F2b: symmetric key cryptography - block cipher

tkkwon@snu.ac.kr

BLOCK CIPHER OVERVIEW

revisit confusion and diffusion

- confusion refers to making the relationship between the ciphertext and the key as complex and involved as possible;
- diffusion means that if we change a character of the plaintext, then several characters of the ciphertext should change, and vice versa
  - Avalanche effect: A small change in plaintext results in the very great change in the ciphertext
    - applies to hash functions as well
  - Completeness: Each bit of ciphertext depends on many bits of plaintext.
- This complexity is generally implemented through a series of substitutions and permutations
  - The simplest way to achieve both diffusion and confusion is to use a substitution-permutation network

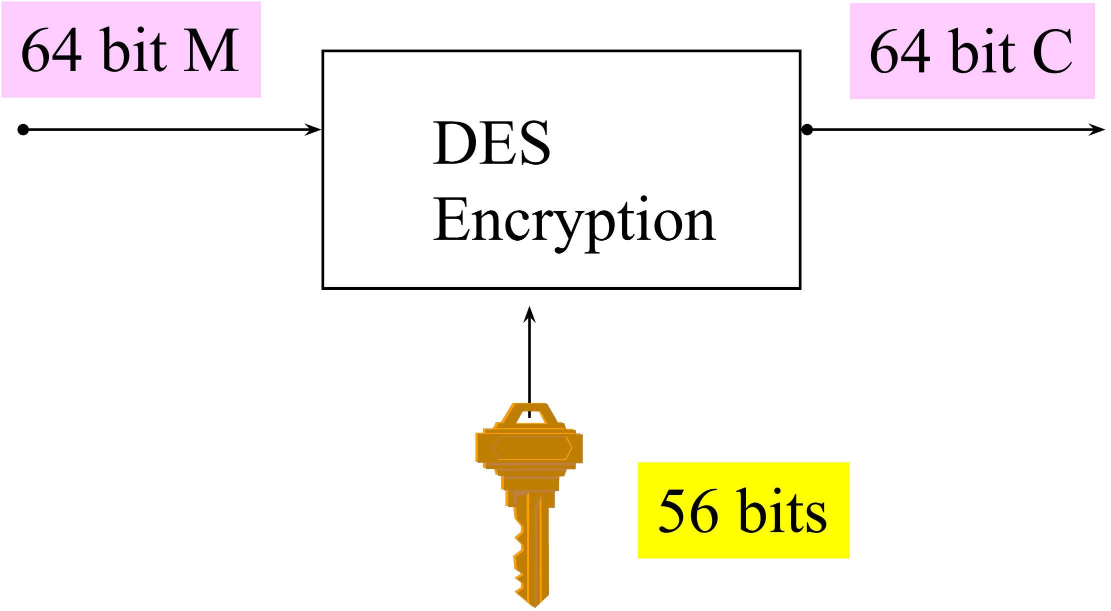
S-Box and P-Box

- S-box (substitution)
  - some number of input bits, m, are replaced by some number of output bits, n, (m and n are equal or different)
  - Usually for confusion
  - diffusion as well
  - m X n lookup box (2^m entries of size n bits)
- P-box (permutation)
  - a method of shuffling used to permute or transpose bits/characters
  - diffusion
  - Expansion, compression, and straight

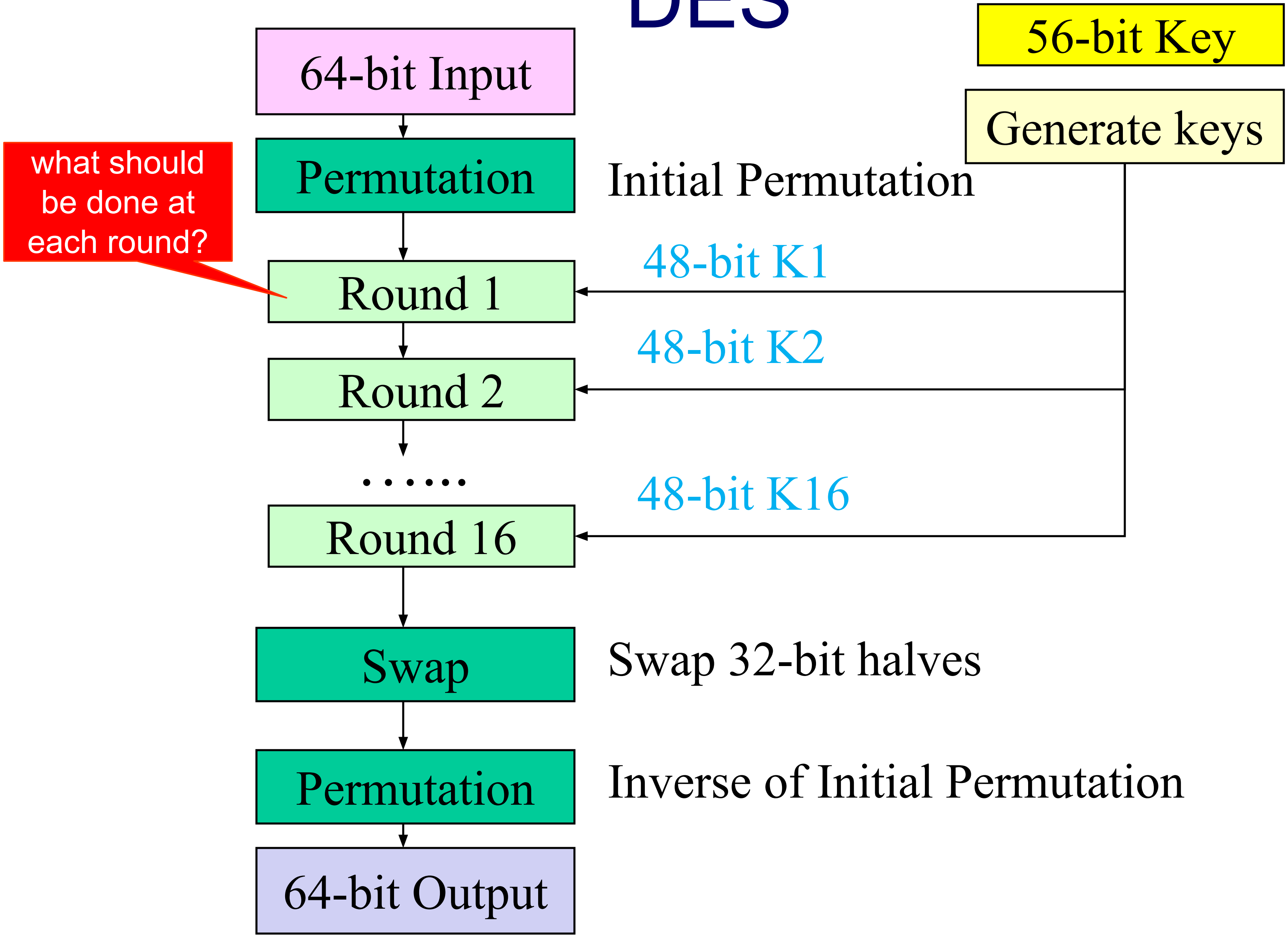
DES

DES

- Published in 1977, standardized in 1979.
- Key: 64 bit quantity=8-bit parity+56-bit key
  - Every 8th bit is a parity bit (odd parity)
- 64 bit input, 64 bit output.



DES



56-bit Key

Generate keys

64-bit Input

Permutation

Initial Permutation

what should be done at each round?

48-bit K1

Round 1

48-bit K2

Round 2

……

48-bit K16

Round 16

Swap

Swap 32-bit halves

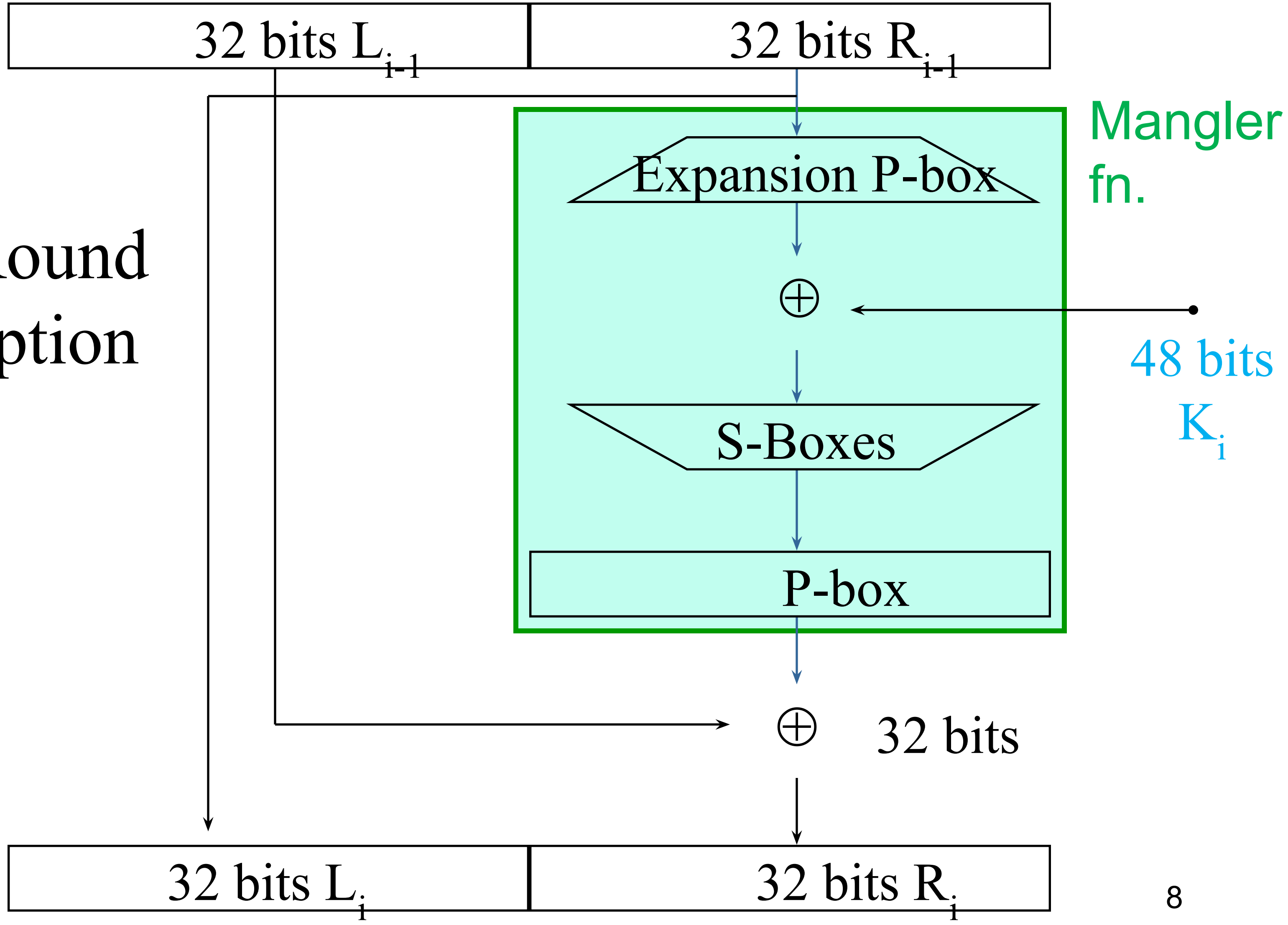
Permutation

Inverse of Initial Permutation

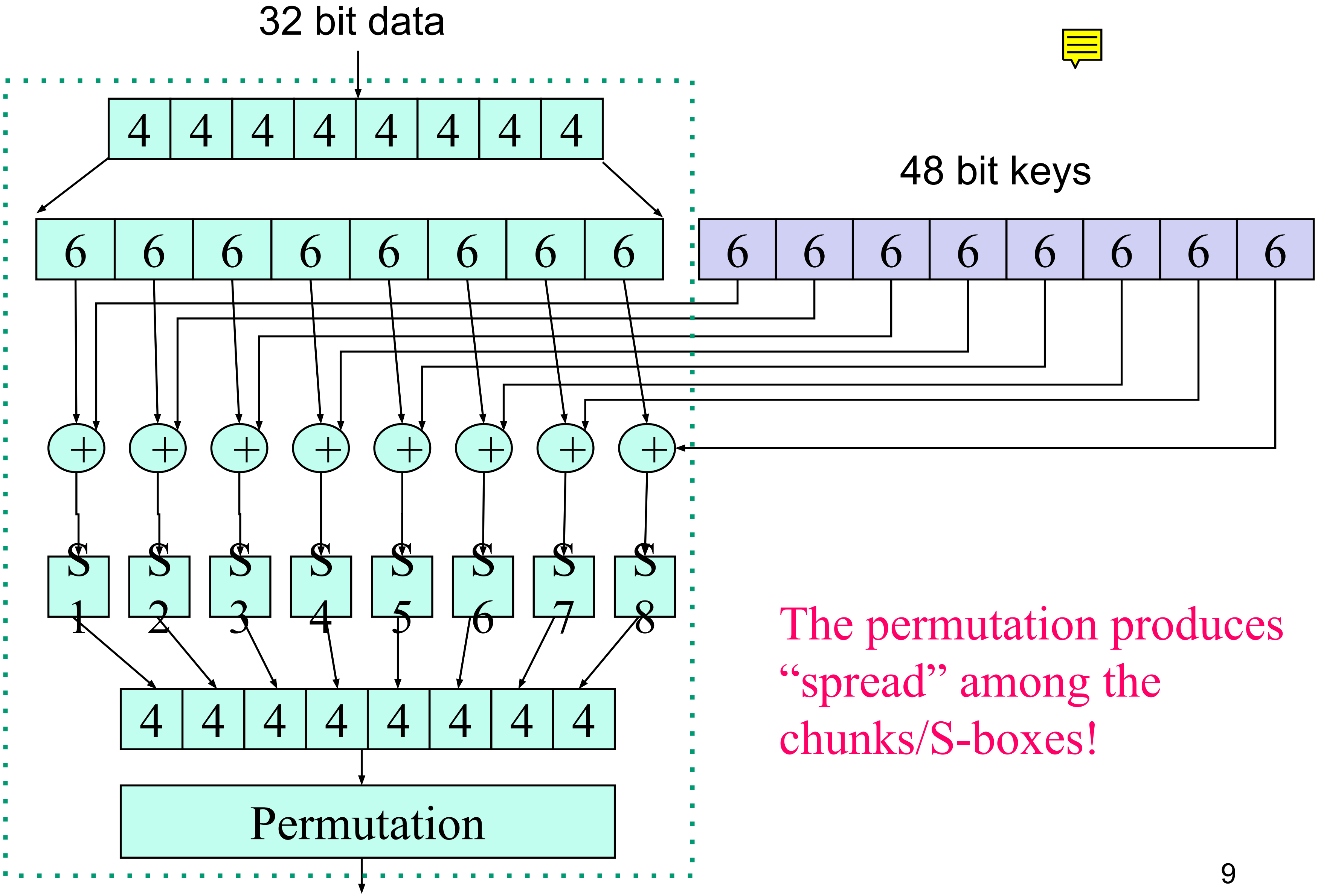
64-bit Output

A round in DES

One Round Encryption



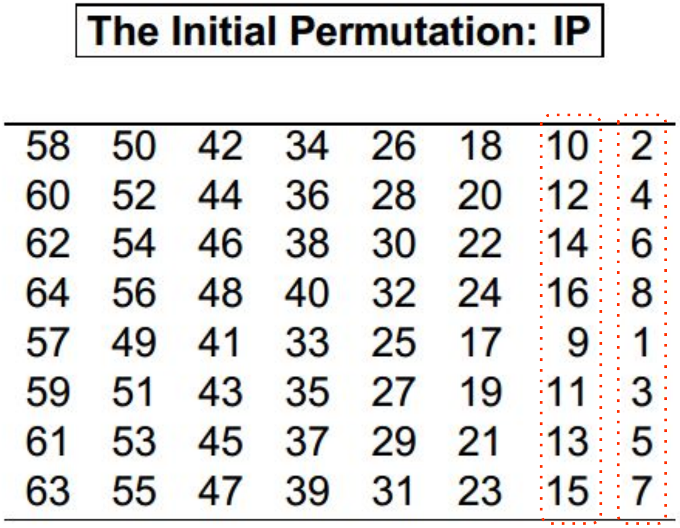
Mangler fn.



The permutation produces “spread” among the chunks/S-boxes!

Questions in DES

- Why initial permutation and inverse of initial permutation?
- Why swap 32bits?
- Is it invertible?



The Initial Permutation: IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

DES decryption

- A Mangler function is not invertible
  - Why?
  - Mangler function with key input is simplified to f below
- Then how to decrypt?
- Each round *i* is expressed by

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \oplus \text{Mangler}(K_i, R_{i-1}) = L_{i-1} \oplus f(R_{i-1})$
- each round function F would be

  $F(L_{i-1} \| R_{i-1}) = R_{i-1} \| L_{i-1} \oplus f(R_{i-1})$
- Let G be a function that maps input $(L_i \| R_i)$ to $R_i \oplus f(L_i) \| L_i$

  $G(L_{i-1} \| R_{i-1}) = R_{i-1} \oplus f(L_{i-1}) \| L_{i-1}$
- Then how about $G(F(L_i \| R_i))$?
- Feistel Cipher: Mangler function f need not be invertible

how are F and G related?

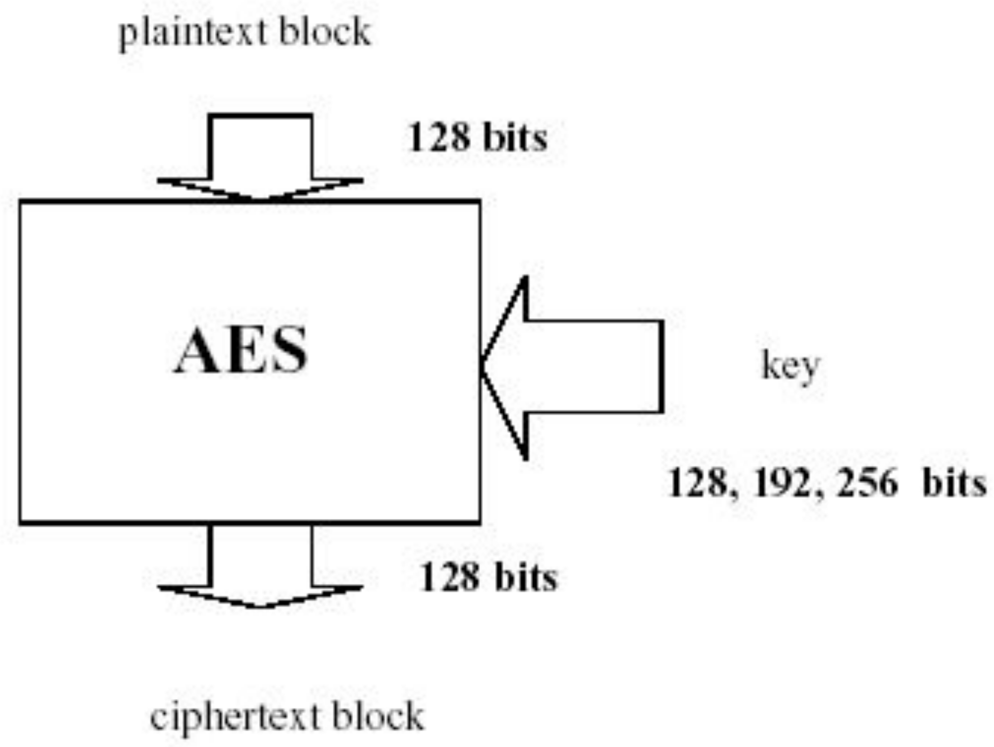
AES

advanced encryption standard (AES)

- DES is broken in 1990s
  - key length is short
- NIST standardized AES in 2001
- based on Rijndael cipher
- 3 different key lengths
- data block during the encryption process is called a state
- has 10 rounds in which the state goes through the following transformations (called `layers'):
  - SubBytes: byte substitution (1 S-box used on every byte)
  - ShiftRows: shift rows (permute bytes between groups/columns)
  - MixColumns: mix columns (uses matrix multiplication in GF(256))
  - AddRoundKey: add round key (XOR state with round key)
- First and last rounds are a little bit different

advanced encryption standard (AES)

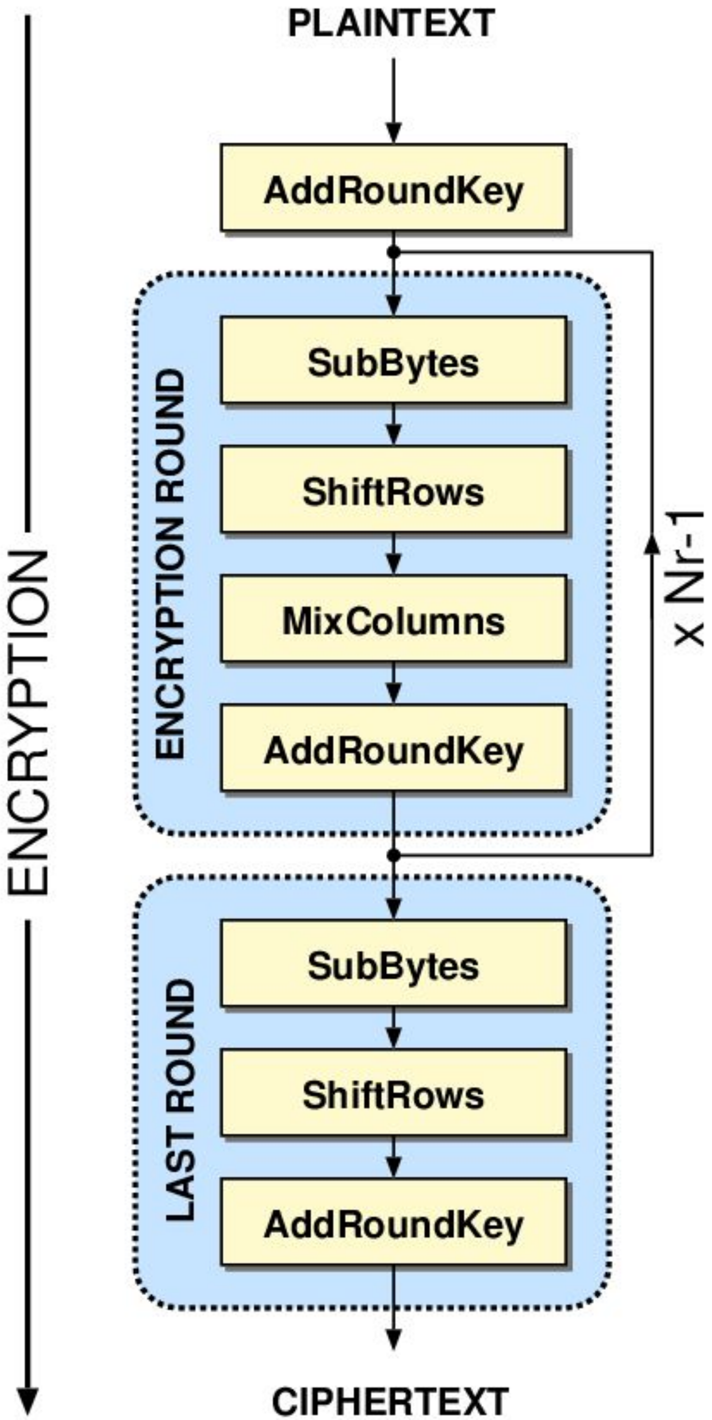
- data block viewed as 4-by-4 table of bytes (i.e. 128 bits)
- Such a table is called the **current state**
- 3 key lengths (128/192/256), 128 bit key is assumed here



state

$$\begin{bmatrix} byte_0 & byte_4 & byte_8 & byte_{12} \\ byte_1 & byte_5 & byte_9 & byte_{13} \\ byte_2 & byte_6 & byte_{10} & byte_{14} \\ byte_3 & byte_7 & byte_{11} & byte_{15} \end{bmatrix}$$

AES: a high level view

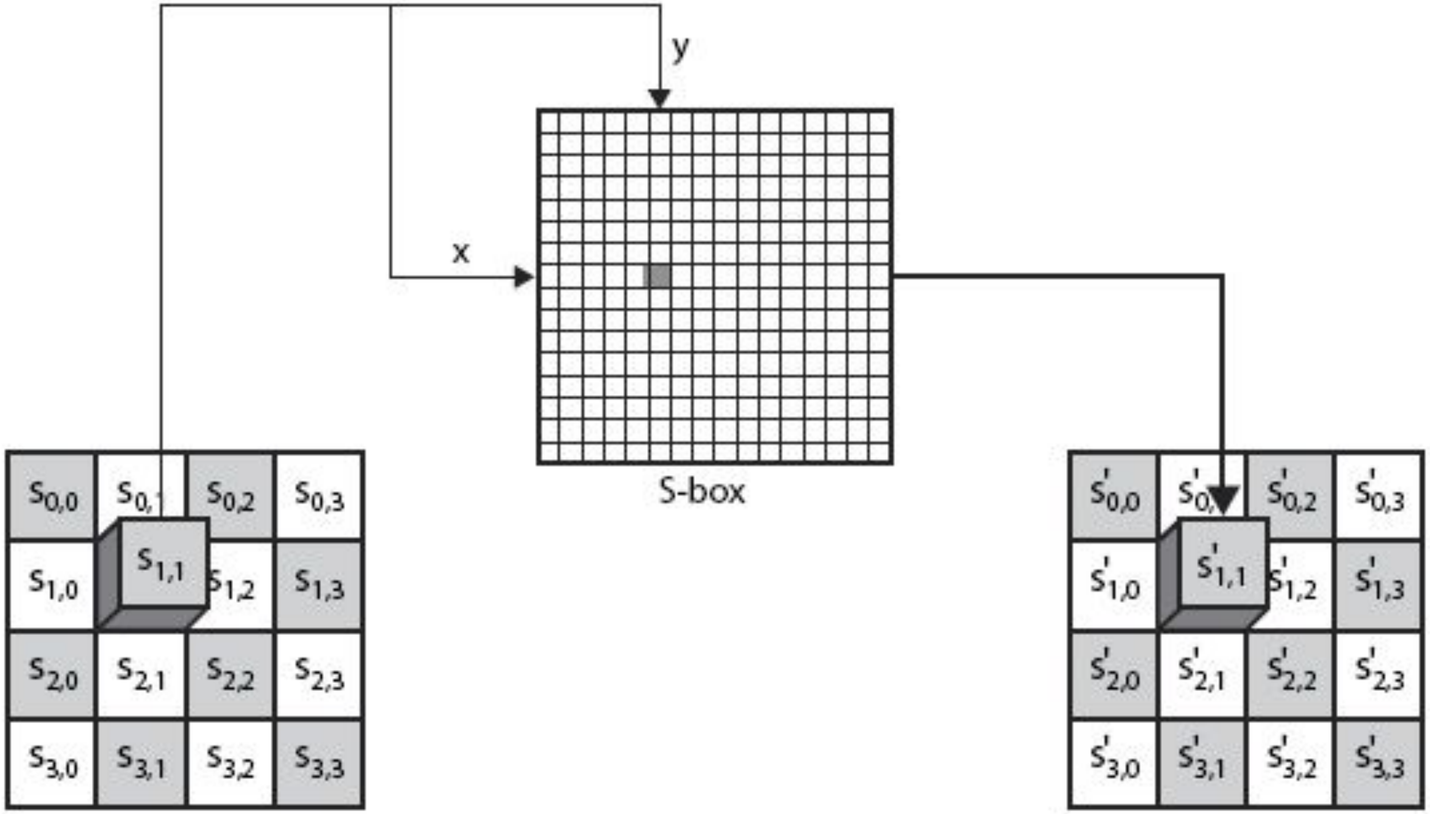


N_r: # of rounds

10/12/14 rounds for 128/192/256 bit keys

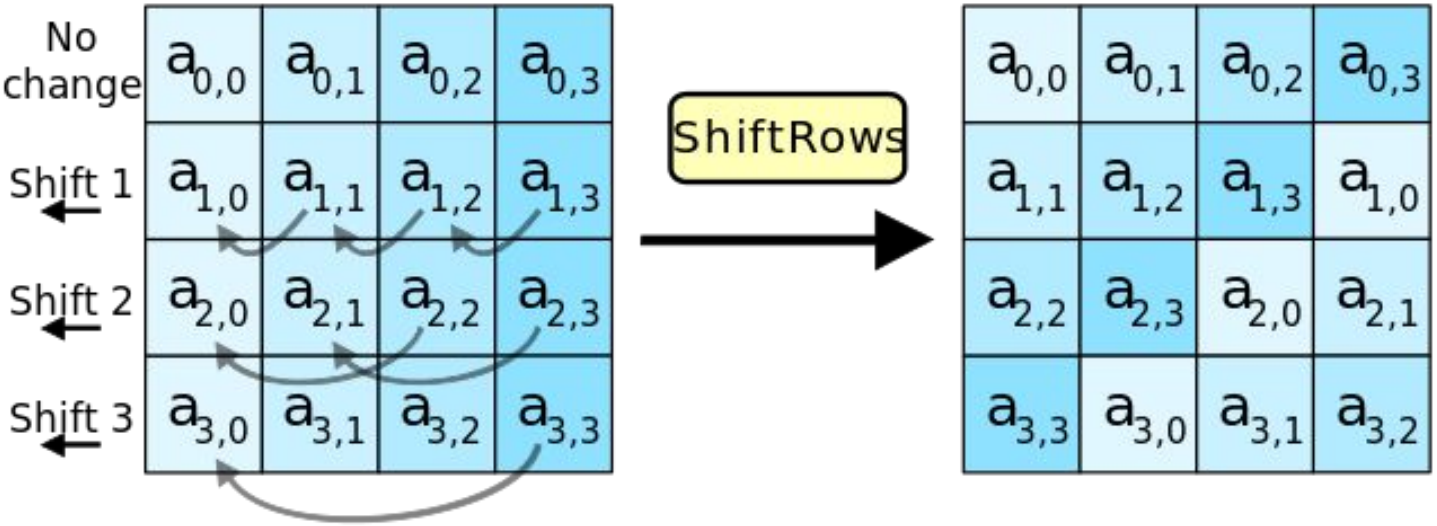
AES: SubBytes (S-box)

- a simple substitution of each byte
- a byte = two nibbles
- S-box has 16x16 entries: all possible 8-bit values
- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)



AES: ShiftRows

- a circular byte shift in each row (permutation)
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left

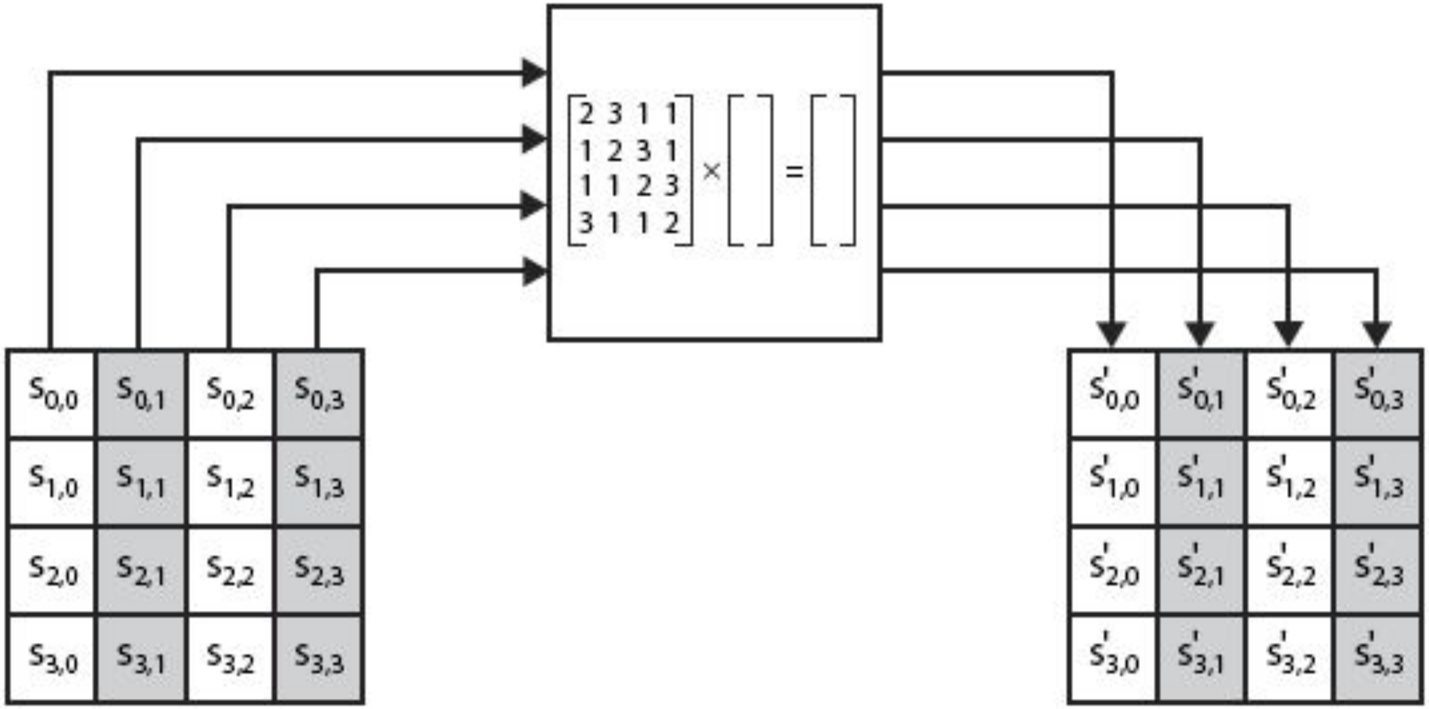


No change	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
Shift 1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
Shift 2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
Shift 3	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

AES: Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication (Hill Cipher)



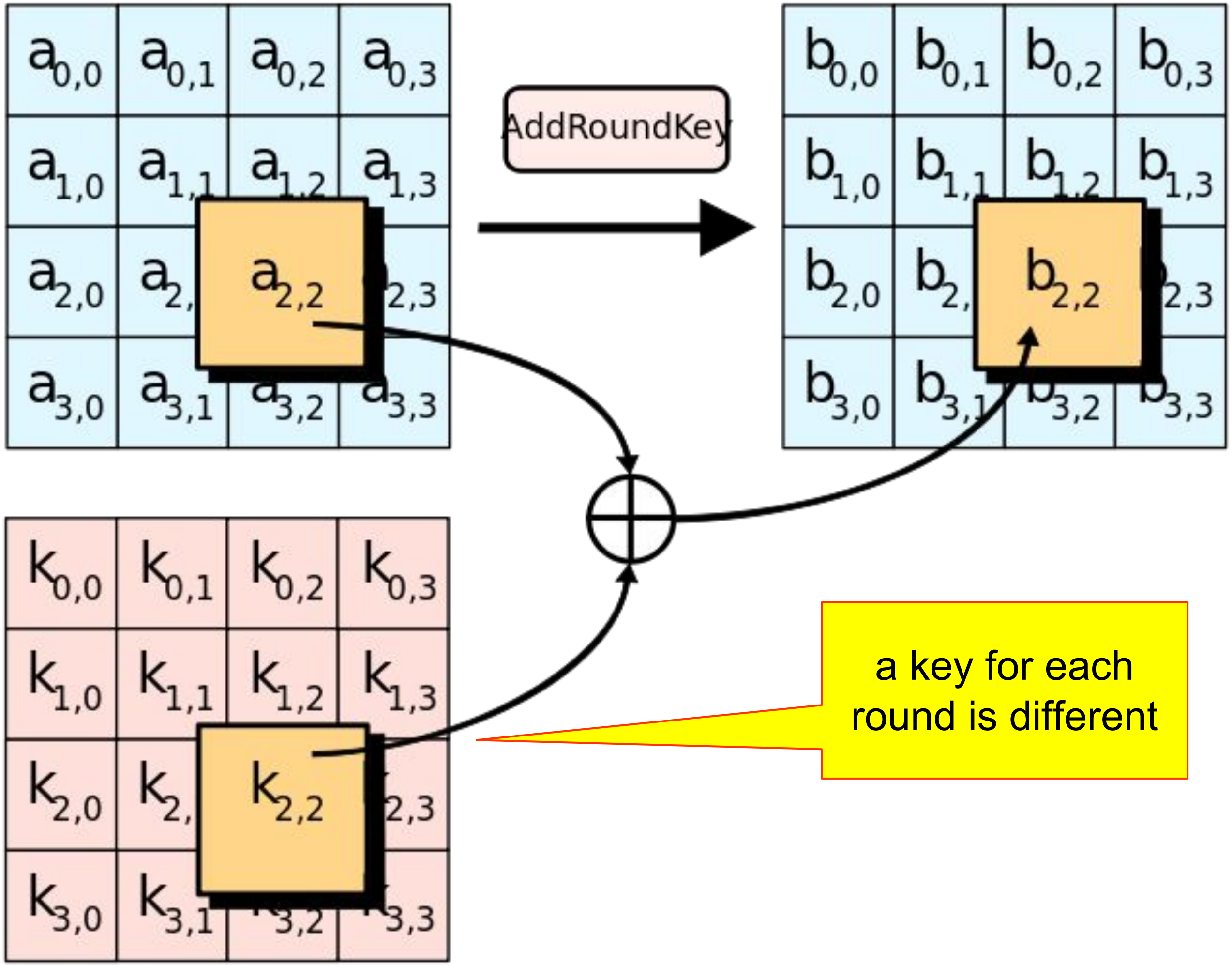
$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}$$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$s'_{0,0}$	$s'_{0,1}$	$s'_{0,2}$	$s'_{0,3}$
$s'_{1,0}$	$s'_{1,1}$	$s'_{1,2}$	$s'_{1,3}$
$s'_{2,0}$	$s'_{2,1}$	$s'_{2,2}$	$s'_{2,3}$
$s'_{3,0}$	$s'_{3,1}$	$s'_{3,2}$	$s'_{3,3}$

AES: Add Round Key

- XOR state with 128-bits of the round key



$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
AddRoundKey
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$\oplus$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,1}$ $k_{2,2}$ $k_{2,3}$
$k_{3,0}$ $k_{3,1}$ $k_{3,2}$ $k_{3,3}$

a key for each round is different

AES

- Objectives:
  - resistance against known attacks
  - speed and code compactness on many CPUs
  - design simplicity
- Every step is invertible
- why there is AddRoundKey at the beginning?
- why the last round is different?

MODES OF OPERATIONS

Modes of Operations

- message is typically longer than the block size
- The relation between the blocks can be various
- DES and AES have multiple modes of operations
- depending on how consecutive blocks are processed
- five modes
    - ECB
    - CBC
    - CFB
    - OFB
    - CTR

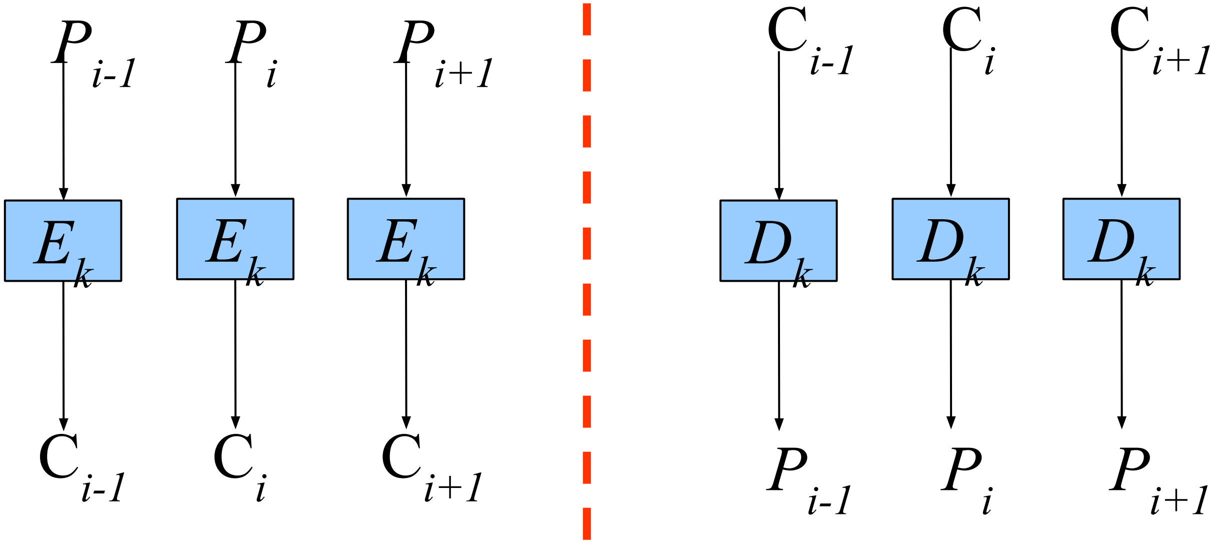
Electronic CodeBook (ECB)

- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks

  $C_i = E_{K1}(P_i)$
- uses: secure transmission of a single value

Electronic Codebook Mode (ECB)

64 bit (8 byte) blocks in DES



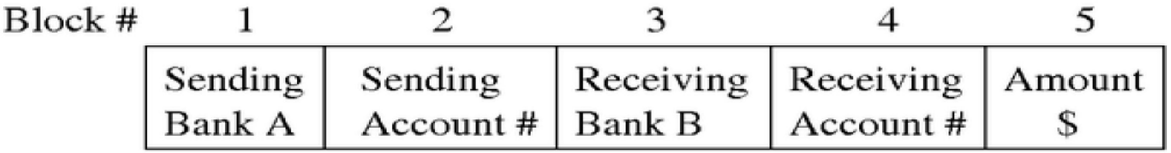
Ciphertext = $(C_0\ C_1 \ldots C_n)$

Advantages and Limitations of ECB

- repetitions in message may appear in ciphertext
  - if aligned with message block
  - particularly with data such as graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness due to encrypted message blocks being independent
- ECB mode is susceptible to cut-and-paste *attacks*
- main use is sending a few blocks of data

cut-and-paste attack

- Once a particular plaintext to ciphertext block mapping Pi → Ci is known, a sequence of ciphertext blocks can easily be manipulated
- Suppose an electronic funds transfer (EFT)



Block #	1	2	3	4	5
	Sending Bank A	Sending Account #	Receiving Bank B	Receiving Account #	Amount $

- Encryption key between banks change not frequently
- Attacker sends $1 from his account in bank A to his account in bank B,
  - Sends $1 from another account to his account in bank B, ...
  - Can change the amount as well
- Intercepting any bank transfer message, replace block4 with the ciphertext for his account in bank B

cut&paste attack is a CCA attack since it can modify ciphertext

Cipher Block Chaining (CBC)

- Two identical plain message blocks produce two different cipher messages.
- This prevents Chosen plaintext attack (CPA).

Cipher Block Chaining (CBC)

- a message is broken into blocks
- but these are linked together in the encryption operation
- each previous cipher block is chained with current plaintext block, hence name
- use Initialization Vector (IV) to start process

  $C_i = E_{K1}(P_i \text{ XOR } C_{i-1})$

  $C_{-1} = IV$
- uses: bulk data encryption, authentication

Cipher Block Chaining Mode (CBC)

Encryption

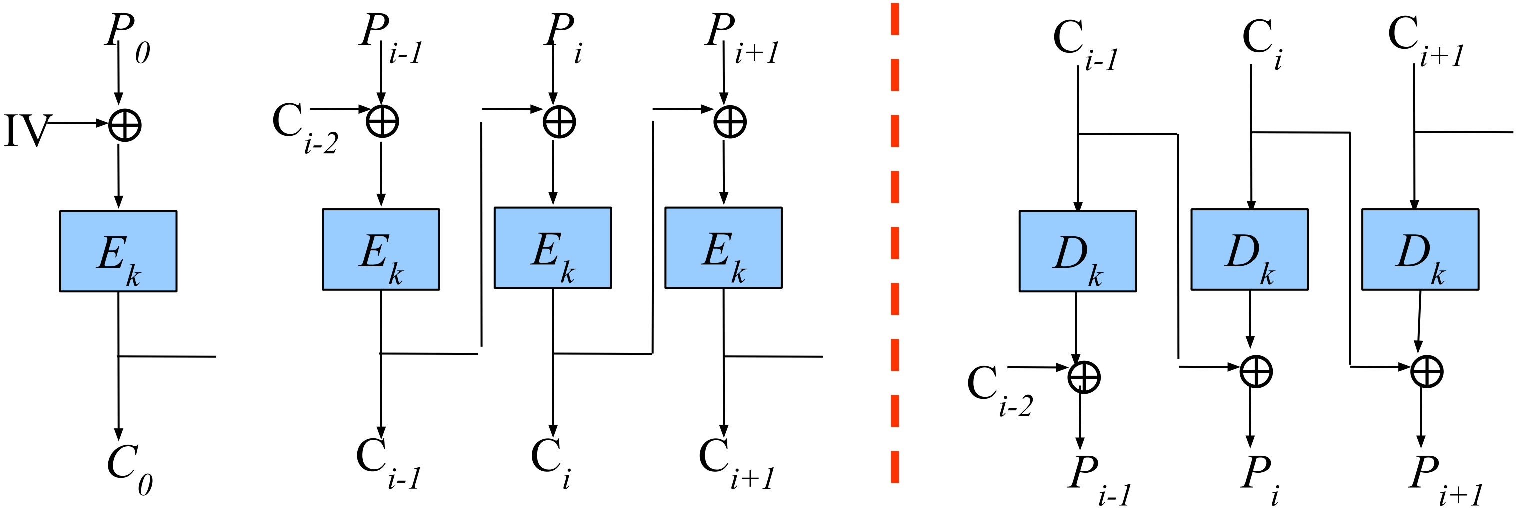
$C_0 = E_k(P_0 \oplus \mathrm{IV})$

$C_i = E_k(P_i \oplus \mathrm{C}_{\mathrm{i-1}})$

Decryption

$P_0 = \mathrm{IV} \oplus \mathrm{D}_k(\mathrm{C}_0)$

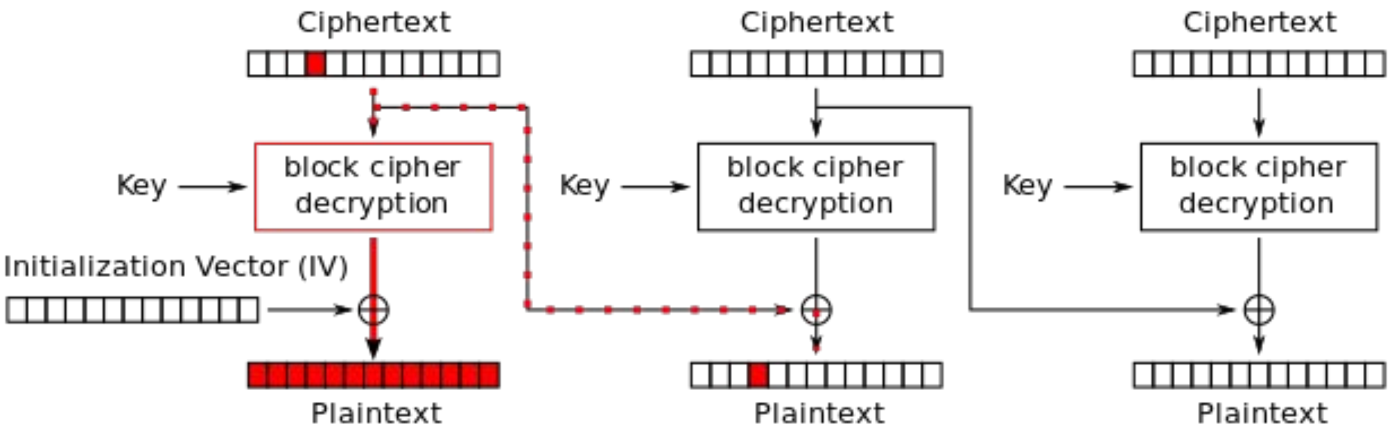
$P_i = \mathrm{C}_{i-1} \oplus \mathrm{D}_k(\mathrm{C}_{\mathrm{i}})$



Ciphertext = $(IV\ C_0\ C_1 \dots C_n)$

CBC – *Error Propagation*

- 1 bit error in the plaintext feed
  - Will affect that block and all others
  - Decryption will correct all but the flipped bit
- 1 bit error in the ciphertext feed
  - Will affect two blocks
  - CBC mode is self recovering



Cipher Block Chaining (CBC) mode decryption

Source: wikipedia

CBC – Initialization Vector (IV)

- If IV is same, every encryption of the same plaintext is the same
- With the IV being a random number, the same plaintext will be all different
- IV can be in public, but should satisfy two req's
  - otherwise, some attacks possible
  - two requirements for IV usage
    - no IV is reused under the same key
    - IV change should be unpredictable

If IV req’t is not satisfied

- IV reuse with the same key
  - using the same key+IV to encrypt two messages sharing a common prefix will generate ciphertexts with the same prefix
  - It will reveal the presence and length of that prefix
- IV is predictable → chosen plaintext attack (CPA)
  - Eve can see any ciphertexts, predict IVs, and wish to guess Alice’ credit level
  - $P_{eve} = IV_{eve} \oplus IV_{alice} \oplus$ “guess“
  - $C_{eve} = E_k(IV_{eve} \oplus P_{eve}) = E_k(IV_{eve} \oplus (IV_{eve} \oplus IV_{alice} \oplus$ “guess"))
    $= E_k(IV_{alice} \oplus$ “guess")
  - Eve can compare C_{eve} and C_{alice}

normally, IV_{eve} is not known in advance since it will be determined when Eve’s message is about to be encrypted

Advantages and Limitations of CBC

- each ciphertext block depends on **all** previous blocks
- encrypting a block requires the finish of encryption of all the previous blocks
  - no parallelism in encryption
- parallelism in decryption
- thwart CPA if IV is random

Cipher FeedBack (CFB)

- a message is treated as a stream of bits
  - can be a stream cipher
- added to the output of the block cipher
  - block cipher (block) length is independent
- result is feedback for next stage (hence name)
- standard allows any number of bits (1,8 or 64 or whatever) to be feedback
  - denoted CFB-1, CFB-8, CFB-64 etc
- is most efficient to use all 64 bits (CFB-64)

$C_i = P_i \text{ XOR } E_{K1}(C_{i-1})$

$C_{-1} = IV$

Cipher Feedback Mode (CFB)

Encryption

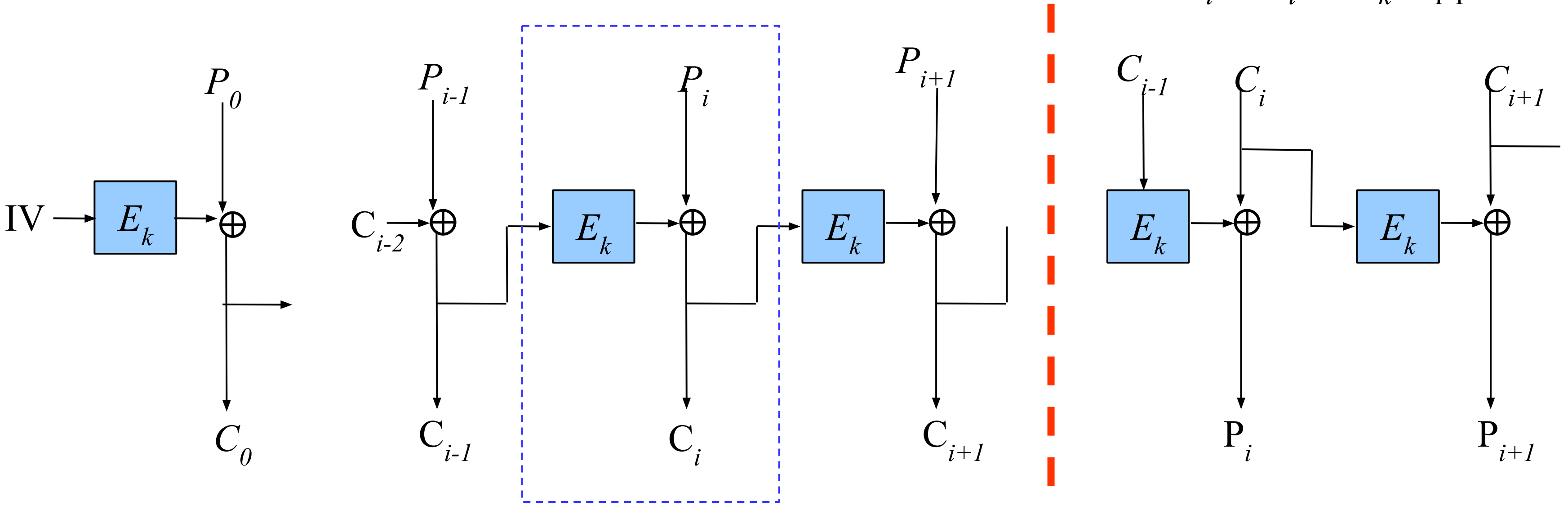
$C_0 = P_0 \oplus E_k(\mathrm{IV})$

$C_i = P_i \oplus E_k(\mathrm{C}_{\mathrm{i-1}})$

Decryption

$P_0 = E_k(\mathrm{IV}) \oplus \mathrm{C}_0$

$P_i = \mathrm{C}_i \oplus \mathrm{E}_k(\mathrm{C}_{\mathrm{i-1}})$



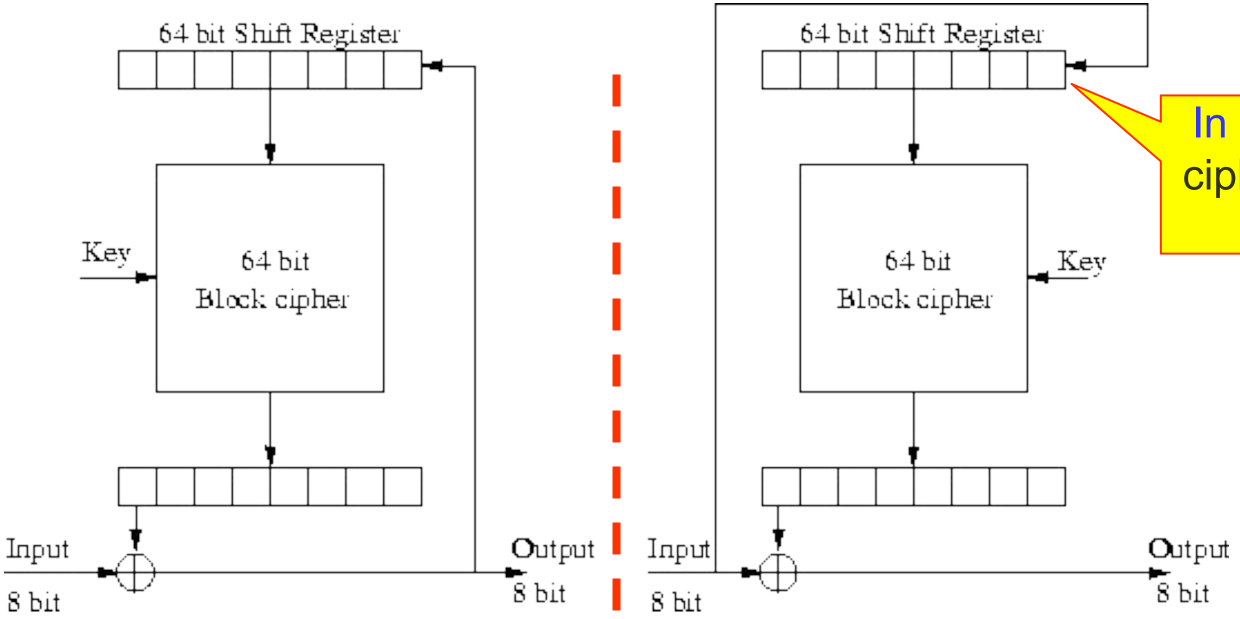
Ciphertext = $(IV\ C_0\ C_1 \ldots C_n)$

CFB – Initialization Vector

- CFB must use an IV
- same requirements as CBC

CFB – *Error Propagation*

- CFB mode is self recovering
- one bit error in ciphertext corrupts some # of blocks
  - CFB-8 -> 8 bytes are garbled



In CFB-8, 8 bits (i.e., ciphertext) are shifted for each round

Bit errors in the incoming cipher block (i.e. a byte) will cause bit errors at the same bit positions in the first plaintext block. This cipher block will then be fed to the shift register and cause bit errors in the plaintext for as long as the erroneous bits stay in the shift register. Hence, for 8-bit CFB, the following 8 bytes will be garbled. After that, the system recovers, and all following bytes is decrypted correctly.

Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes: stream cipher
- One of the most common stream modes
- encryption is not parallelizable
  - keystream cannot be generated in advance
- decryption is parallelizable
- note that the block cipher is used in **encryption** mode at **both** ends

output feedback mode (OFB)

- can be a stream cipher
- IV is used as a seed to generate OTP
- actual encryption/decryption is only X-or
  - fast
- no dependency between consecutive blocks
  - parallel encryption/decryption

Output Feedback Mode (OFB)

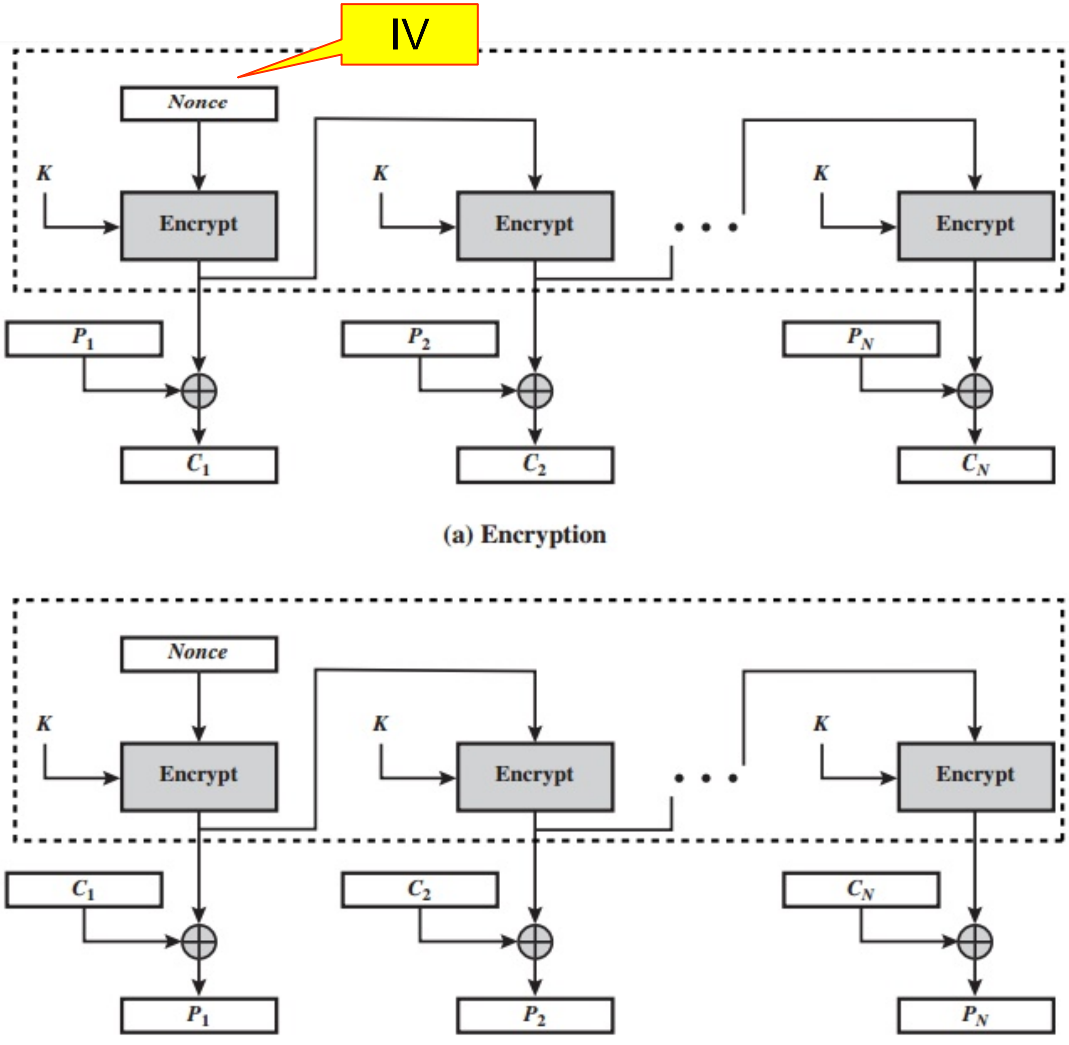
Encryption

$$C_0 = P_0 \oplus E_k(\mathrm{IV})$$
$$C_i = P_i \oplus E_k(\mathrm{S}_{i-1})$$

Keystream

Decryption

$$P_0 = E_k(\mathrm{IV}) \oplus \mathrm{C}_0$$
$$P_i = \mathrm{C}_i \oplus E_k(\mathrm{S}_{i-1})$$



(a) Encryption

(b) Decryption

Note: IV and successive encryptions act as an OTP generator.

OFB: advantages and limitations

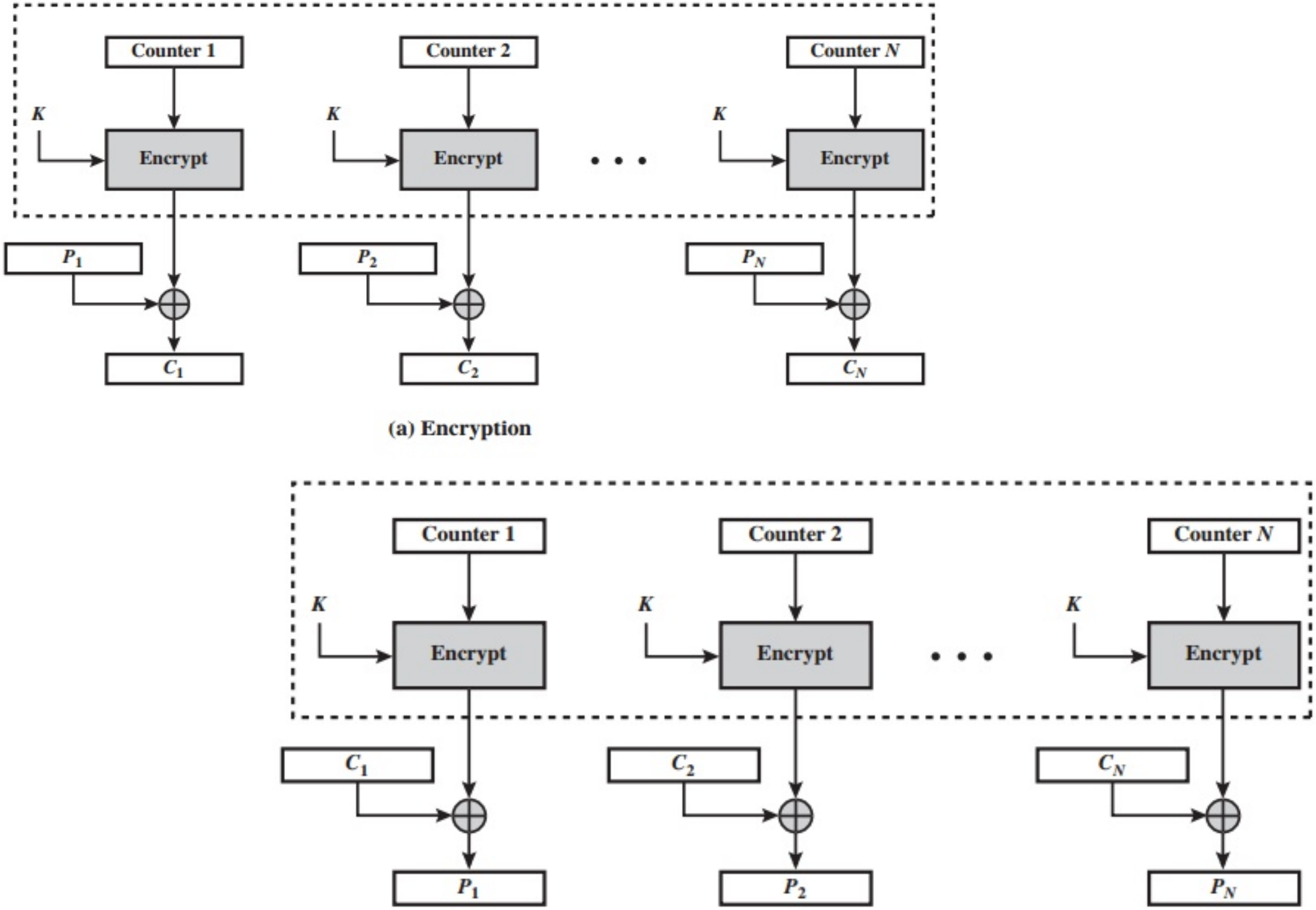
- no error propagation
  - 1 bit error in ciphertext affects only one bit in plaintext
- a (key)stream can be generated in advance
- fast due to parallelism
- if attacker knows plaintext and ciphertext, he can modify the plaintext
- how likely is a keystream S_i repeated?

counter mode (CTR)

- can be a stream cipher
- IV is called a counter
- highly parallelizable
  - no linkage between blocks
  - very fast
- can decrypt from any arbitrary position unlike OFB
- counter should not be repeated for the same key
  - otherwise, attacker can get ⊕ of two plaintext blocks by taking the ⊕ of two ciphertext blocks like OFB

counter (CTR) mode

- counter is typically incremented by 1

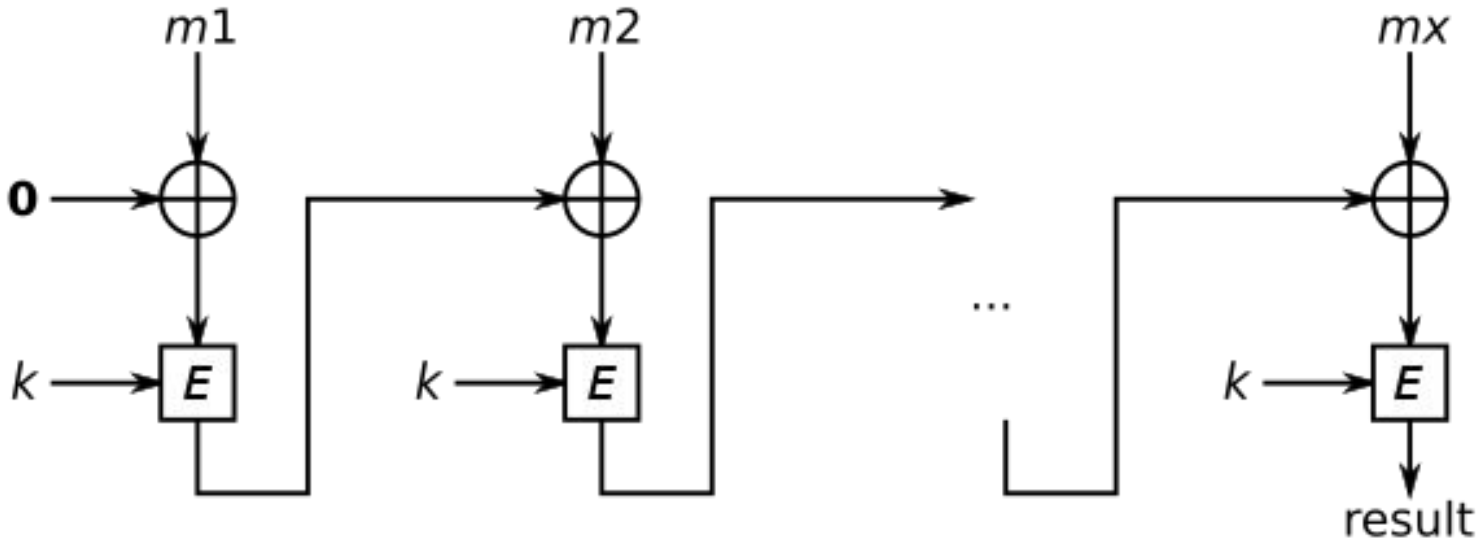


(a) Encryption

(b) Decryption

other usage of modes of operations: message authentication code (MAC)

- CBC can be used to check message integrity


m1
m2
mx
0
k
E
k
E
...
k
E
result