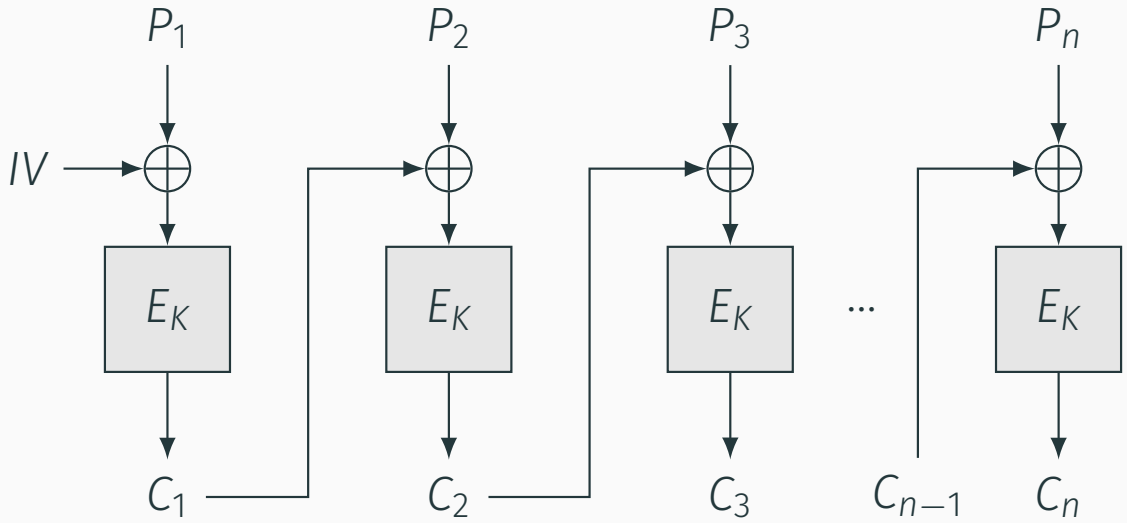
$$m_0 = \underbrace{00\ldots0}_{128}\underbrace{00\ldots0}_{128} \text{ and } m_1 = \underbrace{00\ldots0}_{128}\underbrace{11\ldots1}_{128}.$$

ECB is an example of a deterministic encryption system[6]. In the following slides, we show two randomized encryption modes that (properly used) are CPA-secure.

[6] Many DB systems support deterministic encryption for efficient searching.

Cipher-Block Chaining (CBC) - encryption

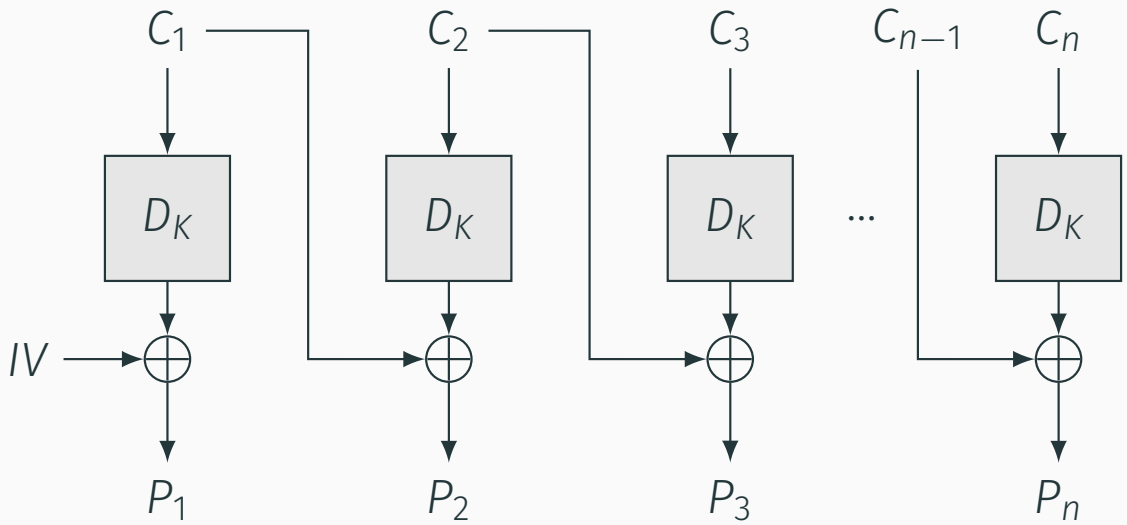
- E_K - block cipher using key K
- IV - initialization vector
- $P = P_1 P_2 \ldots P_n$ - plaintext splitted into blocks P_i
- $C = C_1 C_2 \ldots C_n$ - ciphertext



- $C_1 = E_K(IV \oplus P_1)$ and $C_i = E_K(C_{i-1} \oplus P_i),\ i = 2, \ldots, n$
- In this mode, plaintext P must be padded (dangerous).

Cipher-Block Chaining (CBC) - decryption

- D_K - block cipher using key K (decryption direction)
- IV - initialization vector
- $P = P_1 P_2 \ldots P_n$ - plaintext splitted into blocks P_i
- $C = C_1 C_2 \ldots C_n$ - ciphertext

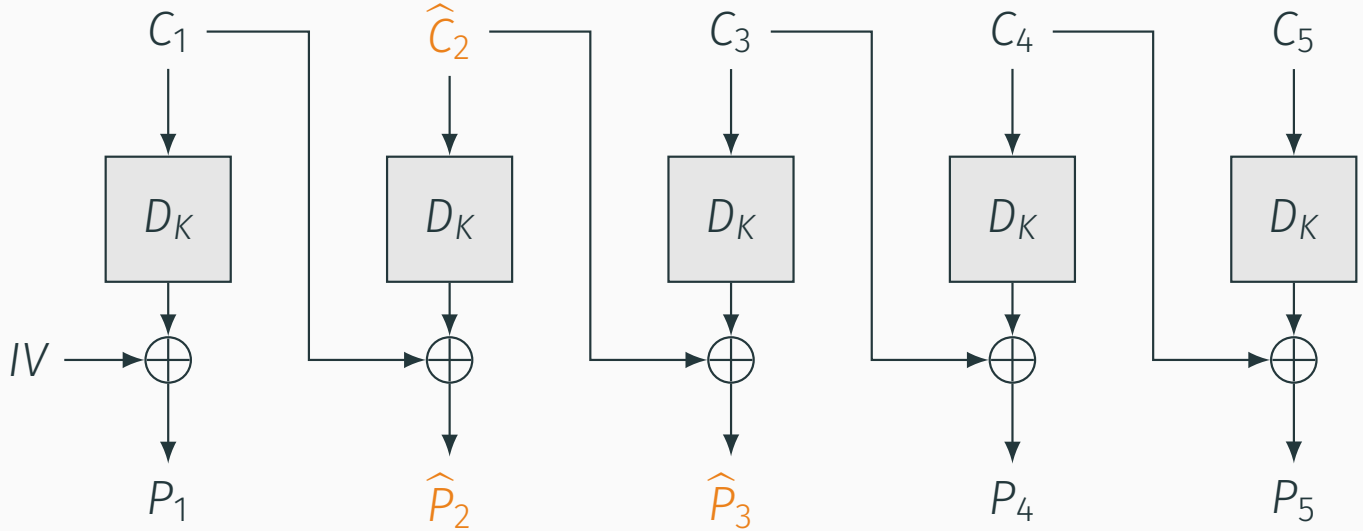


- $P_1 = IV \oplus D_K(C_1)$ and $P_i = C_{i-1} \oplus D_K(C_i),\ i = 2, \ldots, n$

Cipher-Block Chaining (CBC) - error propagation

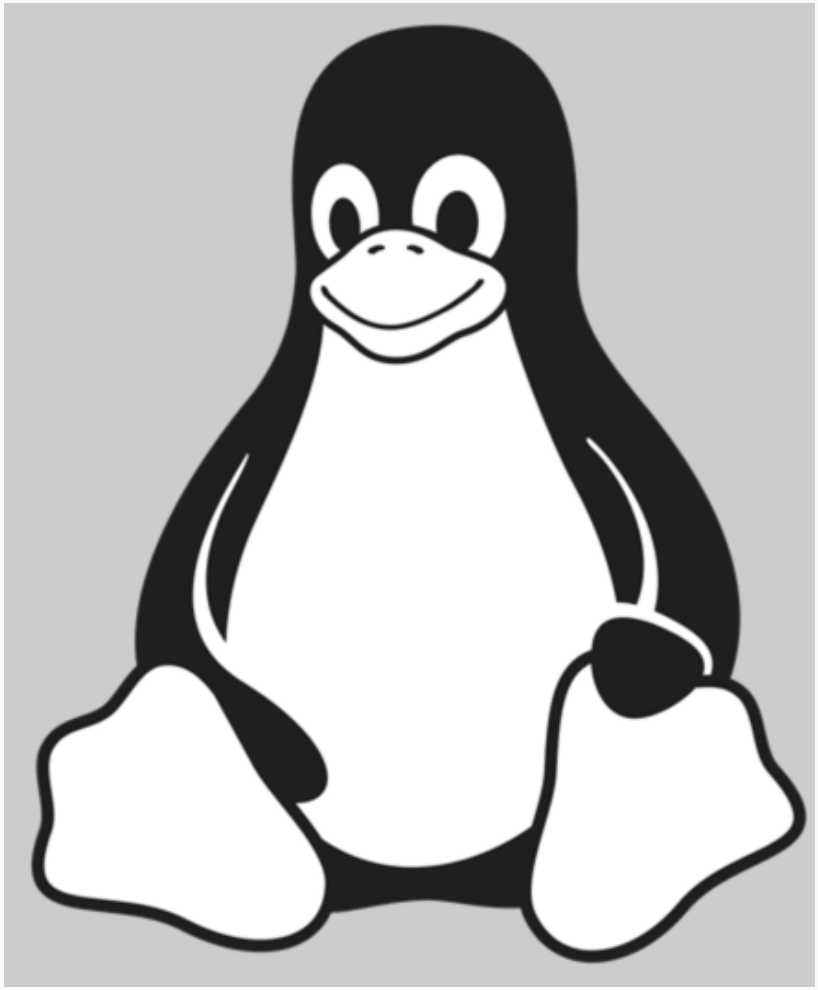
Example

Example Assume that the ciphertext C_2 is modified/changed into $\widehat{C}_2$. How does this affect decrypted plaintext blocks?

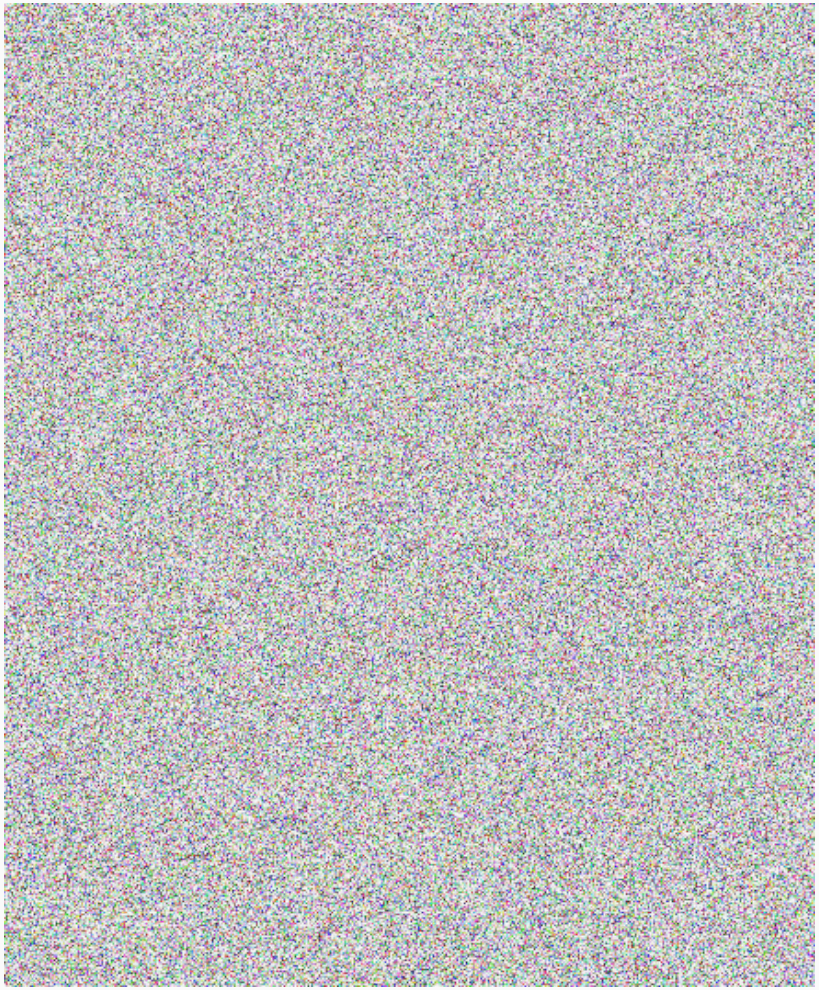


Note that in CBC mode an adversary can cause controlled changes in plaintext blocks. Example: convince yourself that $\widehat{C}_2 = C_2 \oplus 100\ldots0$ results in $\widehat{P}_3 = P_3 \oplus 100\ldots0$.

Cipher-Block Chaining (CBC) Mode



Original Image



Encrypted in CBC mode

Cipher-Block Chaining (CBC) Mode

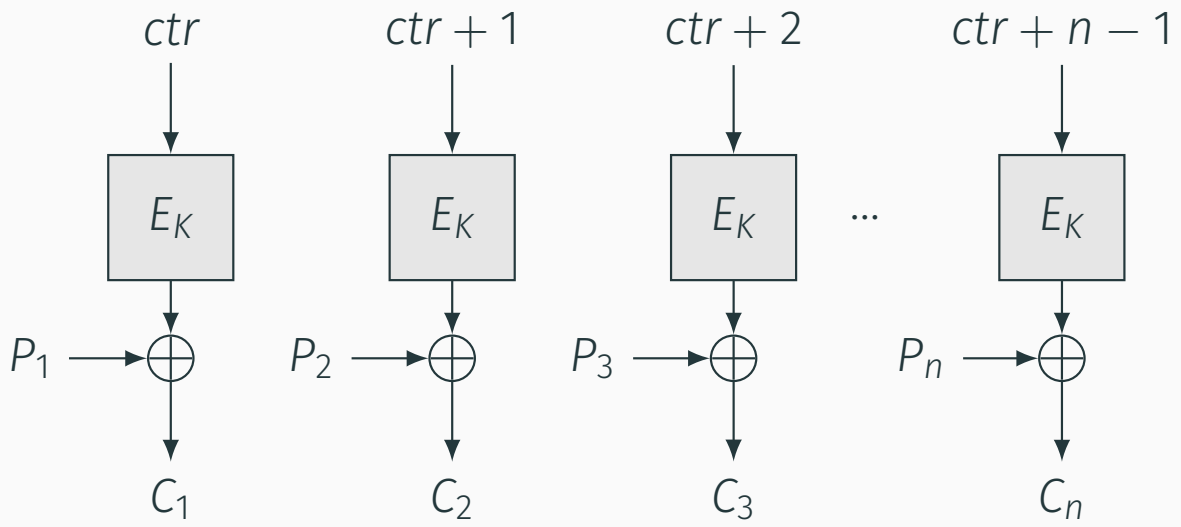
Theorem

CBC encryption mode (used properly) is CPA-secure.

Remember: Initialization vector (IV) must be selected randomly for each new message.

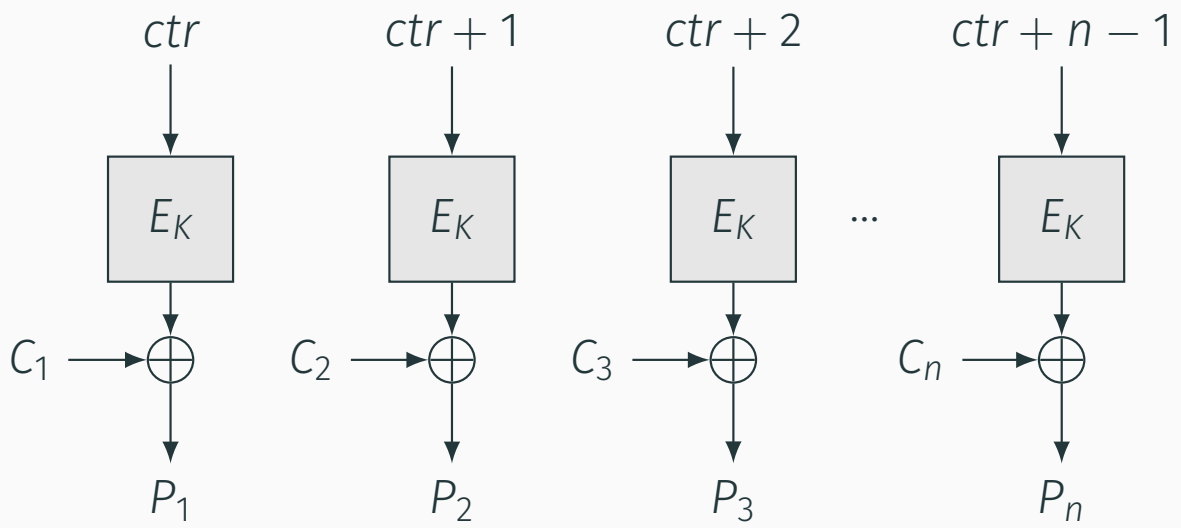
Example usage: your WhatsApp messages are encrypted using CBC mode.

Counter (CTR) Mode - encryption



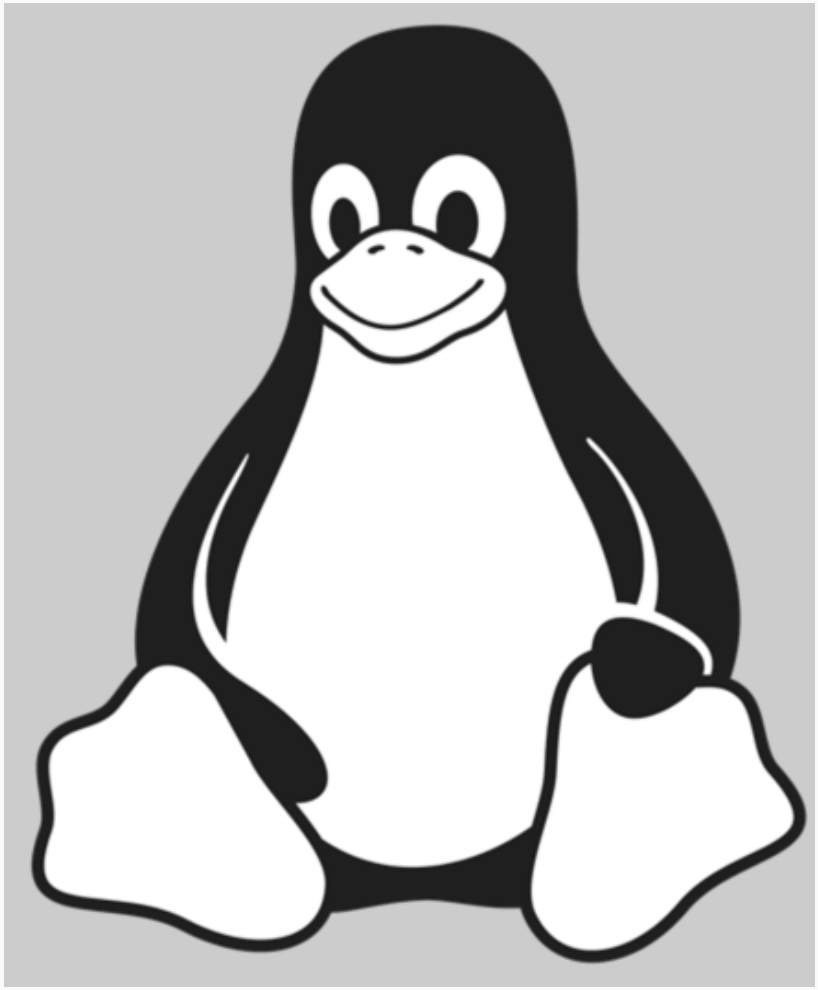
- $C_{i+1} = E_K(ctr + i) \oplus P_{i+1},\ i = 0, 1, \ldots, n-1$
- $K_{i+1} = E_K(ctr + i),\ i = 0, 1, \ldots, n-1$ is called keystream
- *ctr* must not be repeated under the same key K
- CTR mode is a stream cipher built upon a block cipher
- Very efficient, easy to parallelize
- Can encrypt messages as short as 1 bit
- No need for padding overhead (i.e., $\|P\| = \|C\|$)

Counter (CTR) Mode - decryption

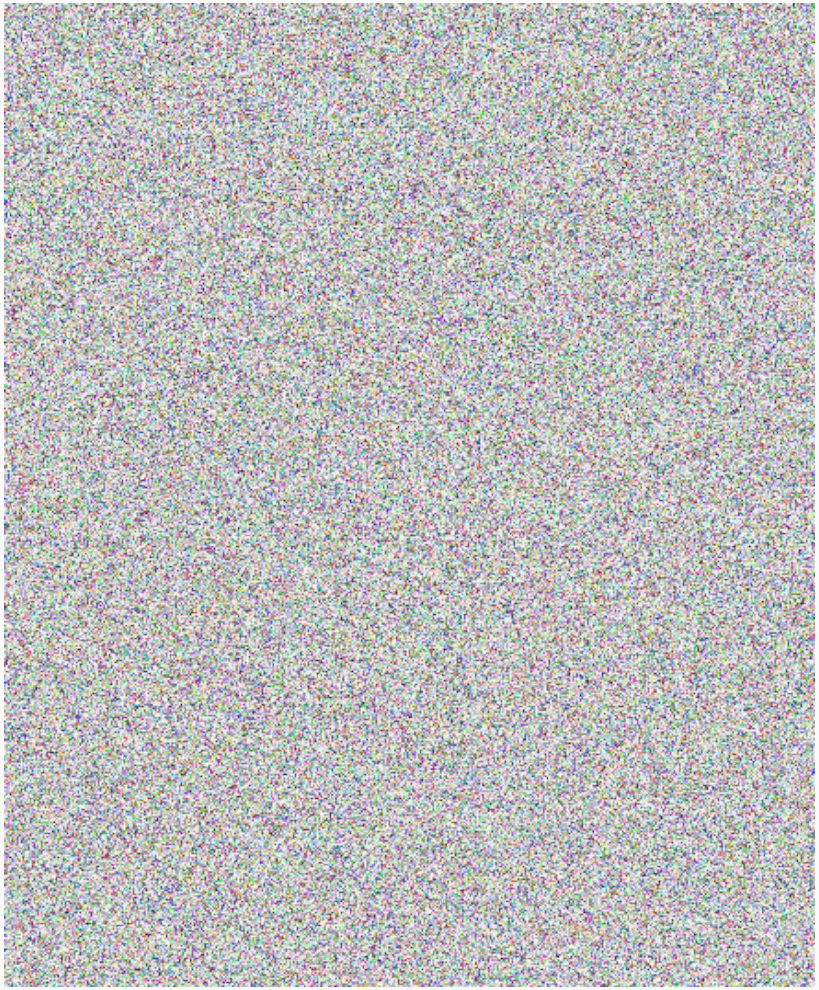


- $P_{i+1} = E_K(ctr + i) \oplus C_{i+1},\ i = 0, 1, \ldots, n-1$

Counter (CTR) Mode



Original Image



Encrypted in CTR mode

Counter (CTR) Mode

Theorem

CTR encryption mode (used properly) is CPA-secure.

Remember: One should never encrypt the same counter value under the same encryption key.

Example usage:

- Most HTTPS secured web communication is protected using AES cipher in Galois/Counter Mode (AES-GCM).
- WiFi also uses AES in CTR mode to protect your communication with an access point: *Counter Mode CBC-MAC Protocol* (in WPA-2, WPA-3) and AES-GCM (in WPA-3).