Modern Cryptography

Symmetric ciphers

Cryptography: terminology (1/2)

Cryptography

- Art or science of hidden writing (confidential writing)
  - from Gr. kryptós, hidden + graph, r. de graphein, to write
- Initially used to maintain confidentiality of information

- **Steganography**: art of concealing data
  - from Gr. steganós, hidden + graph, r. de graphein, to write

Cryptanalysis

- Art or science of breaking cryptographic systems or encrypted information

Cryptology

- Cryptography + cryptanalysis

Cryptography: terminology (2/2)

Cipher

- Specific cryptographic technique

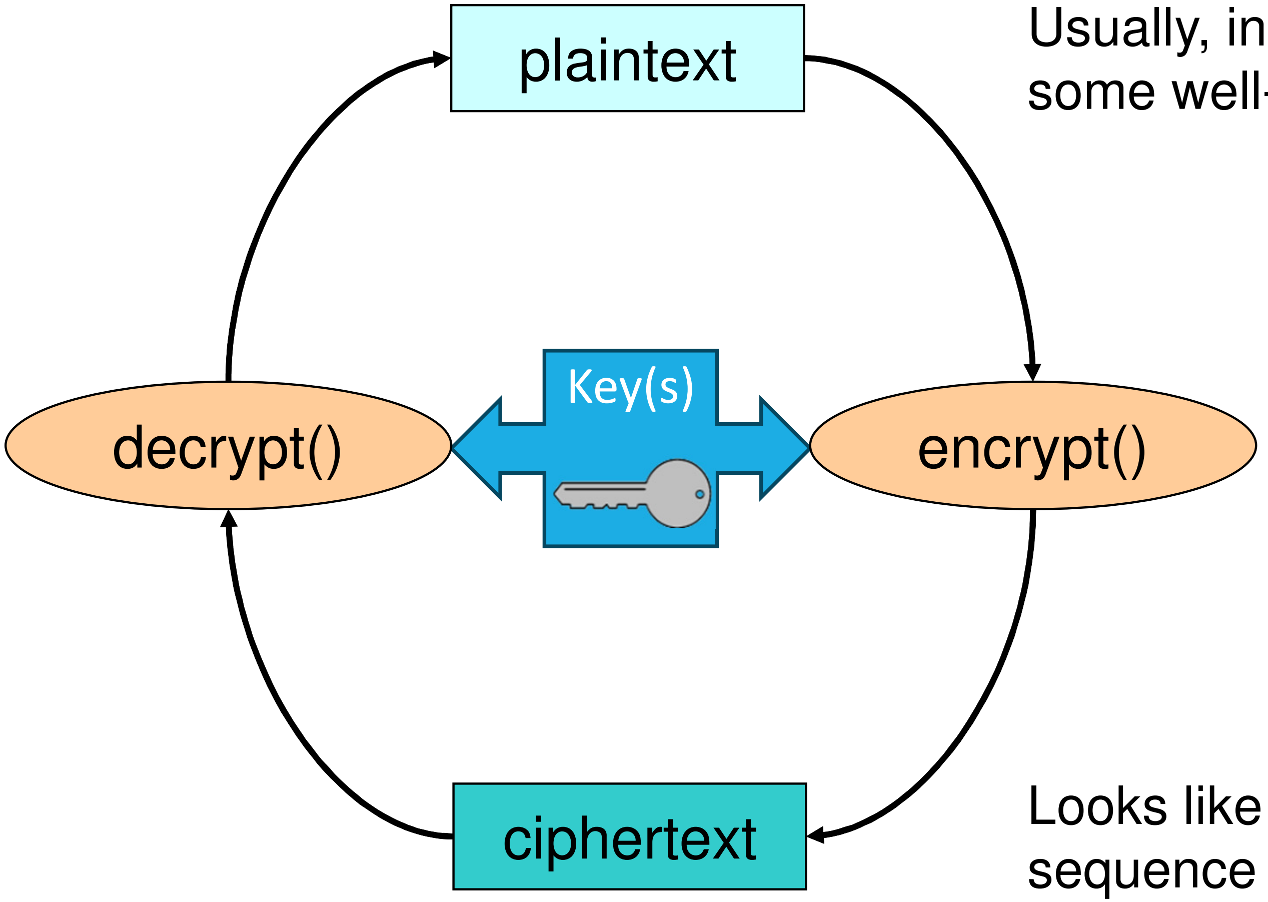
Cipher operation

- **Encryption**: original information → cryptogram
- **Decryption**: cryptogram → original information

- Original information aka plaintext or cleartext
- Cryptogram aka ciphertext

- **Algorithm**: way of transforming data
- **Key**: algorithm parameter
  - Influences algorithm execution

Operations of a Cipher



Usually, information that follows some well-know format

Looks like a random sequence of symbols

Use cases (symmetric ciphers)

Self protection with 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**

Secure communication with 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

Cryptanalysis: goals

Discover original plaintext

- Which originated a given ciphertext

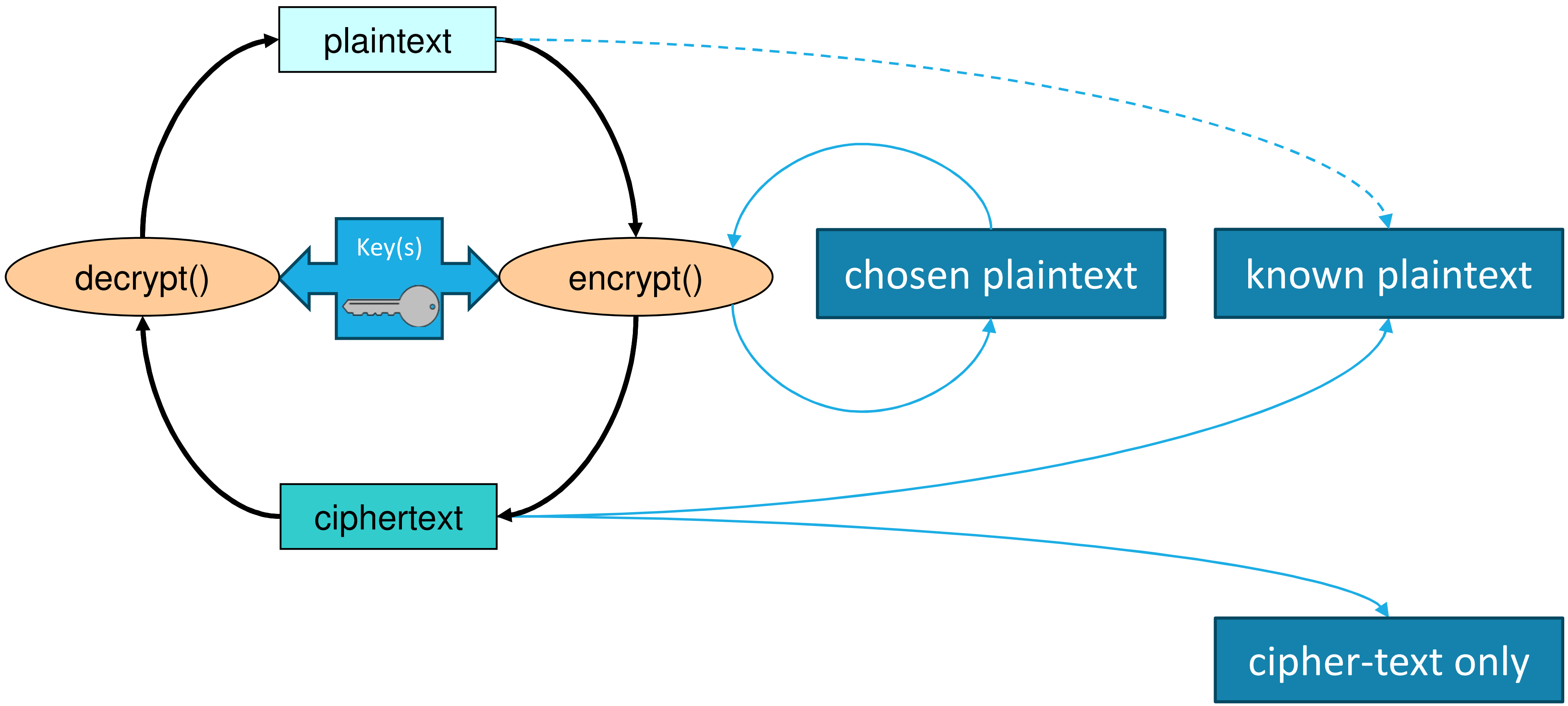
Discover a cipher key

- Allows the decryption of ciphertexts created with the same key

Discover the cipher algorithm

- Or an equivalent algorithm
- Usually algorithms are not secret, but there are exceptions
  - Lorenz, A5 (GSM), RC4 (WEP) , Crypto-1 (Mifare)
  - Algorithms for DRM (Digital Rights Management)
- Using reverse engineering

Cryptanalysis attacks: Approaches



Cryptanalysis attacks: Approaches

Brute force

- Exhaustive search of the key space until finding a match
- Usually unfeasible for a large key space
  - e.g., 128 bits keys have a search space of 2^{128} values
- Randomness is fundamental!

Clever attacks

- Reduce the search space to a smaller set of potential candidates: words, numbers, restricted size or alphabet
- 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

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

  plaintext ⊕ keystream → ciphertext

  ciphertext ⊕ keystream → plaintext

Are polyalphabetic ciphers

- Usually explored in low-level communication protocols

Keystream

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

Computer ciphers: block ciphers

Encrypt/decrypt sequences of blocks

- Symbols <=> fixed-length blocks of bits
- Usually use byte blocks as symbols

Are monoalphabetic ciphers

- Some may be polyphonic ciphers

Computer ciphers: symmetric

Encrypt/decrypt with the same key

- The oldest strategy

Computer ciphers: asymmetric

Encrypt/decrypt with the different, related keys

- Key pair
  - Private component, public component
- An approach that was first proposed in 1978

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 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

- 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.

(Symmetric) stream ciphers: Approach

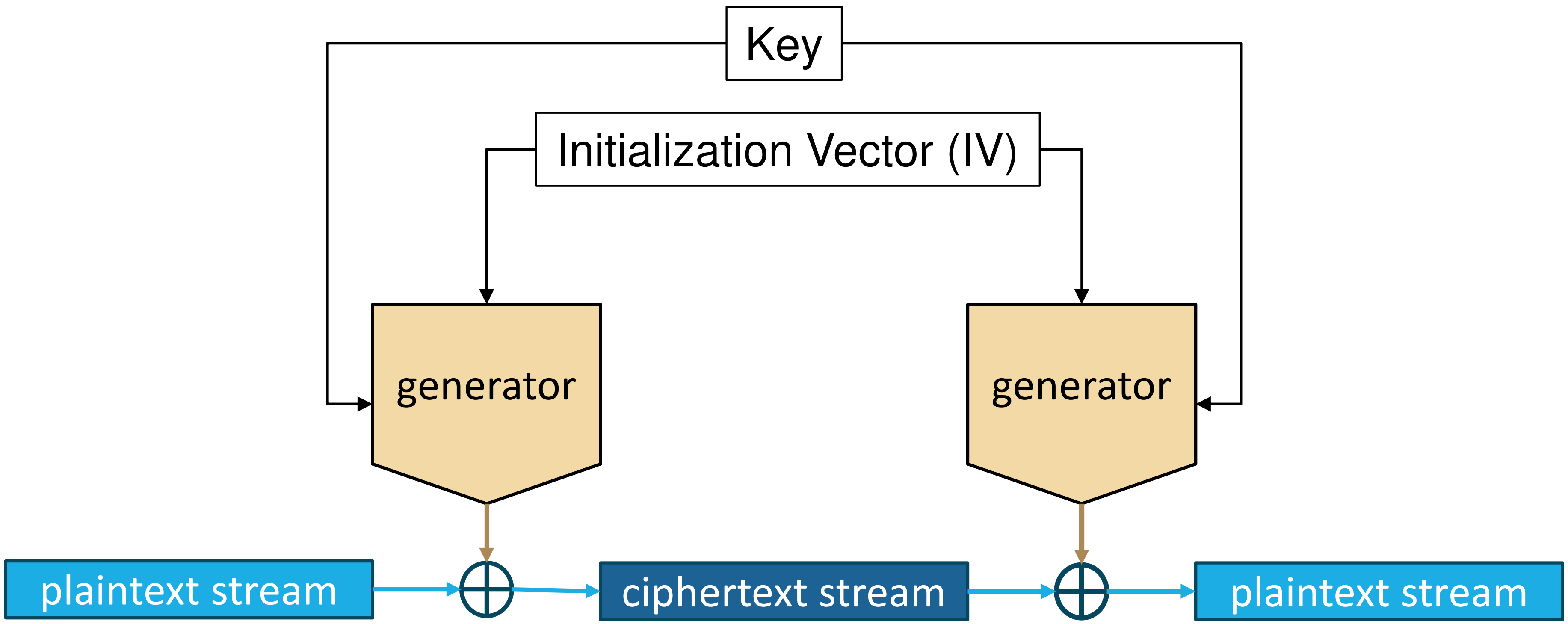
Use a cryptographically secure, pseudo-random bit generator

- This generator produces the keystream
- The generator implements a state machine
- The generator is controlled by two values:
  - **Initialization Vector** (defines the initial state of the state machine)
  - **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:
  - Past keystream values
  - Future keystream values

(Symmetric) stream ciphers: Approach



(Symmetric) stream ciphers: Exploitation considerations

No two messages should be encrypted with the same key and IV

- Because they will be encrypted with the same keystream
- The knowledge about um message reveals the other

$$C1 = P1 \oplus KS$$
$$C2 = P2 \oplus KS \longrightarrow P2 = C2 \oplus KS = C2 \oplus C1 \oplus P1$$

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

Keystreams may be periodic (have a cycle)

- Depends on the generator
- Same problem as the one above
- Plaintext should be shorter than the period length

(Symmetric) stream ciphers: Exploitation considerations

Ciphertexts can be deterministically manipulated

- Each cipher bit depends only on one plaintext bit

  $C' = C \oplus \Delta \quad \rightarrow \quad P' = P \oplus \Delta$

- It is fundamental to have integrity control elements
  - In the ciphertext
  - In the plaintext

Symmetric Block ciphers: Examples

DES (Data Encryption Standard)

- Proposed in 1974, standard in 1977
- Input/output: 64-bit blocks
- Key: 56 bits

AES (Advanced Encryption Standard)

- Proposed in 1998 (Rijndael), standard in 2001
- Input/output: 128-bit blocks
- Key: 128, 192 or 256 bits

Other

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

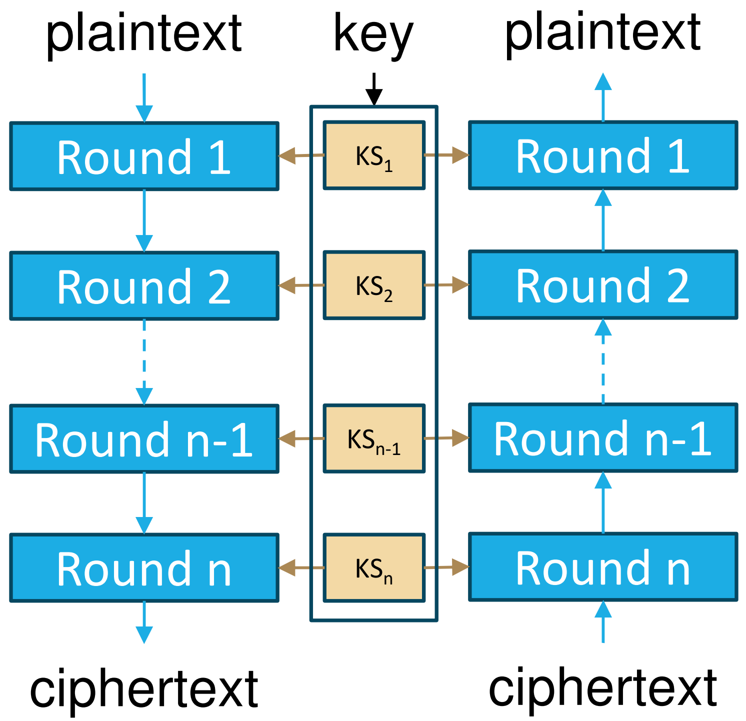
Symmetric block ciphers: Approach

Use a pipeline of transformation rounds

- Each round adds confusion and diffusion
- Each round is usually controlled by a subkey
  - Key schedule
  - A key derived from the key used provided for encryption/decryption

Rounds need to be reversible

- To allow decrypting what was encrypted
- Feistel networks
- Substitution-permutation networks



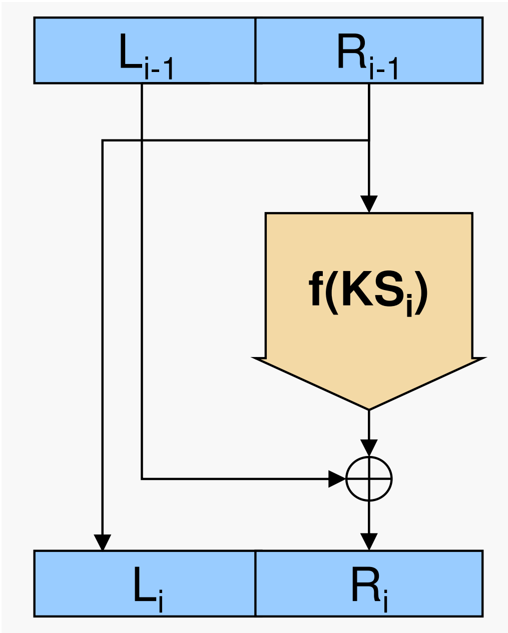
Feistel networks

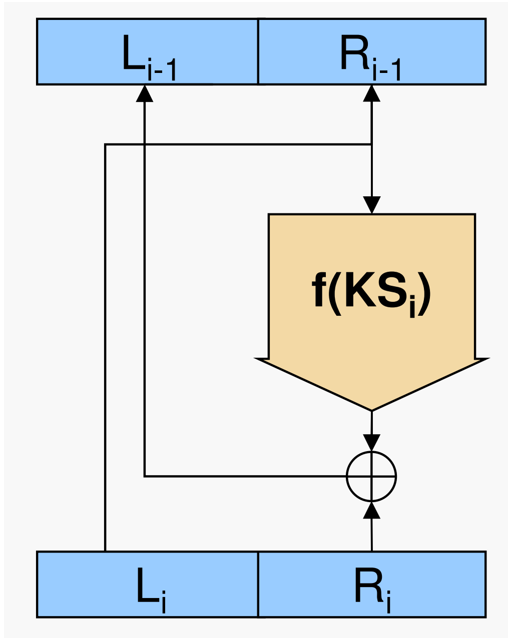
$L_i = R_{i-1}$

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

$R_{i-1} = L_i$

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





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

Substitution-Permutation Network

SBox

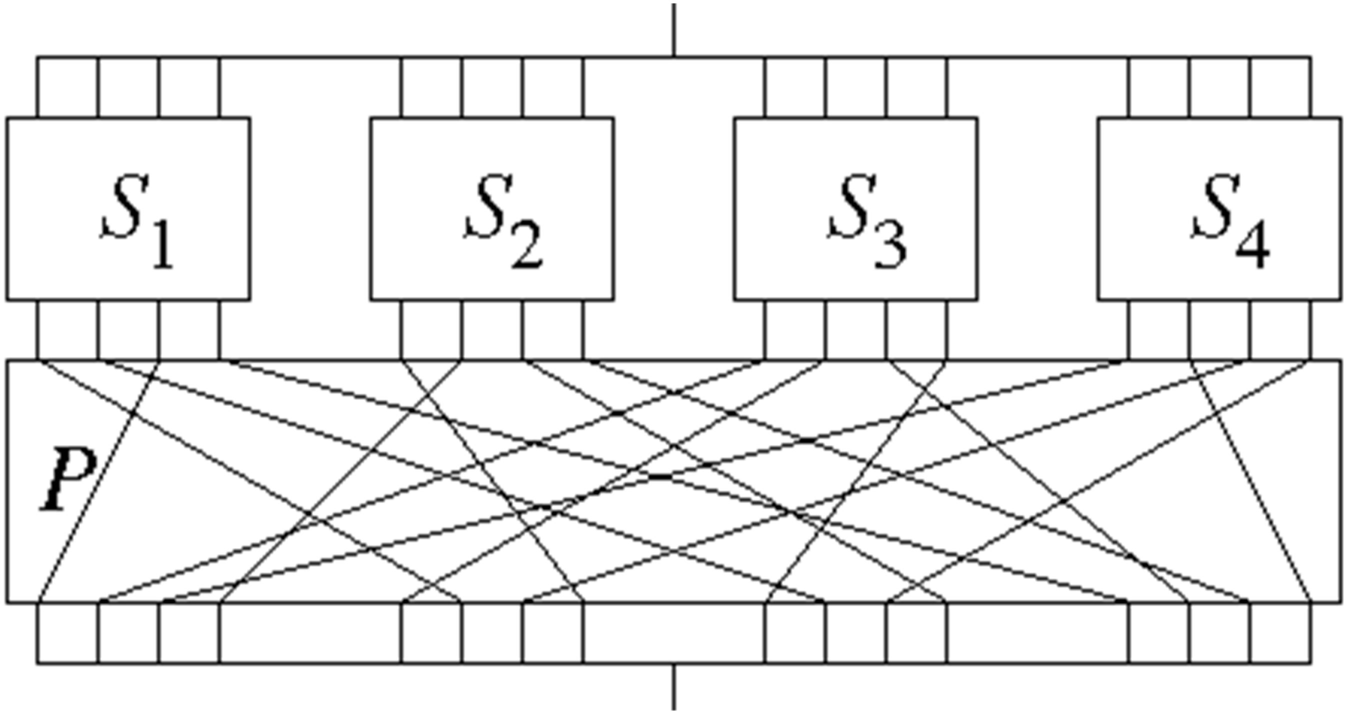
- Table with an output for an input (index)

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

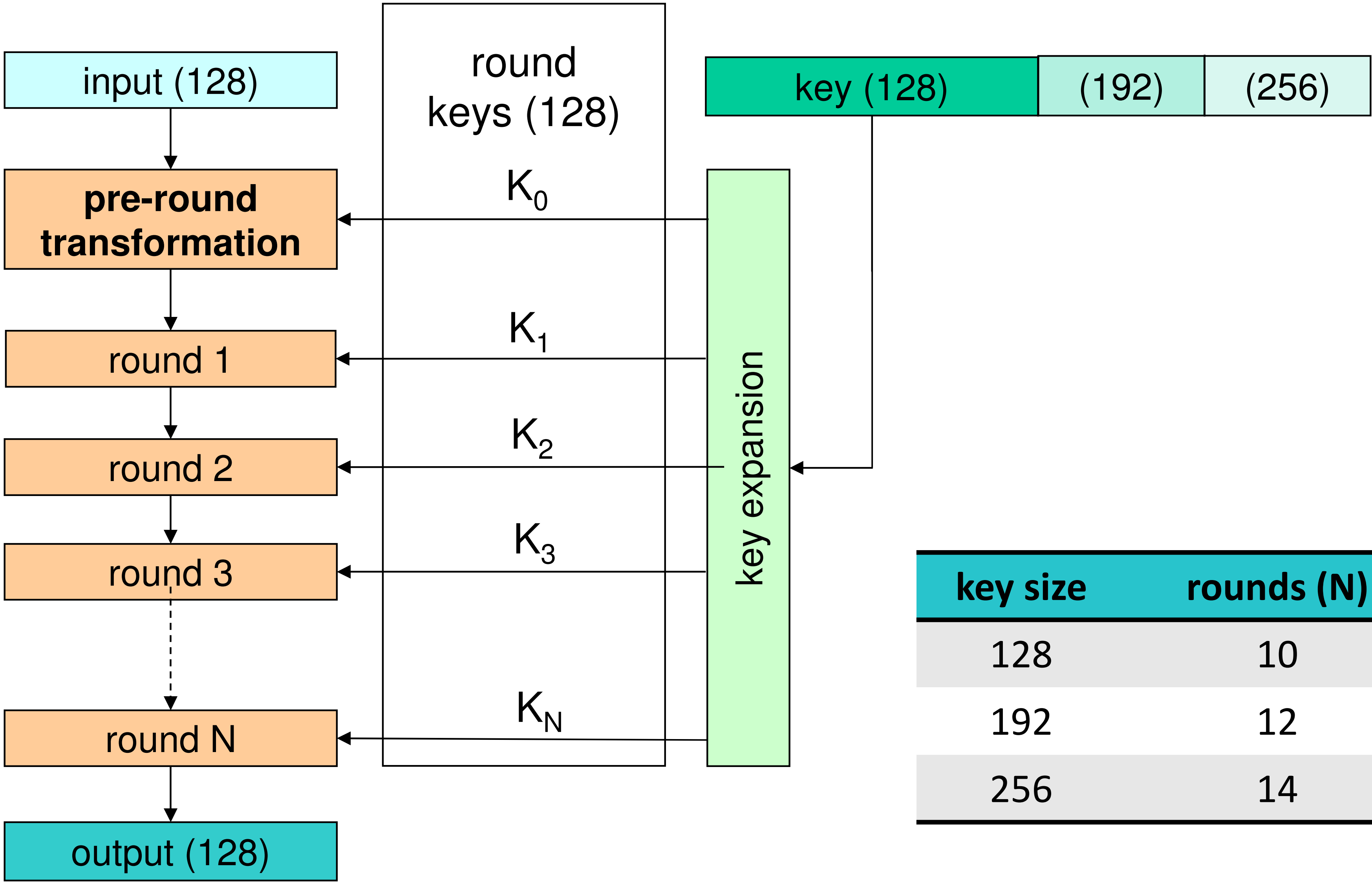
PBox

- Changes the positions of the input bits

Substitution-Permutation Network

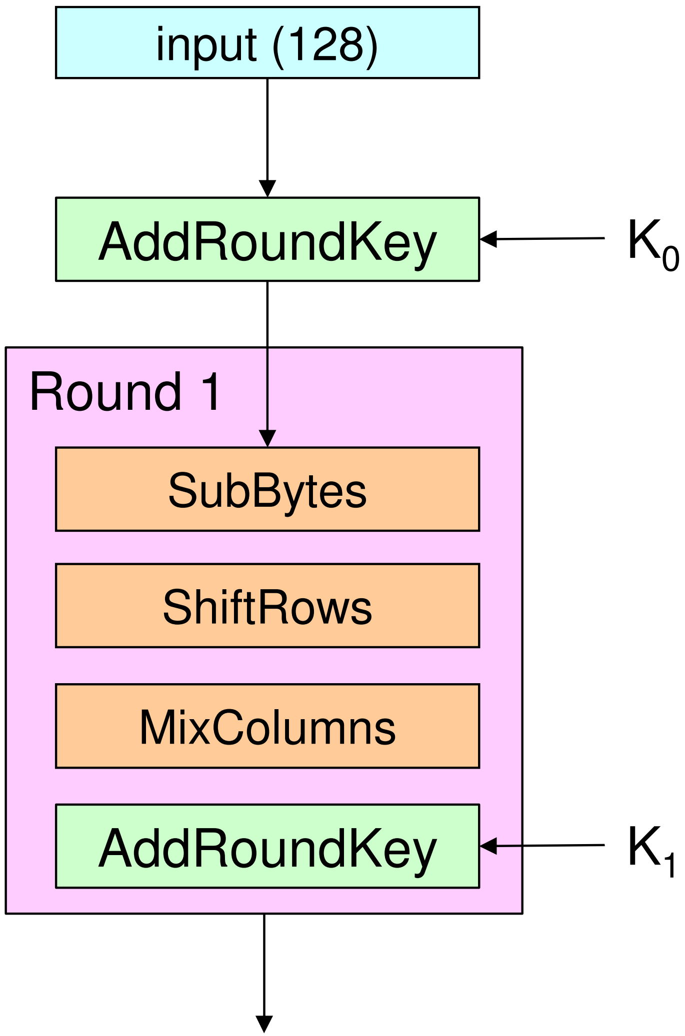


AES architecture



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

AES (encryption) round

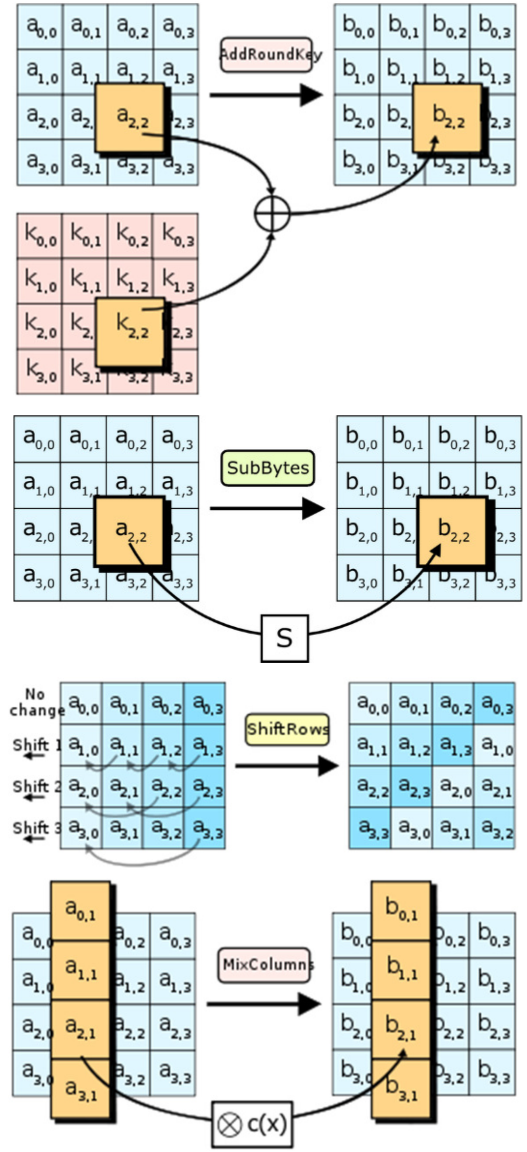


AddRoundKey
- 128-bit XOR
- Output is a 4x4 byte matrix

SubBytes
- 256-element S-box
- Each matrix bytes is substituted

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

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)

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)

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

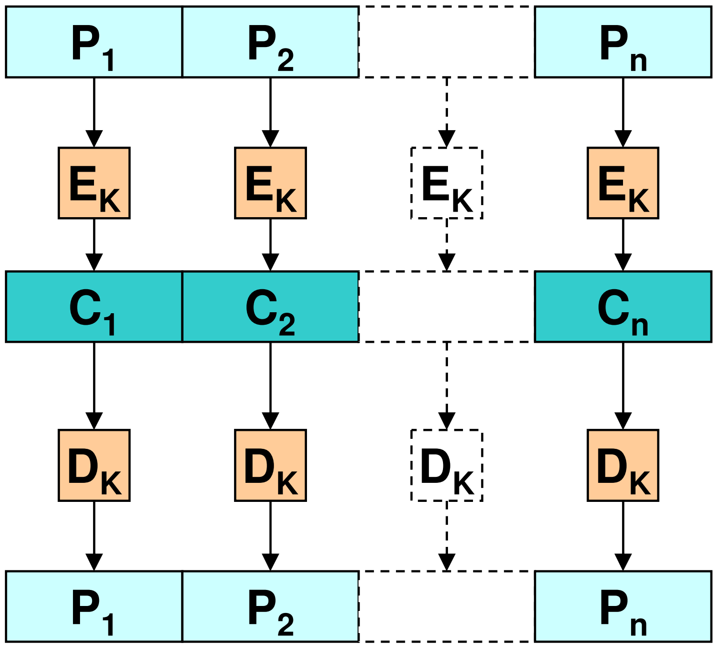
Direct decryption of each block: $P_i = D_K(C_i)$

Blocks are processed independently

- Parallelism is possible
- Uniform random access exists

Problem:

- Pattern exposure
- If $P_1 = P_2$ then $C_1 = C_2$



Cipher Modes: Cipher Block Chaining (CBC)

Encrypt each block T_i with feedback from C_{i-1}

- $C_i = E_K(P_i \oplus C_{i-1})$

Decrypt each block C_i with feedback from C_{i-1}

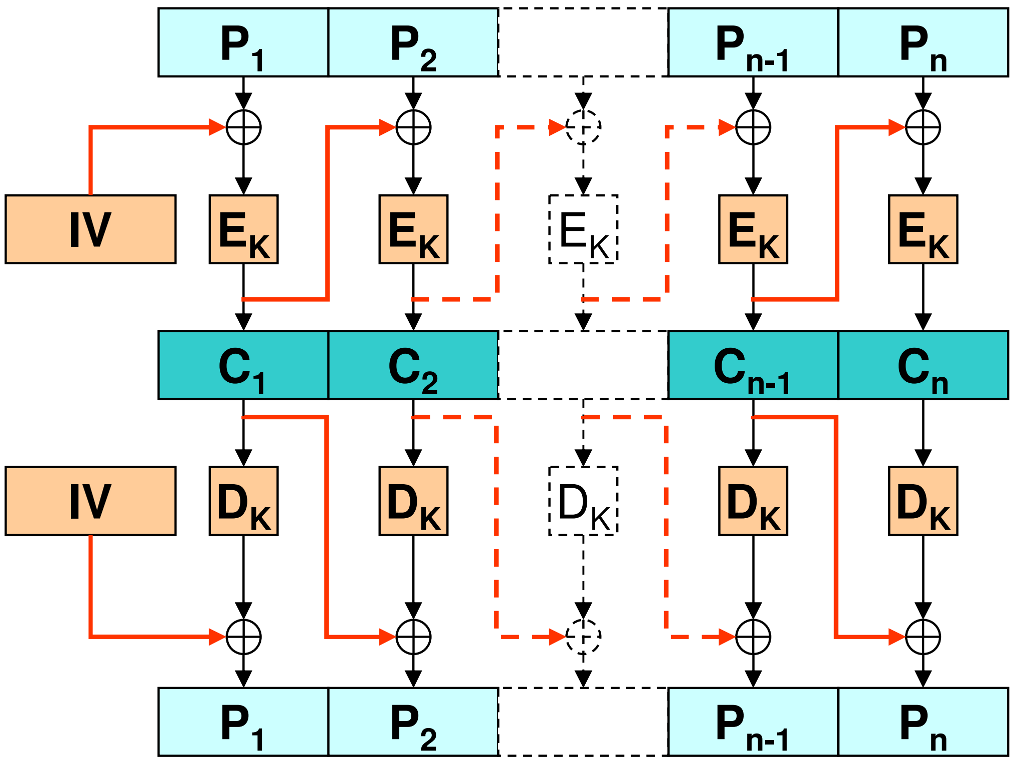
- $P_i = D_K(C_i) \oplus C_{i-1}$
- Parallelism and uniform random access is possible

First block uses an IV

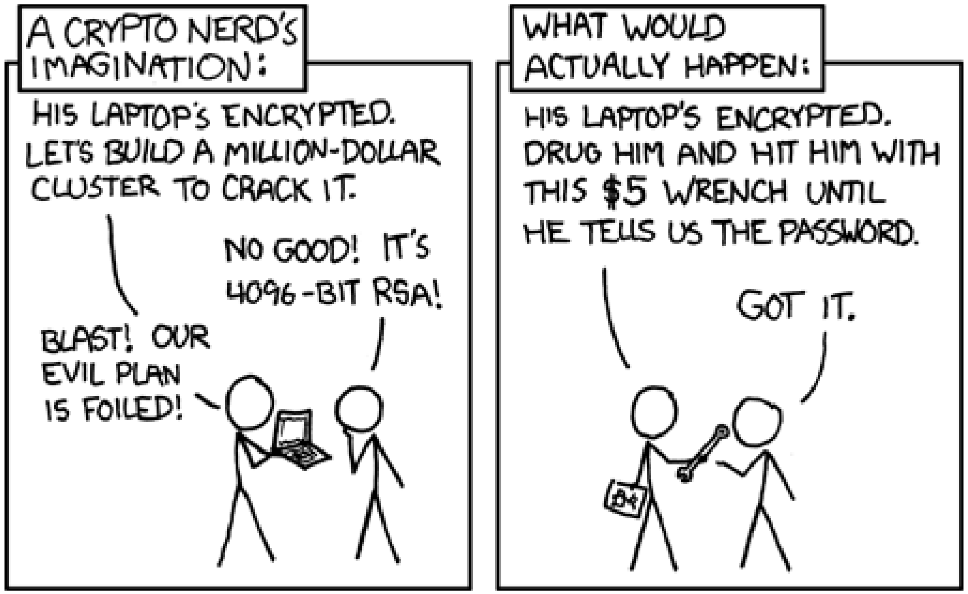
- IV: Initialization Vector
- Better not reuse for the same key
  - Random value, sequence value
- May be sent in clear

Polyalphabetic transformation

- The feedback prevents equal blocks from being equally processed
- Seems like we have a different key per block

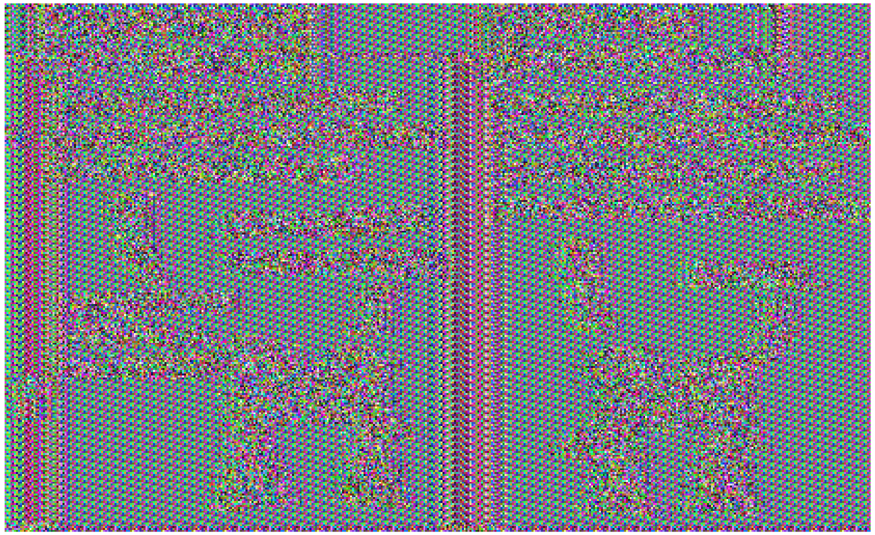


ECB vs CBC: pattern exposure



https://xkcd.com/538/

ECB



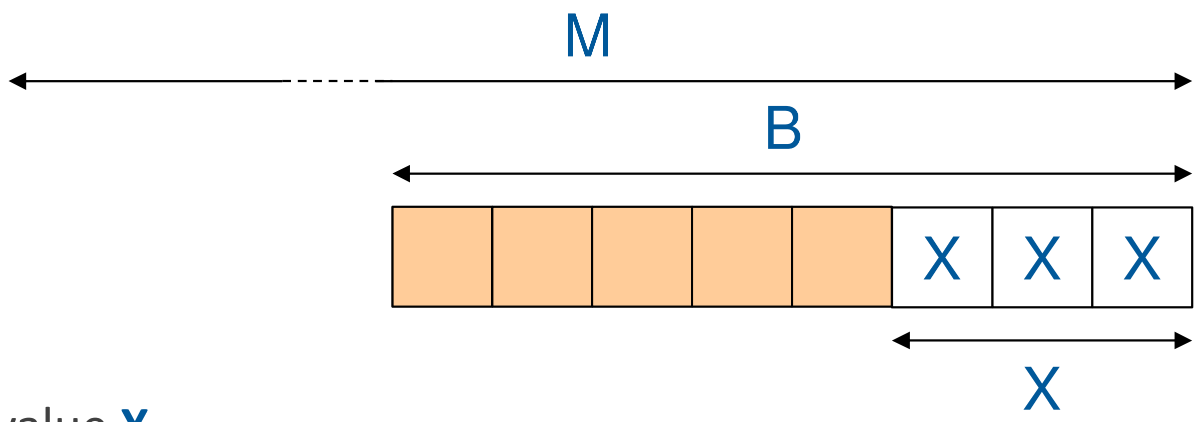
CBC

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
  - **PKCS #7**
    - **X = B – (M mod B)**
    - **X** extra bytes, with the value **X**
    - PKCS #5: Equal to PKCS #7 with B = 8
  - Perfectly aligned inputs get an extra padding block!
- Different processing for the sub-block
  - Adds complexity, rarely used

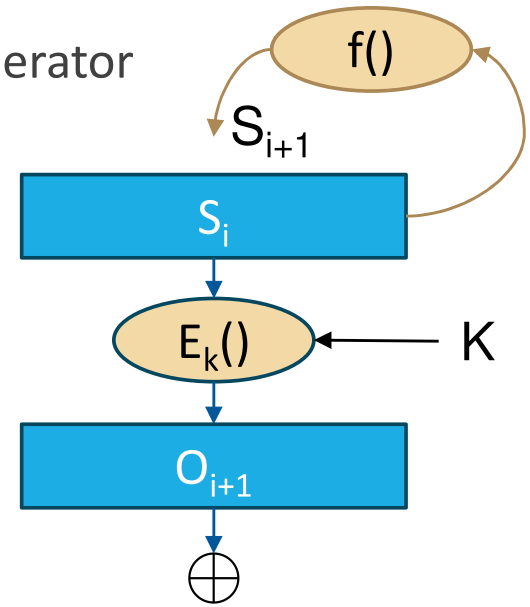
Stream cipher modes

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

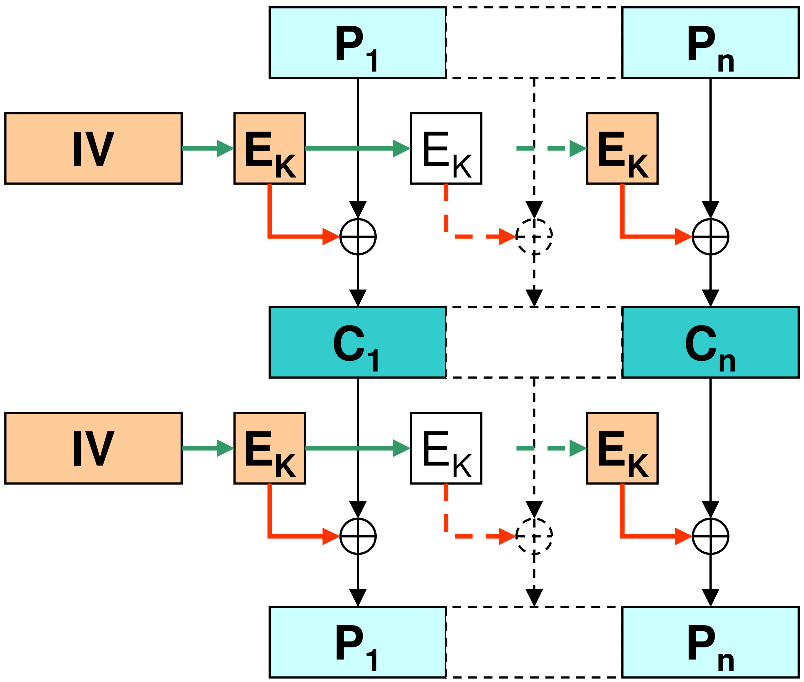
Stream cipher modes

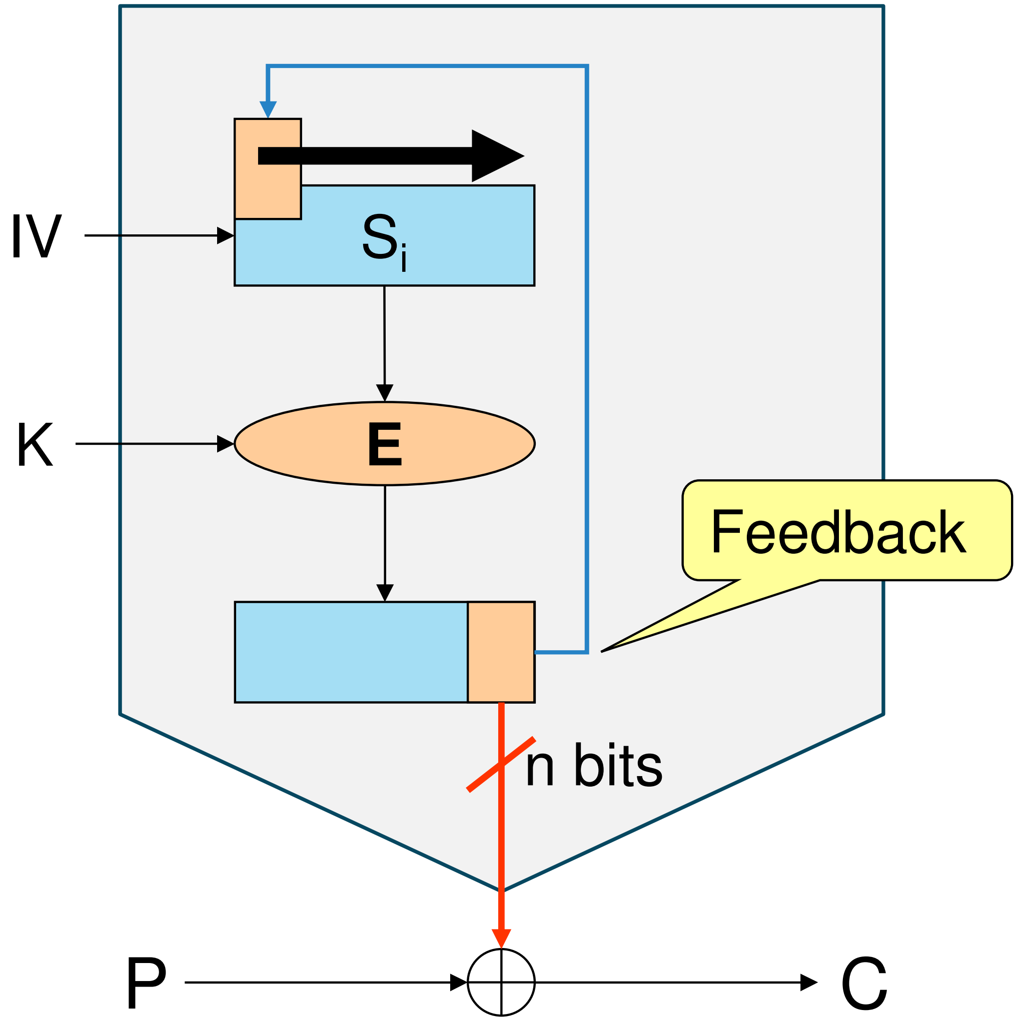
- They use a block cipher to implement a stream cipher generator
- The fundamental idea is:
  - The generator e a state machine with state S_i on iteration i
  - The output of the generator for state S_i is $O_{i+1} = E_K(S_i)$
  - The state S_i is updated to S_{i+1} using some transformation function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



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

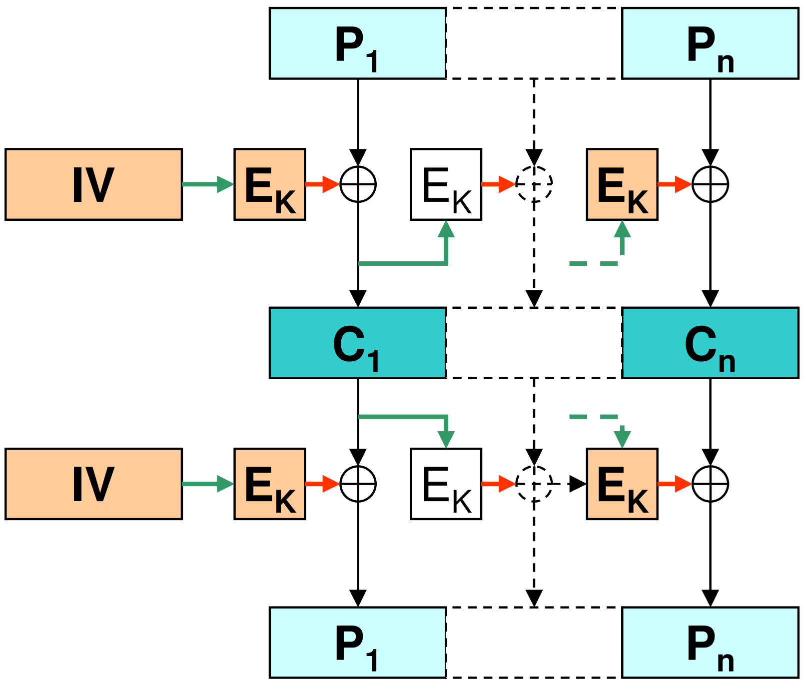
$C_i = T_i \oplus E_K(S_{i-1})$
$T_i = C_i \oplus E_K(S_{i-1})$
$S_{i+1} = f(S_i, E_K(S_i))$
$S_0 = IV$

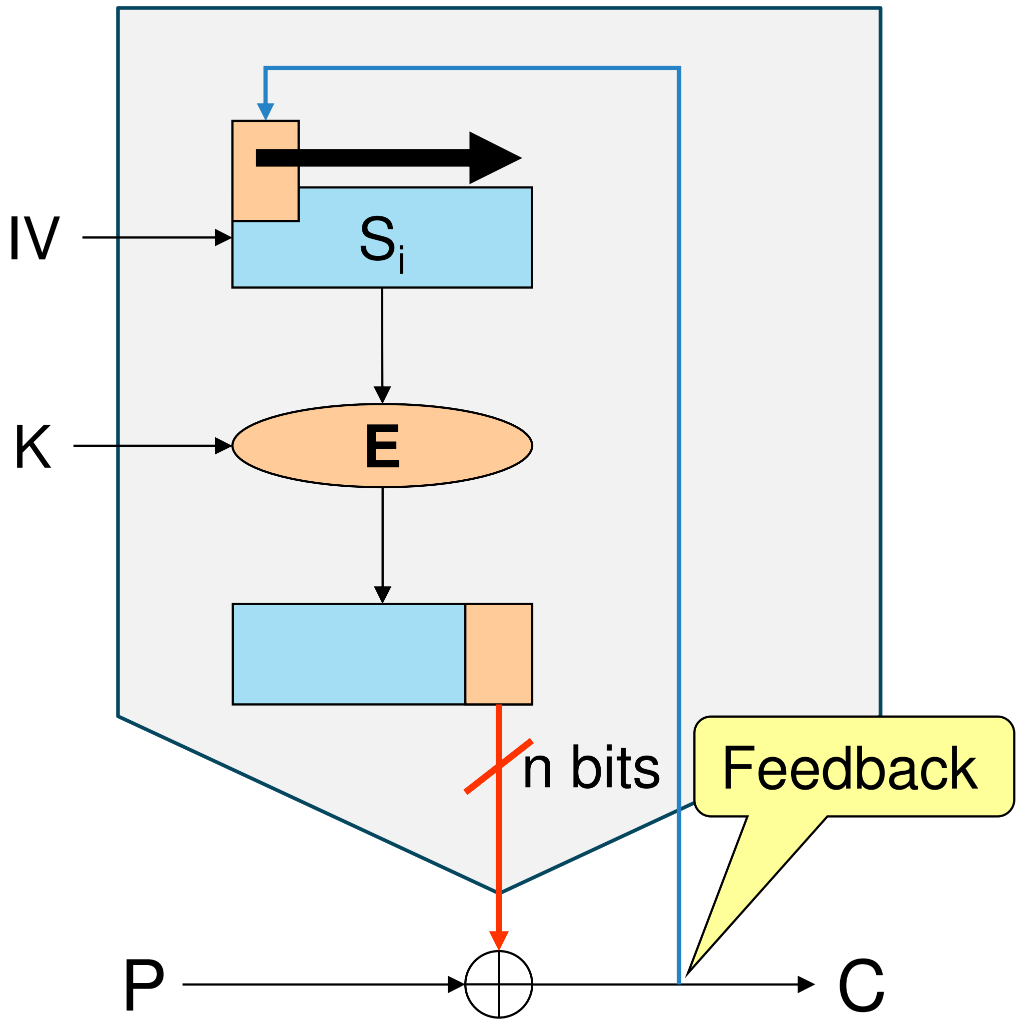




Stream cipher modes: n-bit CFB (Ciphertext Feedback)

$C_i = T_i \oplus E_K(S_{i-1})$
$T_i = C_i \oplus E_K(S_{i-1})$
$S_{i+1} = f(S_i, C_i)$
$S_0 = IV$





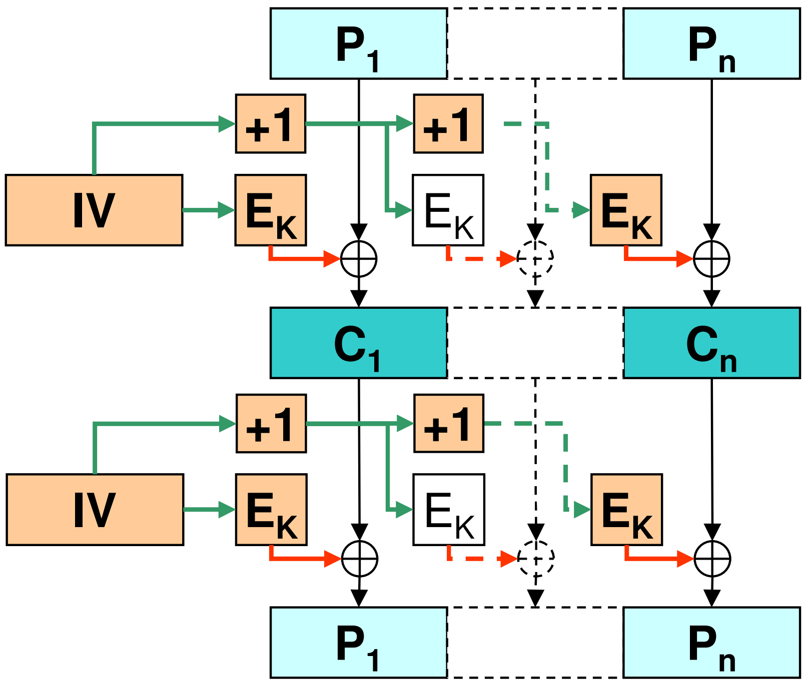
Cipher modes: n-bit CTR (Counter)

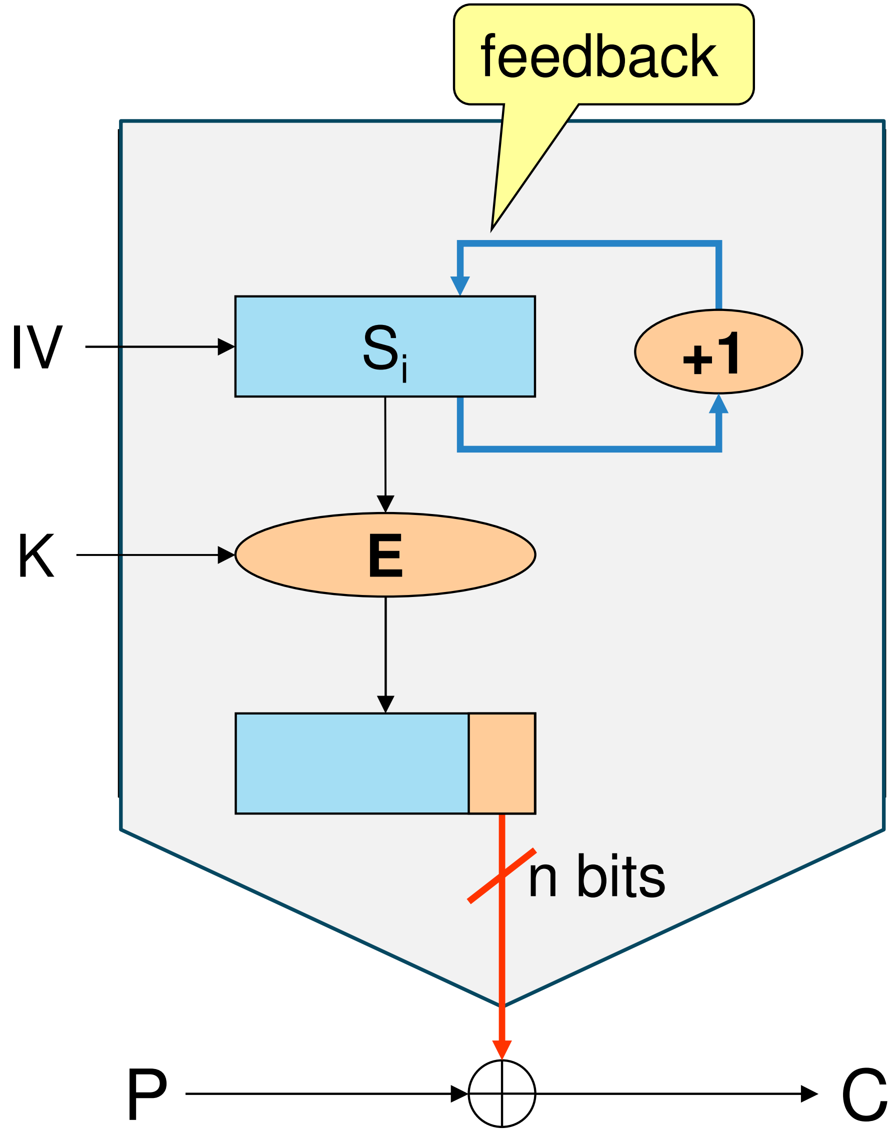
$C_i = T_i \oplus E_K(S_i)$

$T_i = C_i \oplus E_K(S_i)$

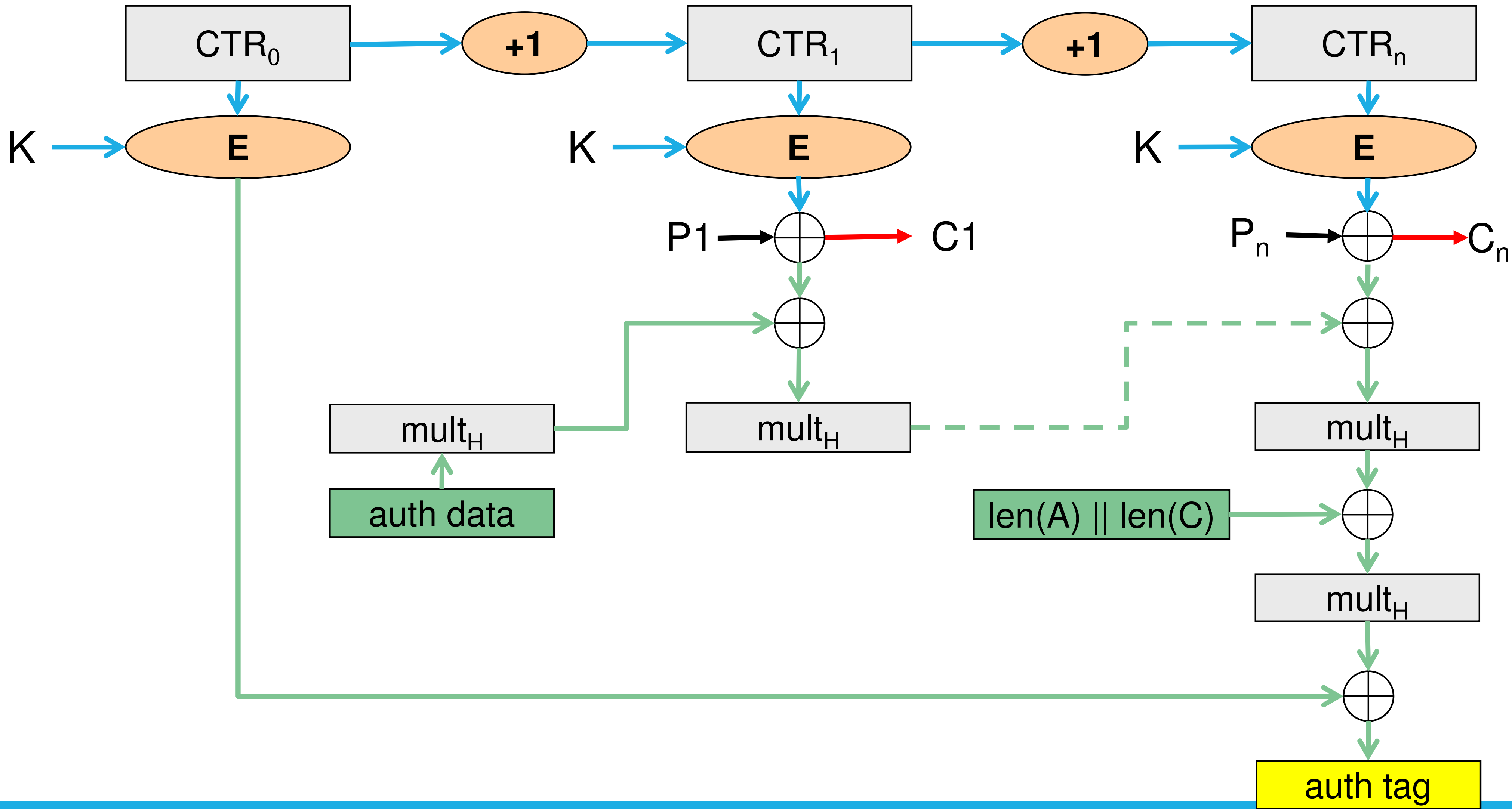
$S_i = S_{i-1}+1$

$S_0 = IV$





Stream cipher modes:
Galois with Counter Mode (GCM)



Cipher Modes: Comparison

<table>
<tr><th></th><th colspan="2">Block</th><th colspan="4">Stream</th></tr>
<tr><th></th><th>ECB</th><th>CBC</th><th>OFB</th><th>CFB</th><th>CTR</th><th>GCM</th></tr>
<tr><td>Input pattern hiding</td><td></td><td>✓</td><td>✓</td><td>✓</td><td>✓</td><td>✓</td></tr>
<tr><td>Same key for different messages</td><td>✓</td><td>✓</td><td>other IV</td><td>other IV</td><td>other IV</td><td>other IV</td></tr>
<tr><td>Tampering difficulty</td><td>✓</td><td>✓ (...)</td><td></td><td>(...)</td><td></td><td>✓</td></tr>
<tr><td>Pre-processing</td><td></td><td></td><td>✓</td><td></td><td>✓</td><td>✓</td></tr>
<tr><td>Parallel processing</td><td rowspan="2">✓</td><td rowspan="2">decrypt</td><td rowspan="2">With pre-proc</td><td rowspan="2">decrypt</td><td rowspan="2">✓</td><td rowspan="2">✓</td></tr>
<tr><td>Uniform random access</td></tr>
<tr><td>Cryptogram single bit error propagation on decryption</td><td>same block</td><td>same & next block</td><td></td><td>a few next bits</td><td></td><td>detected</td></tr>
<tr><td>Capacity to recover from losses</td><td>some</td><td>some</td><td></td><td>some</td><td></td><td>detected</td></tr>
</table>

Cipher modes: multiple encryption

Invented for extending the lifetime of DES

- DES was never cryptanalysed
- But its key was too short (56 bits only)
- Its key could be discovered by brute force

Triple encryption EDE, or 3DES-EDE

- $C_i = E_{K3}(D_{K2}(E_{K1}(P_i)))$
- $P_i = D_{K1}(E_{K2}(D_{K3}(C_i)))$
- 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
- $C_i = E_K(K_1 \oplus P_i) \oplus K_2$
- $P_i = K_1 \oplus D_K(K_2 \oplus C_i)$

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