CS 5173/4173 Computer Security

Topic 3.1 Secret Key Cryptography – Algorithms

Key Sizes

- A Key should be selected from a large potential set to prevent brute force attacks
  - If a key is of 3 bits, what are the possible keys?
    - 000, 001, 010, 011, 100, 101, 110,111
  - Given a pair of (plaintext, ciphertext), an attacker can do a brute force search to find the key
  - If a key is of n bits, how many possible keys does a brute force attacker need to search?

Key Sizes (Cont’d)

- Secret key sizes
  - 40 bits were considered adequate in 70’s
  - 56 bits used by DES were adequate in the 80’s
  - 128 bits are adequate for now
- If computers increase in power by 40% per year, need roughly 5 more key bits per decade to stay “sufficiently” hard to break

Notation

Notation	Meaning
$X \oplus Y$	Bit-wise exclusive-or of X and Y
X | Y	Concatenation of X and Y
K{*m*}	Message *m* encrypted with secret key K

Two Principles for Cipher Design

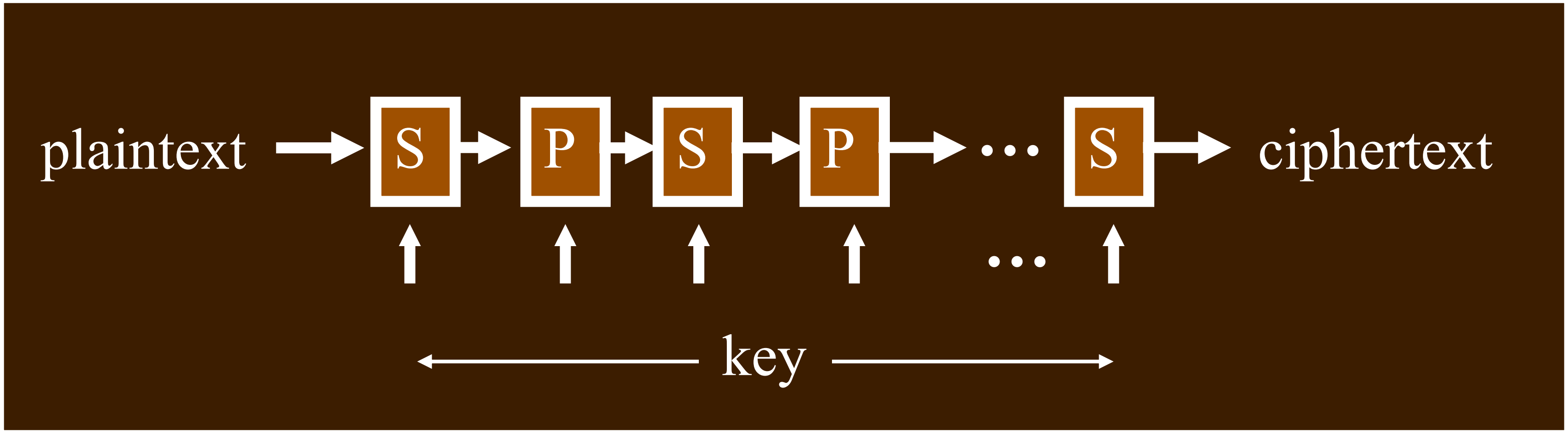
- Confusion:
  - Make the relationship between the <plaintext, key> input and the <ciphertext> output as complex (non-linear) as possible
- Diffusion:
  - Spread the influence of each input bit across many output bits

Exploiting the Principles

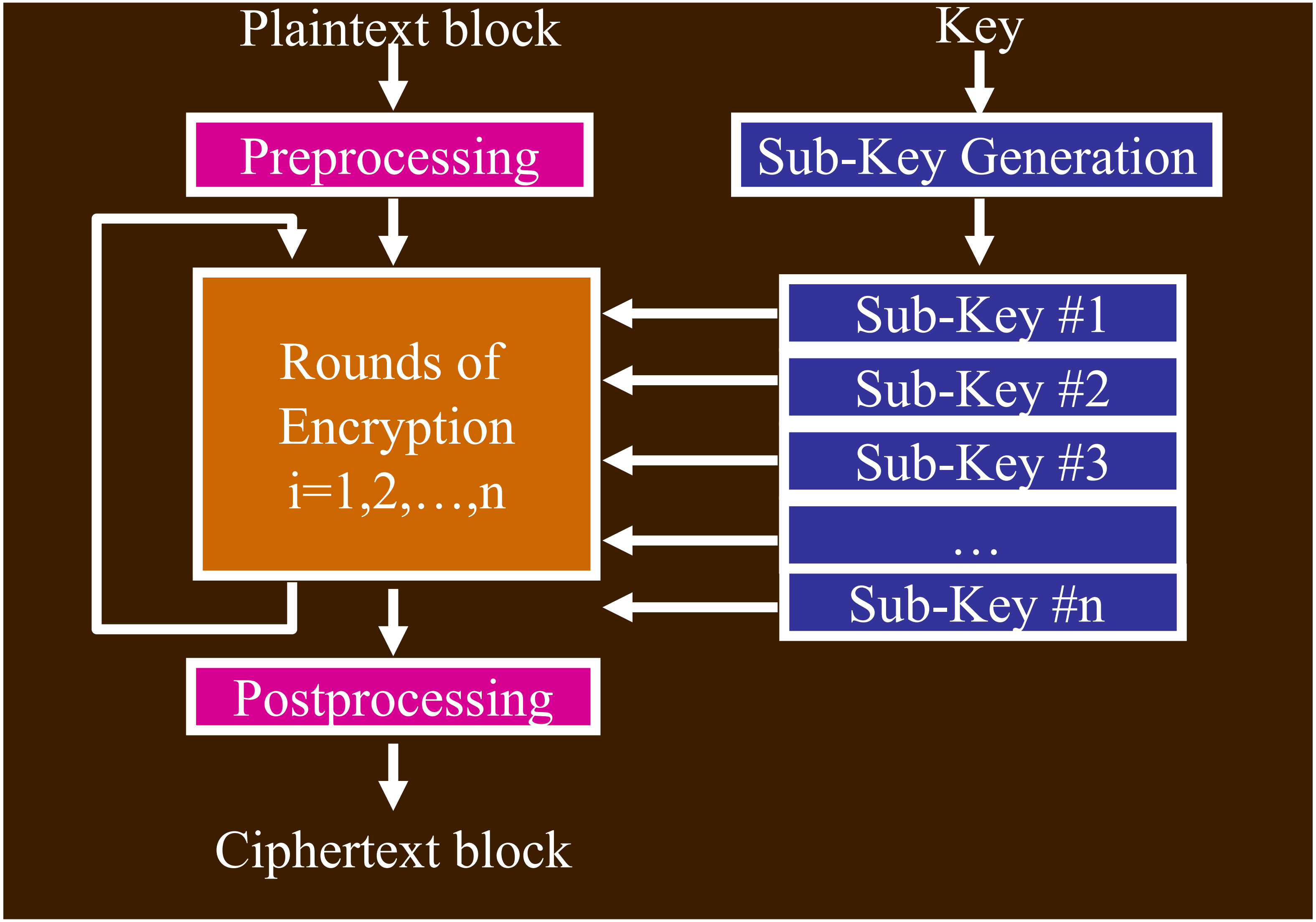
- Idea: use multiple, alternating permutations and substitutions, e.g.,
  - S→P→S→P→S→…
  - P→S→P→S→P→…
- Do they have to alternate? e.g.…
  - S→S→S→P→P→P→S→S→…?
  - Consecutive Ps or Ss do not improve security
- Confusion is mainly accomplished by substitutions
- Diffusion is mainly accomplished by permutations

Secret Key… (Cont'd)

- Basic technique used in secret key ciphers: multiple applications of alternating substitutions and permutations



Basic Form of Modern Block Ciphers



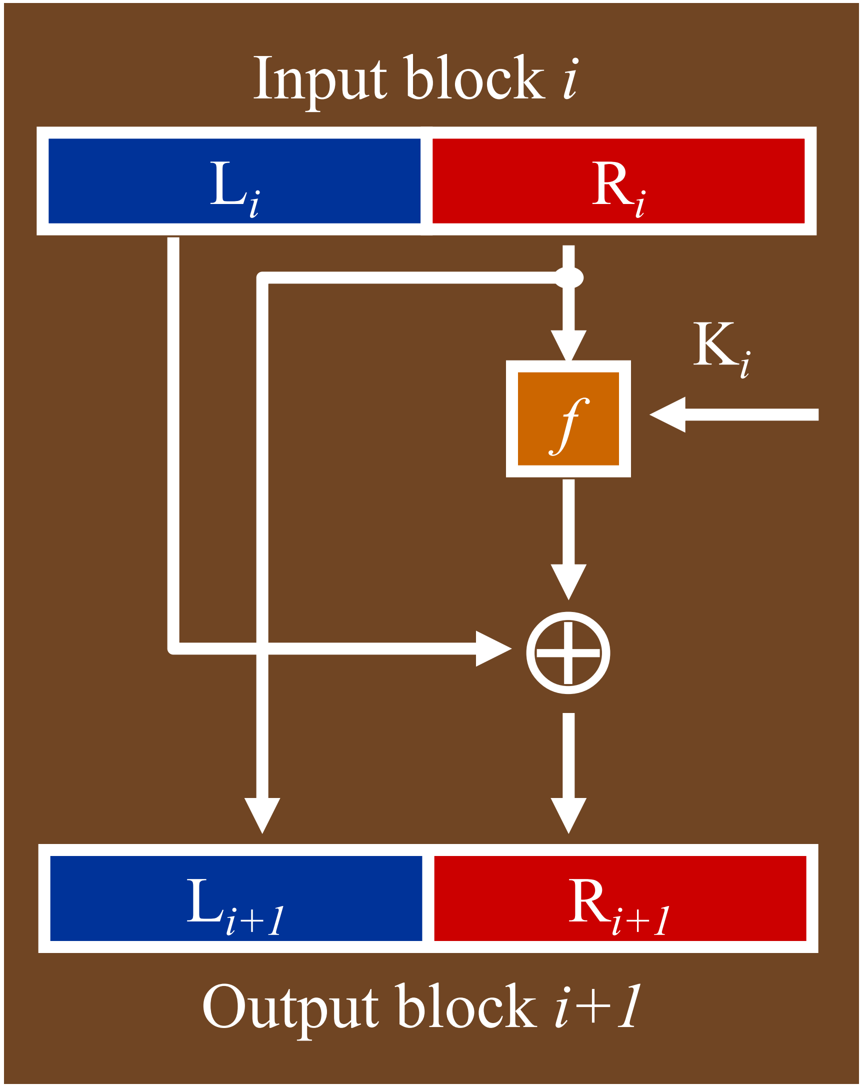
Feistel Ciphers

Feistel Ciphers

- Feistel Cipher has been a very influential "template" for designing a block cipher
- Major benefit: Encryption and decryption take the same time
  - they can be performed on the same hardware
- Examples: DES, RC5

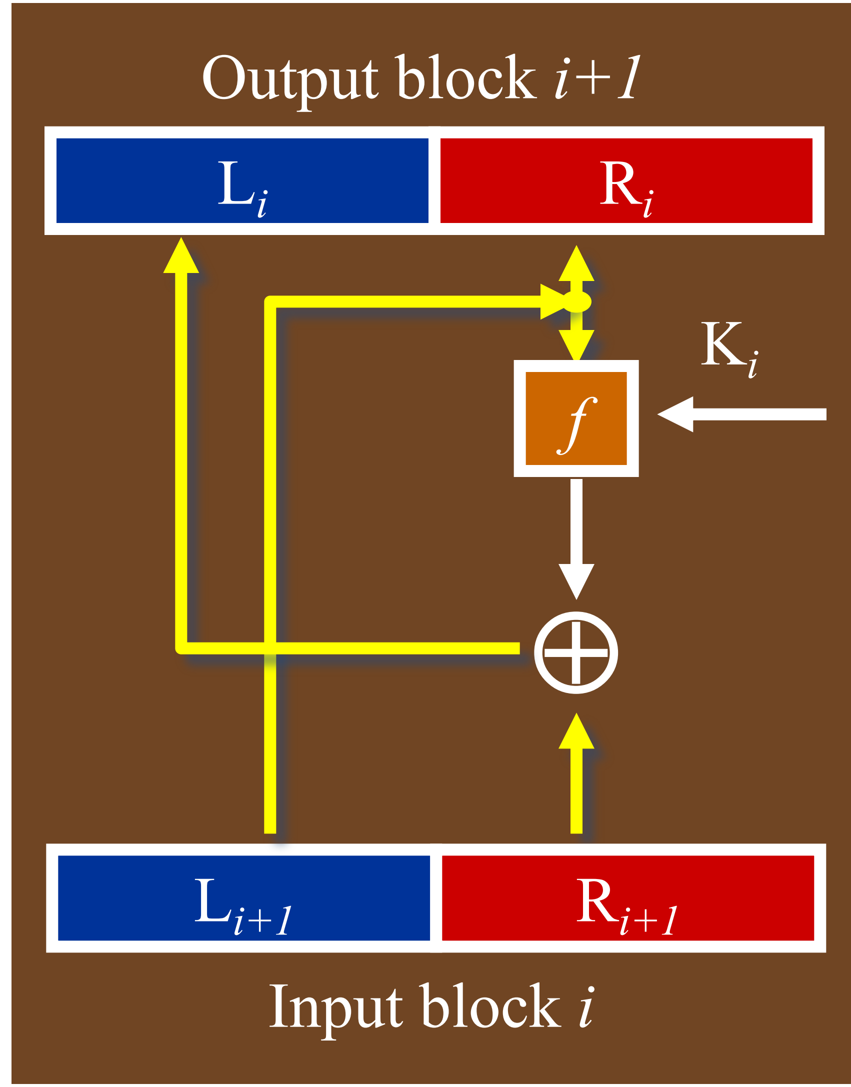
One “Round” of Feistel Encryption

1. Break input block *i* into left and right halves L_i and R_i
2. Copy R_i to create output half block L_{i+1}
3. Half block R_i and key K_i are “scrambled” by function *f*
4. XOR result with input half-block L_i to create output half-block R_{i+1}



One "Round" of Feistel Decryption

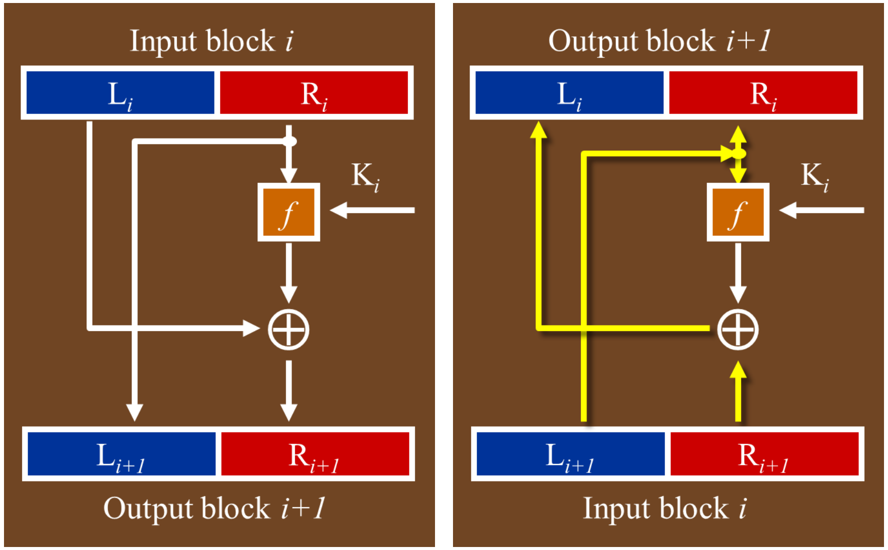
- Just reverse the arrows!
- Why?



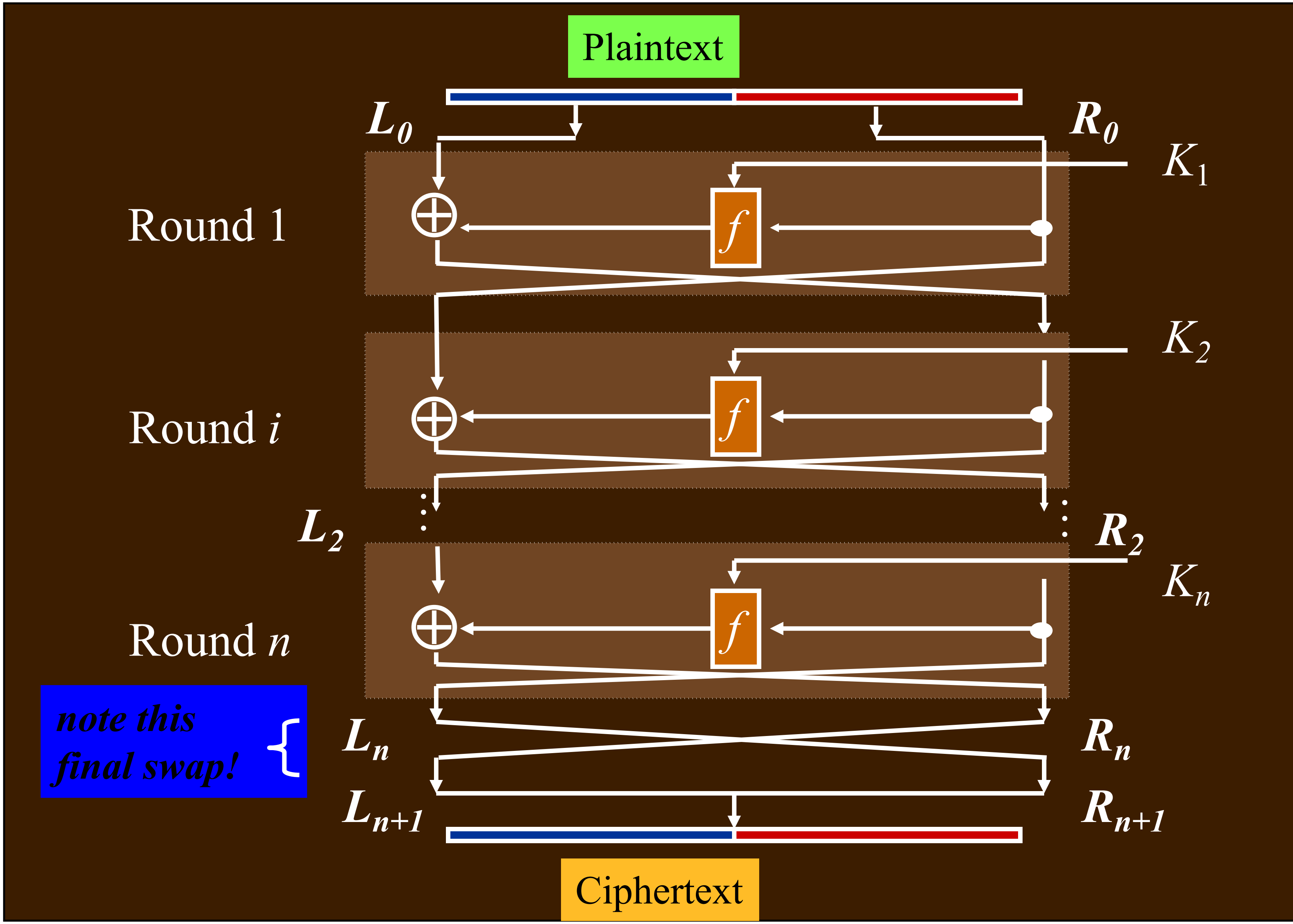
Feistel Cipher: Decryption (cont'd)

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

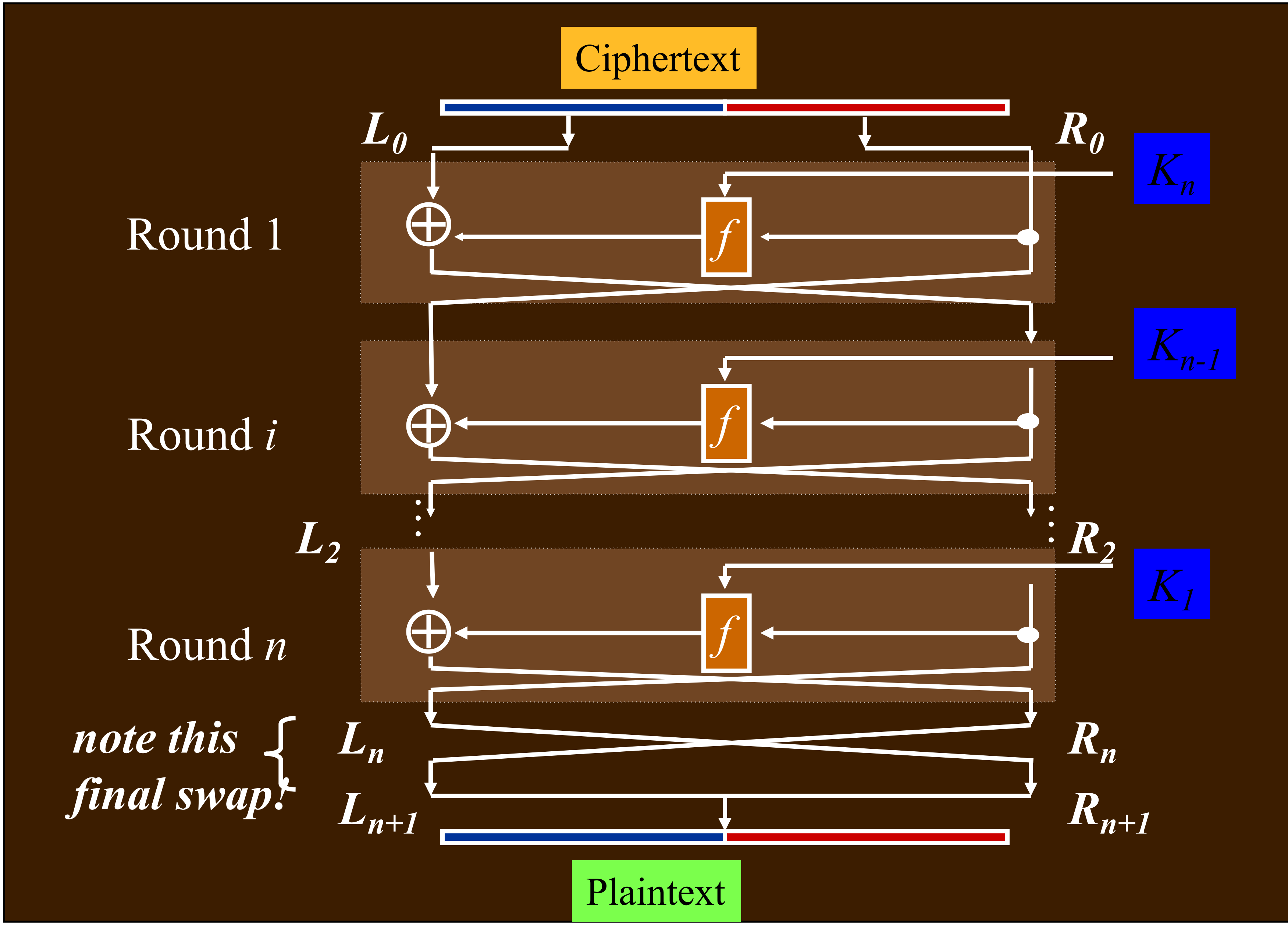
    $= L_i \oplus f(R_i, K_i) \oplus f(R_i, K_i) = L_i$



Complete Feistel Cipher: Encryption

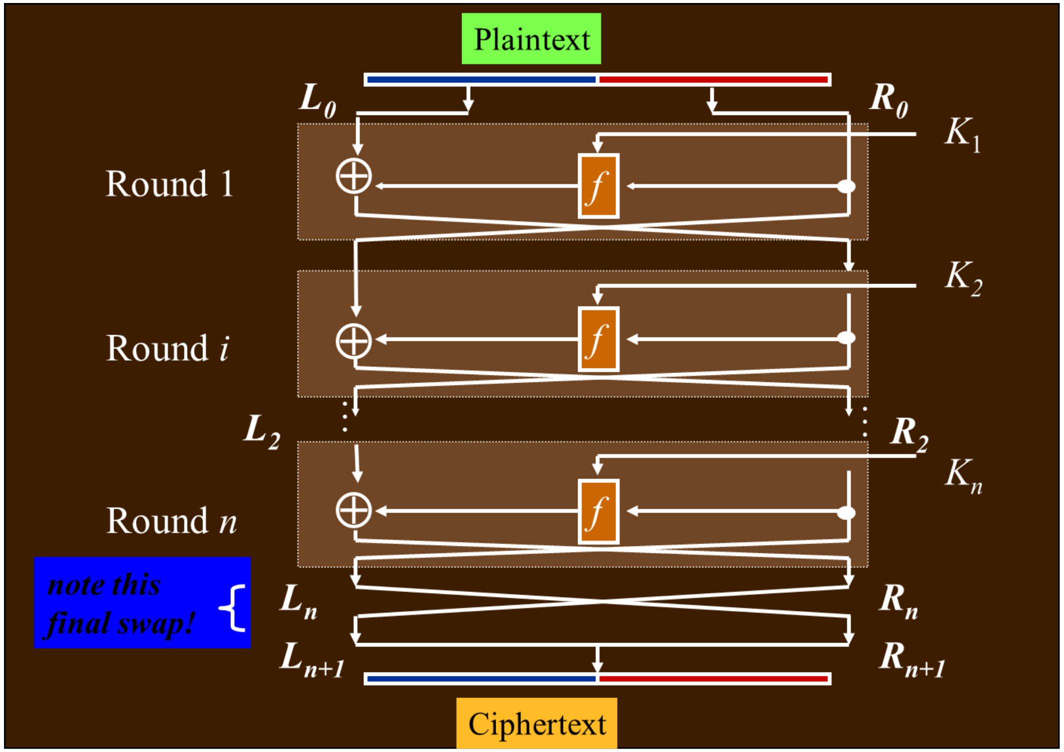


Feistel Cipher: Decryption

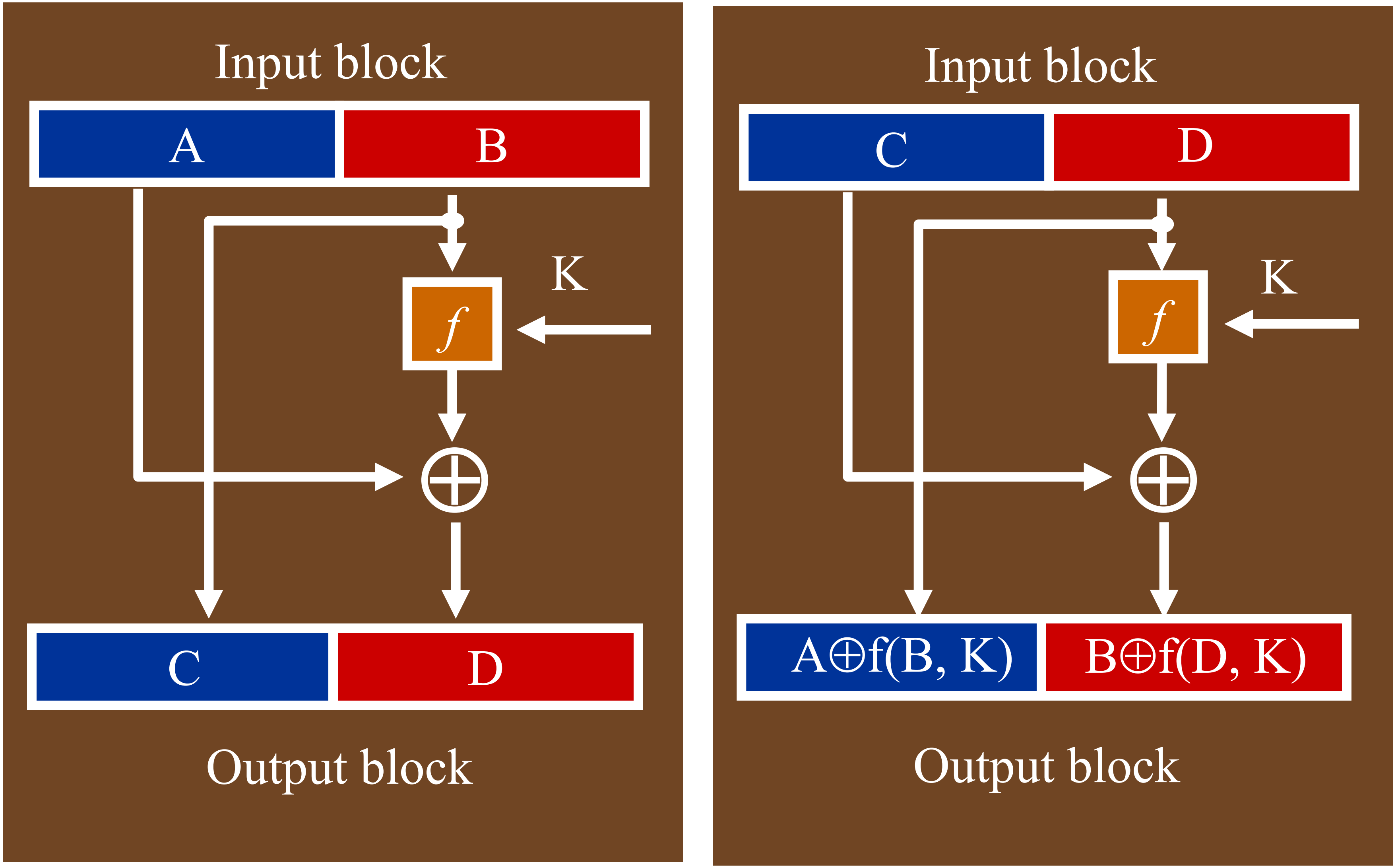


The Swap Operation

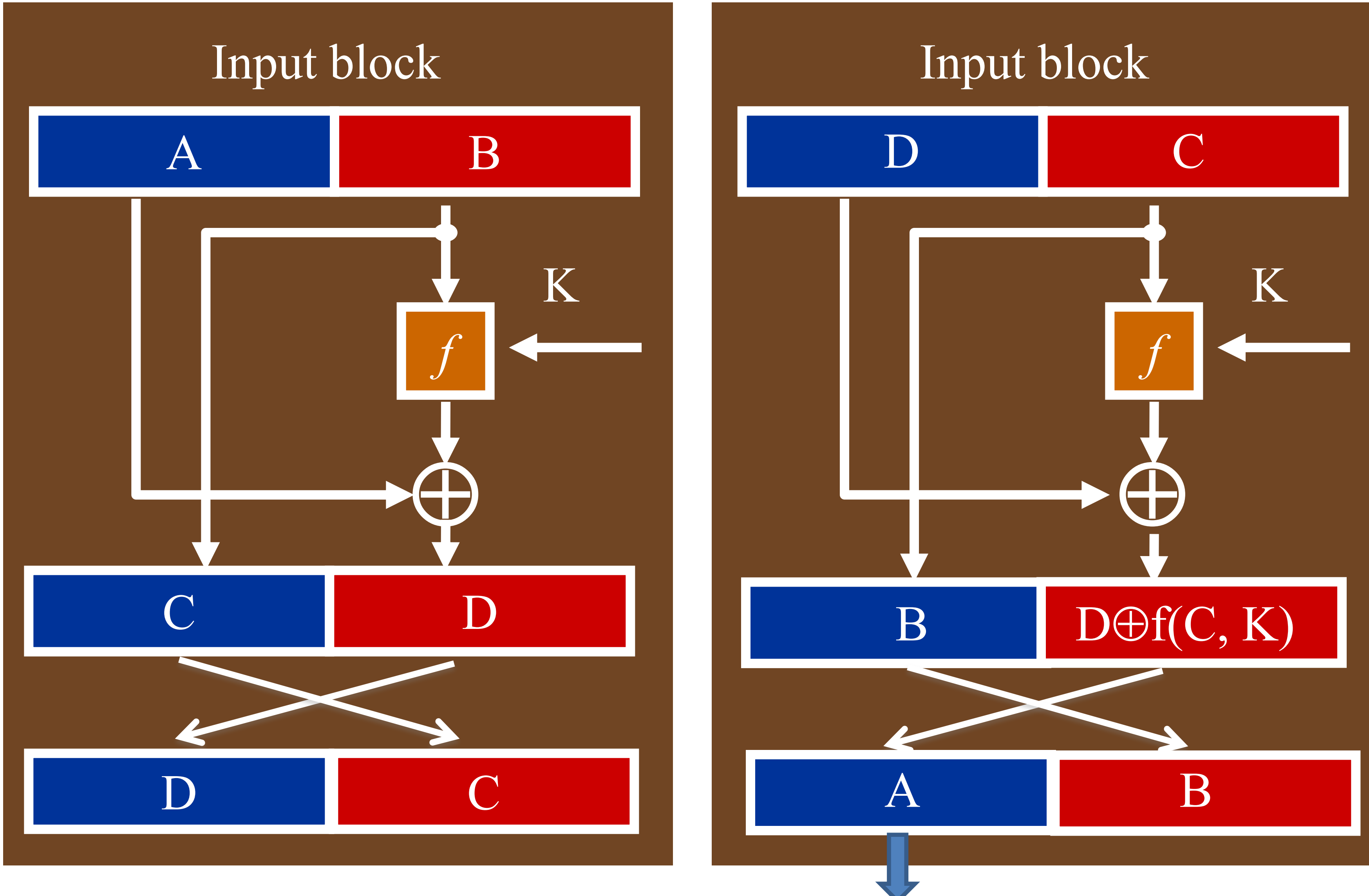
- For the Feistel cipher, why do we need the final swap step?



Without Swap



Without swap, we cannot decrypt to A and B!



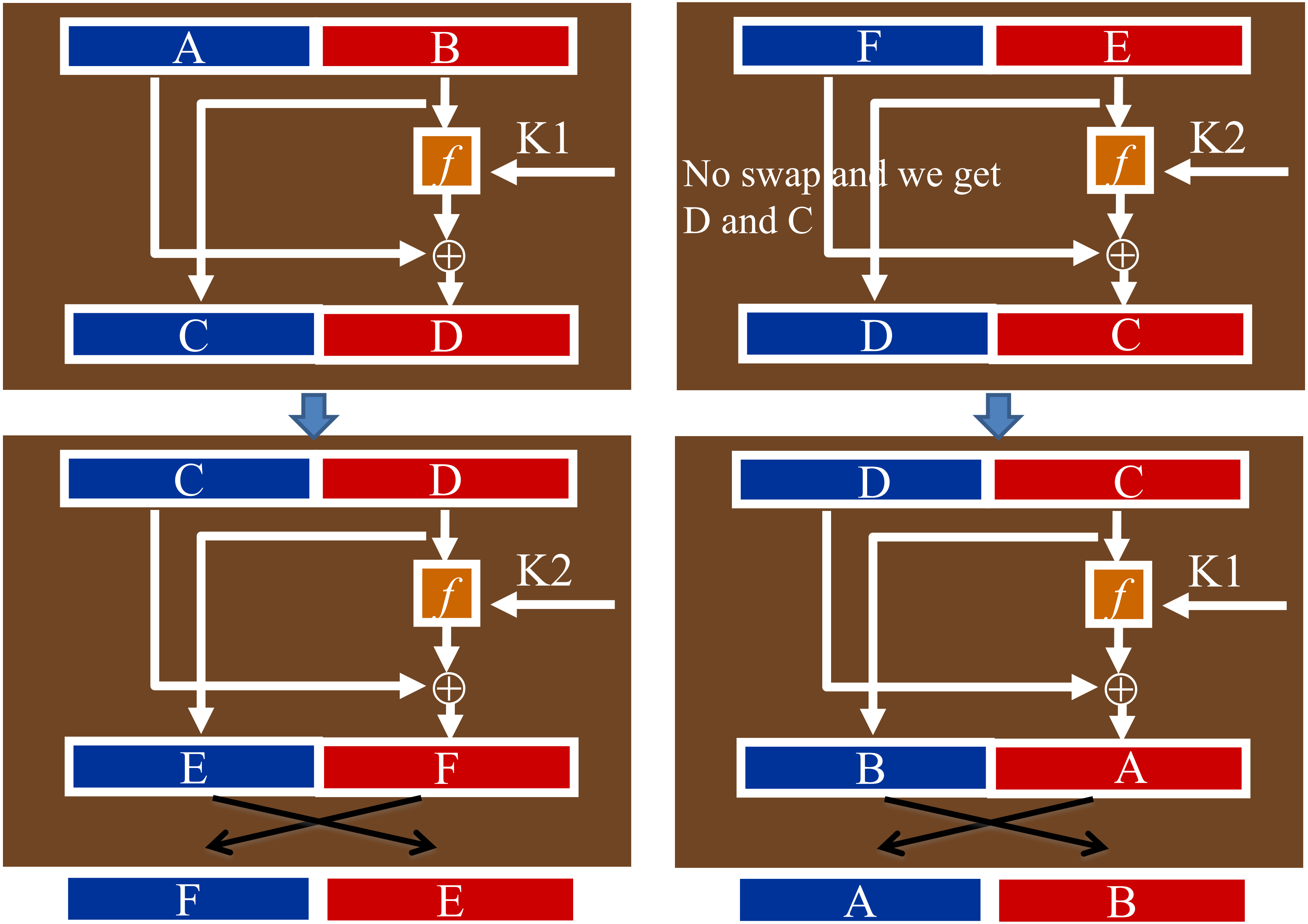
$= D \oplus f(C, K)$

$= A \oplus f(B, K) \oplus f(C, K)$

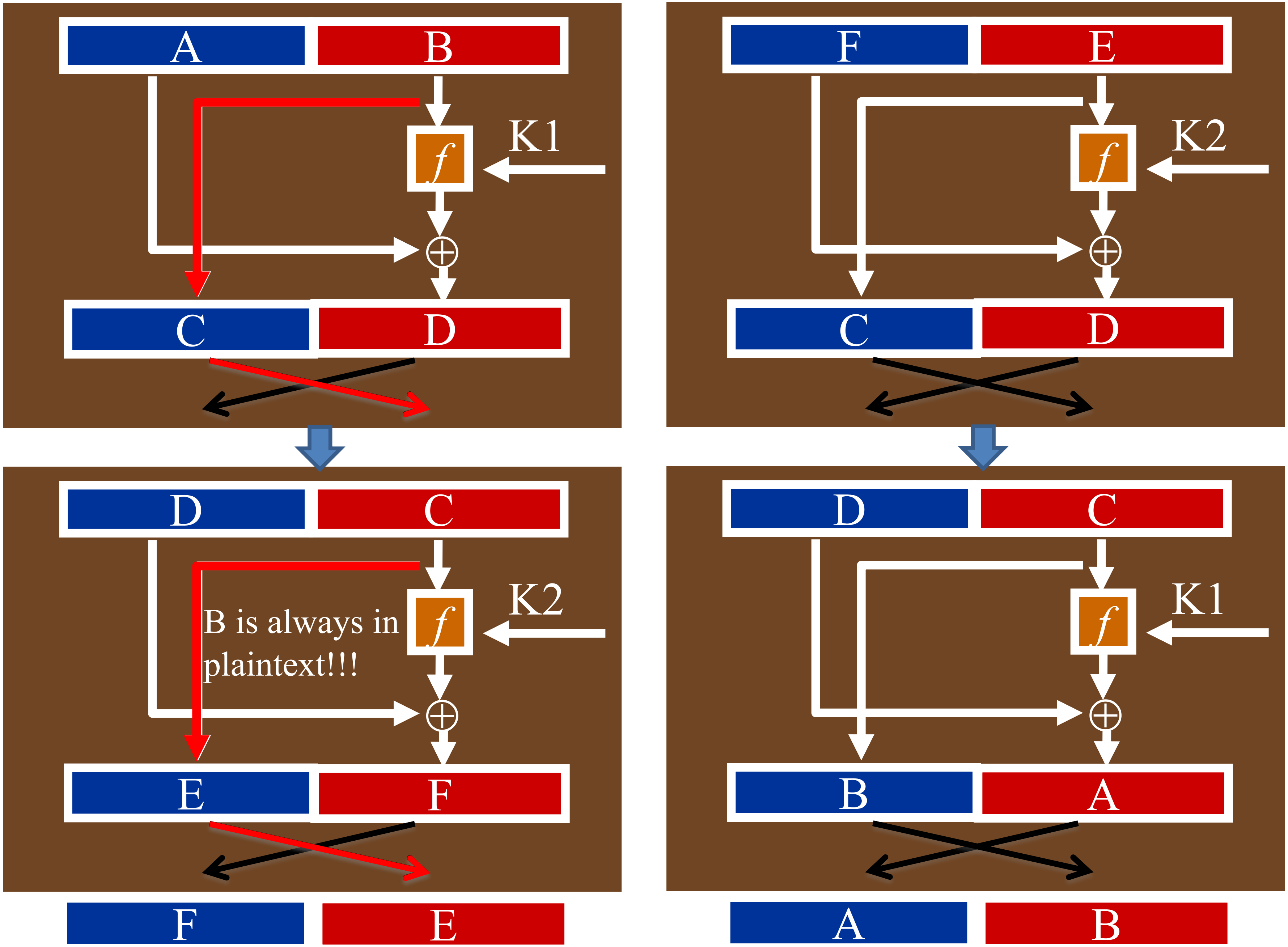
$= A \oplus f(C, K) \oplus f(C, K) = A$

With Swap

With Swap (two rounds)



What's wrong with this scheme?



Parameters of a Feistel Cipher

- Block size
- Key size
- Number of rounds
- Subkey generation algorithm
- "Scrambling" function *f*

Summary

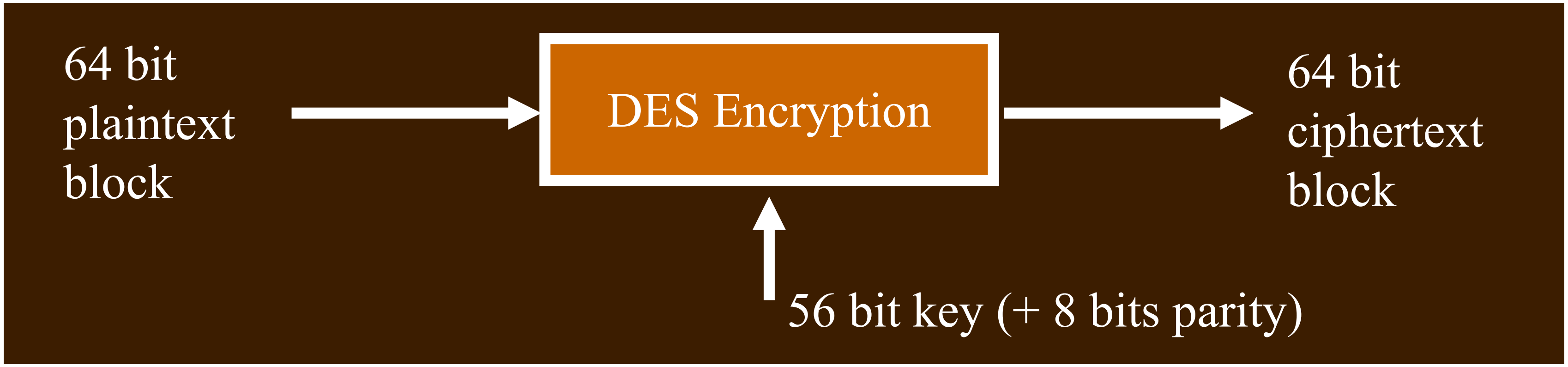
- Decryption is same as encryption, only reversing the order in which round keys are applied
  - Reversability of Feistel cipher derives from reversability of xor $\oplus$
- Function *f* can be anything
  - Hopefully something easy to compute
  - There is no need to invert *f*

DES (Data Encryption Standard)

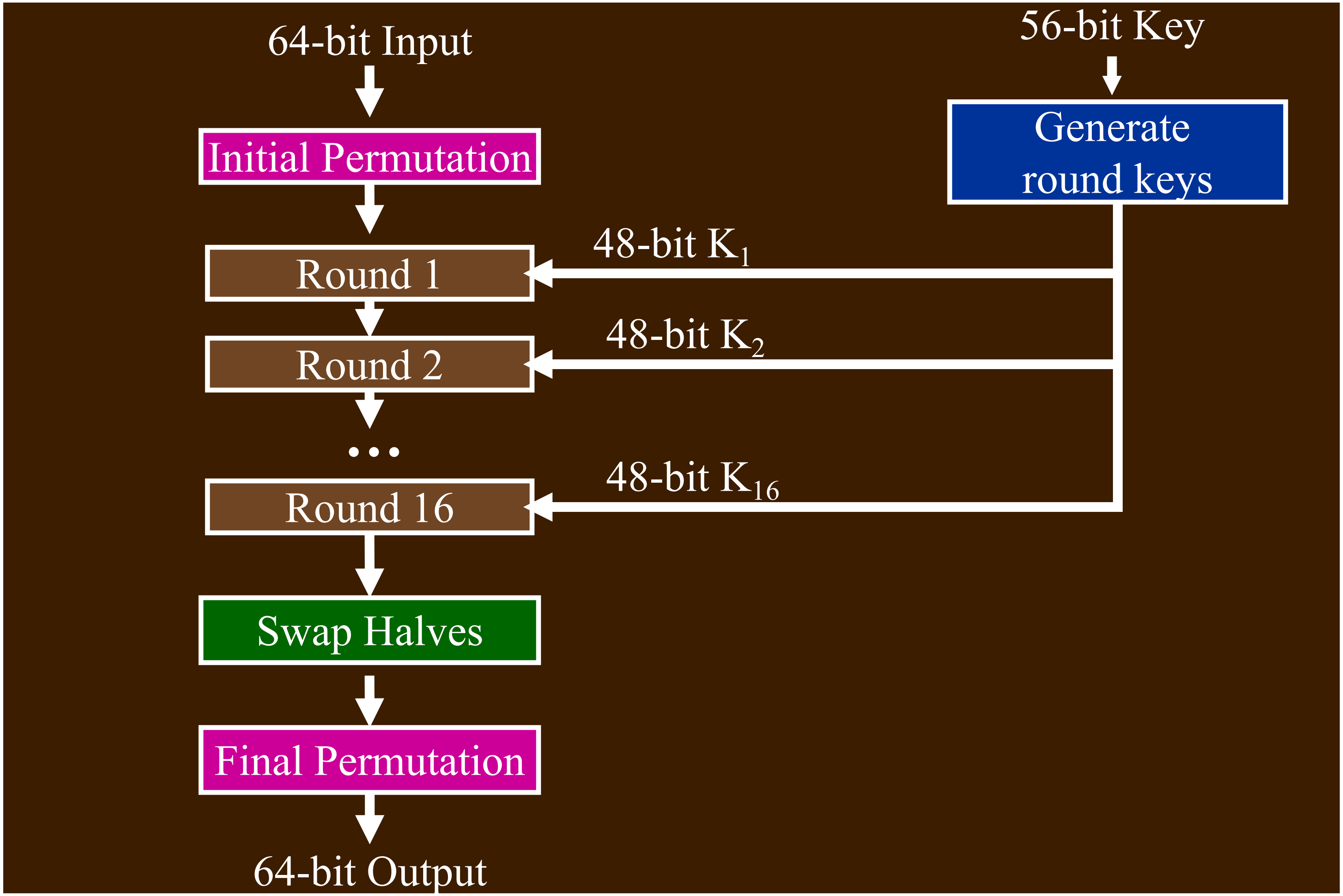
- Standardized in 1976 by NBS
  - proposed by IBM,
  - Feistel cipher
- Criteria (official)
  - provide high level of security
  - security must reside in key, not algorithm
  - not patented
  - efficient to implement in hardware
  - must be slow to execute in software

DES Basics

- Blocks: 64 bit plaintext input, 64 bit ciphertext output
- Rounds: 16
- Key: 64 bits
  - every 8^{th} bit is a parity bit, so really 56 bits long



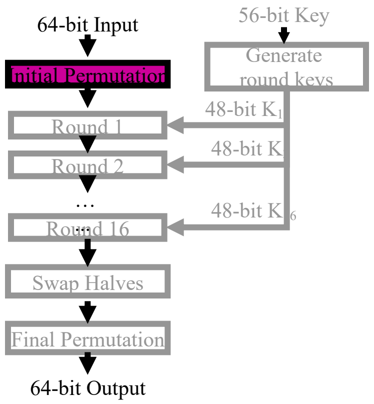
DES Top Level View



Initial and Final Permutations

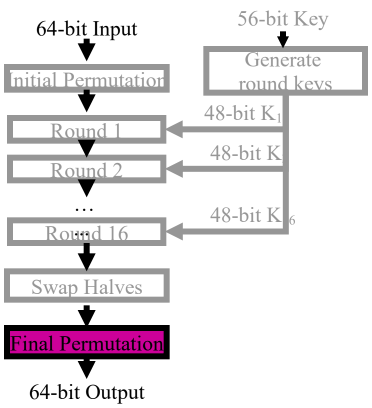
- Initial permutation given below
  - input bit #58→output bit #1, input bit #50→ output bit #2, …

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



Initial… (Cont'd)

- Final permutation is just inverse of initial permutation, i.e.,
  - input bit #1→ output bit #58
  - input bit #2→ output bit #50
  - …

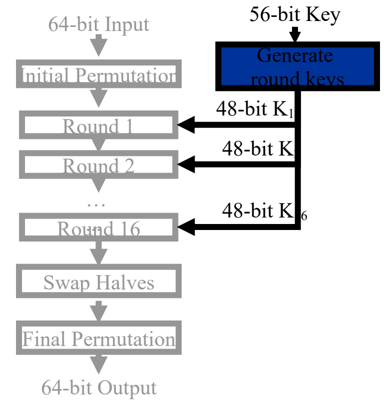


Initial… (Cont'd)

- Note #1: Initial Permutation is fully specified (independent of key)
  - therefore, does not improve security!
  - why needed?
- Note #2: Why is final Permutation needed?
  - to make this a Feistel cipher
    - i.e., the decryption is the reverse of encryption

Key Generation: First Permutation

- First step: throw out 8 parity bits, then permute resulting 56 bits



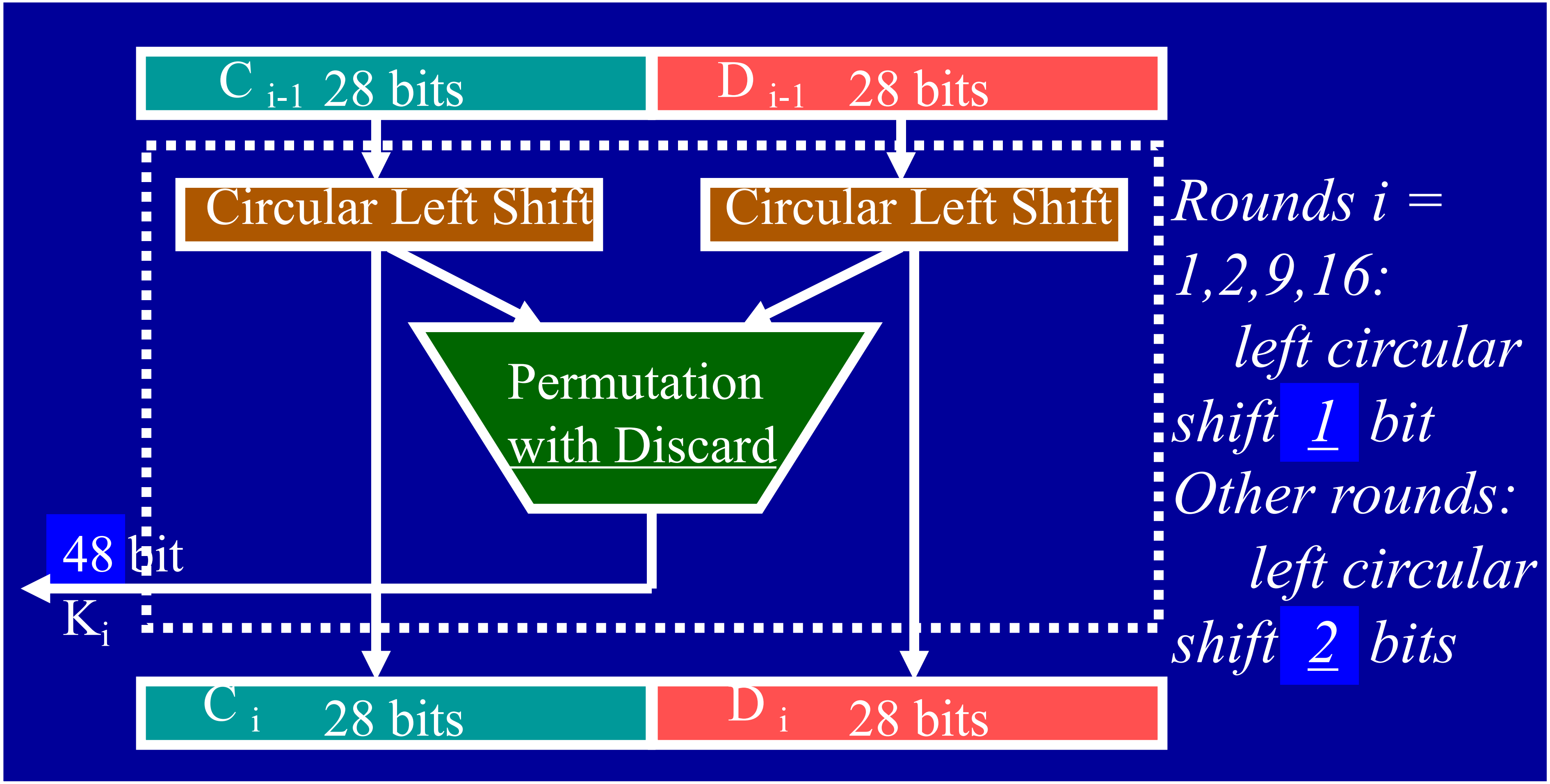
7 columns

8 rows

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

Parity bits left out: 8,16,24,…

KeyGen: Processing Per Round



KeyGen: Permutation with Discard

- 28 bits → 24 bits, each half of key

Left half of K_i = permutation of C_i

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2

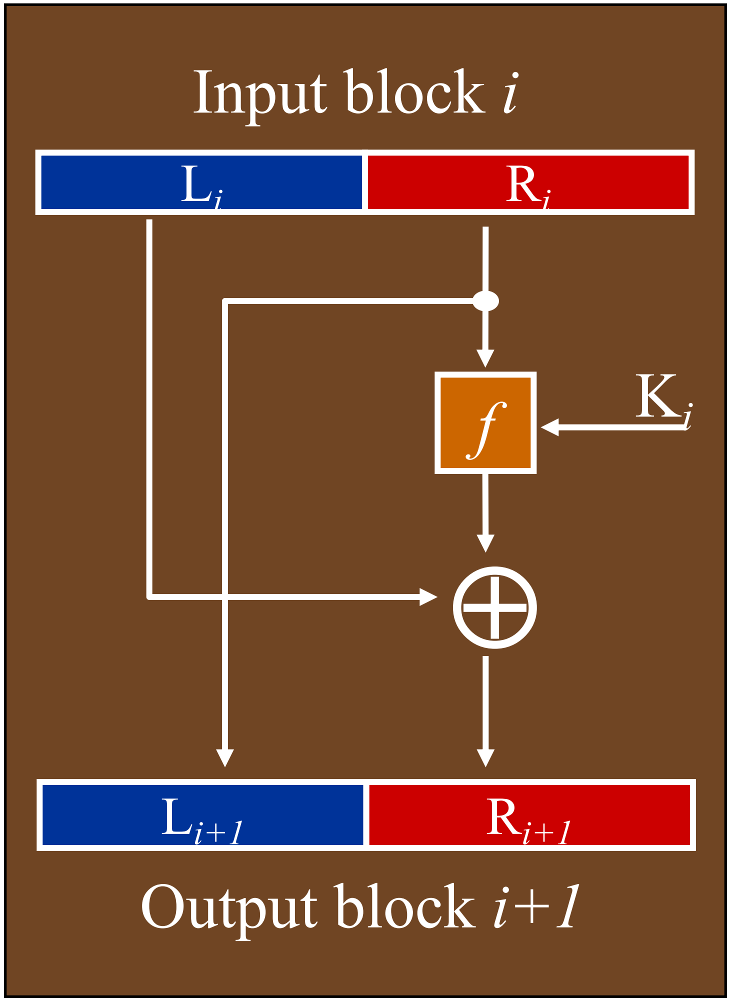
Bits dicarded: 9,18,22,25

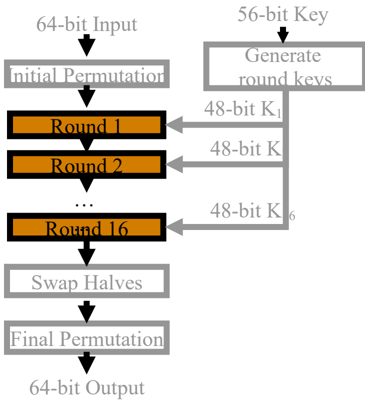
Right half of K_i = permutation of D_i

41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

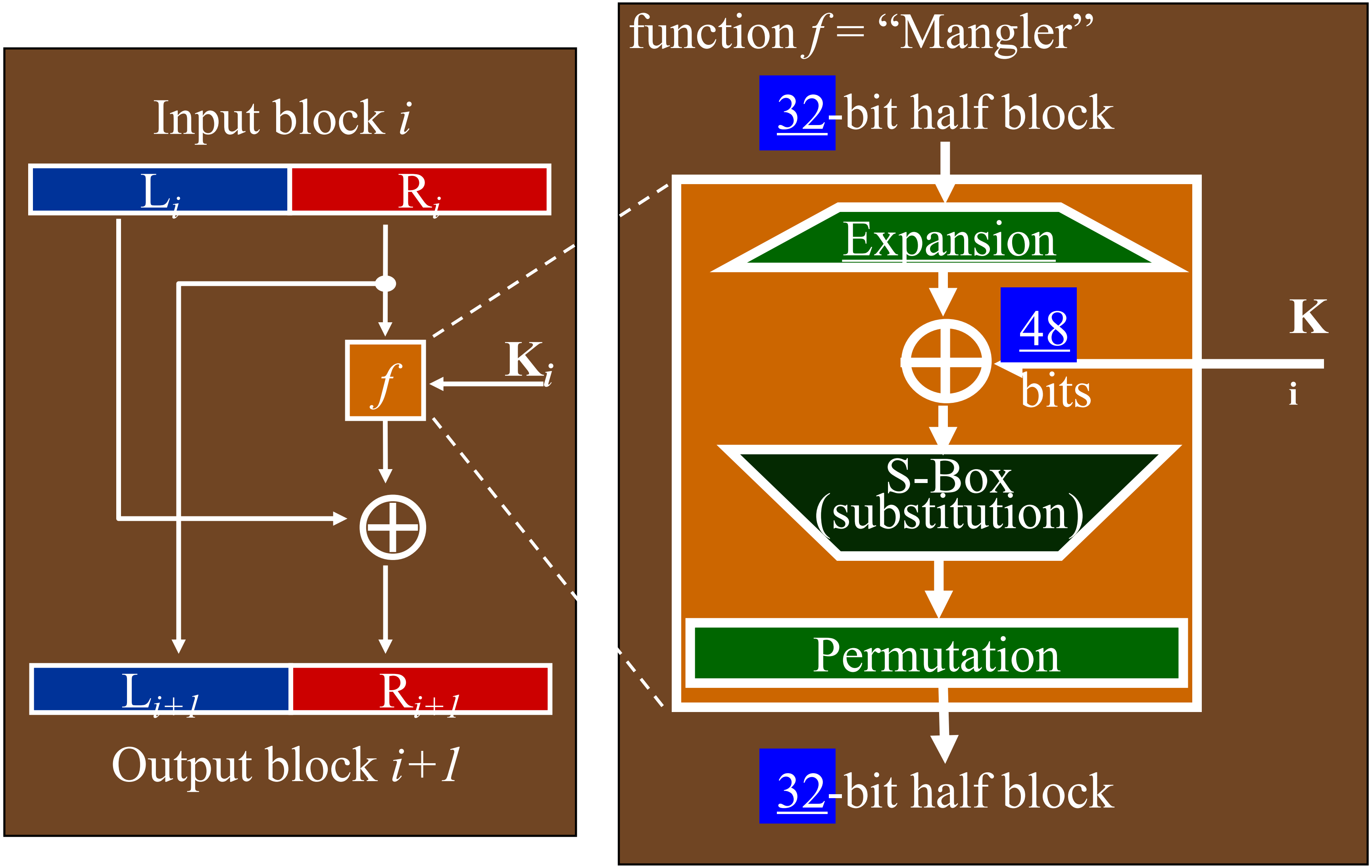
Bits discarded: 35,38,43,54

One DES (Feistel) Round





DES Round: *f* (Mangler) Function



f: Expansion Function

- 32 bits ➔ 48 bits

these bits are repeated

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

f: S_1 (Substitution)

Each row and column contain different numbers

I1/I6 ↓ \ I2/I3/I4/I5 →	0	1	2	**3**	4	5	6	…	F
0	E	4	D	1	2	F	B		
1	0	F	7	4	E	2	D		
2	4	1	E	**8**	D	6	2		
3	F	C	8	2	4	9	1		

Example: input= 100110, output= 1000

input = 101101, out put =?

f: Permutation

- 32bits → 32bits

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

DES Implementation

- That's it!
- Operations
  - Permutation
  - Swapping halves
  - Substitution (S-box, table lookup)
  - Bit discard
  - Bit replication
  - Circular shift
  - XOR
- Hard to implement? HW: No, SW: Yes

Principles for S-Box Design

- S-box is the only non-linear part of DES
- Each row in the S-Box table should be a permutation of the possible output values
- Output of one S-box should affect other S-boxes in the following round

Desirable Property: Avalanche Effect

- Roughly: a small change in either the plaintext or the key should produce a big change in the ciphertext
- Better: any output bit should be inverted (flipped) with probability 0.5 if any input bit is changed
  - What is the expected number of different bits between two length-n random bit streams?
    - 0.5n (Bernoulli distribution)
- f function
  - must be difficult to un-scramble
  - should achieve avalanche effect
  - output bits should be uncorrelated

DES Avalanche Effect: Example

- 2 plaintexts with 1 bit difference:

  0x0000000000000000 and

  0x8000000000000000

  encrypted using the same key:

  0x016B24621C181C32

- Resulting ciphertexts differ in 34 bits (out of 64)
- Similar results when keys differ by 1 bit

Example (cont'd)

- An experiment: number of rounds vs. number of bits difference

Round #	0	1	2	3	4	5	6	7	8
Bits changed	**1**	**6**	**21**	**35**	**39**	**34**	**32**	**31**	**29**

9	10	11	12	13	14	15	16
42	**44**	**32**	**30**	**30**	**26**	**29**	**34**

DES: Keys to Avoid Using

- "Weak keys": These are keys which, after the first key permutation, are:
  - 28 0's followed by 28 0's
  - 28 1's followed by 28 1's
  - 28 0's followed by 28 1's
  - 28 1's followed by 28 0's
- Property of weak keys
  - Easy clue for brute force attacks.
  - Sixteen identical subkeys.
  - Encrypting twice produces the original plaintext.

Weak keys

- Alternating ones + zeros (0x0101010101010101)
- Alternating 'F' + 'E' (0xFEFEFEFEFEFEFEFE)
- 0xE0E0E0E0F1F1F1F1
- 0x1F1F1F1F0E0E0E0E
- DES *weak keys* produce sixteen identical subkeys

More Keys to Avoid!

- "Semi-weak keys": These are keys which, after the first key permutation, are:
  1. 28 0's followed by alternating 0's and 1's
  2. 28 0's followed by alternating 1's and 0's

  …

  12. Alternating 1's and 0's followed by alternating 1's and 0's
- Property of semi-weak keys
  – For a semi-weak key pair (K_1, K_2), $K_1\{K_2\{m\}\} = m$ and $K_2\{K_1\{m\}\} = m$

Semi-weak key pairs

- 0x011F011F010E010E and 0x1F011F010E010E01
- 0x01E001E001F101F1 and 0xE001E001F101F101
- 0x01FE01FE01FE01FE and 0xFE01FE01FE01FE01
- 0x1FE01FE00EF10EF1 and 0xE01FE01FF10EF10E
- 0x1FFE1FFE0EFE0EFE and 0xFE1FFE1FFE0EFE0E
- 0xE0FEE0FEF1FEF1FE and 0xFEE0FEE0FEF1FEF1
- $K_1\{K_2\{m\}\} = m$ and $K_2\{K_1\{m\}\} = m$

DES Key Size

- 56 bits is currently too small to resist brute force attacks using readily-available hardware
- Ten years ago it took $250,000 to build a machine that could crack DES in a few hours
- Now?

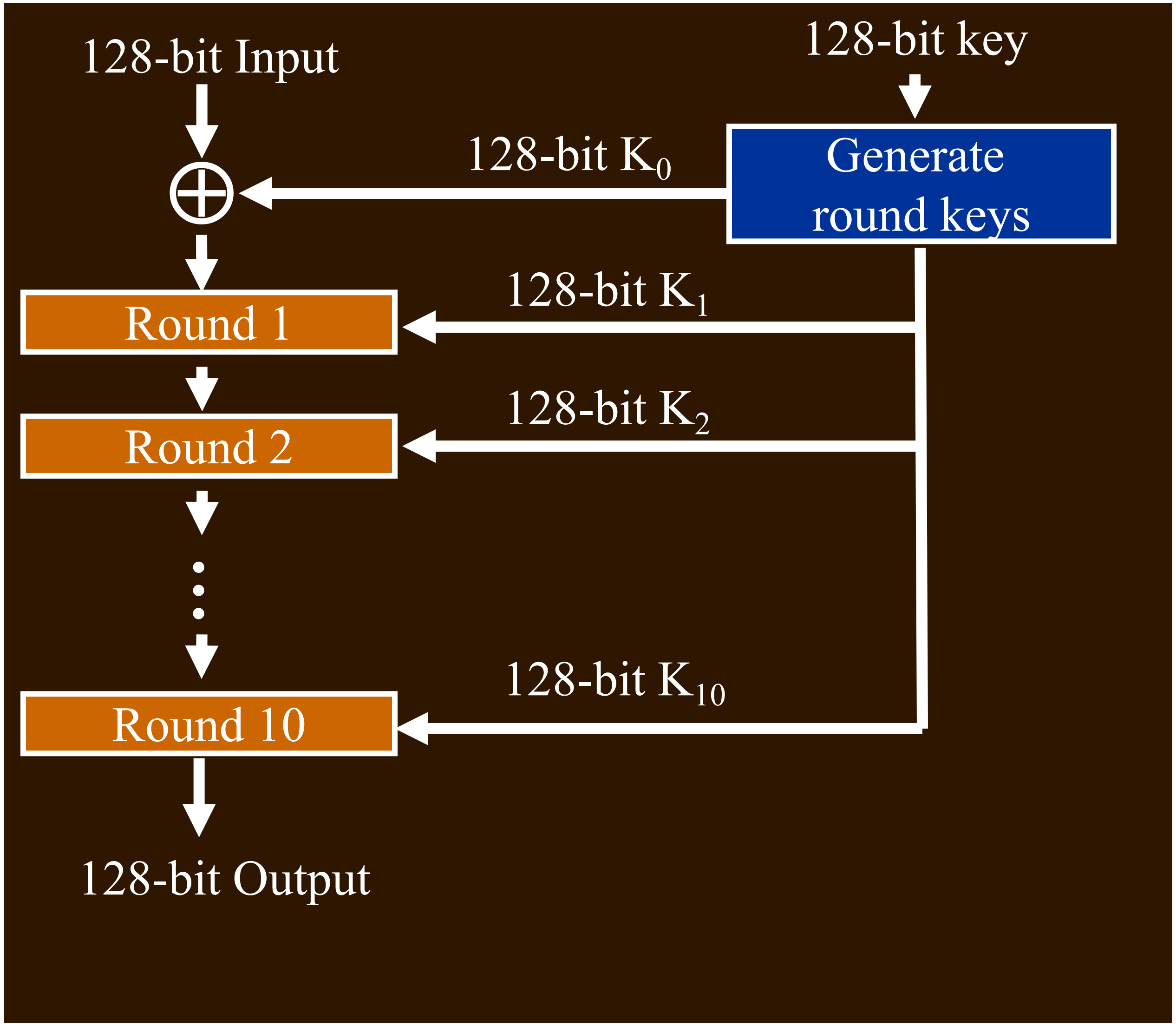
Cryptanalysis of DES

- Differential cryptanalysis exploits differences between encryptions of two different plaintext blocks
  - provides insight into possible key values
- Linear cryptanalysis requires known plaintext / ciphertext pairs, analyzes relationships to discover key value
- No attacks on DES so far are significantly better than brute force attacks, for comparable cost

Advanced Encryption Standard (AES)

- Selected from an open competition, organized by NSA
  - winner: Rijndael algorithm, standardized as AES
- Some similarities to DES (rounds, round keys, alternate permutation+substitution)
  - but not a Feistel cipher
- Block size = 128, or 192, or 256 bits
- Key sizes = 128 (10 rounds), 192 (12 rounds), or 256 (14 rounds)

AES-128 Overview



Does AES Have Avalanche Property?

Examples of AES Encryption

Input String	Key	Output String (HEX)
ABCDEFGHIJKLMNOP	11223344556677889900AABBCCDDEEFF	BC4784A37D6F46452656B993D53393F5
ABCDEFGHIJKLMNOP	01223344556677889900AABBCCDDEEFF	855866490543FDF6504FC84088FEDCA0
ABCDEFFHIJKLMNOP	11223344556677889900AABBCCDDEEFF	372CCA446C0D391C4381392344630EE1

Input String(HEX)	Key	Output String (HEX)
00000000000000000000000000000000	00000000000000000000000000000000	66E94BD4EF8A2C3B884CFA59CA342B2E
00000000000000000000000000000000	00000000000000000000000000000001	0545AAD56DA2A97C3663D1432A3D1C84
00000000000000000000000000000001	00000000000000000000000000000001	A17E9F69E4F25A8B8620B4AF78EEFD6F

AES Assessment

- No known successful attacks on full AES
  - Best attacks work on 7–9 rounds
- For brute force attacks, AES-128 needs much more effort than DES

Attacks on AES

- Differential Cryptanalysis: based on how differences in inputs correlate with differences in outputs
  - greatly reduced due to high number of rounds
- Linear Cryptanalysis: based on correlations between input and output

Attacks on AES (Cont'd)

- Side Channel Attacks
- Timing Attacks: measure the time it takes to do operations
  - some operations, with some operands, are much faster than other operations, with other operand values
  - provides clues about what internal operations are being performed, and what internal data values are being produced

Attacks on AES (Cont’d)

- Side Channel Attacks
- Power Attacks: measures power to do operations
  - changing one bit requires considerably less power than changing many bits in a byte