Modern Symmetric Cryptography

SIO

deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

André Zúquete

Terminology

- Cryptography
  - Art or science of hidden writing (confidential writing)
    - From Gr. kryptós, hidden + graph, r. de graphein, to write
  - Initially used to enforce the confidentiality of information
  - Steganography: art of concealing data → escondidos em imagens...
    - From Gr. steganós, hidden + graph, r. de graphein, to write

- Cryptanalysis
  - Art or science of breaking cryptographic systems or encrypted information

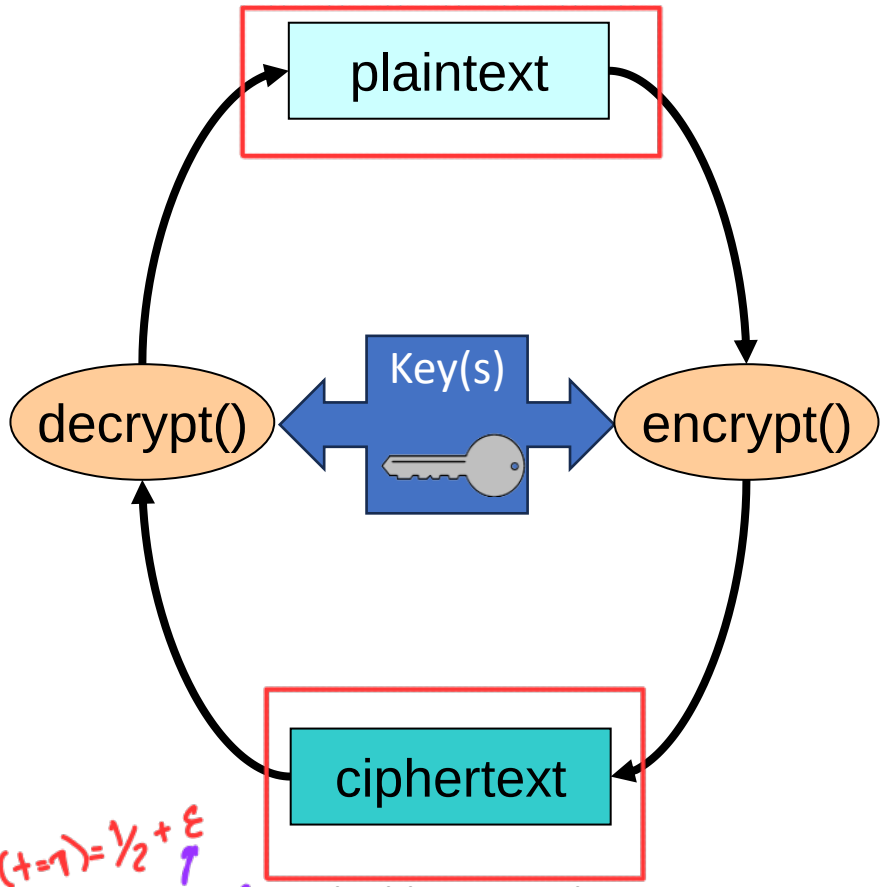
- Cryptology
  - Cryptography + cryptanalysis

ALL

Cryptography: how does it work?

- Select a cipher (or cipher algorithm)
  - Specific cryptographic technique
- Apply the cipher with a key
  - **Encryption**: original information → cryptogram
  - **Decryption**: cryptogram → original information
  - **Key**: algorithm parameter
    - Influences algorithm execution
  - Original information aka plaintext or cleartext
  - Cryptogram aka ciphertext

Usually, information that follows some well-know format



$P(t=0)=P(t=1)=\frac{1}{2}+\varepsilon$ (negligible)

Looks like a random sequence of symbols

Use cases for (symmetric) ciphers

- Self protection with secret key K
  - Alice encrypts plaintext **P** with key **K** → Alice: $C = \{P\}_k$
  - Alice decrypts ciphertext **C** with key **K** → Alice: $P' = \{C\}_k$
  - **P'** should be equal to **P** (requires checking)
  - Only Alice needs to know **K** ← encriptar ficheiros

- Secure communication with secret key K
  - Alice encrypts plaintext **P** with key **K** → Alice: $C = \{P\}_k$
  - Bob decrypts cyphertext **C** with key **K** → Bob: $P' = \{C\}_k$
  - **P'** should be equal to **P** (requires checking)
  - **K** needs to be known by Alice & Bob

Shared secret

Goals of cryptanalysis

- Reveal the plaintext hidden in a ciphertext
  - Usually requires discovering the key the produced the cyphertext
- Sometimes requires discovering the cipher algorithm
  - Usually algorithms are not secret, but there are exceptions
  - Sometimes using reverse engineering
  - Lorenz, A5 (GSM), RC4 (WEP) , Crypto-1 (Mifare)
  - Algorithms for DRM (Digital Rights Management)

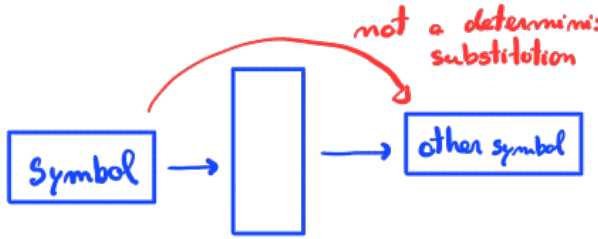
Cryptanalysis attacks

- Brute force
  - Exhaustive search of the key space until finding a match
    - GPUs are great for this!
  - Usually unfeasible for large key spaces
    - Key space: set of all possible keys with the same size
    - e.g., 128-bit keys allow a key space of 2^{128} values ← we use 2^{4096}
  - Key randomness is fundamental!
    - Means that any key as the same probability of being the right one

Rainbow Tables

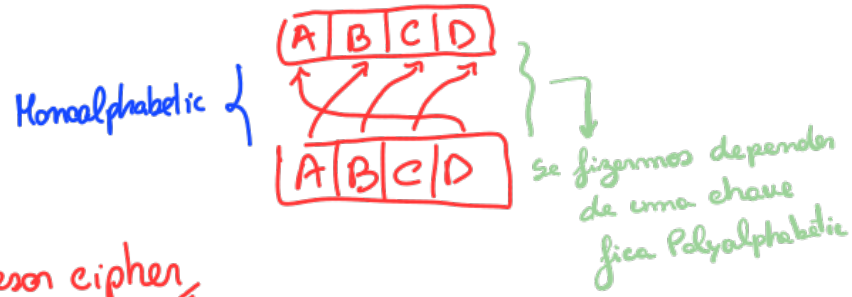
- Clever attacks
  - Reduce the search space to a smaller set of potential candidates
    - Words, numbers, restricted size or alphabet → words list
  - Identify patterns in different operations, etc.

Computer ciphers



- Operate by making substitutions
  - Original information is a sequence of **symbols**
  - Each symbol is replaced by a **substitution symbol**
    - Usually with the same size
    - **Polyphonic substitution**: several, larger substitution symbols for each original symbol
  - Substitution symbols are picked from a **substitution alphabet**

- Usual symbols
  - Bit
  - Block of bits



- Strategies
  - **Monoalphabetic** substitution: key → one substitution alphabet
  - **Polyalphabetic** substitution: key → several substitution alphabets (one for each symbol)

Caeson cipher

↳ More than 1 Key

Computer ciphers: stream ciphers

- Encrypt/decrypt by mixing streams
  - They consider the data to cipher or decipher as a bit stream
  - Each plaintext/ciphertext bit is XORed (⊕) with each keystream bit
  - Usually explored in low-level communication protocols

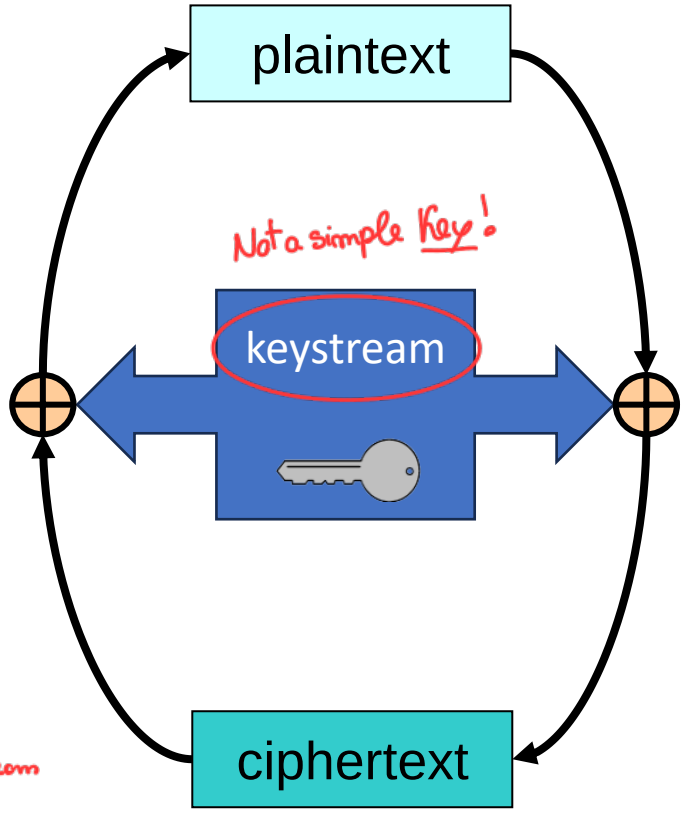
bit-by-bit

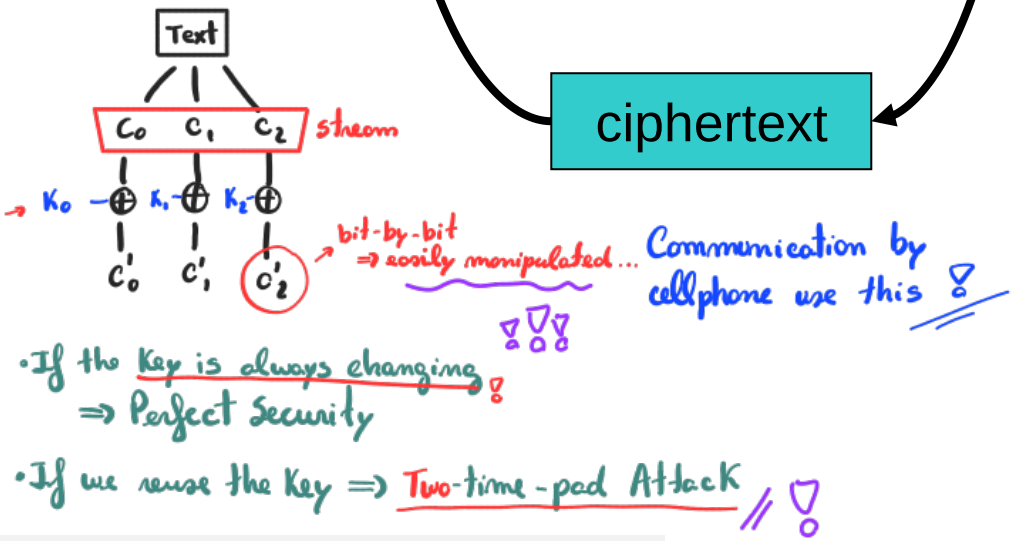
- Polyalphabetic ciphers
  - Each bit (0 or 1) is not always encrypted the same way

- Keystream
  - Randomly produced, as long as the processed data
    - Vernam cipher (or **one-time pad**)
    - The only perfect cipher (but rarely used, very unpractical)
  - Pseudo-randomly produced from a limited key
    - Ordinary stream ciphers

TESTÉ
Provado

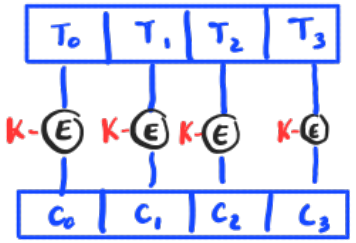
must be a pseudo random generator (for we to use the same seed)





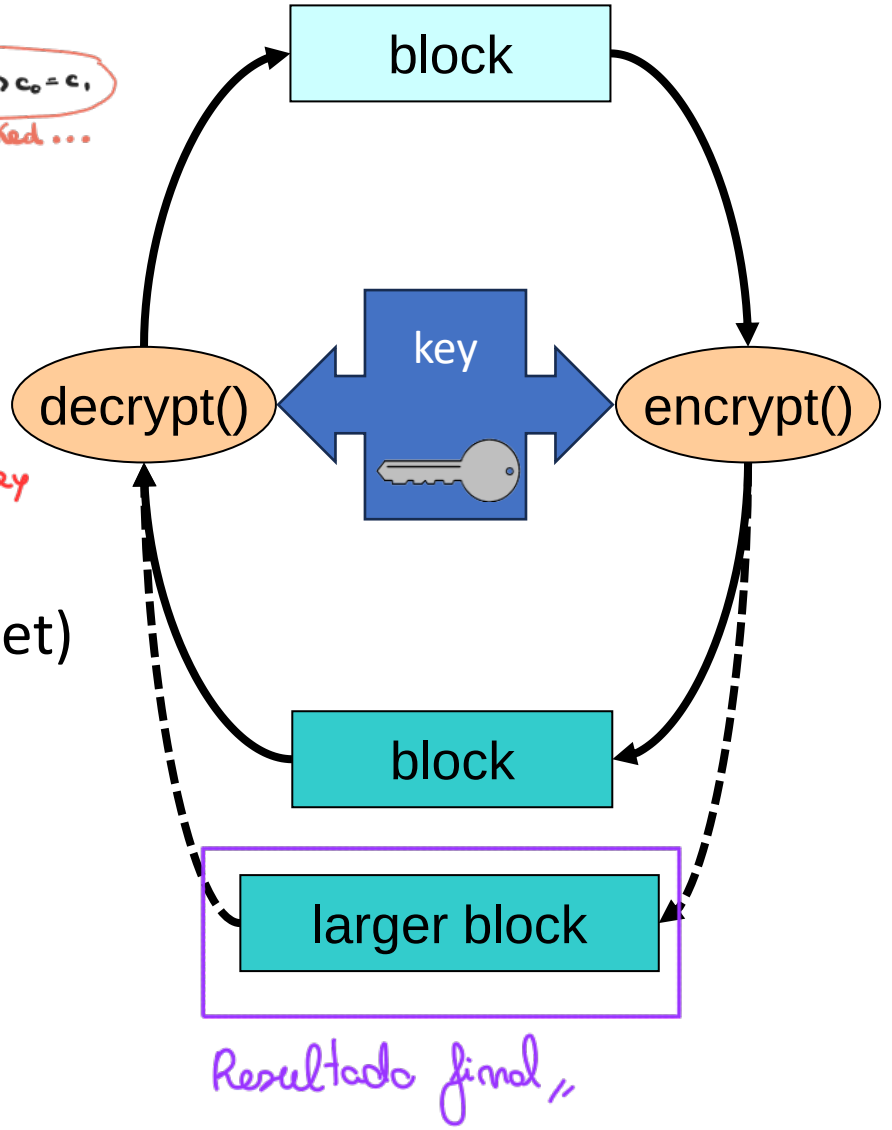
Computer ciphers: block ciphers

- Encrypt/decrypt sequences of blocks
  - Symbols are fixed-length blocks of bits
  - Usually use byte blocks as symbols



- Are monoalphabetic ciphers
  - Transform each symbol into another symbol
  - The key defines the transformation (substitution alphabet)

- Some may be polyphonic ciphers
  - Ciphertext blocks longer than plaintext blocks
  - Used in **randomized ciphers**



Computer ciphers: symmetric

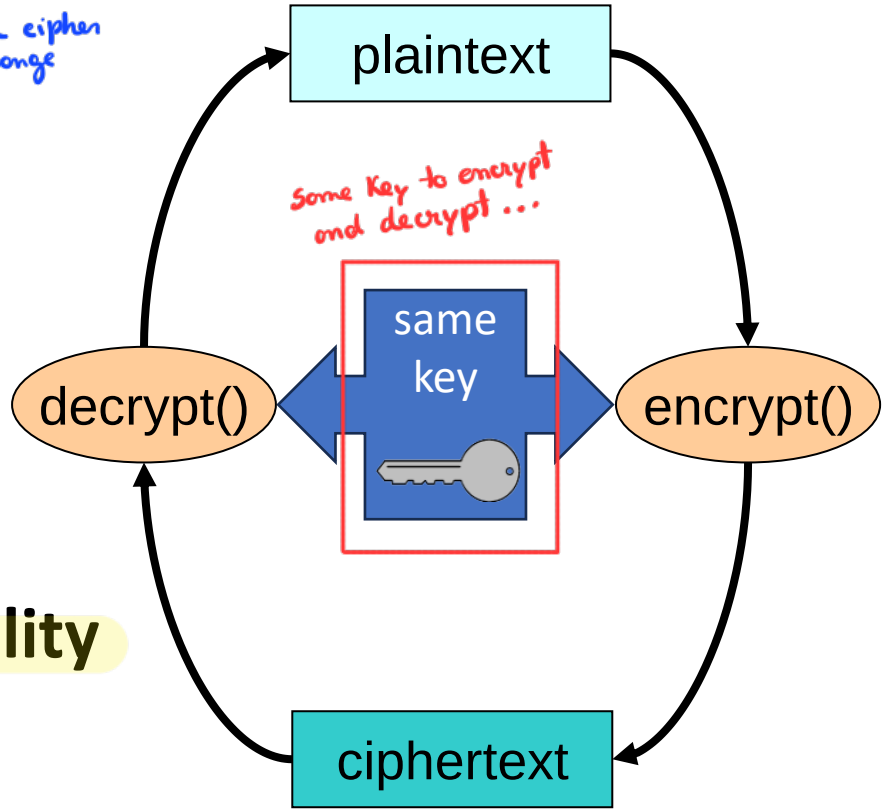
$m = D(E(m,K),K)$

1 bit change => ALL cipher change

- Encrypt/decrypt with the same key
  - The oldest strategy

Shared Key

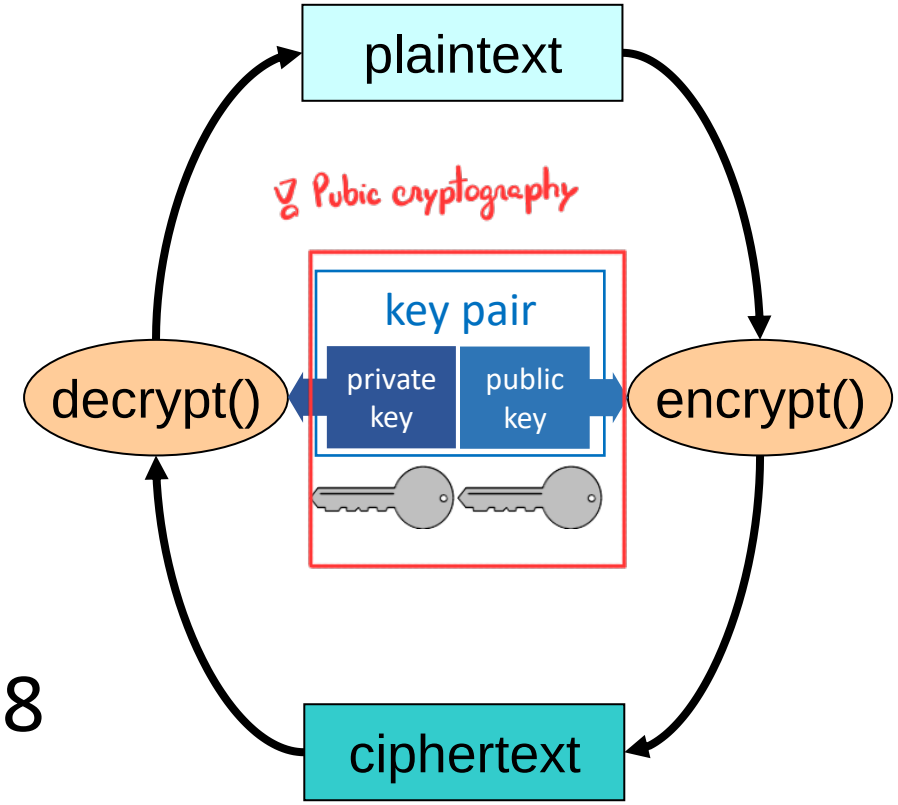
- Also called **secret key ciphers**



- Most common mechanism to provide **confidentiality**
  - Relatively simple to be implemented in software and hardware
  - Very good performance
  - Widely available across systems and platforms

Computer ciphers: asymmetric

- Encrypt/decrypt with two different keys
  - Key pair
- Key Pair
  - Private component, public component
  - Public computed from private (or from secret data)
- An approach that was first proposed in 1978
- Different algorithms work in different ways

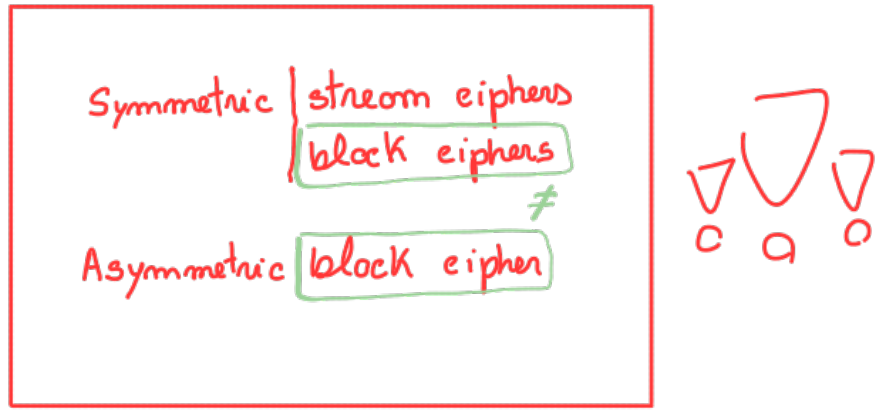


Computer ciphers: combinations

- (Symmetric) stream ciphers
  - Polyalphabetic ciphers
  - Keystream defined by the key
  - Keystream and XOR implement a polyalphabetic transformation

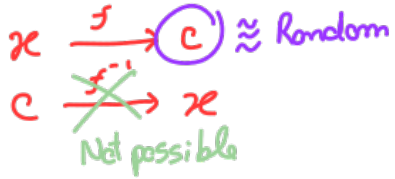
- Symmetric block ciphers
  - Monoalphabetic ciphers
  - Substitution alphabet is defined by the algorithm & key

- Asymmetric (block) ciphers
  - Polyphonic ciphers
    - Not by nature, but for security reasons
  - The functionalities of these ciphers are not homogeneous



Techniques used by ciphers

- Confusion
  - Complex relationship between the key, plaintext and the ciphertext
  - Output bits (ciphertext) should depend on the input bits (plaintext + key) in a very complex way



- Diffusion

The cipher is not related with text → 1 bit change → ALL ciphertext changes

  - Plaintext statistics are dissipated in the ciphertext
    - If one plaintext bit toggles, then the ciphertext changes substantially, in an unpredictable or pseudorandom manner
  - Avalanche effect

(Symmetric) stream ciphers: examples

- A5/1, A5/2
  - Cellular communications
  - Initially secret, reverse engineered
  - Explored in a weak fashion (64-bit keys w/ 10 bits stuck at zero)

- E0
  - Bluetooth communications
  - Keys up to 128 bits

- RC4
  - Wi-Fi communications (WEP, deprecated)
  - Initially secret, reverse engineered, never officially published
  - Keys with 40 to 2048 bits

- Other
  - Salsa20, Chacha20, etc.

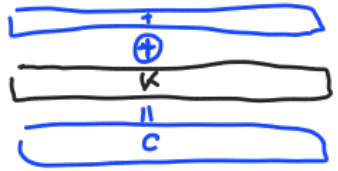
• Encode audio very rapidly...

A5/1, A5/2
E0
RC4
Salsa20
Chacha20

(Symmetric) stream ciphers: approach

the pseudo-random generator must be very good ... CAN'T REPEAT

- Use a cryptographically secure, pseudo-random bit **generator**
  - This generator produces the keystream
  - The generator implements a state machine
  - The generator is usually controlled by two values:
    - **Initialization Vector** (defines the initial state of the state machine) IV
    - **Key** (defines how one state moves to the next to produce the keystream)



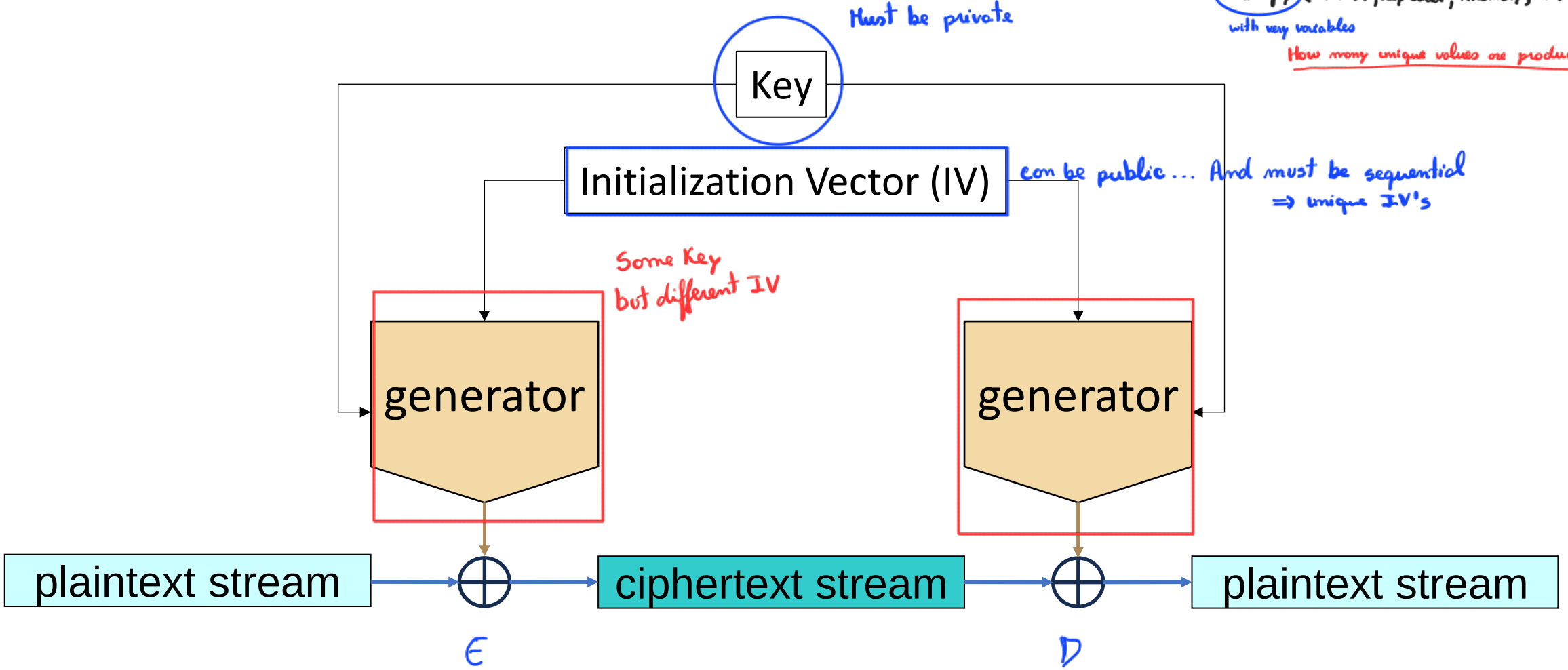
- Cryptographically secure, pseudo-random means:
  - Statistically, the keystream looks like a totally random sequence of zeros and ones
  - If an attacker learns a part of the keystream, it cannot infer: → it cannot find the rest
    - Past keystream values
    - Future keystream values

In python random this is not true...

(Symmetric) stream ciphers: approach

How to generate random values in IoT?
⇒ is always the same state...

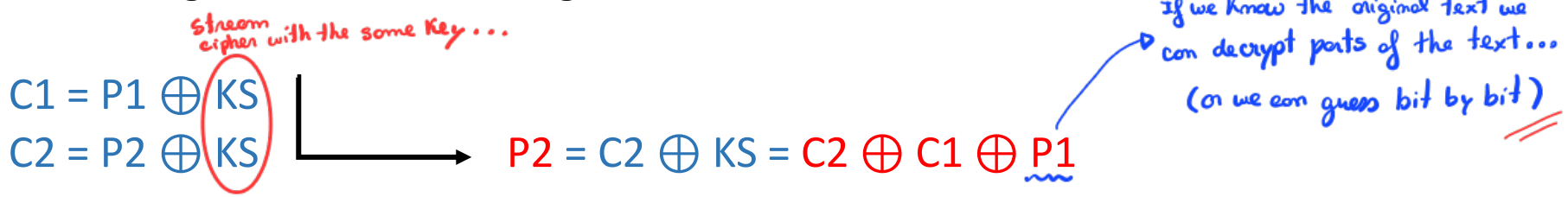
In linux:
→ entropy (mouse, Key board, memory, ...)
with very variables
How many unique values are produced?



Stream ciphers: exploitation considerations

Two-time-pad Attack

- No two messages (P1,P2) should be encrypted with the same key and IV
  - Because they will be encrypted with the same keystream (KS)
  - The knowledge about one message reveals the other



  - Knowledge about P1 => immediate knowledge about P2
  - Known/chosen-plaintext attacks become very effective!

Not reuse Keys!

- Keystreams may be periodic (have a cycle)
  - Depends on the type and quality of the generator
  - Same problem as the one above (KS is reused)
  - Plaintext should be shorter than the period length

Stream ciphers: exploitation considerations

- Ciphertexts can be deterministically manipulated
  - Stream ciphers are simple and have no capability to detect manipulation
  - Each cipher bit depends only on one plaintext bit

- It is fundamental to have integrity control elements
  - In the ciphertext; or
  - In the plaintext
  - Objective is to detect accidental or malicious changes to P

Symmetric Block ciphers: examples

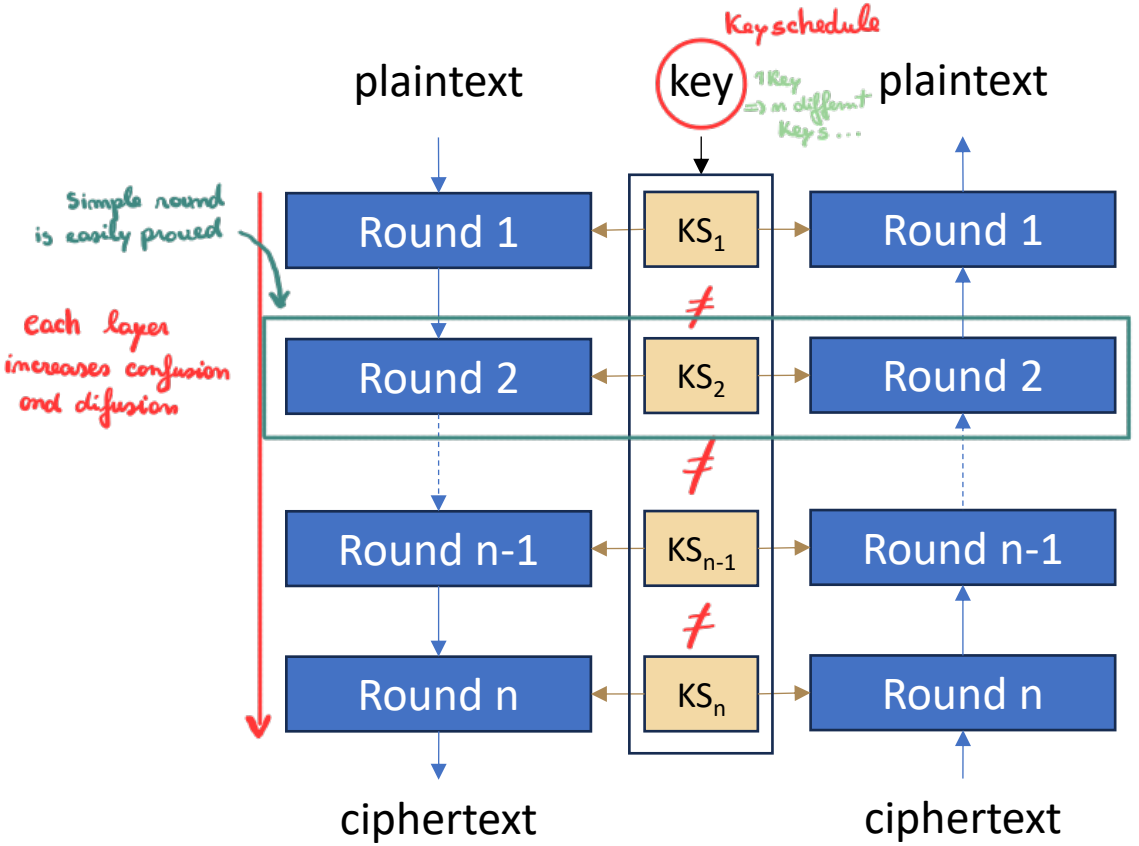
- DES (Data Encryption Standard)
  - Proposed in 1974, standard in 1977, nowadays deprecated
  - Input/output: 64-bit blocks
  - Key: 56 bits

- AES (Advanced Encryption Standard)
  - Proposed in 1998 (Rijndael), standard since 2001
  - Input/output: 128-bit blocks
  - Key: 128, 192 or 256 bits
  - Mosty commonly used symmetric cipher in applications

- Other
  - IDEA, CAST, Twofish, Blowfish, RC5, RC6, Kasumi, etc.

Symmetric block ciphers: approach

- Use a pipeline of transformation **rounds**
  - Each round adds confusion and diffusion
  - Each round is usually controlled by a **subkey**
    - Aka **key schedule**
    - A value derived from the encryption/decryption key
- Rounds need to be reversible
  - To allow decrypting what was encrypted
  - Make use of well-known structures:
    - Feistel networks
    - Substitution-permutation networks



Feistel network

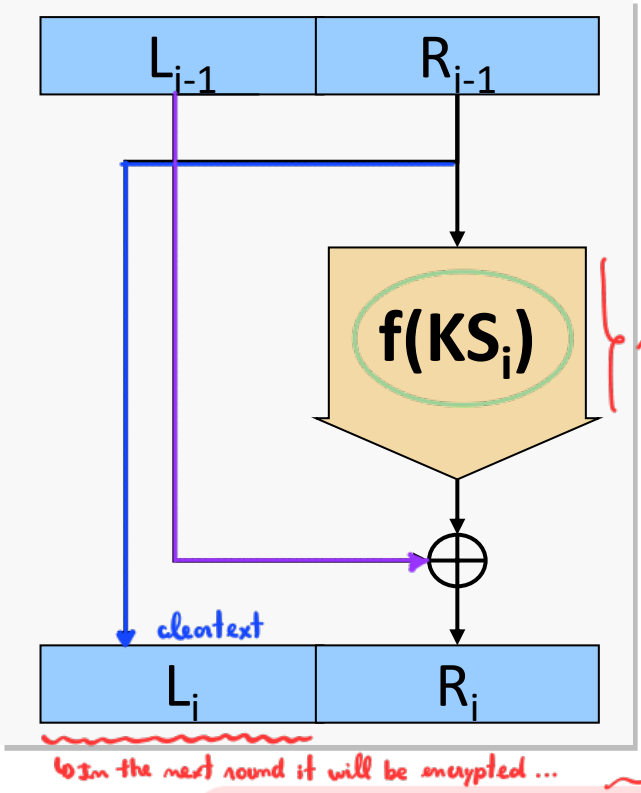
→ Pipeline of transformation rounds

$L_i = R_{i-1}$

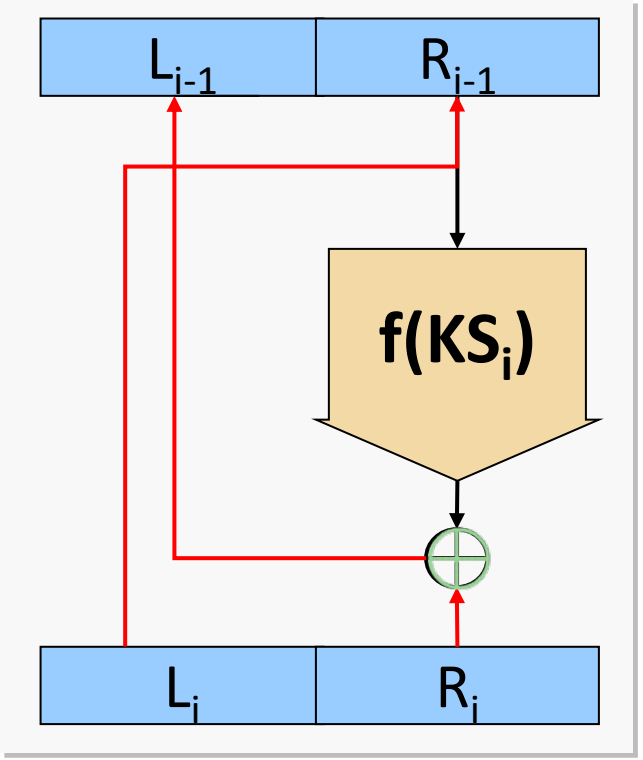
$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$ ← don't need to be reversible

$R_{i-1} = L_i$

$L_{i-1} = R_i \oplus f(L_i, K_i)$ } Decryption



like a permutation function ...



The function **$f(KS_i)$** doesn't need to be reversible!

Substitution-Permutation network

For the function f

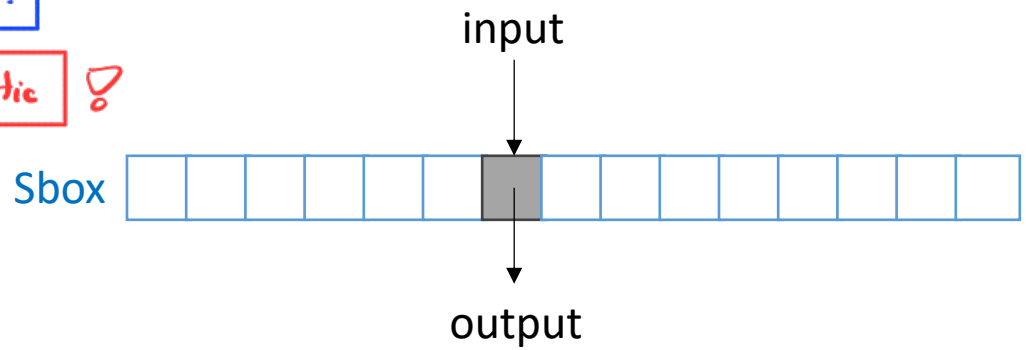
can be a look-up table ...

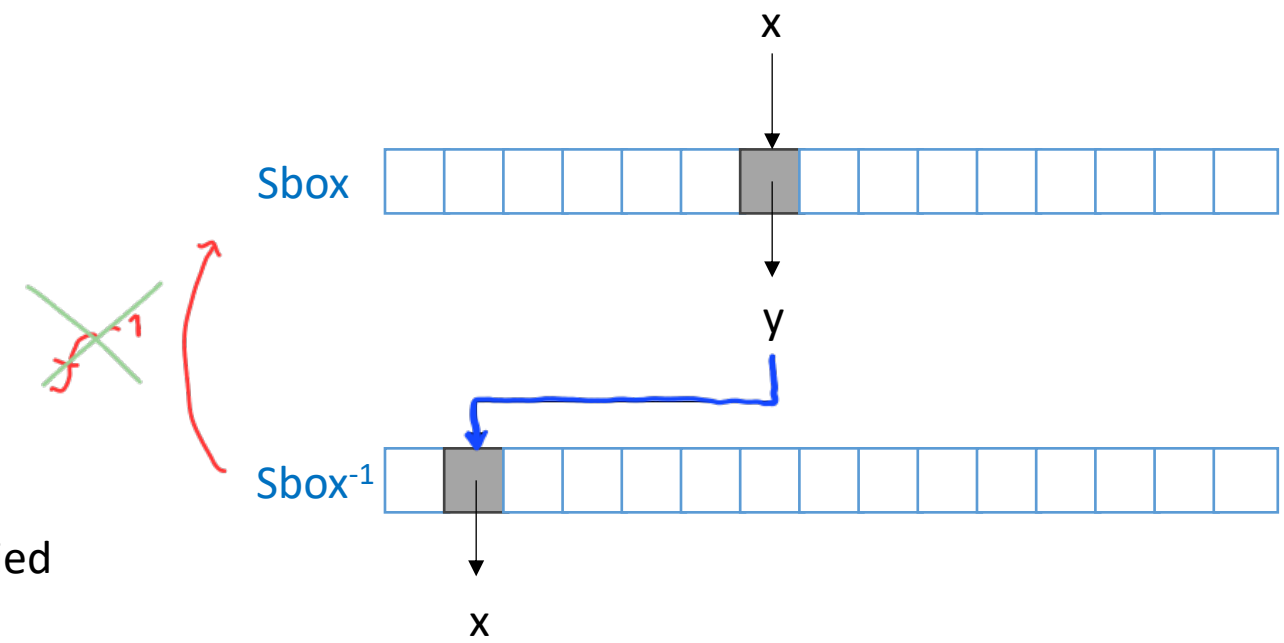
Must be deterministic

- SBox – Substitution Box
  - Table with an output for an input (index)

    output = SBox[input]
  - SBoxes may be constant or key-dependent
    - DES and AES use constant Sboxes
    - Blowfish and Twofish use variable, key-dependent SBoxes
  - In SP networks, SBoxes must be reversible
    - Bijective transformations

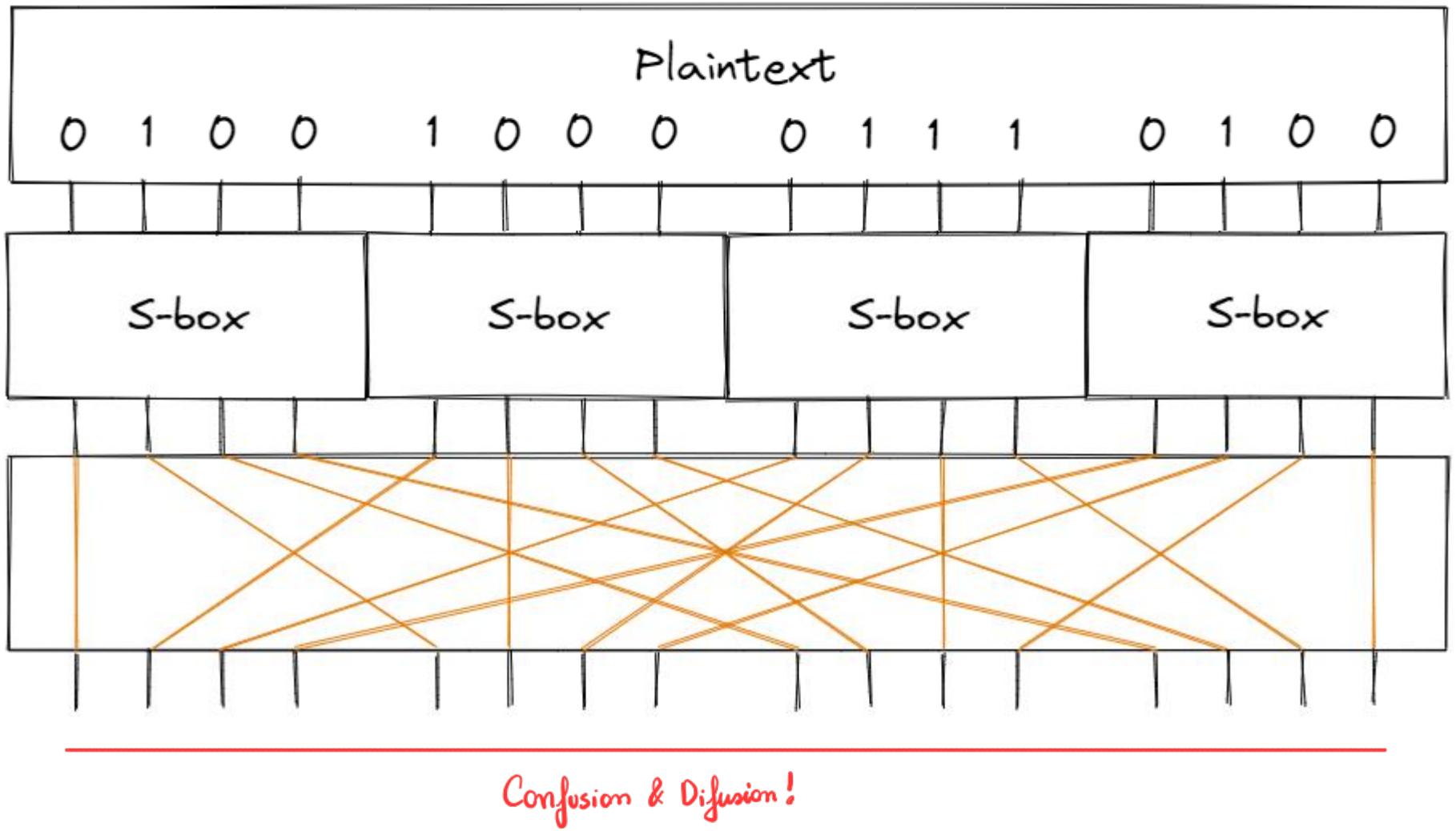
      $y = SBox[x]$ $x = SBox^{-1}[y]$





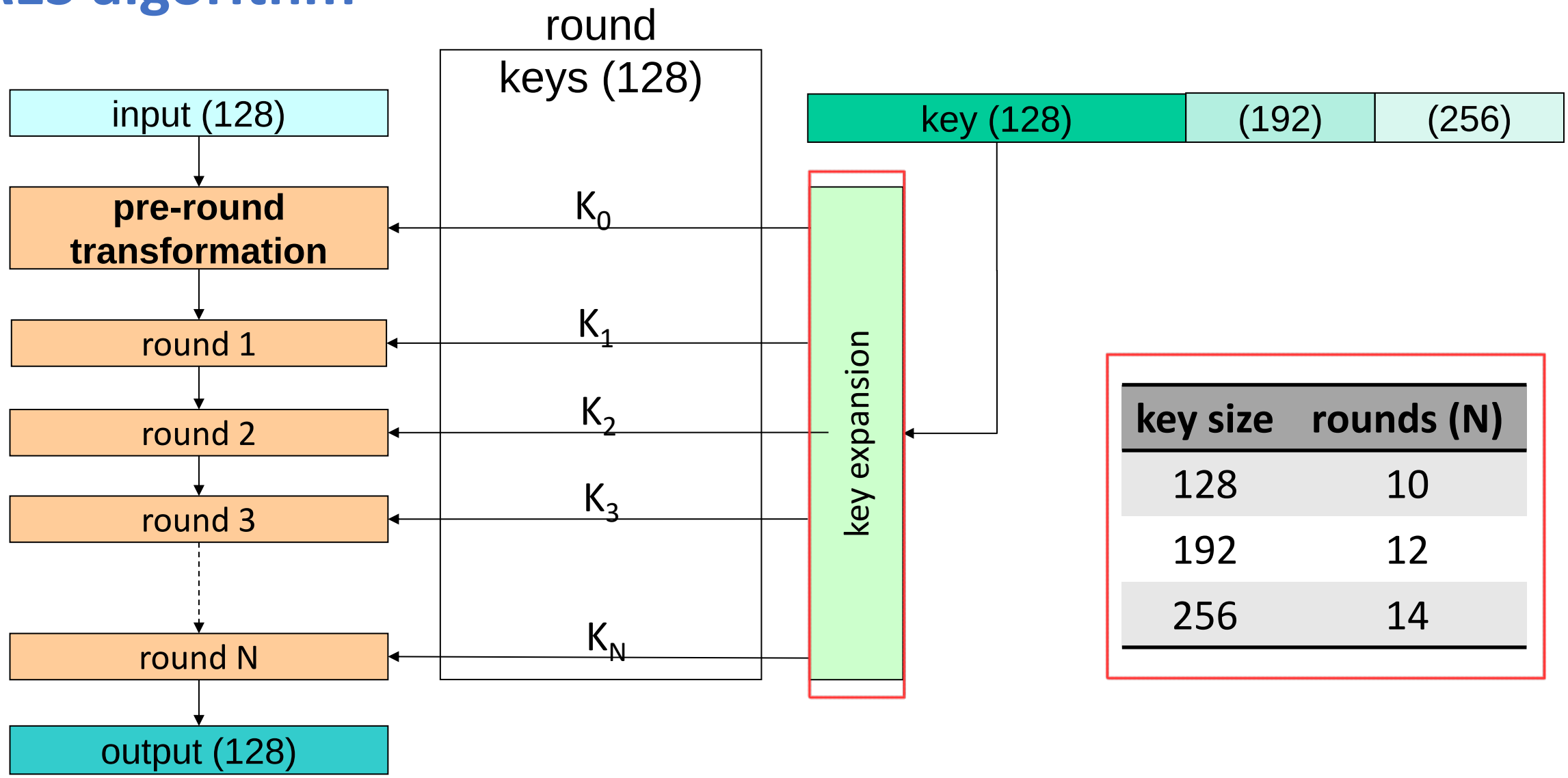
- PBox – Permutation Box
  - Changes the positions of the input bits
    - Bits are not modified, only the position is modified

Substitution-Permutation network



AES algorithm

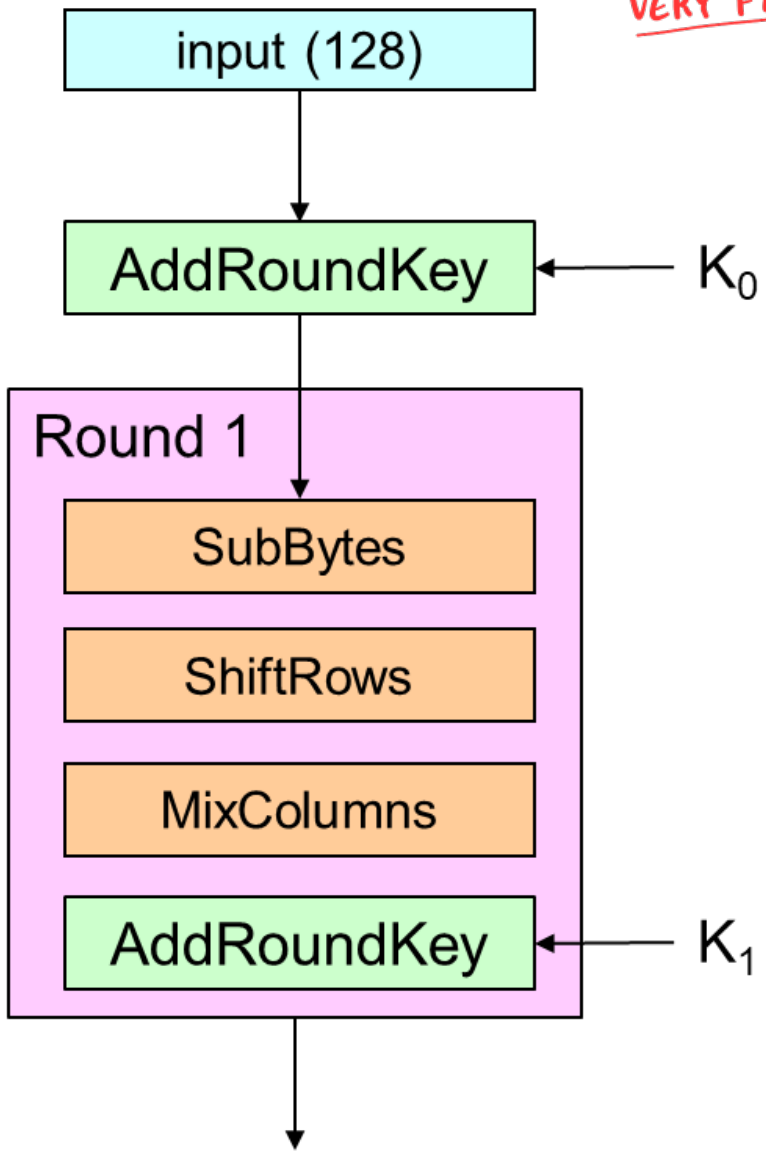
✗ Not details...



key size	rounds (N)
128	10
192	12
256	14

AES (enccryption) round

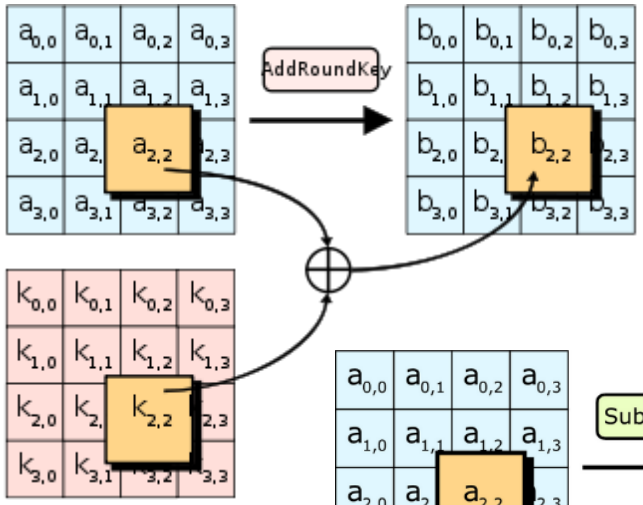
X Not details...



VERY FAST

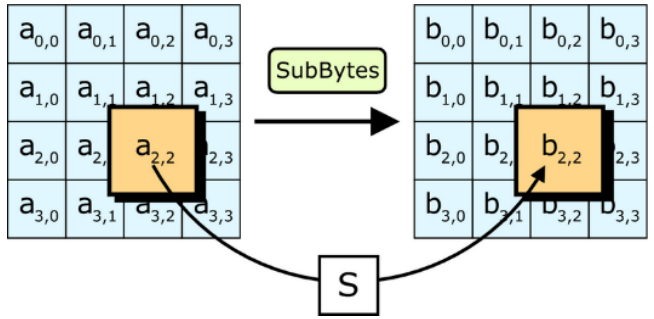
AddRoundKey

- 128-bit XOR
- Output is a 4x4 byte matrix



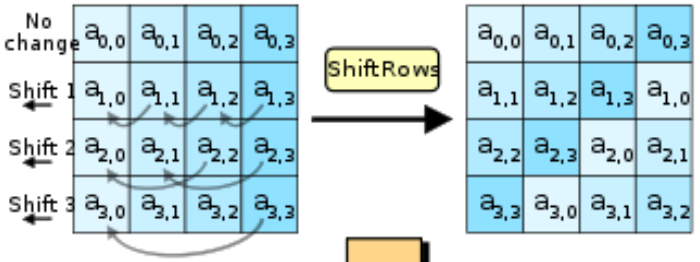
SubBytes

- 256-element S-box
- Each matrix byte is substituted



ShiftRows

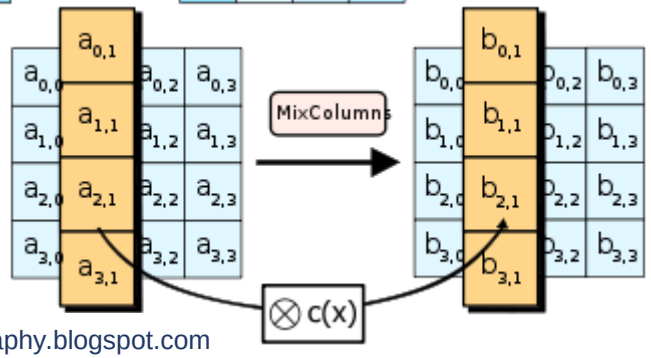
- Rows are rotated left
- Byte shifts vary (0, 1, 2 & 3)



Lookup-tables

MixColumns

- Each column is transformed
- Not performed in the last round



https://aescryptography.blogspot.com

AES in CPU instruction sets

- Intel AES New Instructions (AES-NI)

The impact o performance is very low...
BitLocker in windows use this!

AESENC	Perform one round of an AES encryption flow
AESENCLAST	Perform the last round of an AES encryption flow
AESDEC	Perform one round of an AES decryption flow
AESDECLAST	Perform the last round of an AES decryption flow
AESKEYGENASSIST	Assist in AES round key generation
AESIMC	Assist in AES Inverse Mix Columns

- ARMv8 Cryptographic Extension
  - … and other

Cipher Modes: Electronic Code Book (ECB)

How to not do it in large scale...!

X Not semantically Secure

- Direct encryption of each block: $C_i = E_K(P_i)$
- Direct decryption of each block: $P_i = D_K(C_i)$

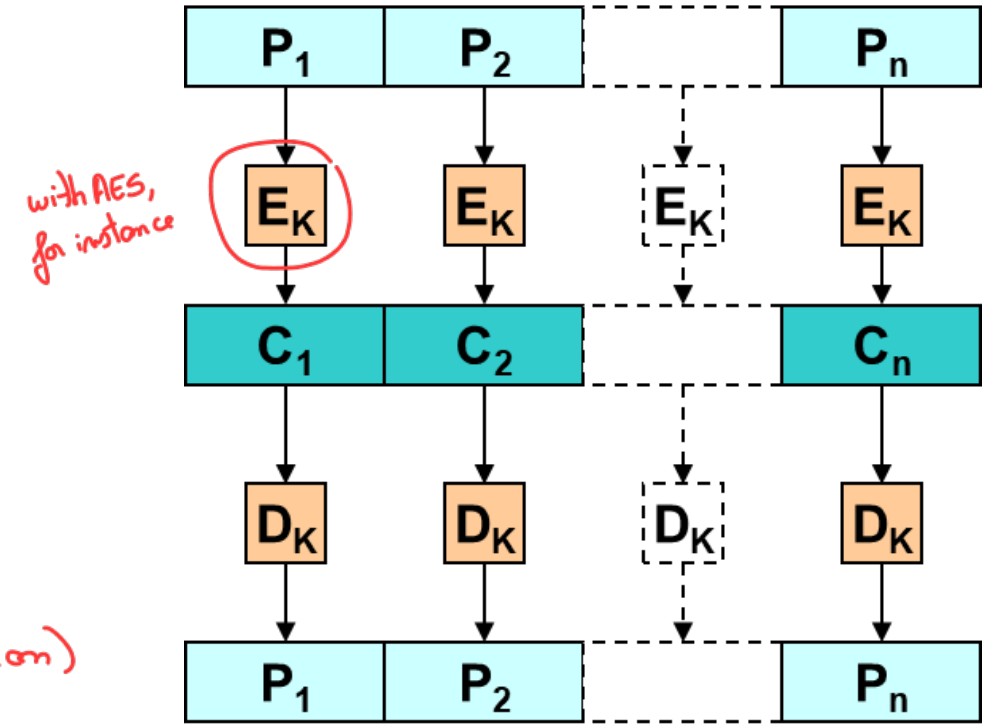
Parallelism!

Monoalphabetic

- Blocks are processed independently
  - Parallelism is possible
  - Uniform random access exists
- Problem:
  - Exposes patterns existing in the clear text
  - If $P_i = P_j$ then $C_i = C_j$

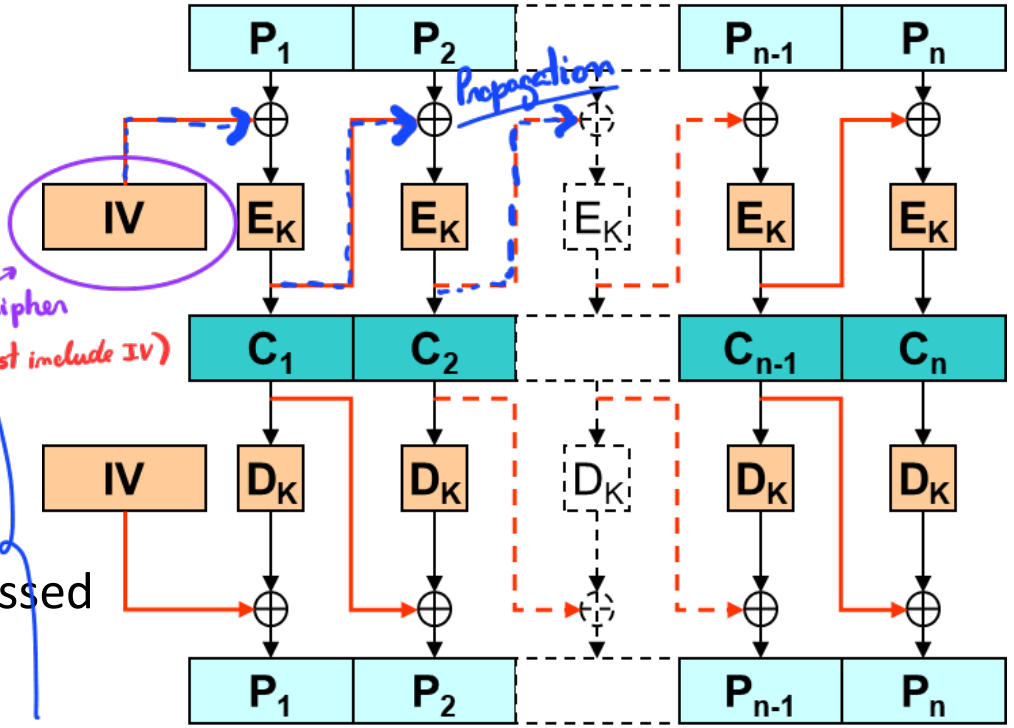
Patterns

← Block cipher typical problem (No propagation)

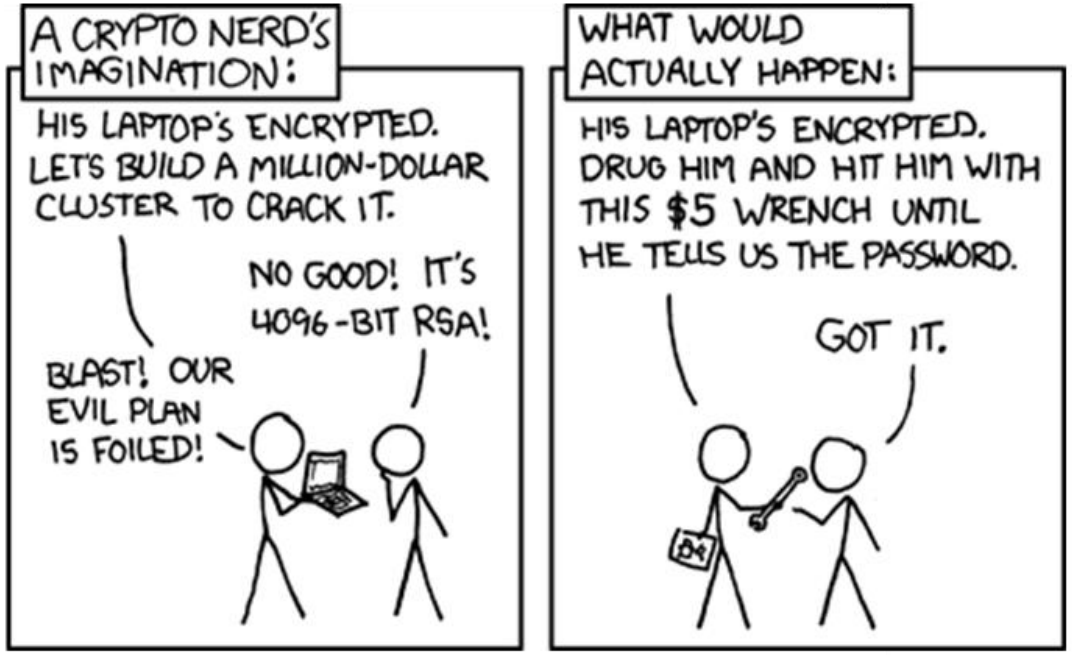


Cipher Modes: Cipher Block Chaining (CBC)

- Encrypt each block T_i combined with C_{i-1}: $C_i = E_K(P_i \oplus C_{i-1})$
- Decrypt each block C_i combined C_{i-1}: $P_i = D_K(C_i) \oplus C_{i-1}$
  - Parallelism and uniform random access is possible
- First block uses an IV (Initialization Vector)
  - Better not reuse for the same key
  - Random value, sequence value, etc.
  - May be sent openly
- Polyalphabetic transformation
  - The feedback prevents equal blocks from being equally processed
  - Seems like we have a different key per block

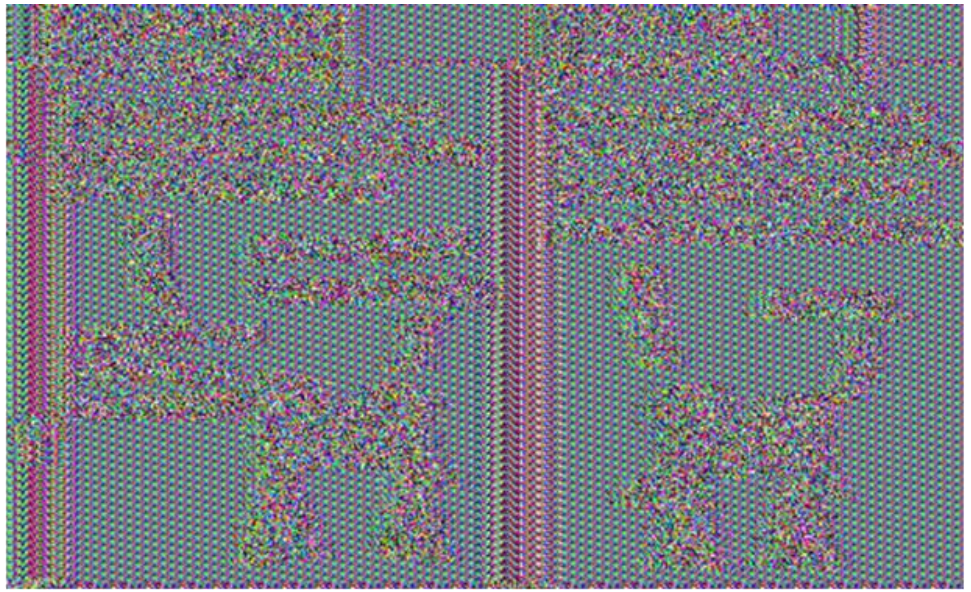


ECB vs CBC: pattern exposure



https://xkcd.com/538/

ECB



Patterns

Because $P_0 = P_1 \Rightarrow C_0 = C_1$

Not semantically secure

should not happen ...

CBC

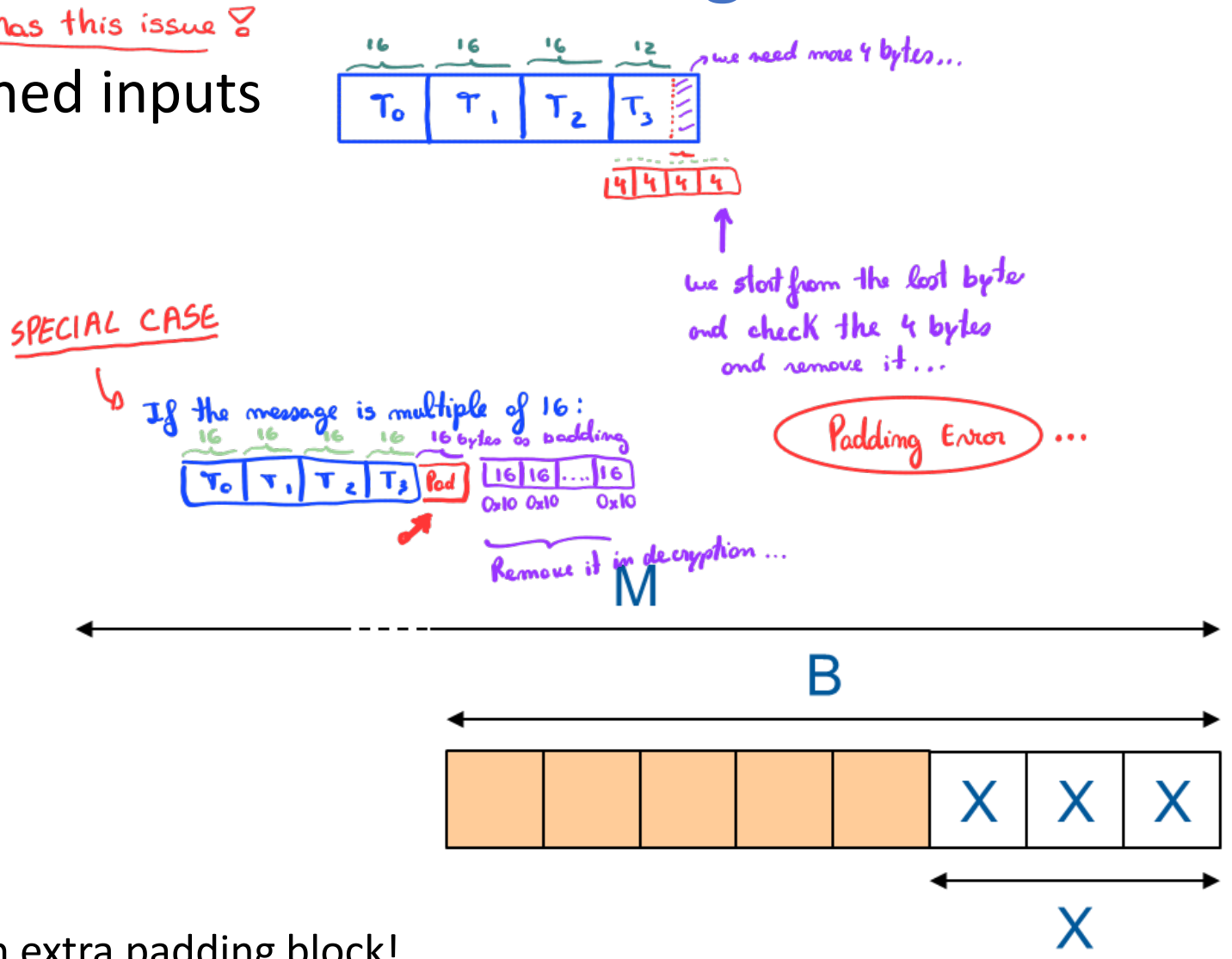
we need to complement with MAC for Authentication ...

ECB/CBC cipher modes: contents not block-aligned

- ECB and CBC require block-aligned inputs
  - Final sub-blocks need special treatment

- Alternatives
  - Padding
    - Of last block, identifiable
  - Different processing for the sub-block
    - Adds complexity, rarely used

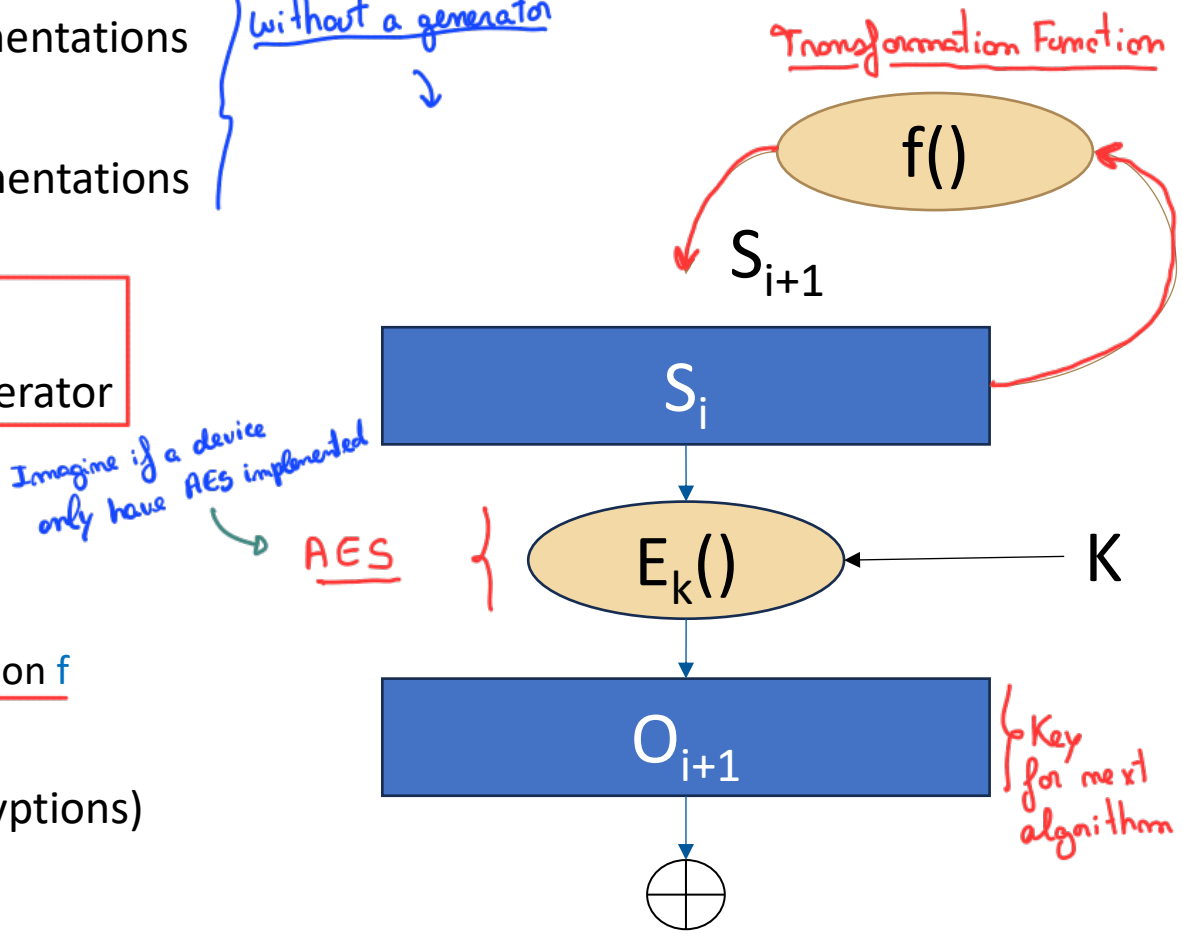
- PKCS #7 padding
  - **X = B – (M mod B)**
  - **X** extra bytes, with the value **X**
  - PKCS #5: Equal to PKCS #7 with **B = 8**
  - Drawback: perfectly aligned inputs get an extra padding block!



Stream cipher modes

- Stream ciphers use a pseudorandom generator
  - There are multiple techniques to implement them
  - Some techniques are specially suited for hardware implementations
    - Typically used in mobile, battery-powered devices
  - Other techniques are more suitable for CPU-based implementations

- Stream cipher modes
  - They use a block cipher to implement a stream cipher generator
  - The fundamental idea is:
    - The generator is a state machine with state S_i on iteration i
    - The output of the generator for state S_i is $O_{i+1} = E_K(S_i)$
    - The state S_i is updated to S_{i+1} using some transformation function f
    - S_0 is defined by an IV
  - The generator only uses block cipher encryptions (or decryptions)



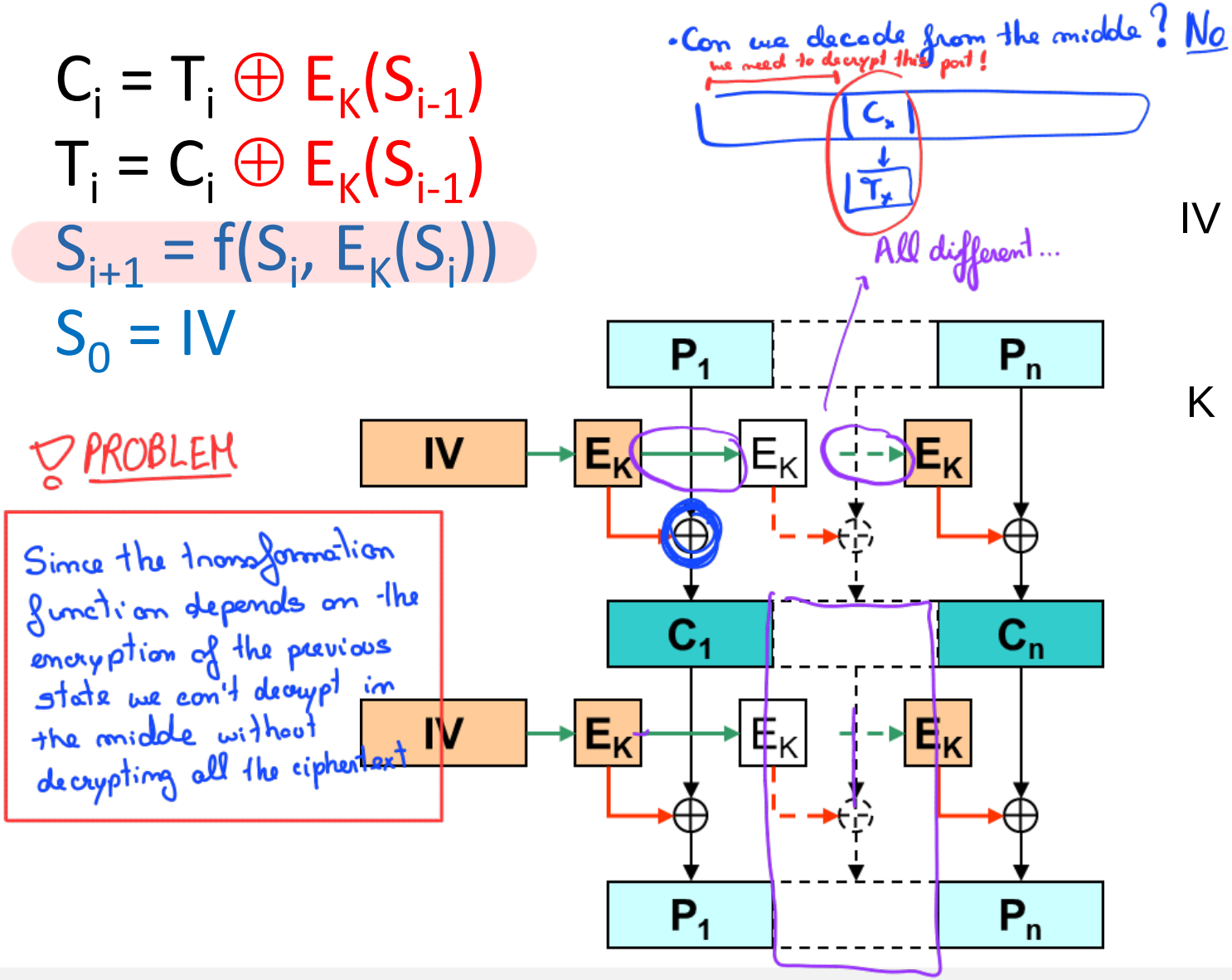
Stream cipher modes: n-bit OFB (Output Feedback)

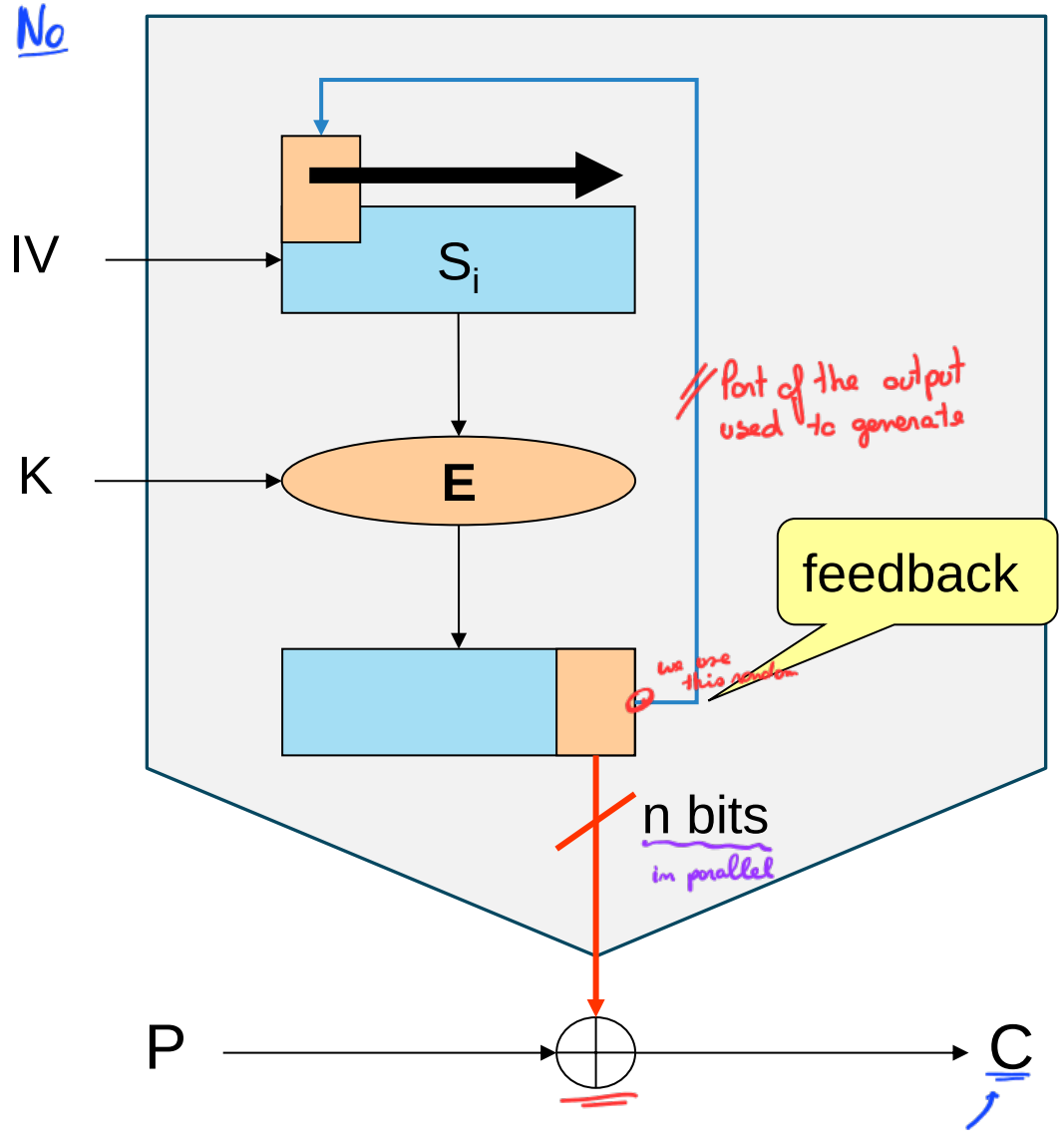
$C_i = T_i \oplus E_K(S_{i-1})$

$T_i = C_i \oplus E_K(S_{i-1})$

$S_{i+1} = f(S_i, E_K(S_i))$

$S_0 = IV$





Stream cipher modes: n-bit CFB (Ciphertext Feedback)

$C_i = T_i \oplus E_K(S_{i-1})$

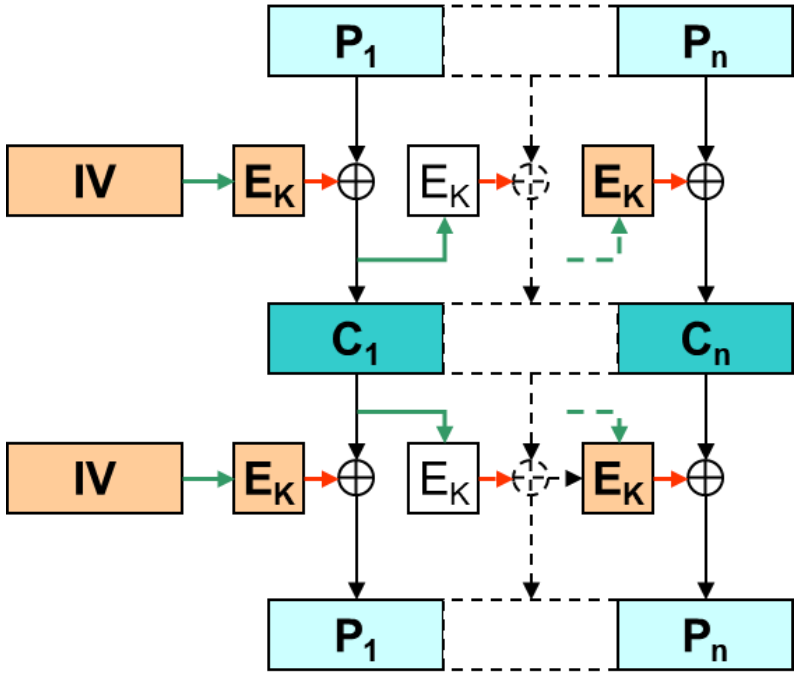
$T_i = C_i \oplus E_K(S_{i-1})$

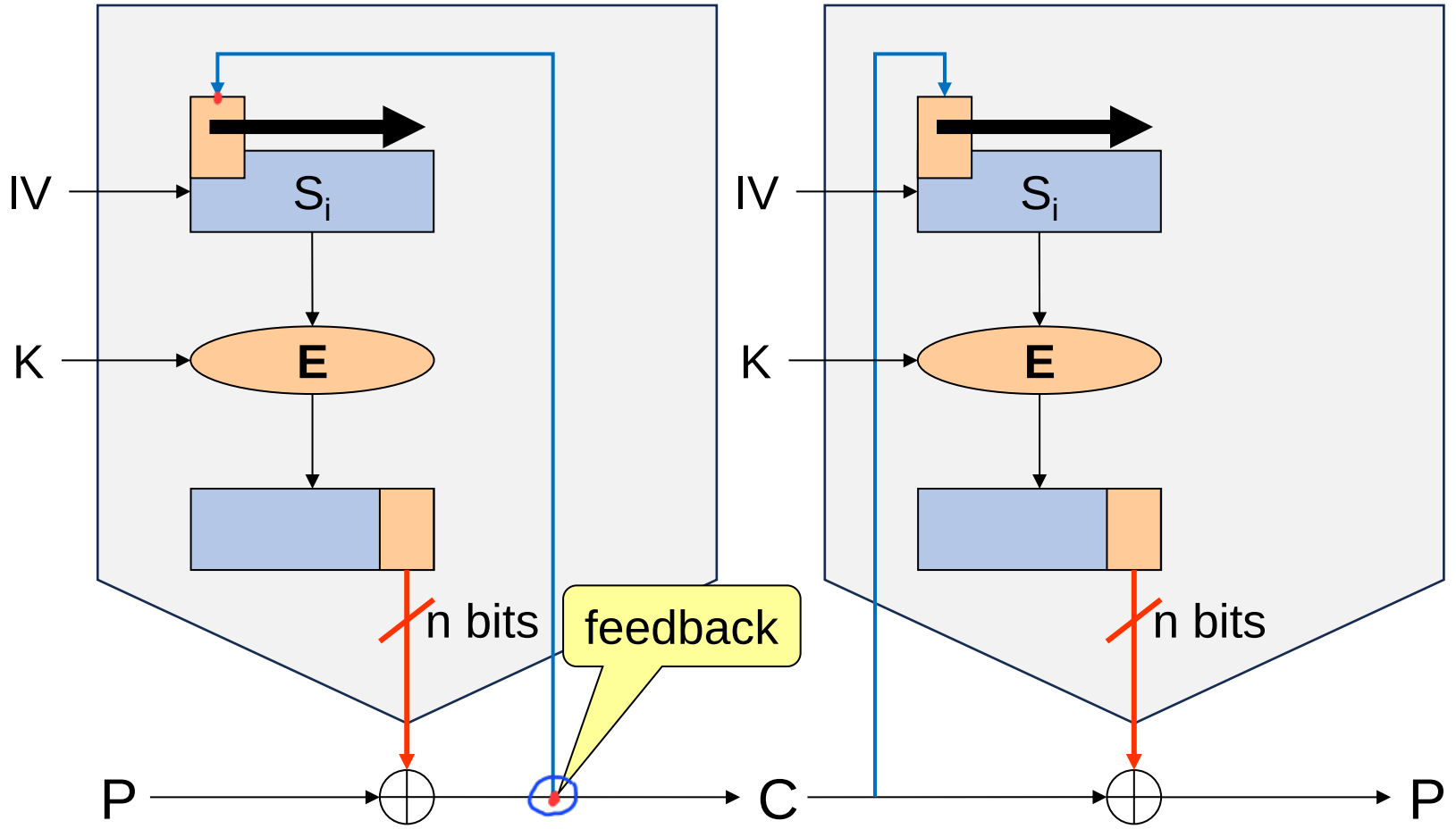
$S_{i+1} = f(S_i, C_i)$

$S_0 = IV$

PROBLEM

with one bit corrupted in the ciphertext the all message is impossible to decrypt ... (from there on ...





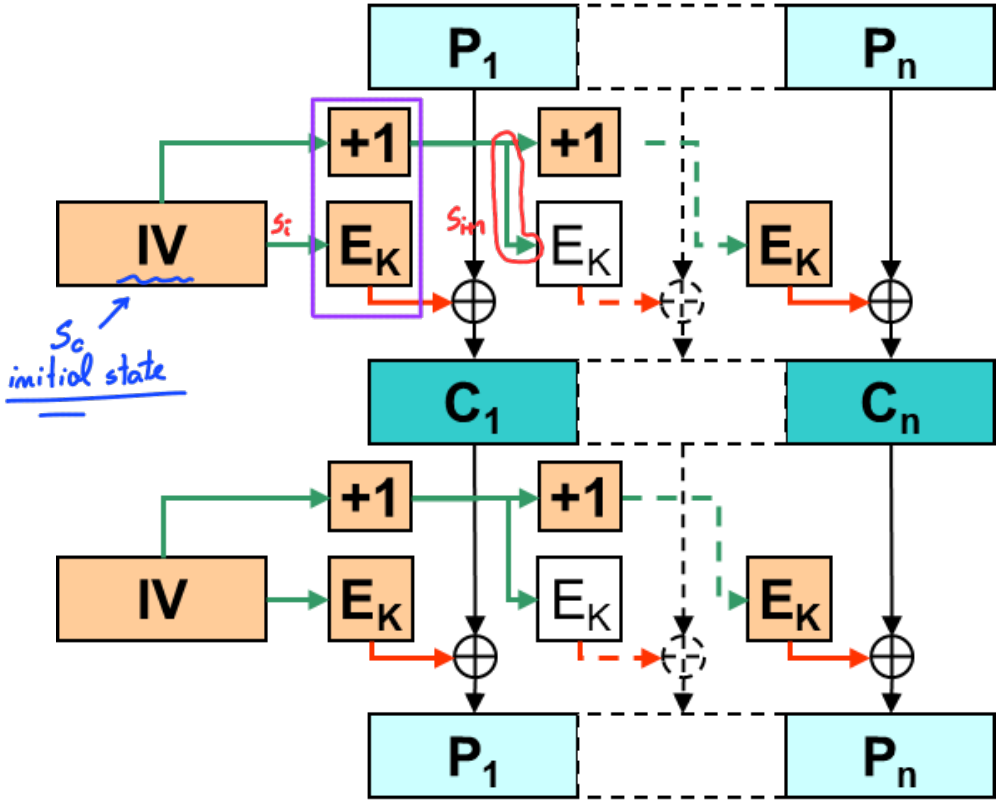
Cipher modes: n-bit CTR (Counter)

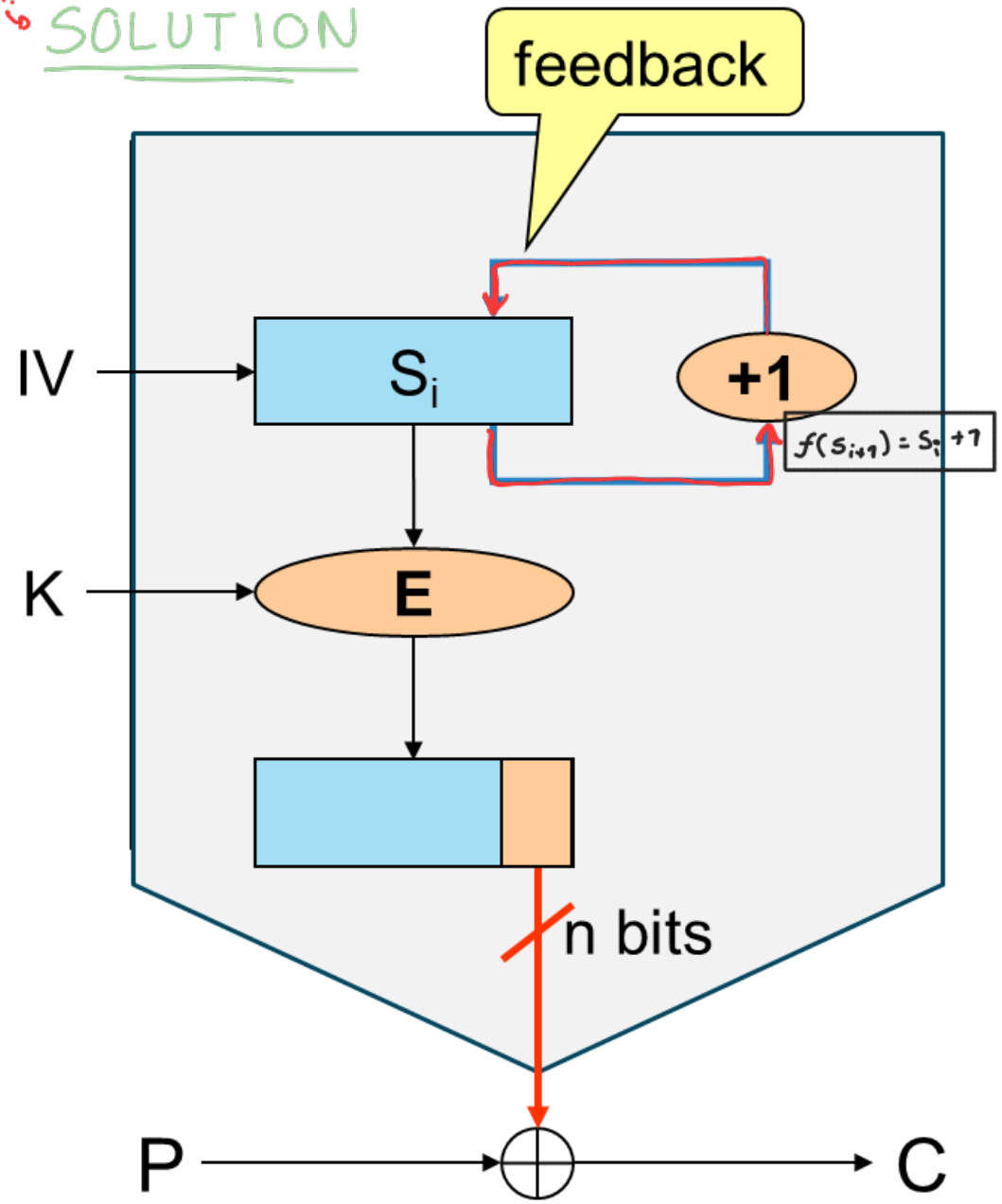
$C_i = T_i \oplus E_K(S_{i-1})$

$T_i = C_i \oplus E_K(S_{i-1})$

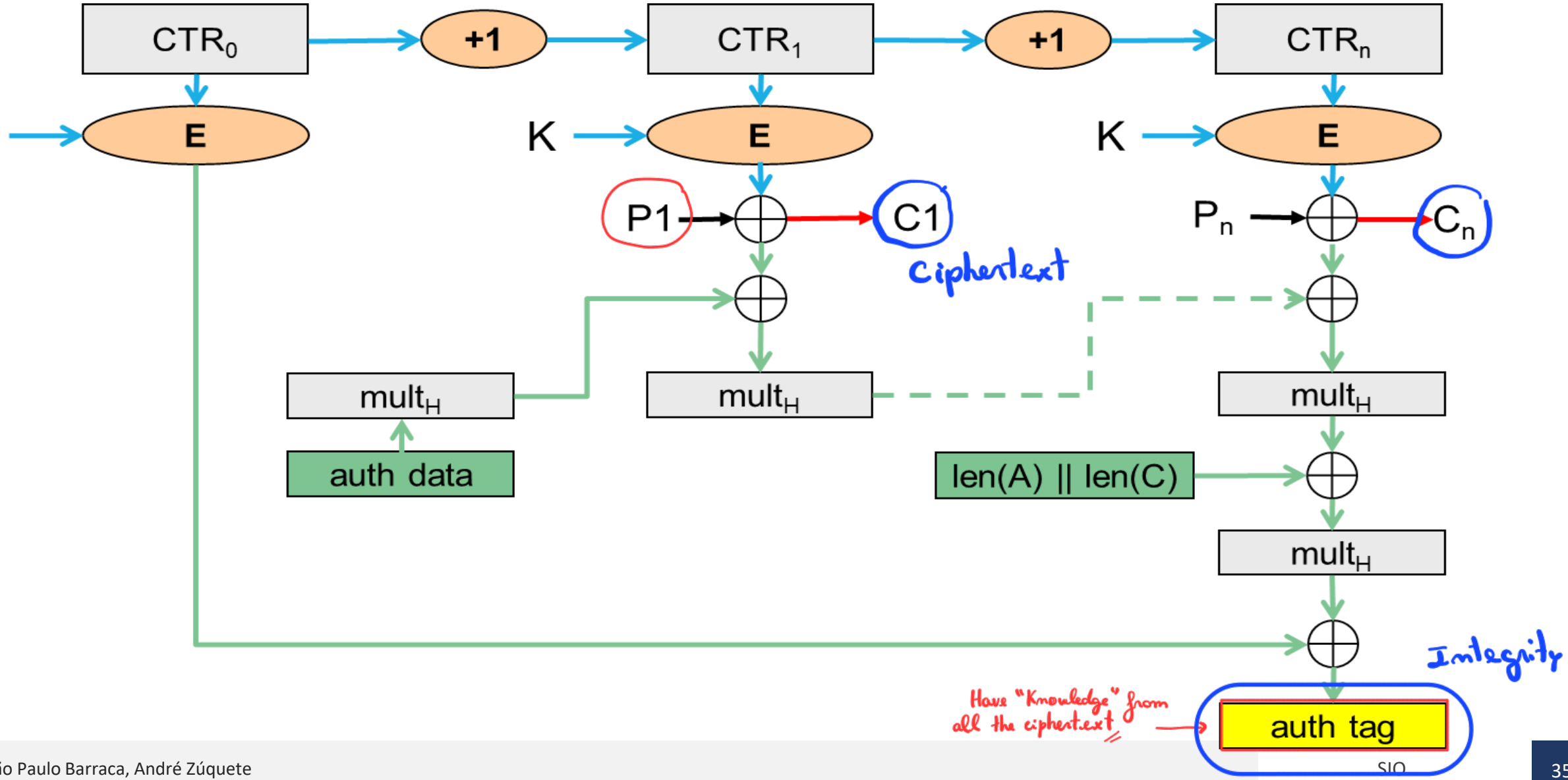
$S_{i+1} = S_i + 1$

$S_0 = IV$





Stream cipher modes: Galois with Counter Mode (GCM)



Cipher Modes: comparison

	Block		Stream			
	ECB	CBC	OFB	CFB	CTR	GCM
Input pattern hiding	✗	✓	✓	✓	✓	✓
Same key for different messages	✓	✓	other IV			
Tampering difficulty	✓	✓ (...)		(...)		✓
Pre-processing			✓		✓	✓
Parallel processing	✓	decrypt	With pre-proc	decrypt	✓	✓
Uniform random access						
Cryptogram single bit error propagation on decryption	same block	same & next block		a few next bits		detected
Capacity to recover from losses	some	some		some		detected

Cipher Modes: multiple encryption

- Invented for extending the lifetime of DES
  - DES was never cryptanalyzed
  - But its key was too short (56 bits only)
  - Its key could be discovered by brute force

TESTE

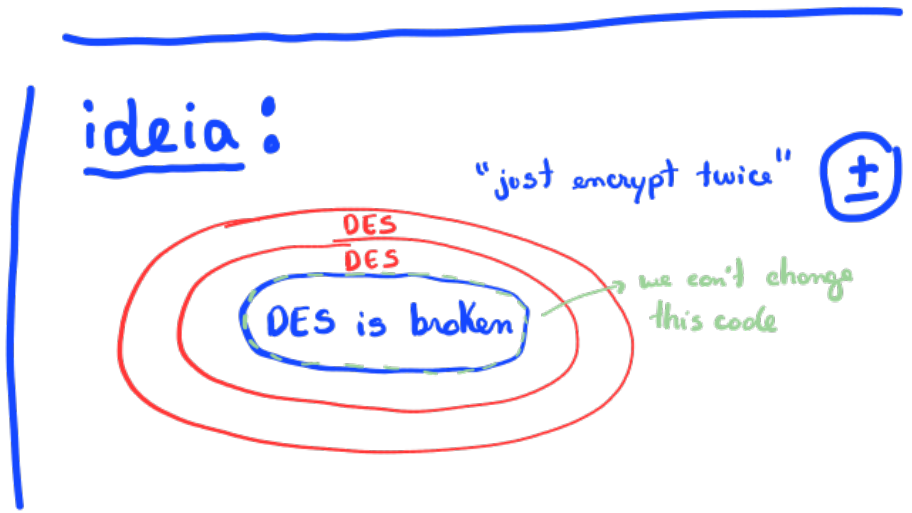
Encrypt - Decrypt - Encrypt

- Triple encryption EDE, or 3DES-EDE

$C_i = E_{K3}(D_{K2}(E_{K1}(P_i)))$ $P_i = D_{K1}(E_{K2}(D_{K3}(C_i)))$

chaves diferentes

  - With K1 ≠ K2 ≠ K3, it uses a 168-bit key
  - With K1 = K3 ≠ K2, it uses a 112-bit key
  - If K1 = K2 = K3 , then we get simple encryption
  - In all cases, **3 times slower than DES**



Cipher Modes: DESX

- Another solution for extending the lifetime of DES
  - Much faster than 3DES
  - Two extra keys are used to add confusion
    - Before the cipher input
    - After the cipher output

$C_i = E_K(K_1 \oplus P_i) \oplus K_2 \qquad P_i = K_1 \oplus D_K(K_2 \oplus C_i)$

Reuse the algorithms

- The equivalent key length is 184 bits
  - 64 + 64 + 56 bits
  - More than with 3DES