Block Ciphers and Modes of Operation

- Readings
  - Sections 3.3, 4.1, 4.2, 4.4

Block Cipher

- A **block cipher** $E_\pi(\bullet)$ is a (parametrized) deterministic function mapping *n*-bit plaintext blocks to *n*-bit ciphertext blocks. The value *n* is called the **blocklength**.
  - It is essentially a simple substitution cipher with character set = $\{0, 1\}^n$.
  - Example for a 64 bit block:

01011100 … 10101…….

(64 bits) (64 bits)

Are there any restrictions on this function for it to be a cipher?

Counting the Number of Functions

Consider a mapping f: N → N, N a finite set

Let |N| be the size of the set N.

Then there are $|N|^{|N|}$ such functions

If one considers only 1-1 functions, (injective), then there are |N|! such functions

If |N| is 2^{64} then there are $2^{64}!$ one-one (injective) functions.

Note: Since N is a finite set, an injective function over N to itself is also bijective

- Injective and bijective functions on wikipedia

Specifying the Functions

- Specifying an arbitrary function on 64-bit blocks (or even just an arbitrary bijective function) takes too many bits.
  - For an arbitrary function of *k* bits, it takes $k2^k$ bits to specify it directly.
  - For 64 bit blocks, this is $64 \cdot 2^{64}$ or 2^{70}.
  - Even specifying a 1-1 function of k bits takes about the same number of bits.
- Note that we can use Stirling's approximation to estimate n! if needed:

$$\boxed{n! \approx \sqrt{2\pi n} \bullet \left(\frac{n}{e}\right)^n}$$

The Key to the Cipher

- The parameter **key** is a *k*-bit binary string.
  - It may be that the set of all keys, the **keyspace** *K,* is a proper subset of all *k*-bit binary strings. In that case, we say that the **effective key size,** or **security parameter**, provided by the cipher is $log_2|K|$
- The keyed block cipher $E_K(\bullet)$ is a bijection, and has a unique inverse: the decryption function $D_K(\bullet)$.
  - Alternative notation: $K\{\bullet\}$ and $K^{-1}\{\bullet\}$

Using simple transformations on block subcomponents: substitution

- Substitution: changing each input subblock to some output subblock.
- Example 8 bit block:

“*xor* with 11101011 = y”

Let an input block be m = 01100100

Then, the output of the “substitution” is

$m \oplus y = 10001111 = c$

Note: is this mapping 1-1 onto?

Using simple transformations on block subcomponents: permutation

- A permutation in this context is simply a shuffling of the bits of the subblock.
- Example 8-bit block

"define where each bit of the shuffled block comes from"

Bit 1 → to position 5

Bit 2 → to position 6

Bit 3 → to position 2

…

Bit 8 → to position 1

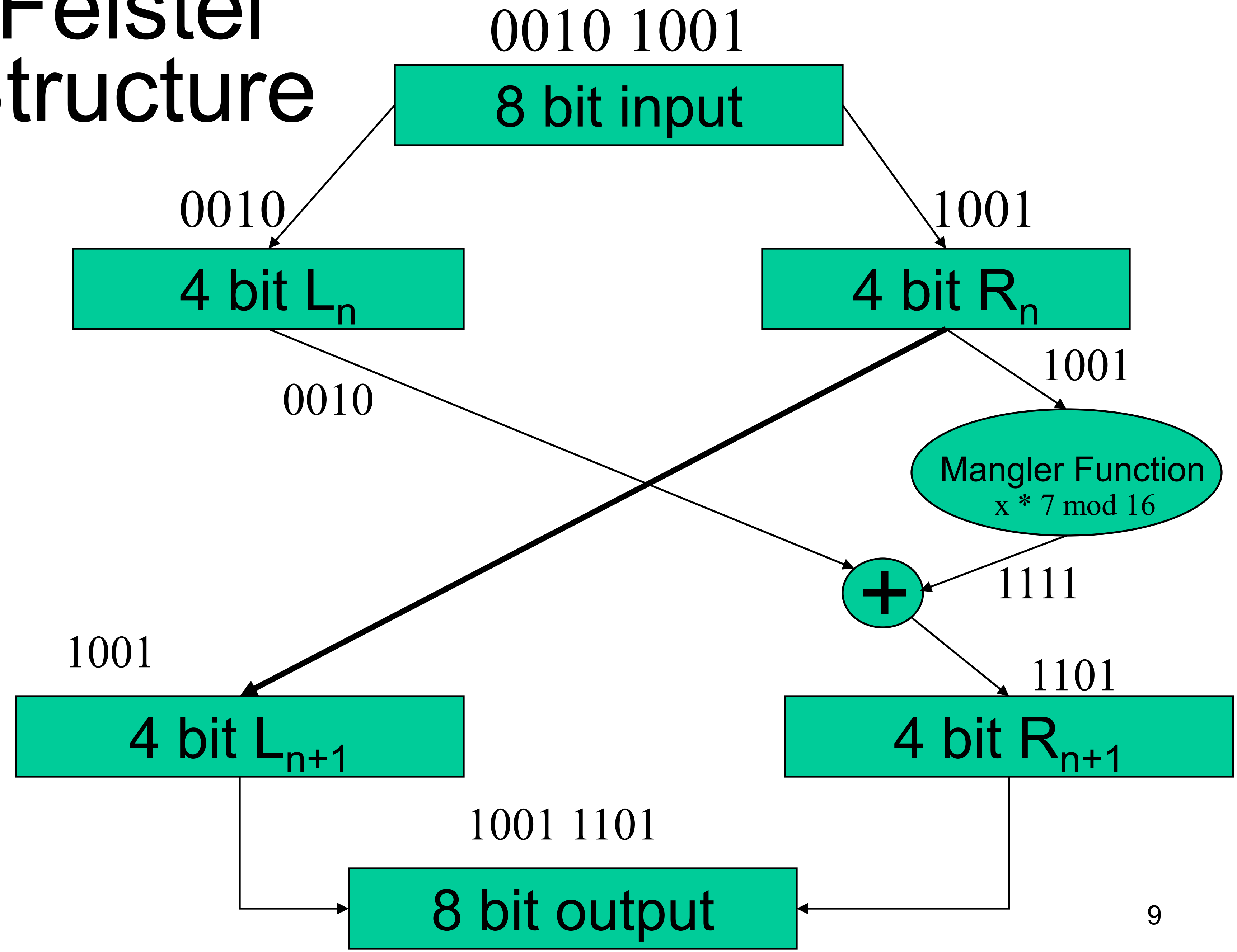
8 3 5 4 1 2 6 7

01100100 → 01000110

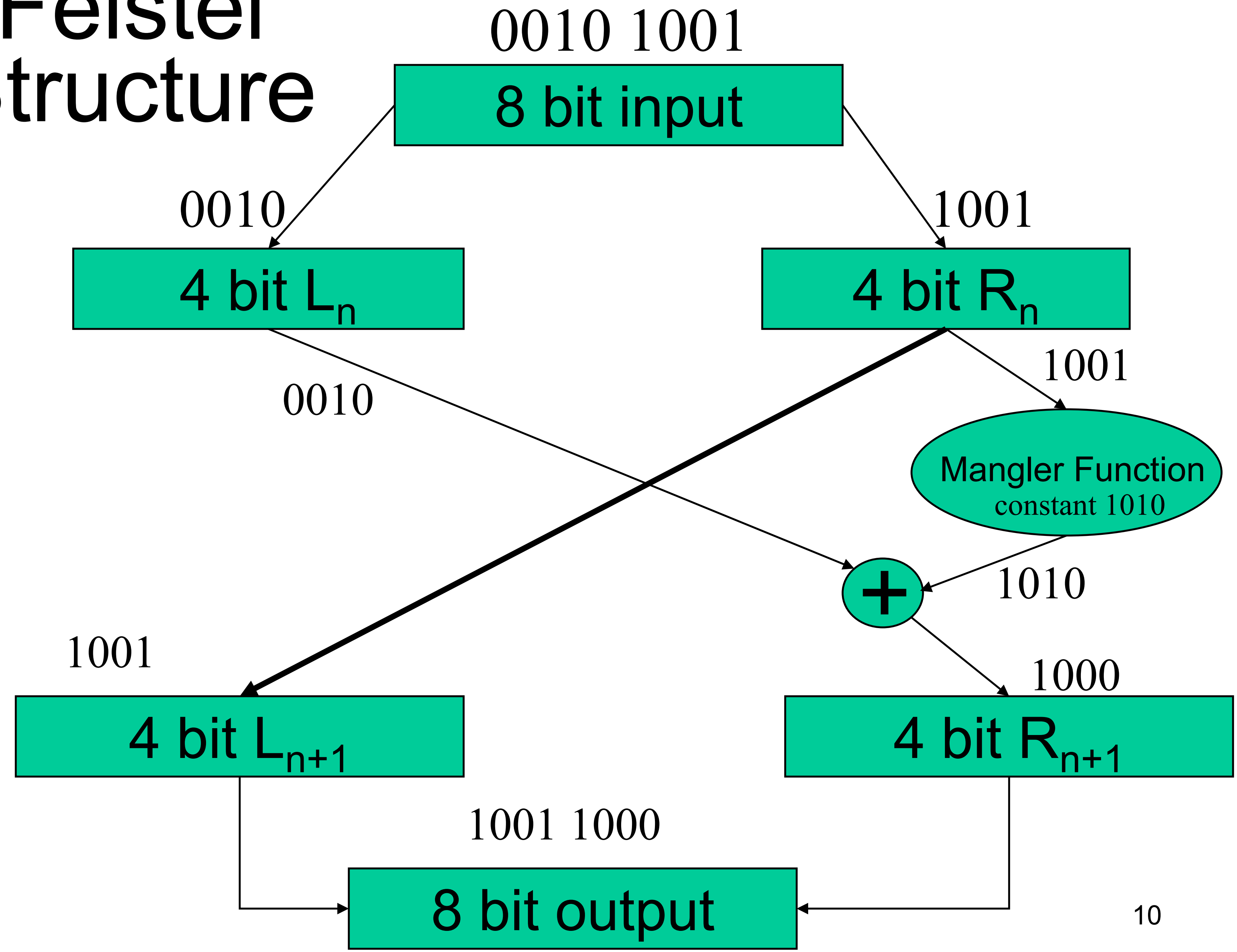
Feistel Structures

- Technique for scrambling data
- Scrambles a block at a time
- Based on the reversible properties of the *XOR* function

Feistel Structure



Feistel Structure



DES

- DES uses a 56 bit key to guide the encryption, which works roughly as follows:
  - An initial permutation is done on the 64-bit input
  - A 56-bit key is used to generate 16 subkeys used in 16 rounds (subkey generation is complex in itself)
  - Rounds can be viewed as doing substitutions and permutations in each round, based on the subkey (these are the real "scrambling the data" rounds)
  - A final permutation is done that is the inverse of the initial permutation
  - Developed by NSA with industry input

The Initial and Final Permutations

Original order

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

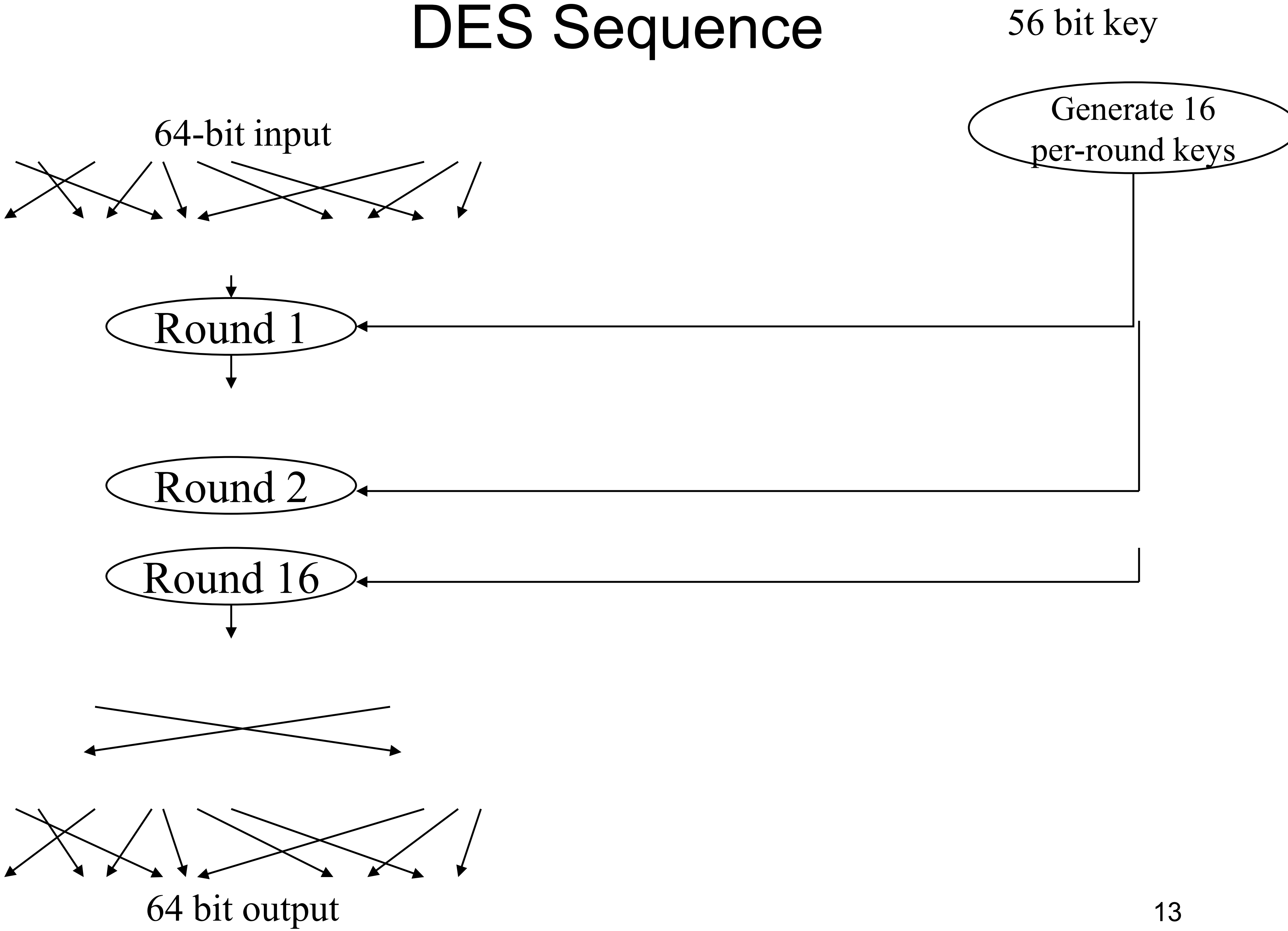
Initial Permutation

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

Final Permutation

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

DES Sequence



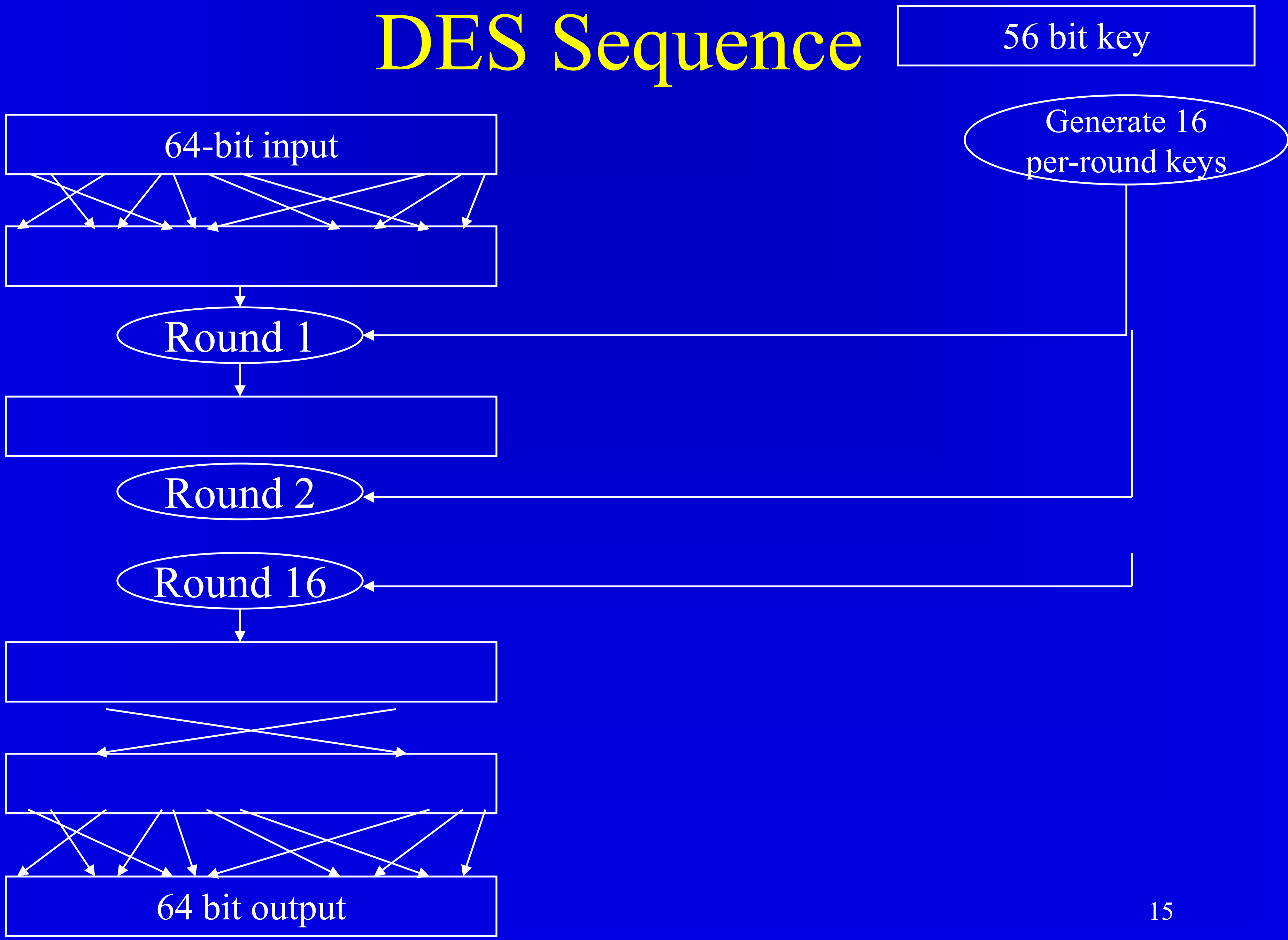
Generate Sixteen 48 Bit Keys

- *Permute initial DES key* (64 bits with parity):
  - Extracts 56 of 64 key bits (omits parity bits) using a given permutation called *permuted choice 1* resulting in two 28 bit sub-keys called C_0 and D_0 . Next do:
- *16 rounds of the following cascading process*

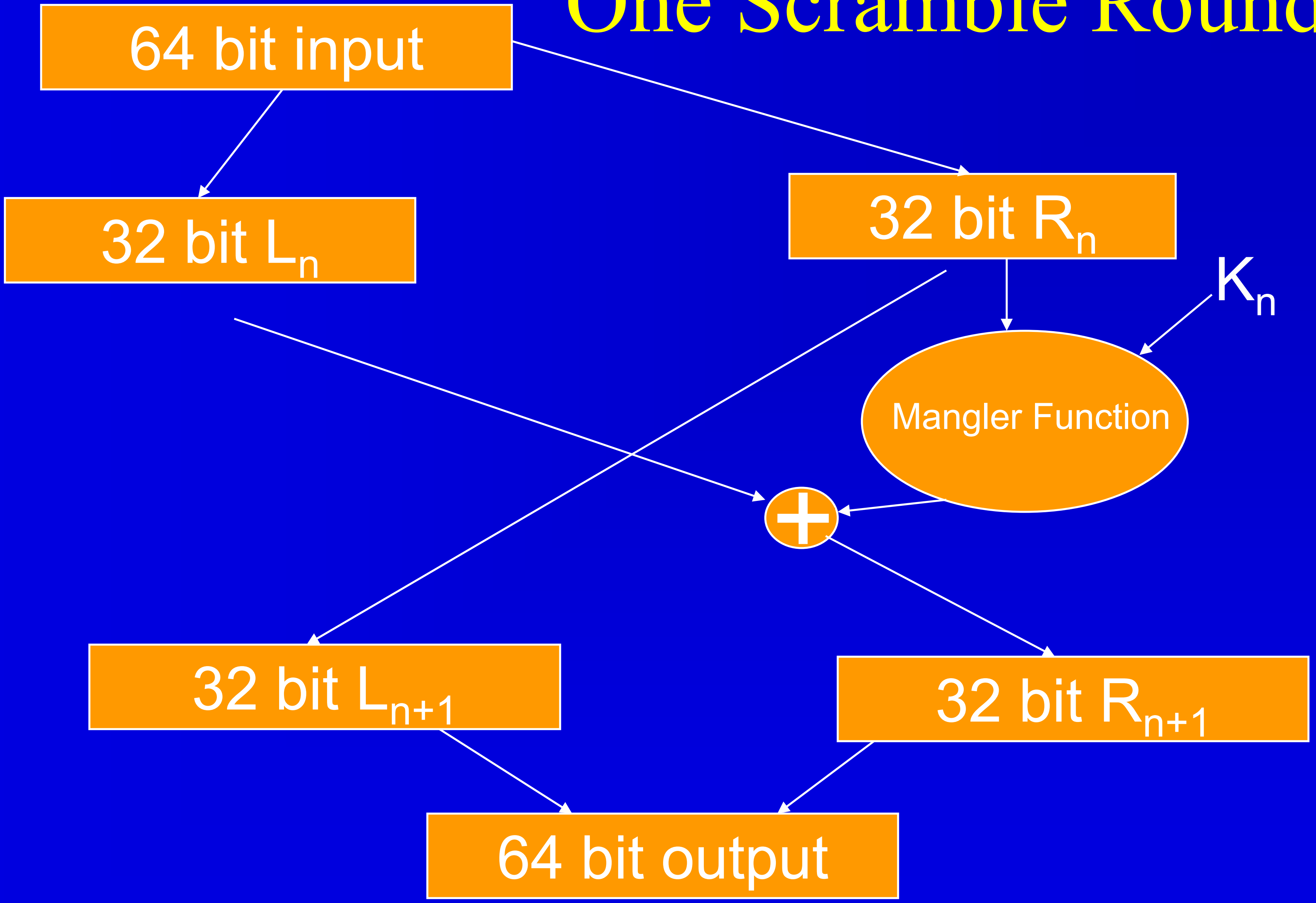
1. Shift the 28 bits of each half (C_{i-1} and D_{i-1})
   - In rounds 1, 2, 9, and 16 single shift left
   - Other rounds, two-bit rotate left
   - The output feeds back into step 1 of the next round and step 2 below
2. Permute each half defined by *permuted choice 2* which does not use 8 of the bits (positions 9, 18, 22, 25 and 35, 38, 43, 54)
3. Concatenate the two halves into a 48 bit key k_i

Note: The actual permuted choice 1 and 2 are shown in text

DES Sequence



One Scramble Round



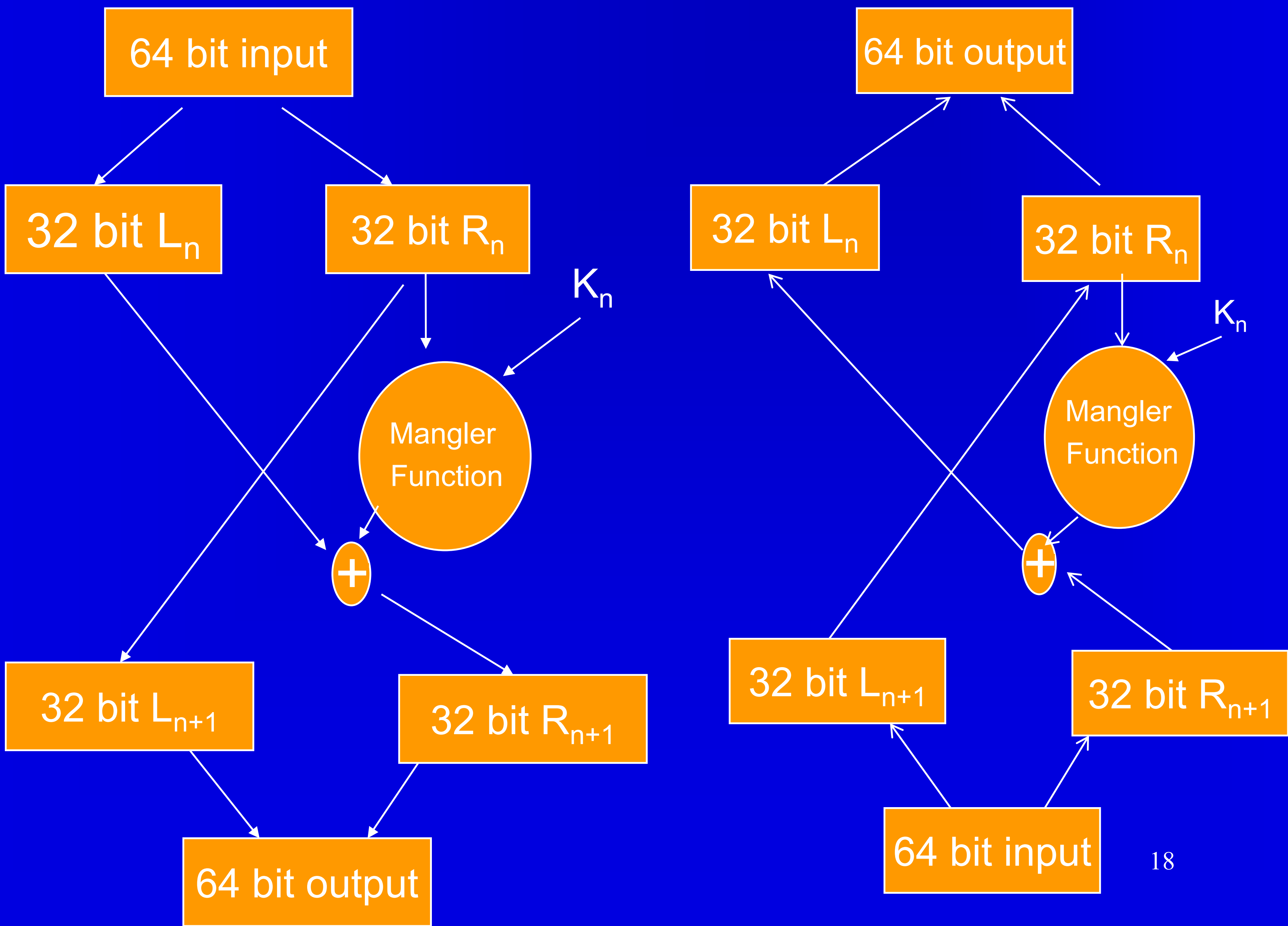
Mangler Function

Combine 32 bit input and 48 bit key into 32 bit output

1. Expand 32 bit input to 48 bits by adding a bit to the front and end of each 4 bit segment. (These bits are taken from adjacent bits of the 4-bit segment) to get R_1 to R_8.
2. XOR each 6 bit R_i of input with 6 bits of key K_i to get V_i.
3. Feed each 6 bit V_i result into an S_i box process.
4. The output of each S_i box process is a *4-bit result.*
5. Combine the S_i box processes into a 32 bit result and do a defined permutation (see text).

Encrypt round n

Decrypt round n

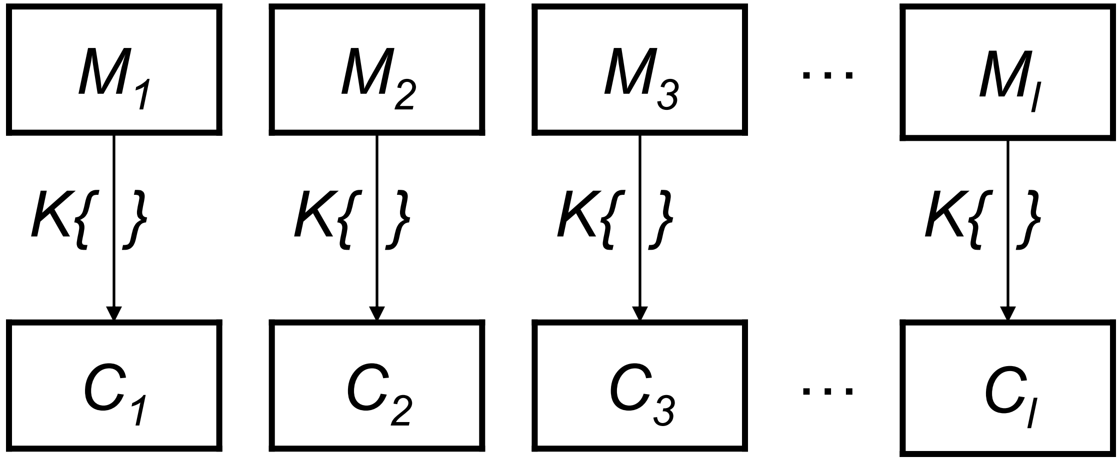


Using a Block Cipher

- Assuming one can encrypt a 64-bit block with a cipher such as DES or 3DES (triple DES), how do you use this capability?
  - Messages are longer than 64 bits
  - They may not be a multiple of 64 bits
  - What are the security implications of the encryption / decryption methods on these messages

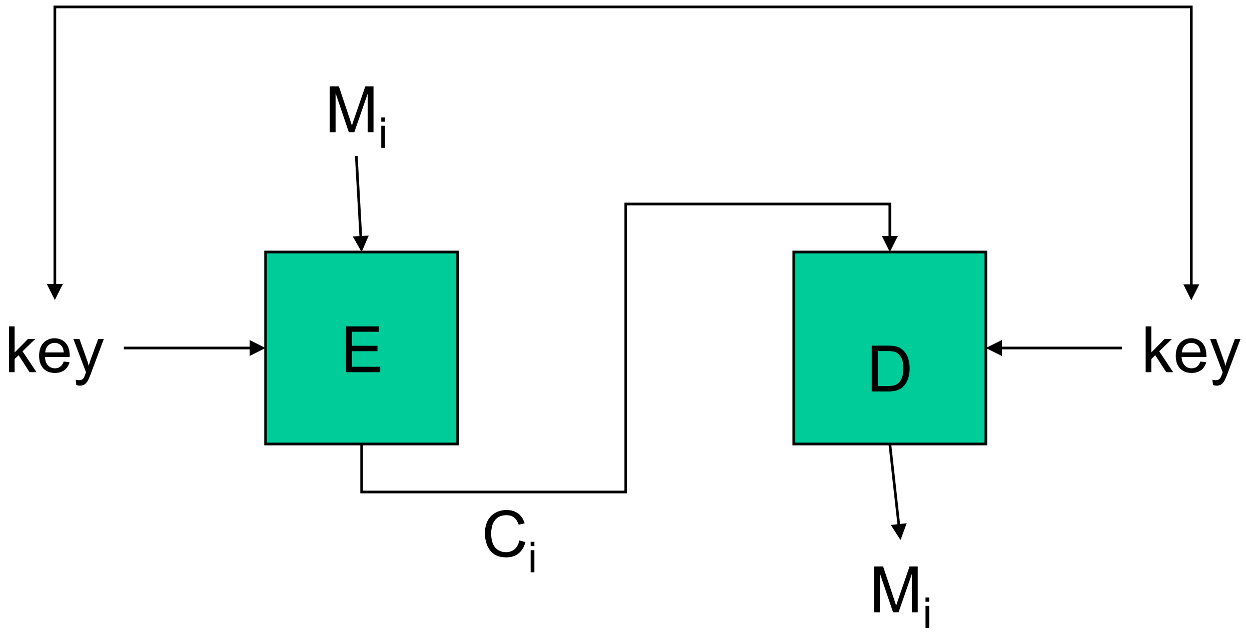
Modes of Operation

- Clearly, the block cipher can be used exactly as a substitution cipher, i.e., by encrypting each block of plaintext independently using the same key. This is called the **Electronic Codebook Mode**, or **ECB**:



ECB (continued)

- Decryption also works block by block (inverse substitution):

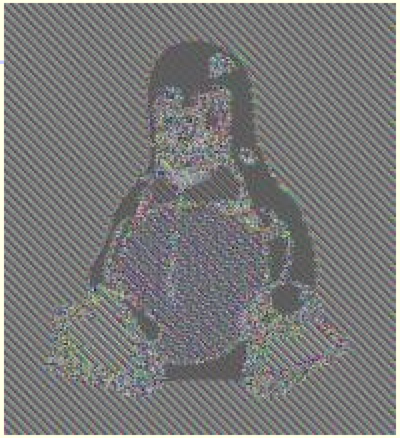


ECB Limitations

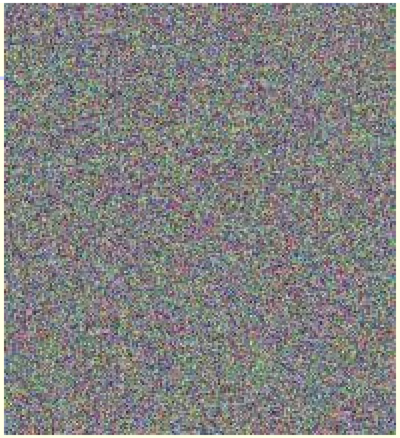
- If a message has two identical blocks, the ciphertext will be two identical blocks
- Blocks can be rearranged by an adversary to his advantage
- Message information is not sufficiently diffused
- Thus ECB use is limited, such as for transmitting an IV vector

Pictures from
http://en.wikipedia.org/wiki/Cipher_Block_Chaining

Original



Encrypted using ECB mode

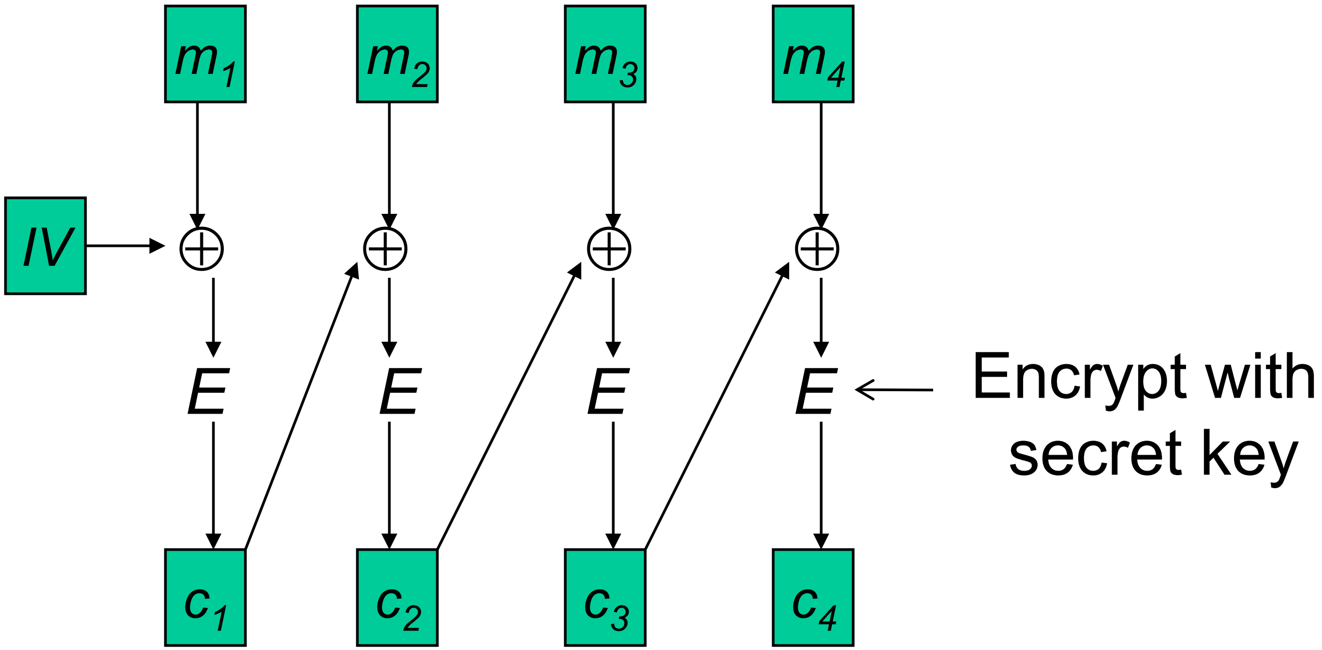


Encrypted using other modes

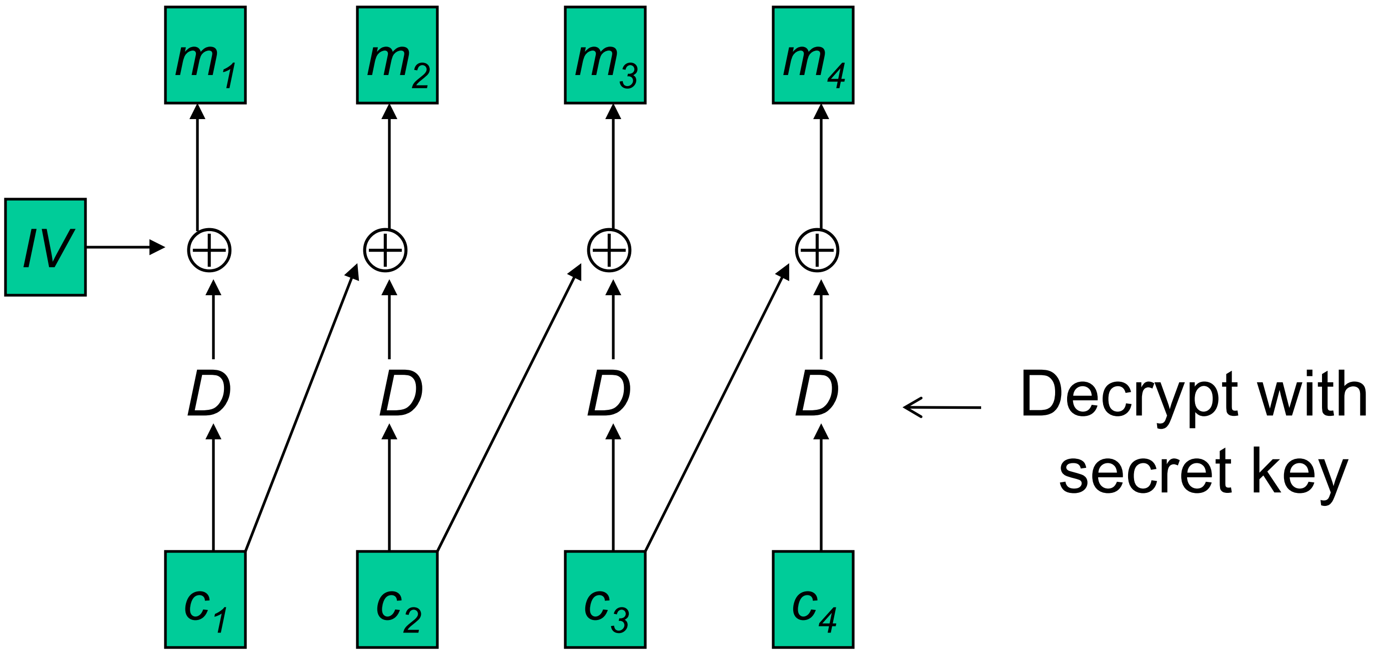
Cipher Block Chaining (CBC)

- An initial vector (IV) is *xored* into the first block before encryption:
  - $C_1 = E_k(IV \oplus M_1)$
- Subsequent blocks are first *xored* with the previous cipherblock before encrypting:
  - $C_{i+1} = E_k(C_i \oplus M_{i+1})$
- The encrypted message is transmitted as
  - $IV, C_1, \ldots, C_l$

Encryption using CBC

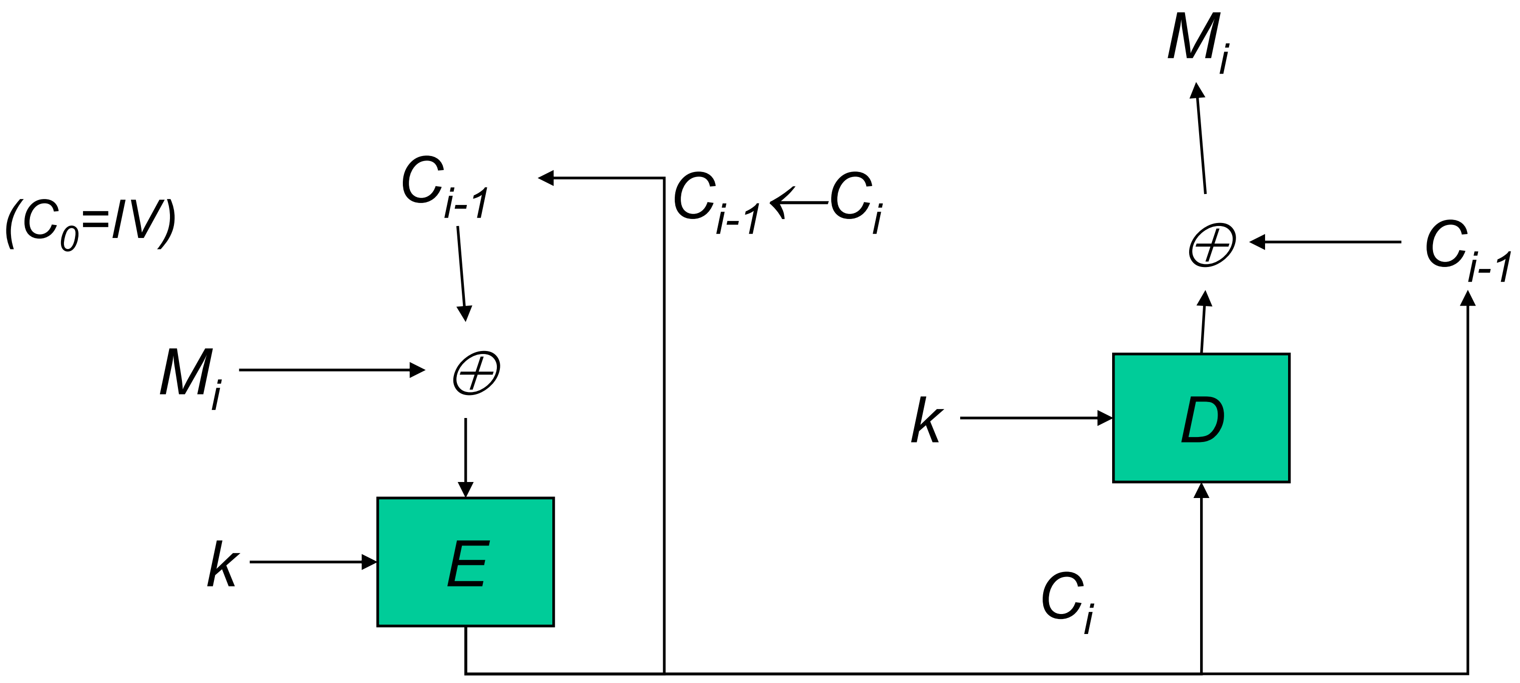


Decryption using CBC



CBC (continued)

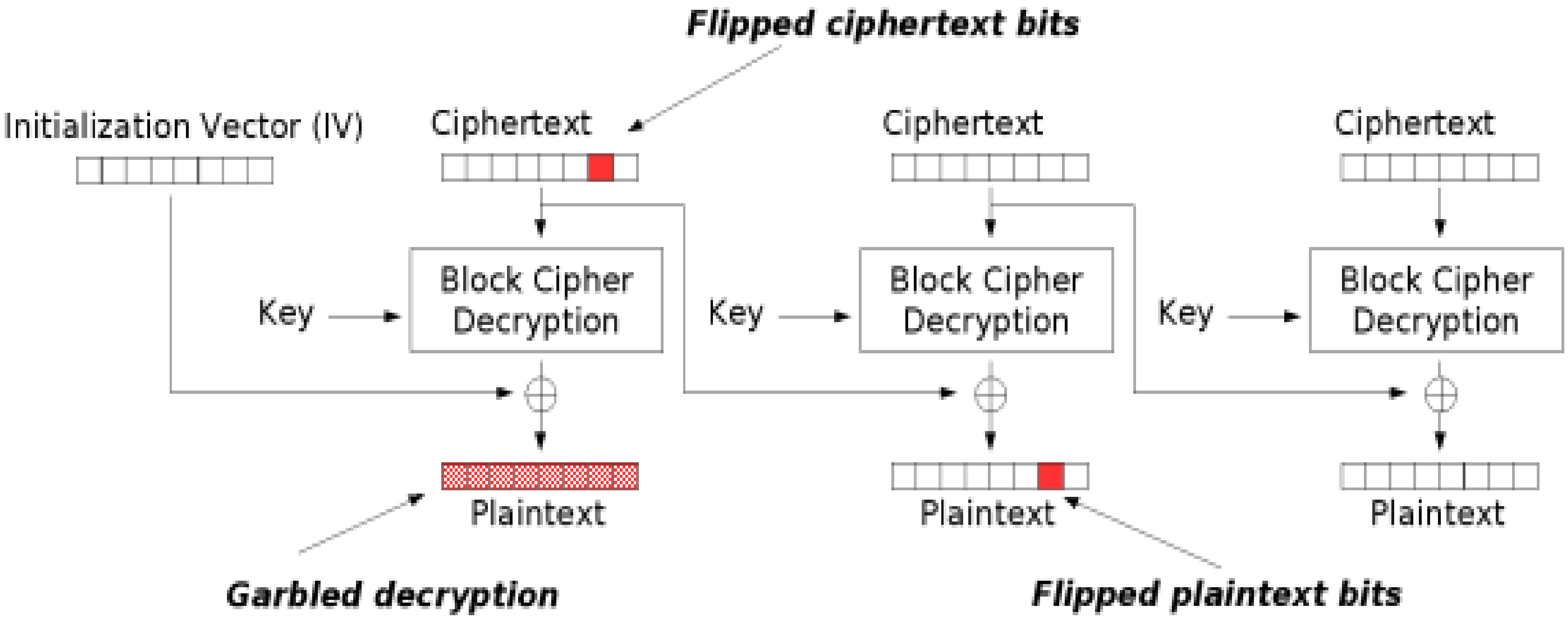
- Decryption of C_i uses knowledge of C_{i-1} (where $C_0 = IV$):
  - $M_i = D_k(C_i) \oplus C_{i-1}$



CBC issues

- Not parallelizable (for encryption)
- A single-bit transmission error in ciphertext block C_i results in whole plaintext block P_i and the same bit in plaintext block P_{i+1} being corrupted.
- The IV should be integrity-protected
- The IV can be sent in the cleartext.

CBC Error Propagation



Modification attack or transmission error for CBC

http://en.wikipedia.org/wiki/Cipher_Block_Chaining

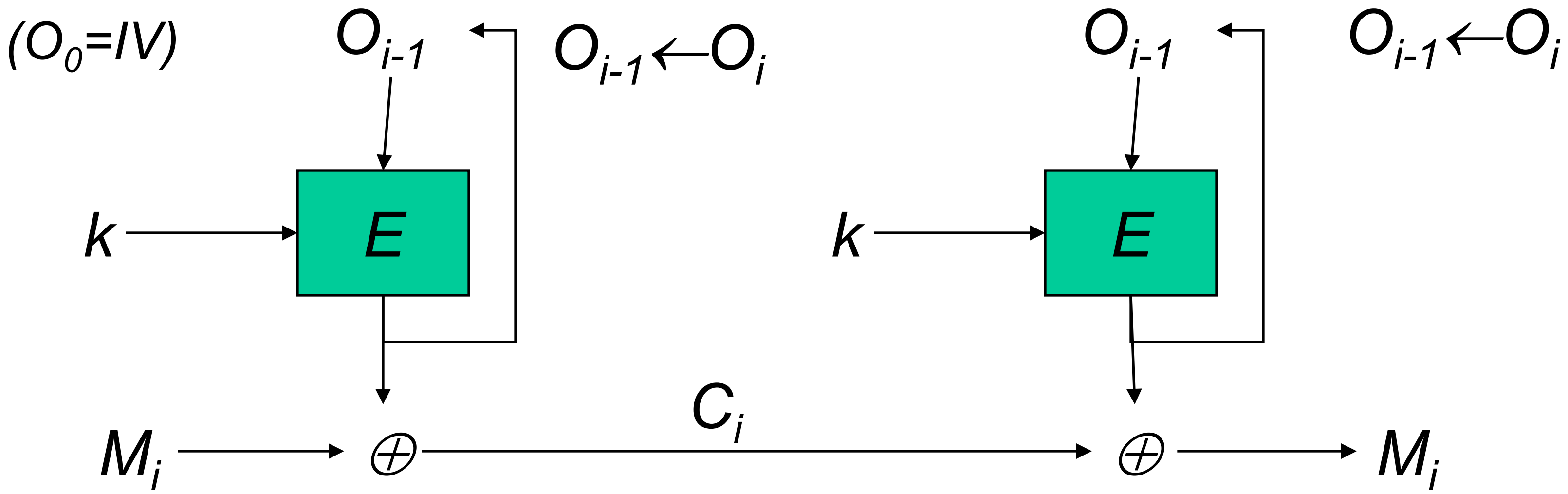
Block Ciphers as Stream Ciphers

- Two modes of operation of a block cipher implement a stream cipher:
  - **Output Feedback Mode (OFB),** a **Key-auto-key** stream cipher **(KAK)**
  - **Cipher Feedback Mode (CFB)**, a **Ciphertext-auto-key** stream cipher **(CTAK)**

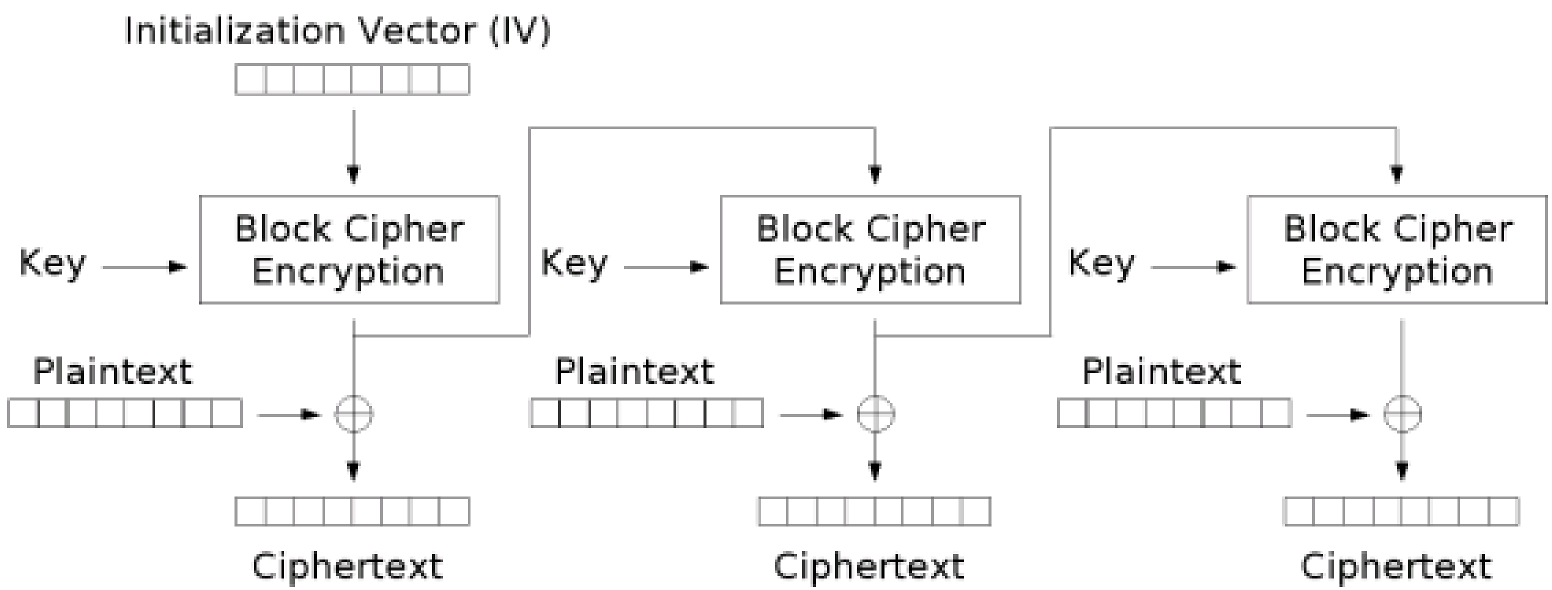
  - In both cases encryption is obtained by xoring a keystream with the plaintext.
    - OFB: Keystream depends on previous keystream
    - CFB: Keystream depends on previous ciphertext

OFB

- The keystream (output of encryption) is xored into plaintext to obtain ciphertext. The keystream is also the input for next chained encryption.
  - $C_i = M_i \oplus O_i$; $O_i = E(O_{i-1})$ **(encryption)**
  - $M_i = C_i \oplus O_i$; $O_i = E(O_{i-1})$ **(decryption)**

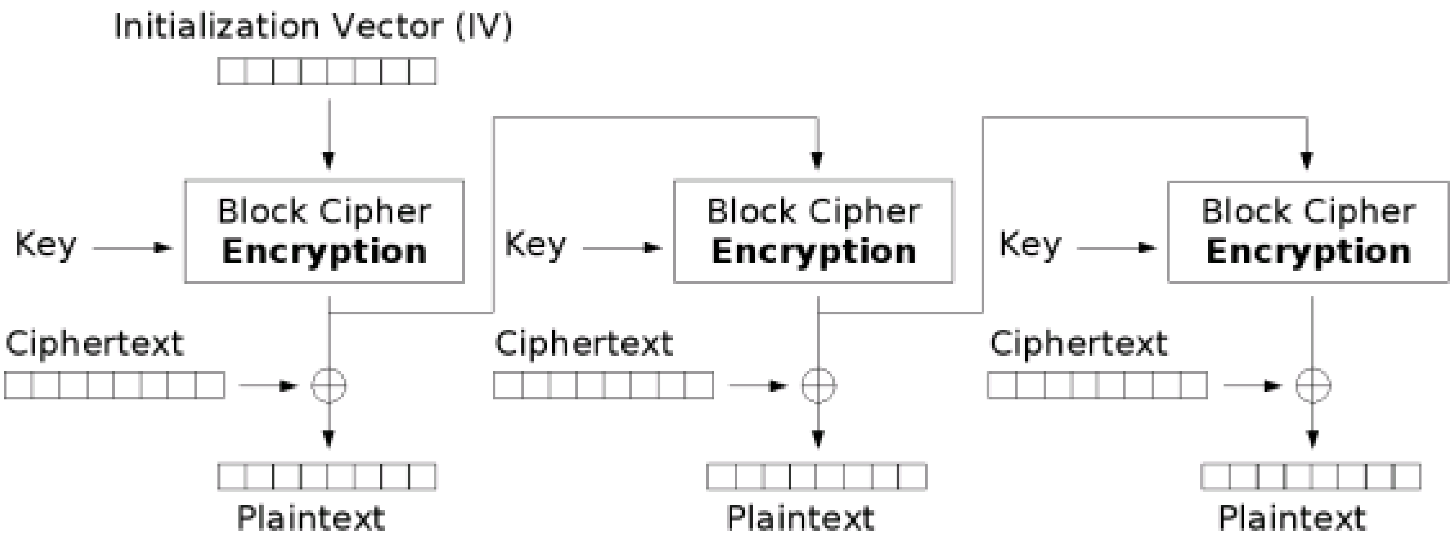


OFB Encryption



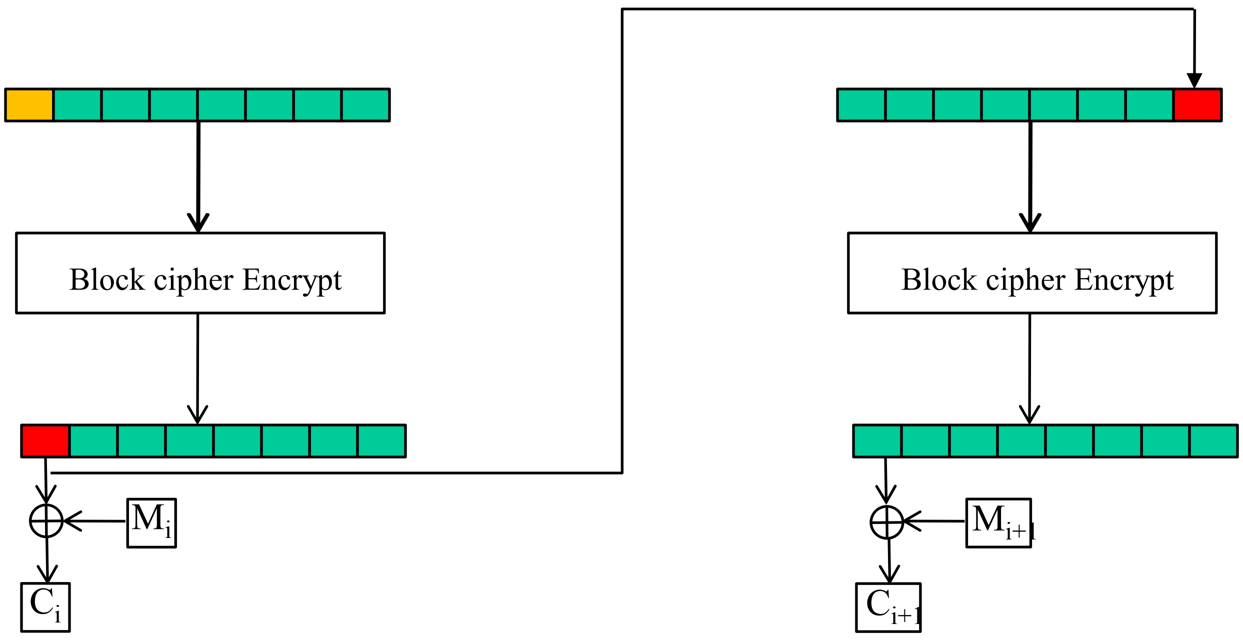
Output Feedback (OFB) mode encryption

OFB Decryption



Output Feedback (OFB) mode decryption

K-bit OFB mode



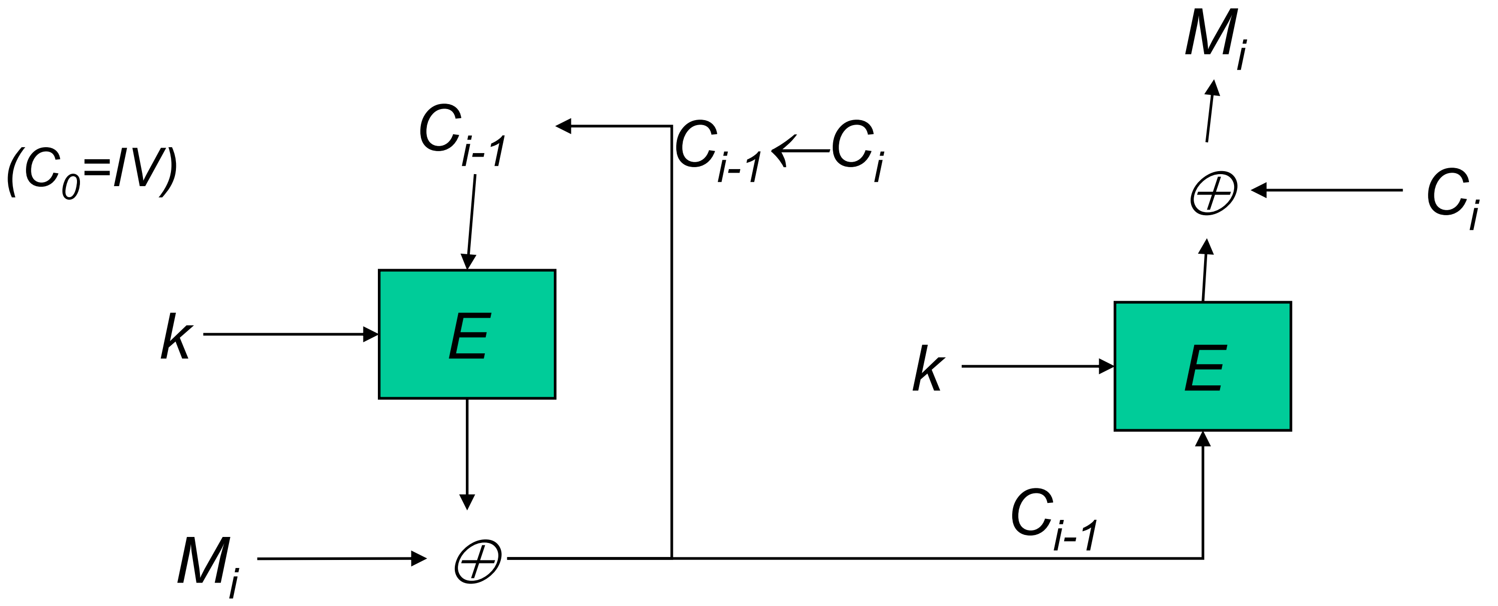
Example is OFB-8 (8 bits). The keystream input is encrypted. First 8 bits are used to encode 8 bits of plaintext. The keystream input at the next phase is the current input, left shifted by 8 bits, plus the first 8 bits of the encrypted previous phase input.

OFB issues

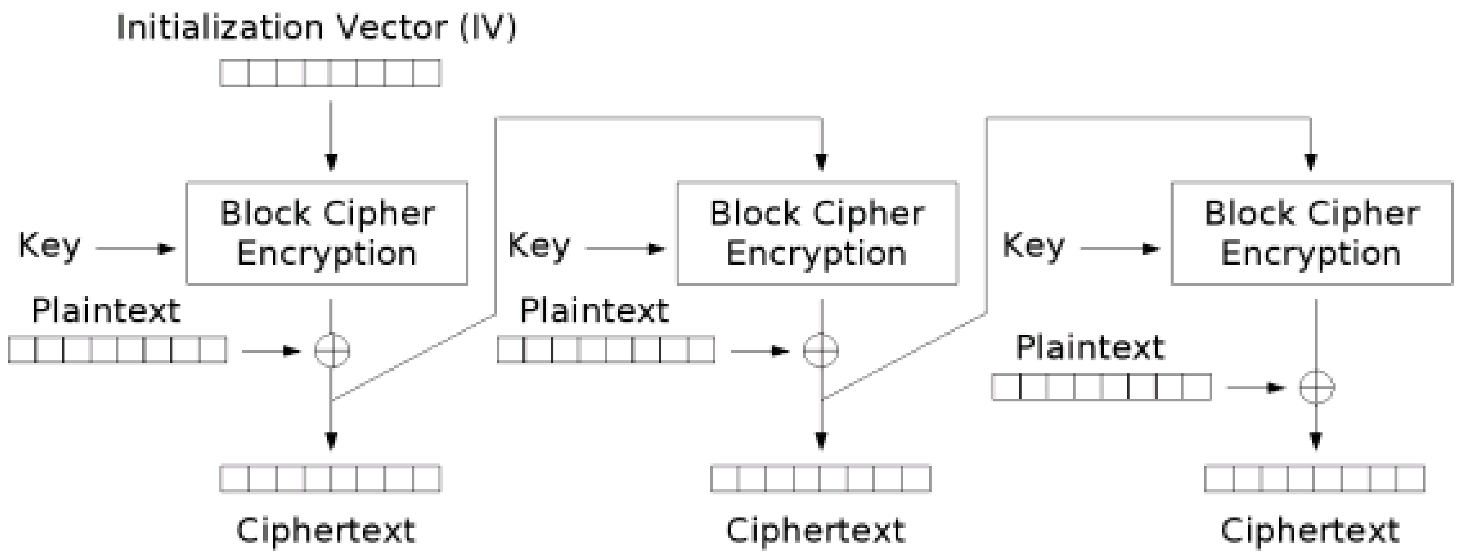
- IV repetition completely compromises security
- More parallelizable than CBC---the **key stream** is independent of the ciphertext, and can be pre-computed to enable random-access to plaintext.
- The operation of encryption and decryption must be synchronous---if a ciphertext “block” (8 bit, 16 bit, 64 bit) is missed, the two operations will not fall back in synch.

CFB

- The keystream (output of encryption) is xored into plaintext to obtain ciphertext. The ciphertext is the input for next chained encryption.
  - **$C_i = M_i \oplus E(C_{i-1})$ (encryption)**
  - **$M_i = C_i \oplus E(C_{i-1})$ (decryption)**

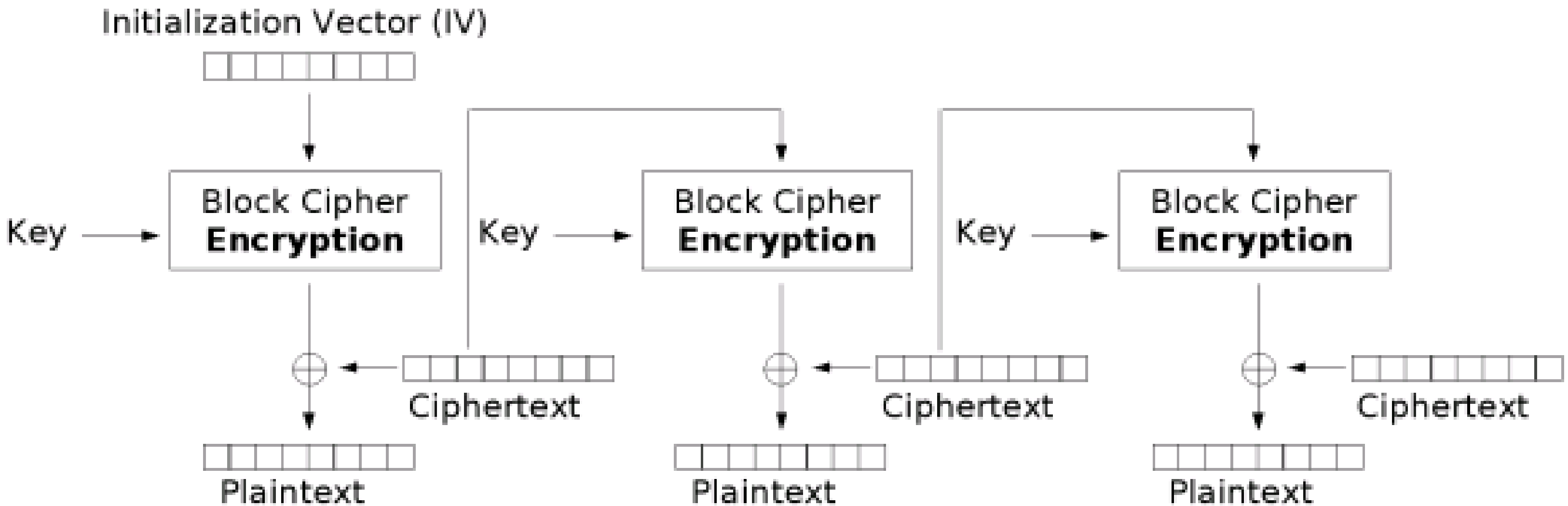


CFB Encryption



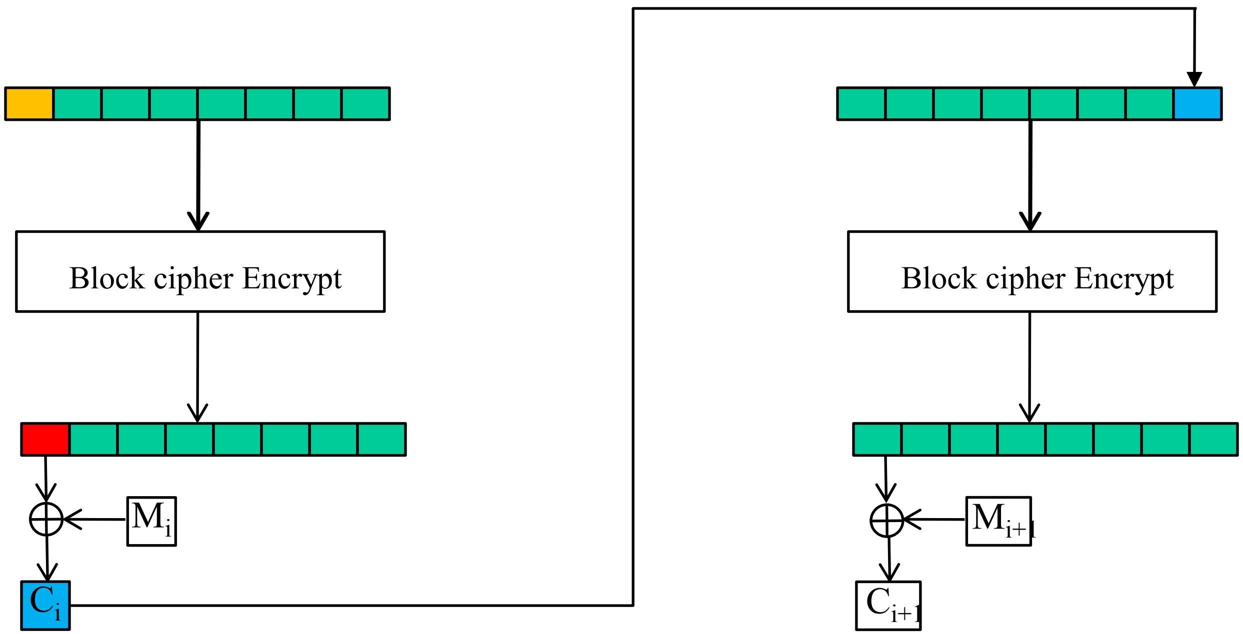
Cipher Feedback (CFB) mode encryption

CFB Decryption



Cipher Feedback (CFB) mode decryption

k-bit CFB mode

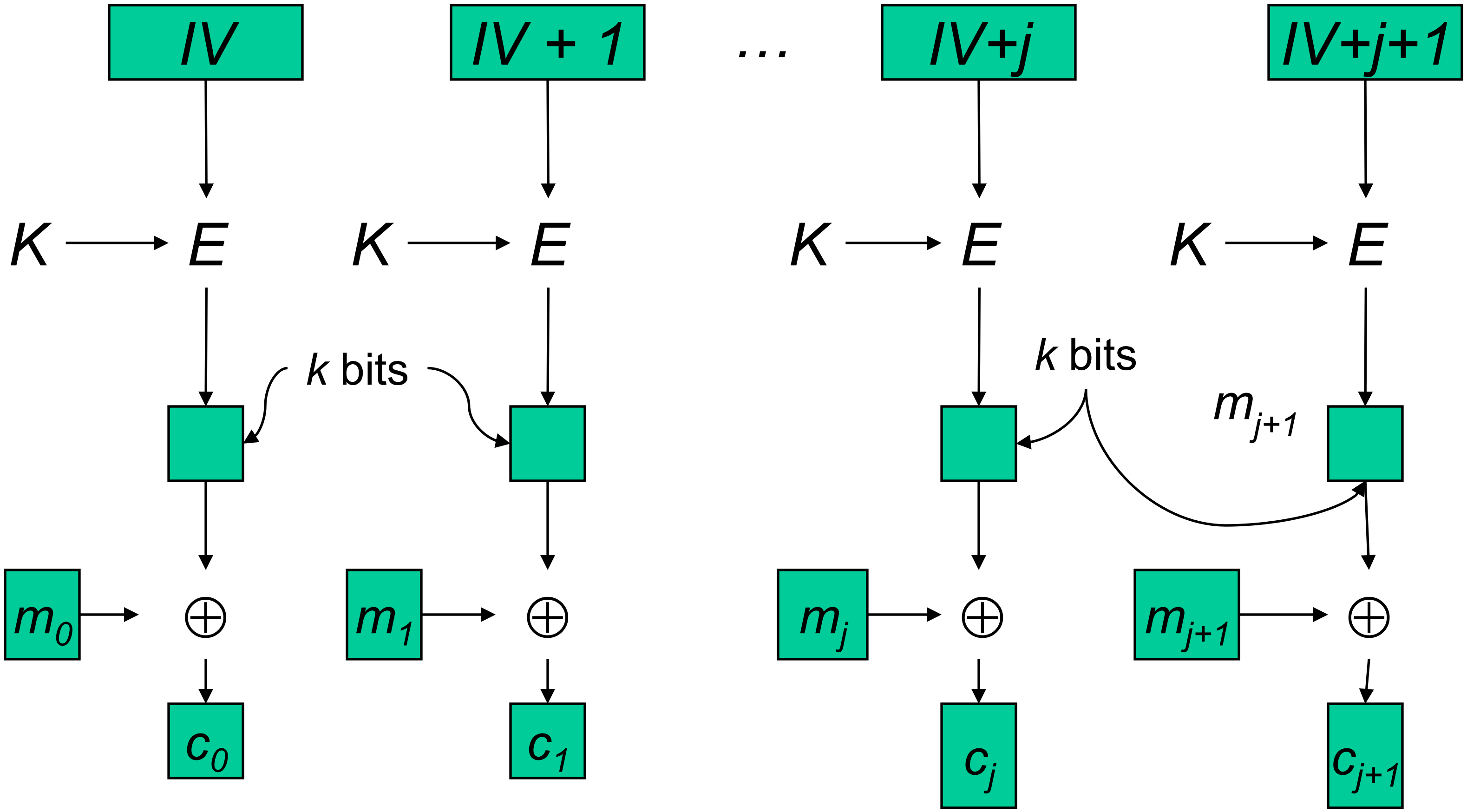


Example is CFB-8 (8 bits). The keystream input is encrypted. First 8 bits are used to encode 8 bits of plaintext. The keystream input at the next phase is the current keystream input, left shifted by 8 bits, plus the 8 bit previous cipher text.

CFB Issues

- The IV must be generated in a strongly pseudo-random fashion
- Not parallelizable (similar to CBC)
- Similar analysis of error propagation as CBC.
- Self-synchronizing
  - Under CFB-64, if a ciphertext block is missing, that block is lost and the following will decrypt incorrectly.
  - Analysis for CFB-8 and CFB-16 is similar.

Counter Mode



Reading Assignments

- Section 3.6
- Stream cipher A5/1
  - http://en.wikipedia.org/wiki/A5/1
- Wired Equivalent Privacy
  - http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy