

CSCE 465 Computer & Network Security

Instructor: Abner Mendoza

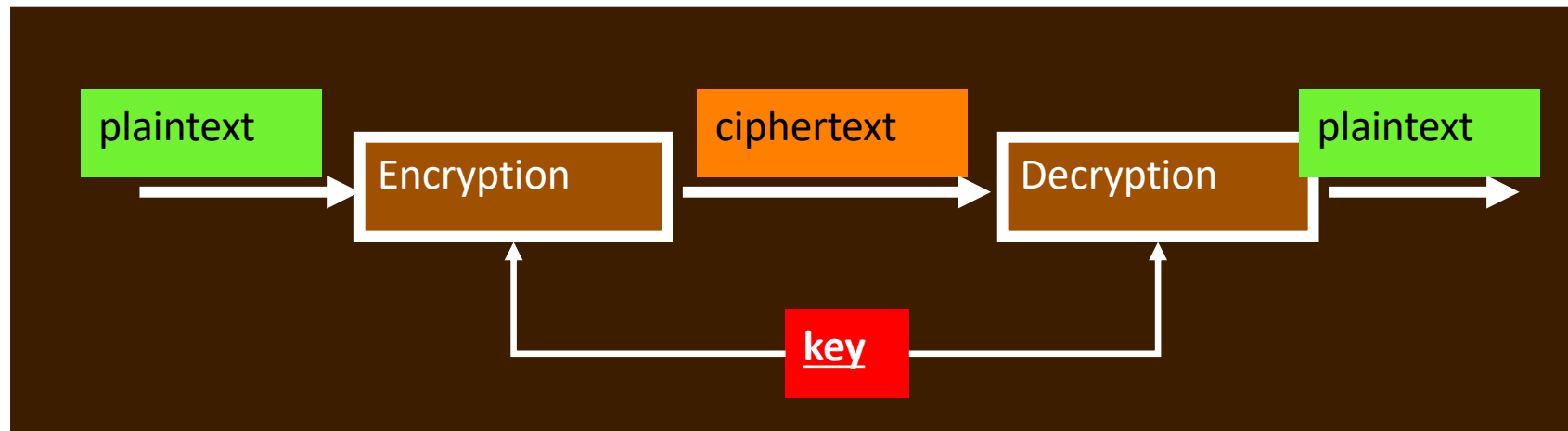
Secret Key Cryptography (I)

Roadmap

- Overview
- Feistel Cipher
- DES
- AES

Introduction

Secret Key Cryptography



- **Same** key is used for both encryption and decryption
 - this one key is **shared** by two parties who wish to communicate securely
- Also known as **symmetric key** cryptography, or **shared key** cryptography

Applications of Secret Key Crypto

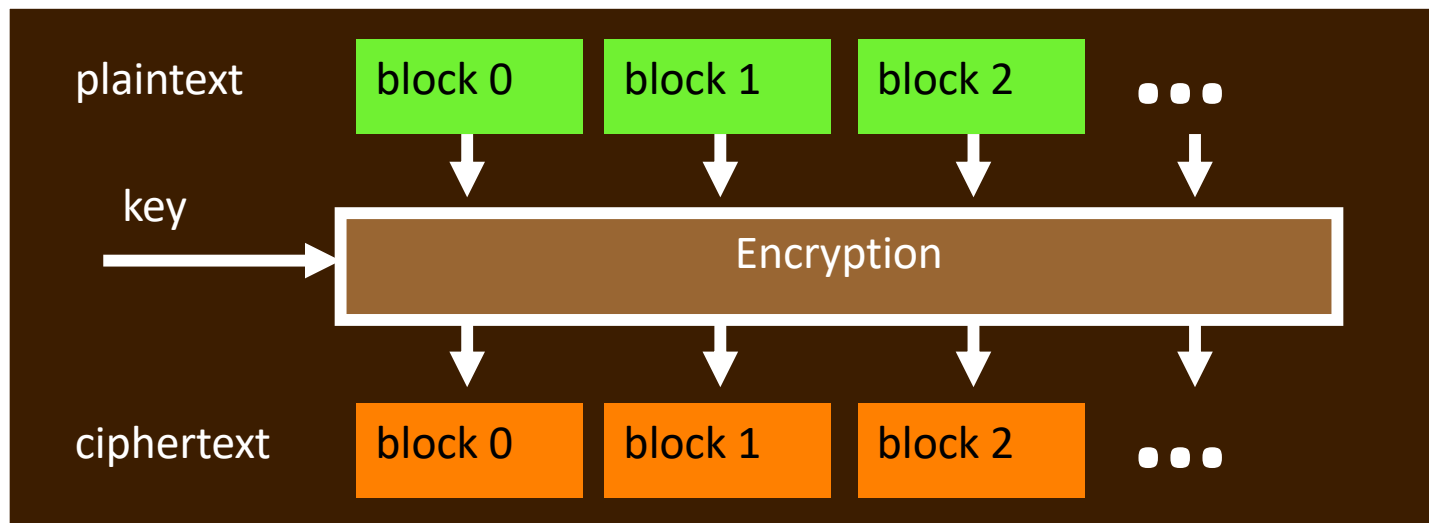
- **Communicating securely** over an insecure channel
 - Alice encrypts using shared key
 - Bob decrypts result using same shared key
- **Secure storage** on insecure media
 - Bob encrypts data before storage
 - Bob decrypts data on retrieval using the same key

Applications... (Cont'd)

- *Message integrity*
 - Alice computes a *message integrity code* (MIC) from the message, then encrypts with shared key
 - Bob decrypts the MIC on receipt, and verifies that it agrees with message contents
- *Authentication*
 - Bob can verify Alice sent the message
 - how is that possible?

Generic Block Encryption

- Converts one input plaintext **block of fixed size k bits** to an output ciphertext block also of k bits
- Benefits of large k ? of short k ?



Key Sizes

- Keys should be selected from a large potential set, to prevent brute force attacks
- 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
$X \mid Y$	Concatenation of X and Y
$K\{m\}$	Message m encrypted with secret key K

Two Principles for Cipher Design

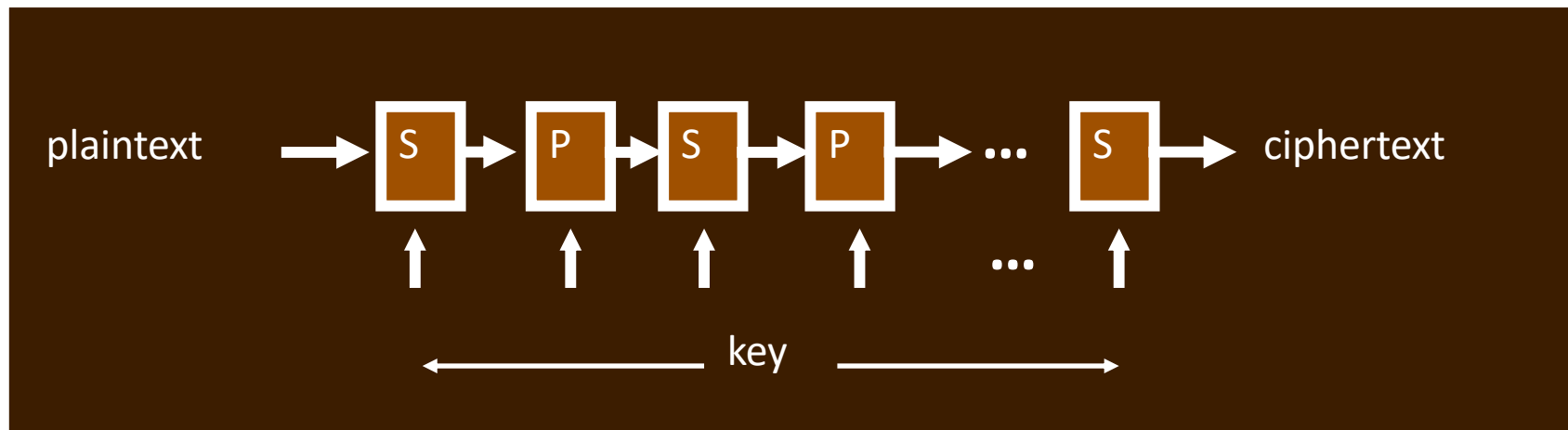
- **Confusion:**
 - Make the relationship between the <plaintext, key> input and the <ciphertext> output as complex (non-linear) as possible
- **Diffusion:**
 - Spread the influence of each input bit across many output bits

Exploiting the Principles

- Idea: use **multiple, alternating** permutations (P) and substitutions (S), e.g.,
 - $S \rightarrow P \rightarrow S \rightarrow P \rightarrow S \rightarrow \dots$
 - $P \rightarrow S \rightarrow P \rightarrow S \rightarrow P \rightarrow \dots$
- Do they have to alternate? e.g....
 - $S \rightarrow S \rightarrow S \rightarrow P \rightarrow P \rightarrow P \rightarrow S \rightarrow S \rightarrow \dots??$
- Confusion is mainly accomplished by **substitutions**
- Diffusion is mainly accomplished by **permutations**
- Example ciphers: **DES, AES**

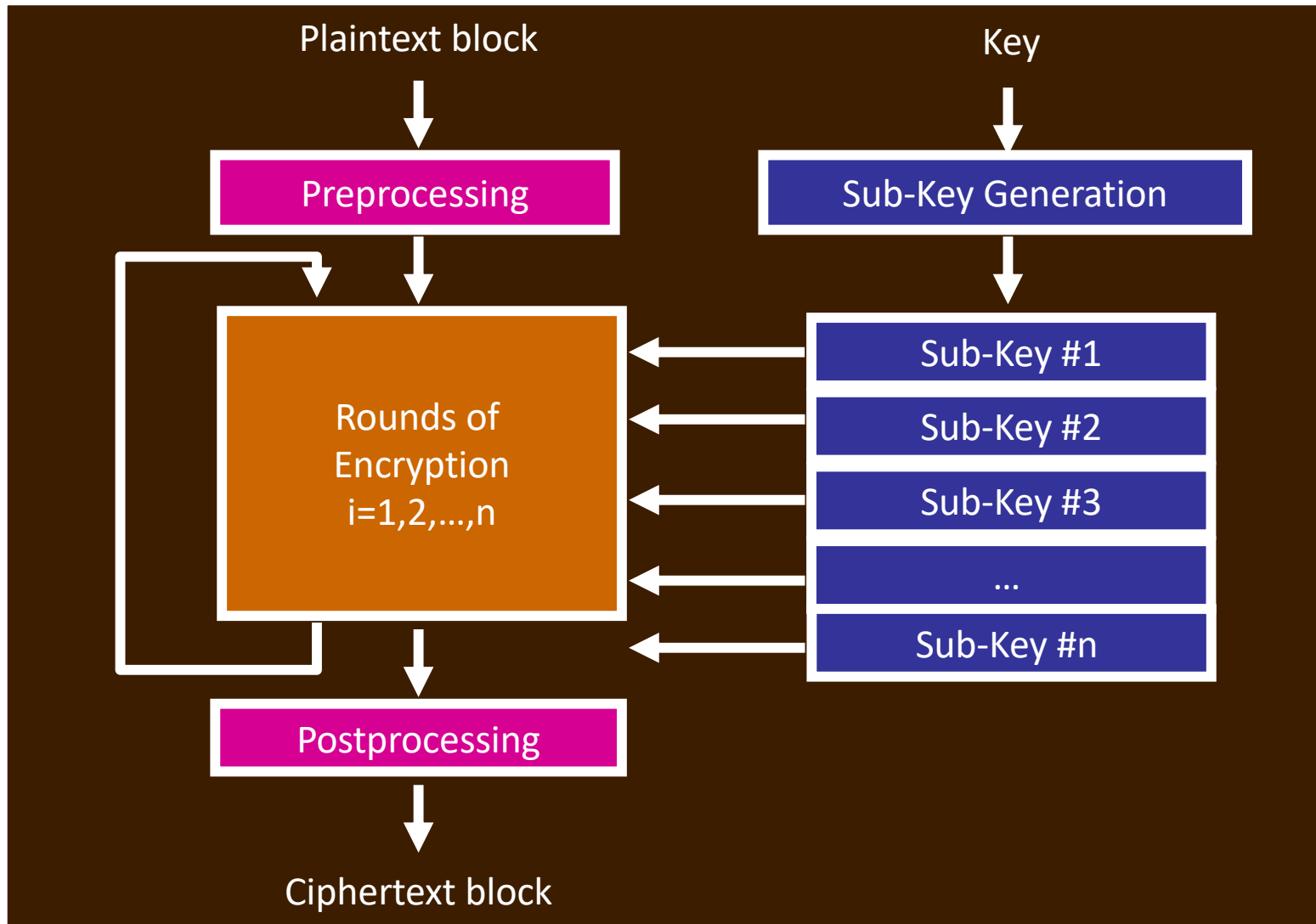
Secret Key... (Cont'd)

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



Well-known examples: **DES, AES**

Basic Form of Modern Block Ciphers



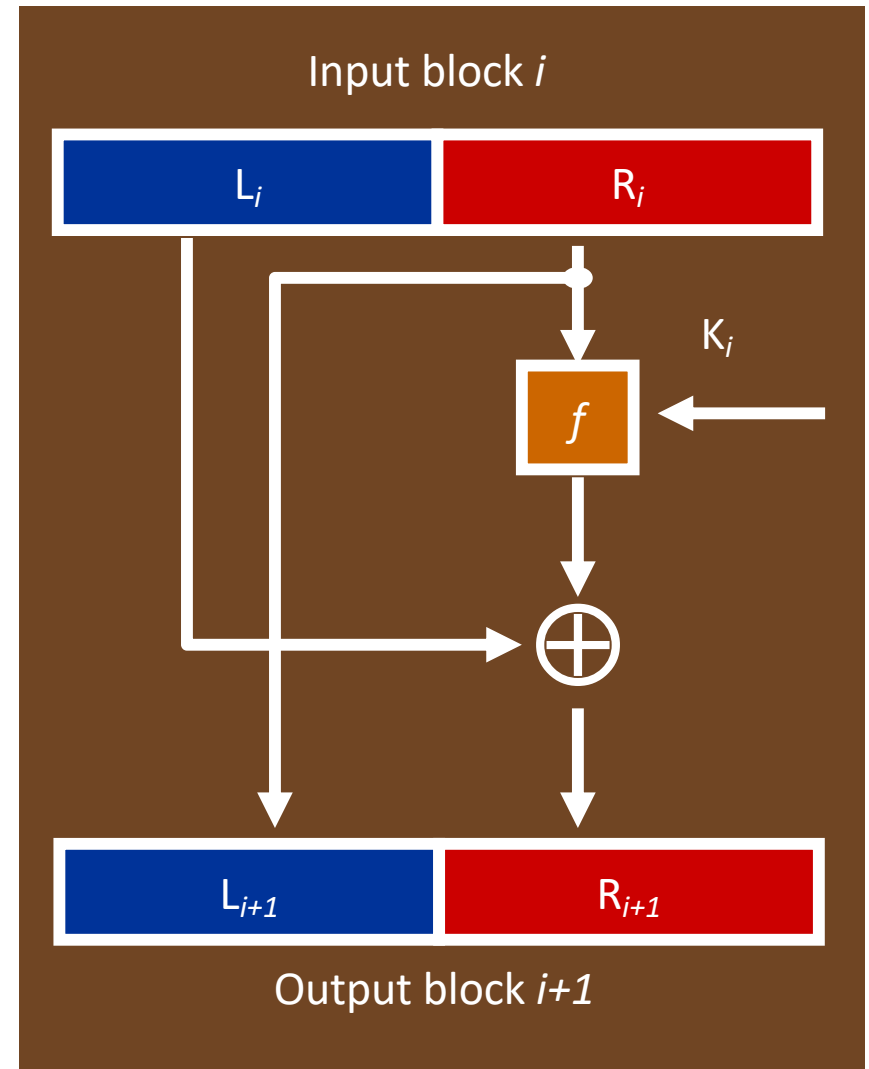
Feistel Ciphers

Overview

- Feistel Cipher has been a very influential “template” for designing a block cipher
- Major benefit: can do encryption and decryption **with the same hardware**
- Examples: DES, RC5

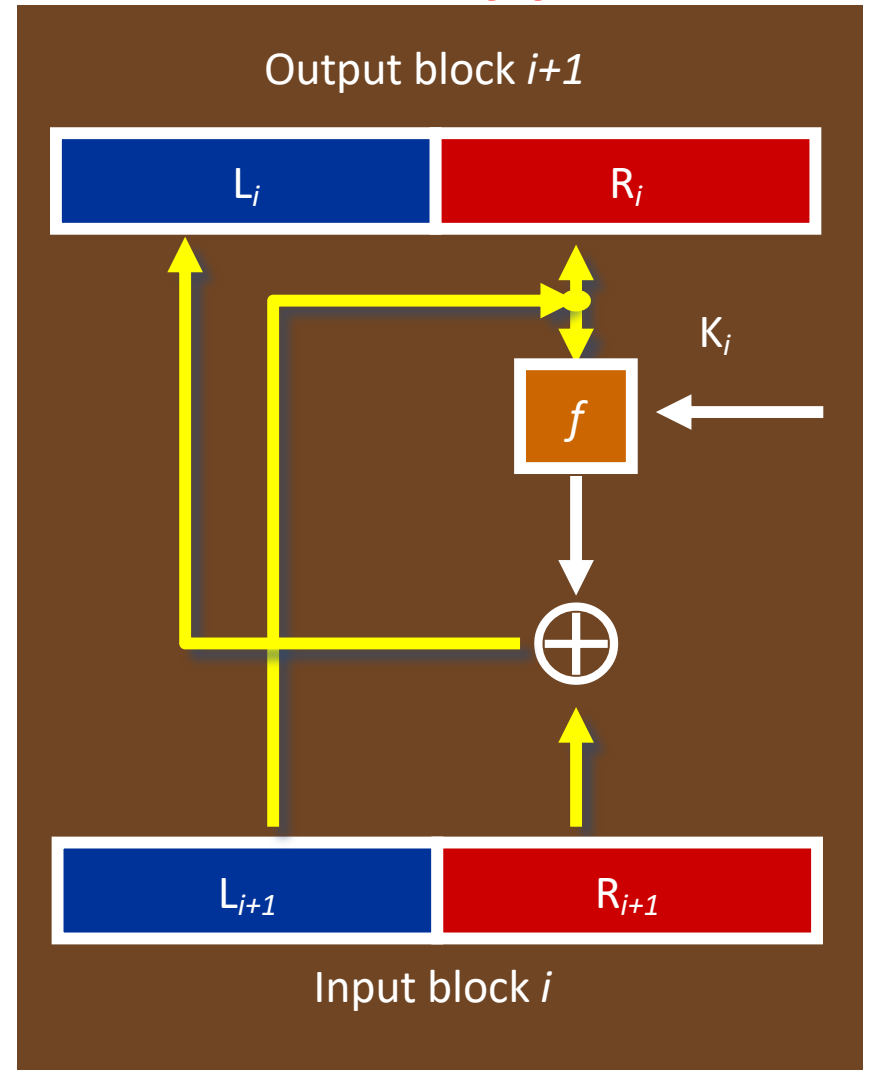
One “Round” of Feistel Encryption

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}

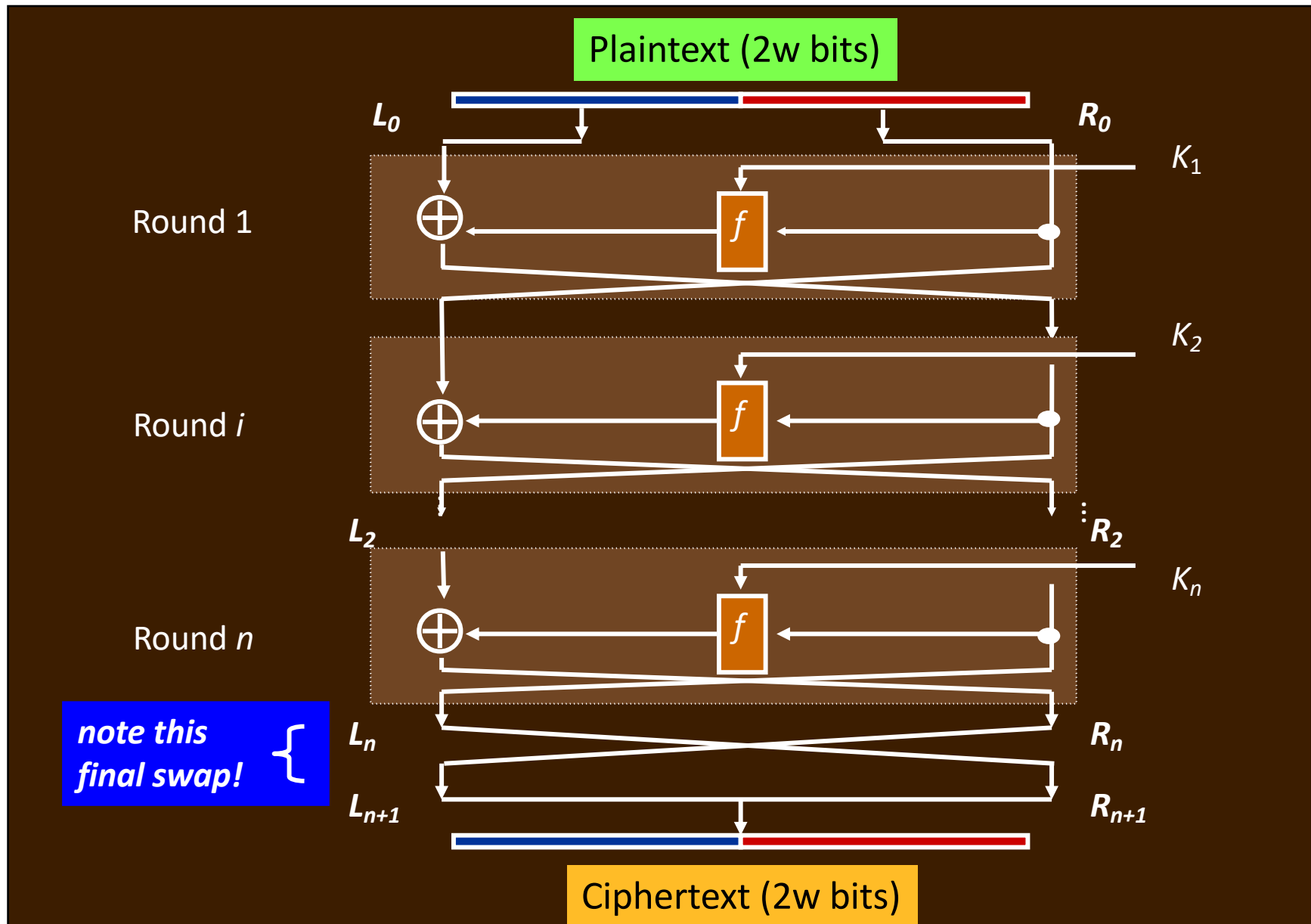


One “Round” of Feistel Decryption

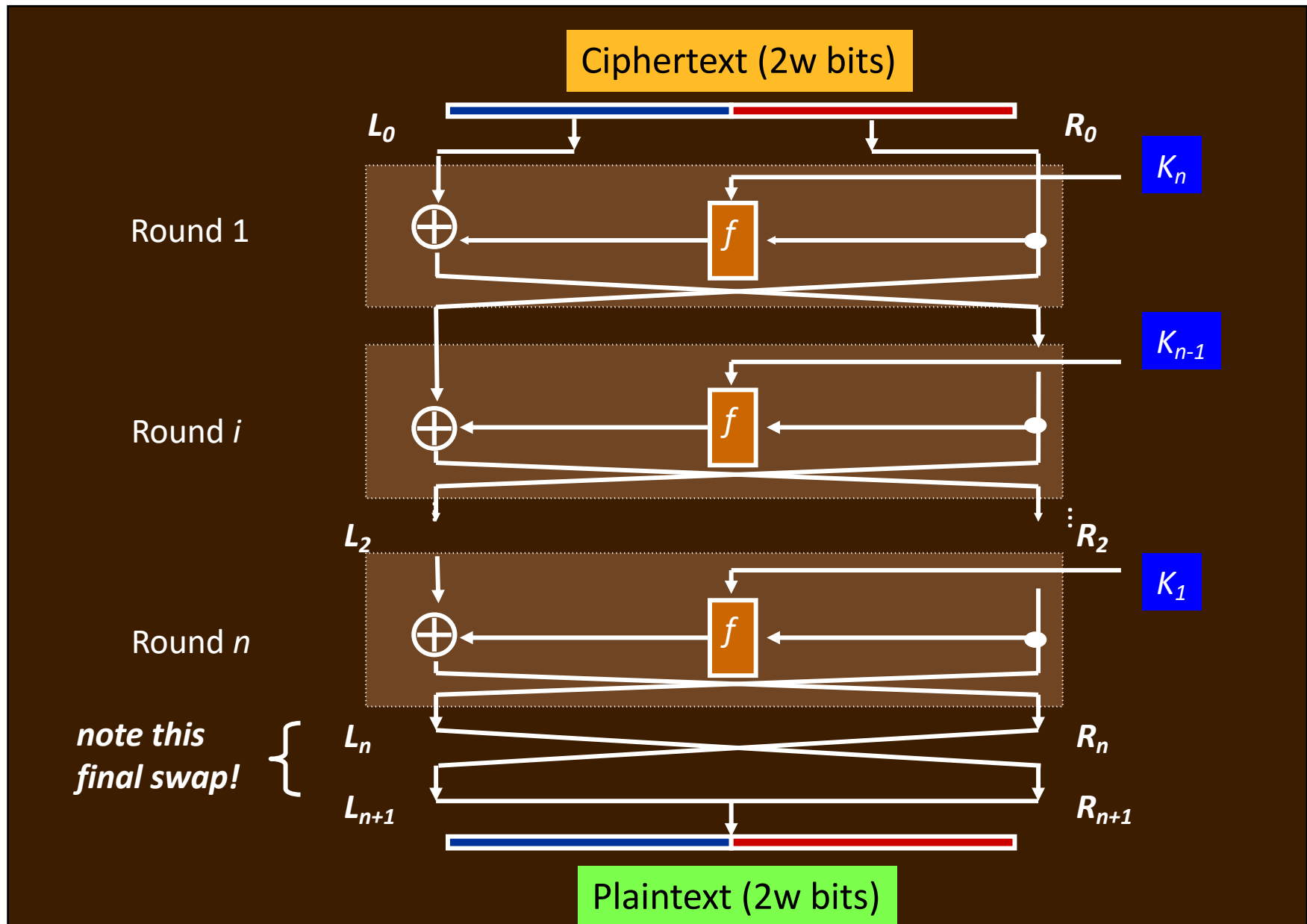
- Just reverse the arrows!



Complete Feistel Cipher: Encryption



Feistel Cipher: Decryption



Parameters of a Feistel Cipher

- Block size
- Key size
- Number of rounds
- Subkey generation algorithm
- “Scrambling” function f

Comments

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

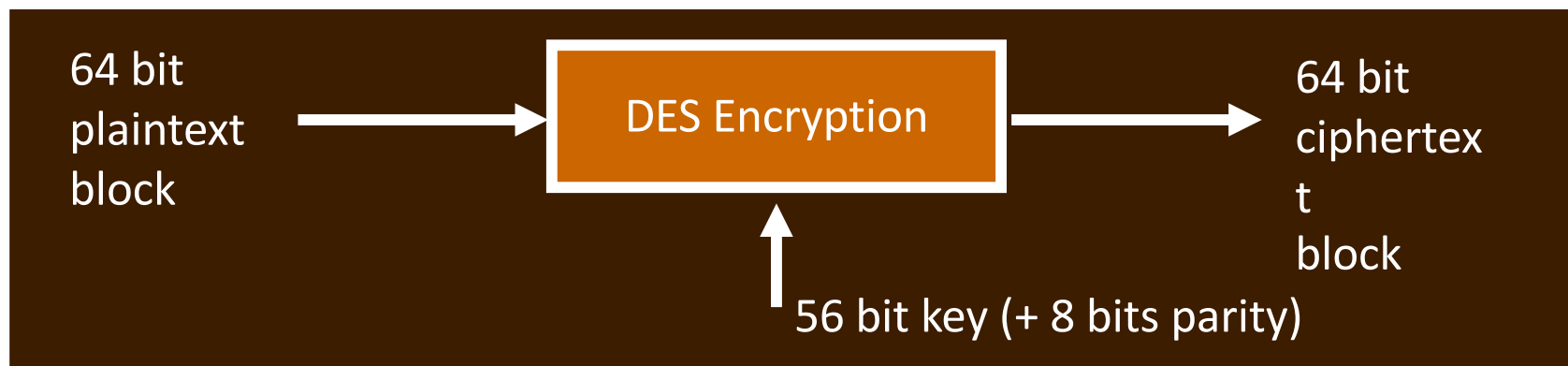
DES (Data Encryption Standard)

DES (Data Encryption Standard)

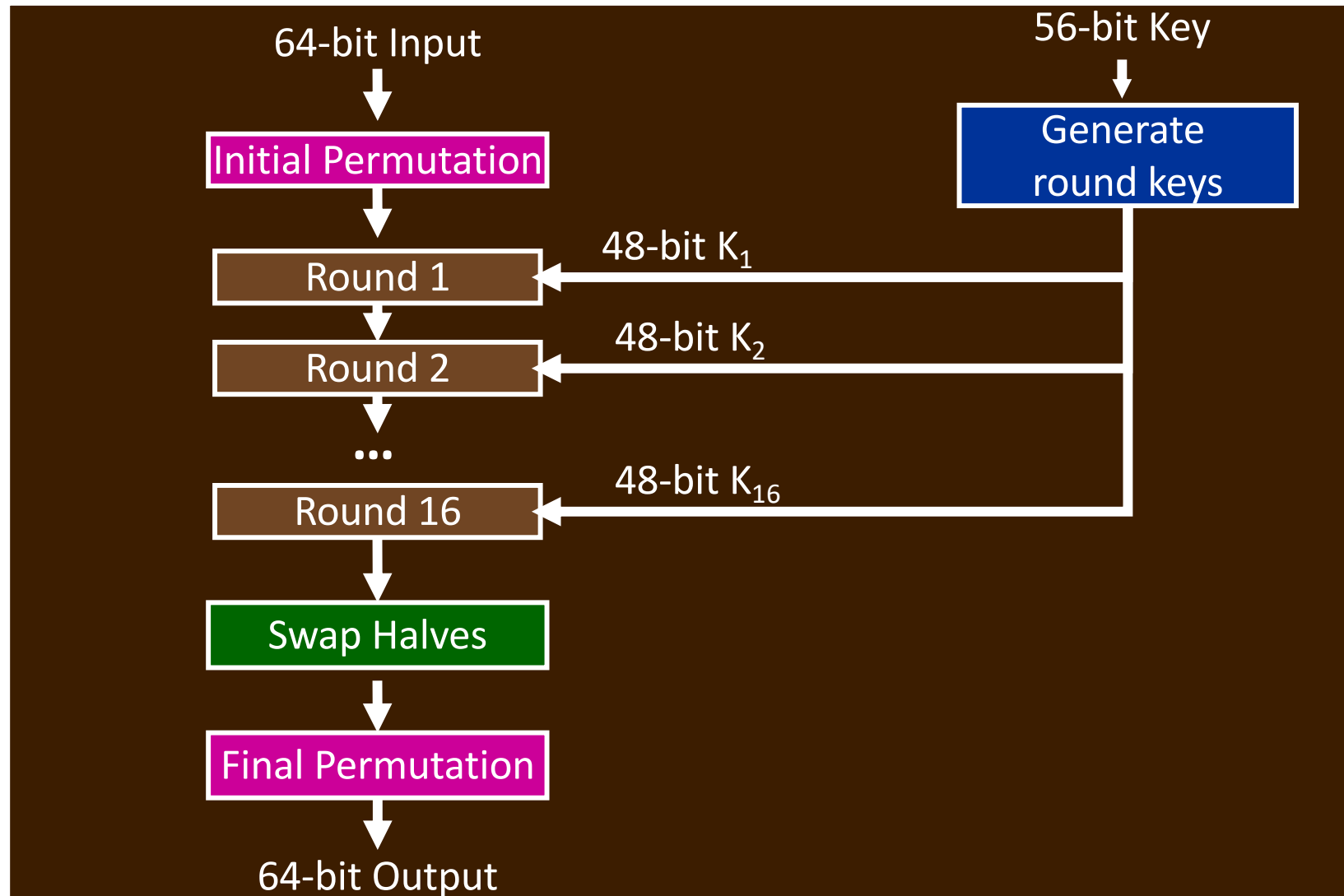
- Standardized in 1976 by NBS
 - proposed by IBM,
 - Feistel cipher
- Criteria (**official**)
 - provide high level of security
 - security must reside in key, not algorithm
 - not patented
 - must be exportable
 - efficient to implement in hardware
- Criteria (unofficial)
 - must be slow to execute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

DES Basics

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



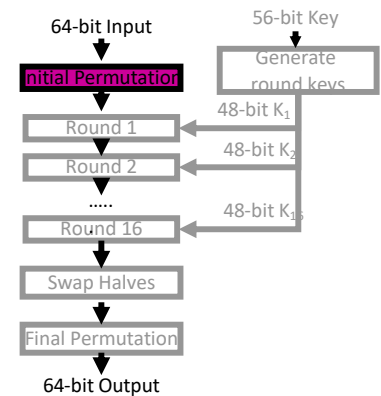
DES Top Level View



Initial and Final Permutations

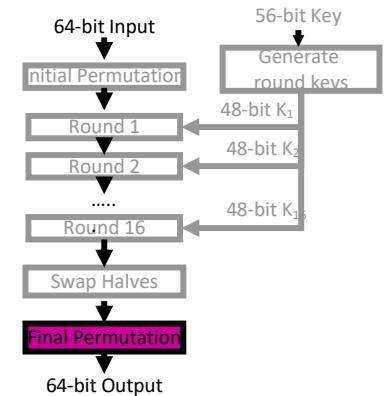
- **Initial** permutation given below
 - input bit #58 → output bit #1, input bit #50 → output bit #2, ...

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: Final Permutation is needed to make this a Feistel cipher
 - i.e., can use same hardware for both encryption and decryption

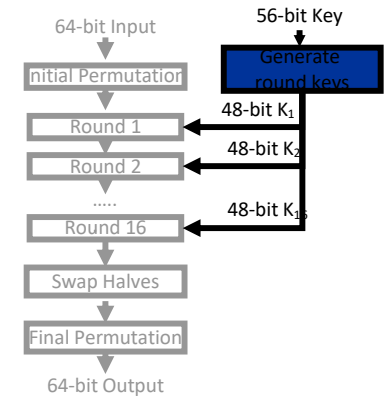
Key Generation: First Permutation

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

8 rows

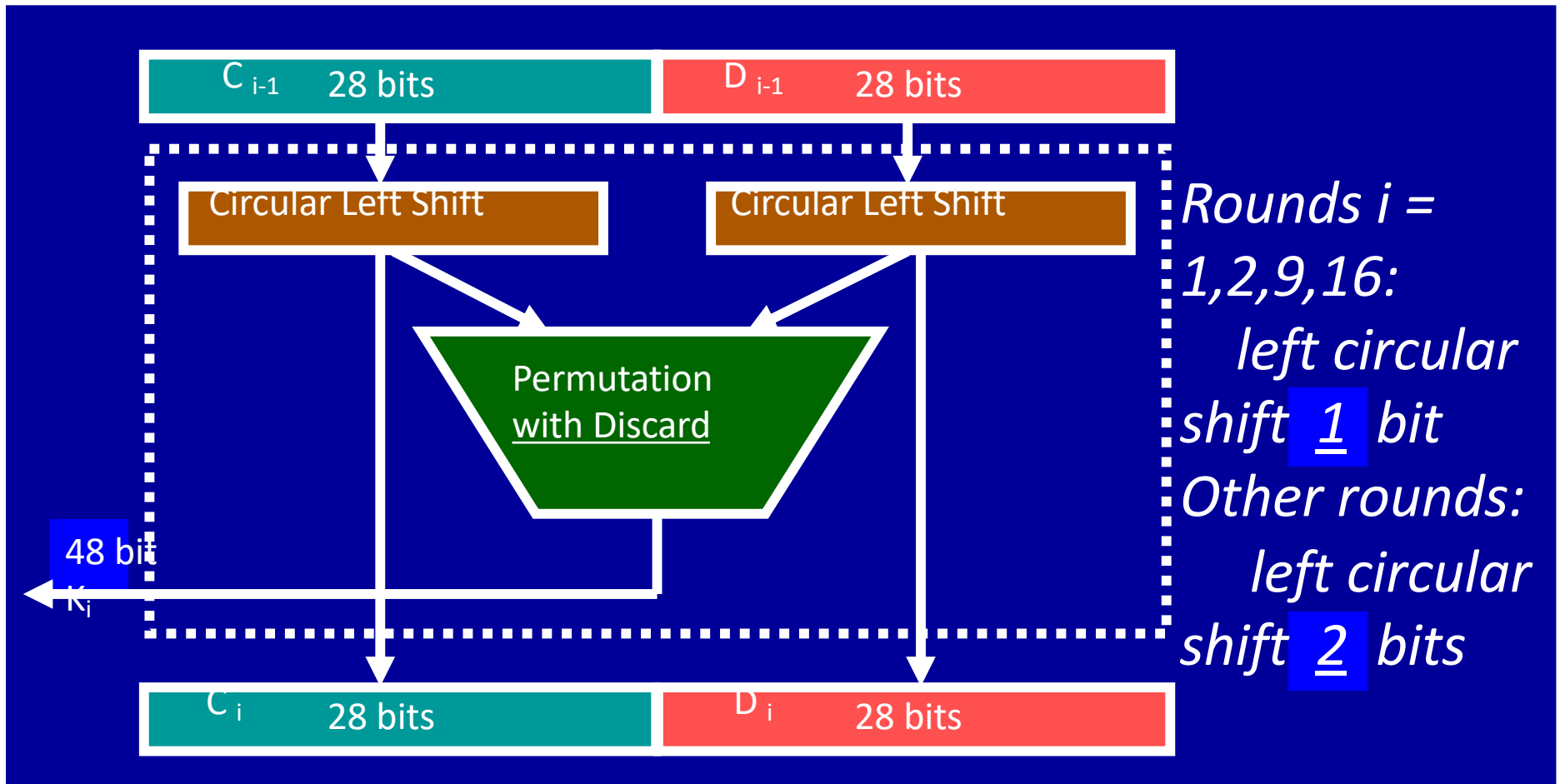
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

7 columns



*Parity bits left out:
8,16,24,...*

KeyGen: Processing Per Round



KeyGen: Permutation with Discard

- 28 bits \rightarrow 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

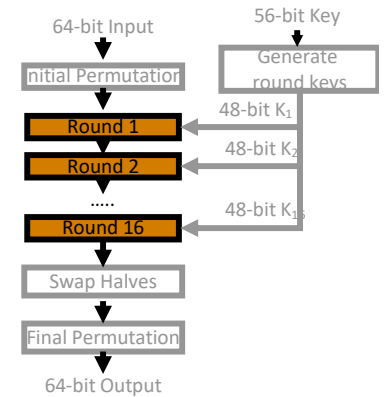
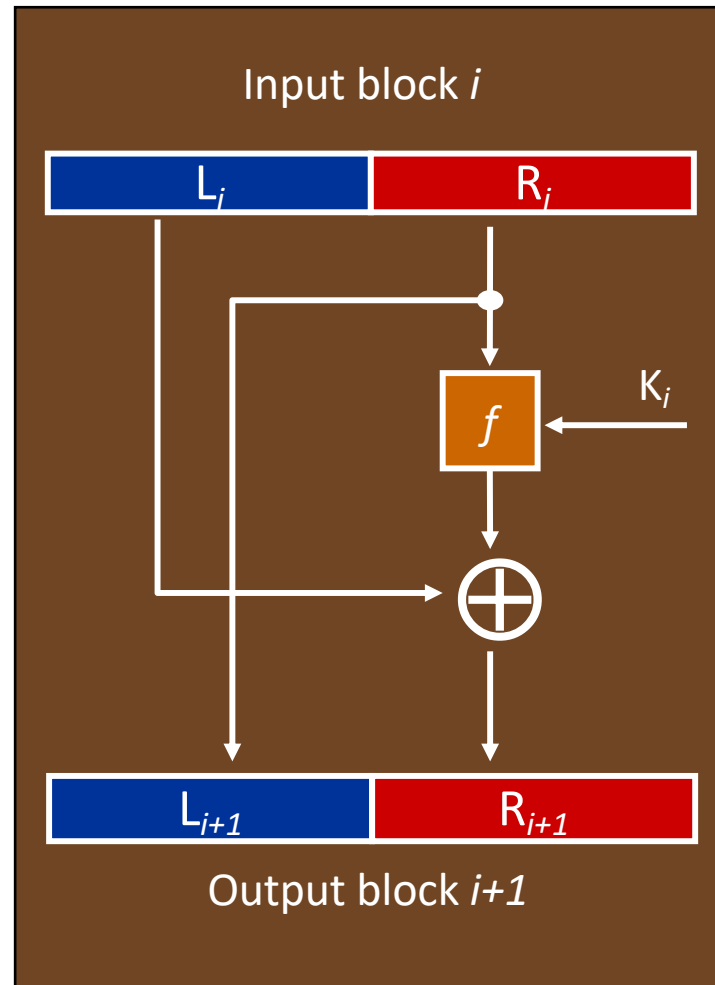
Bits left out:
9,18,22,25

Right half of K_i = permutation of D_i

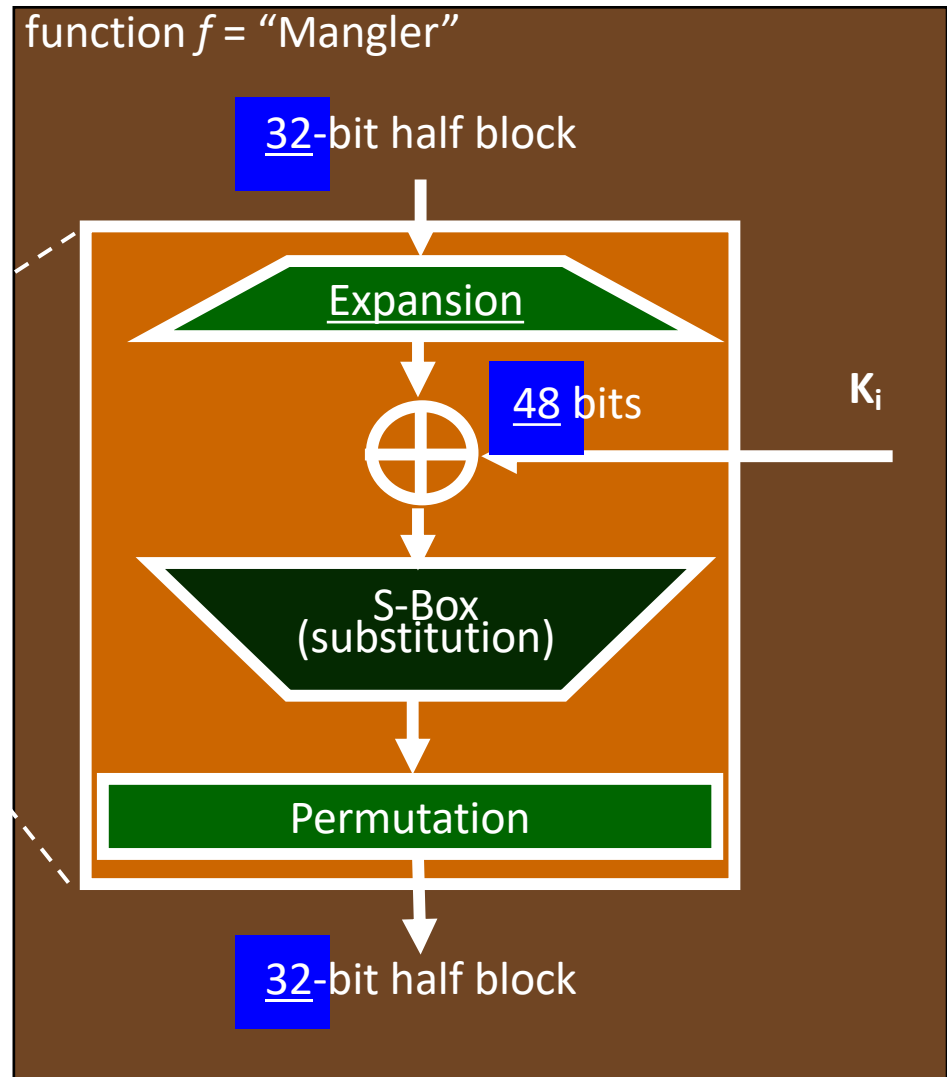
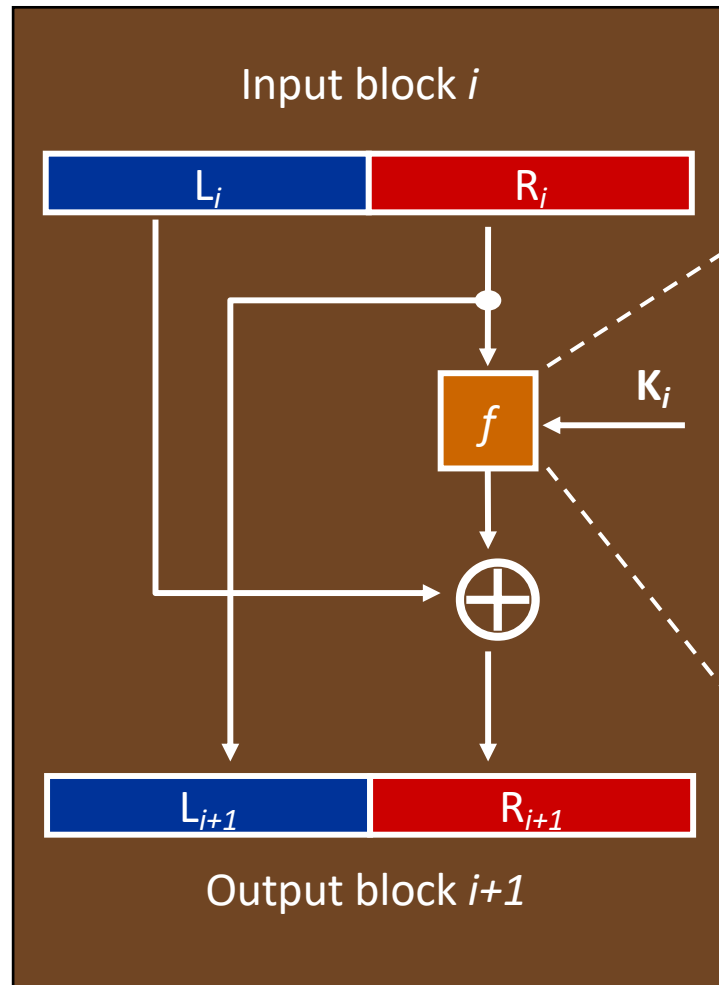
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Bits left out:
35,38,43,54

One DES (Feistel) Round



DES Round: f (Mangler) Function



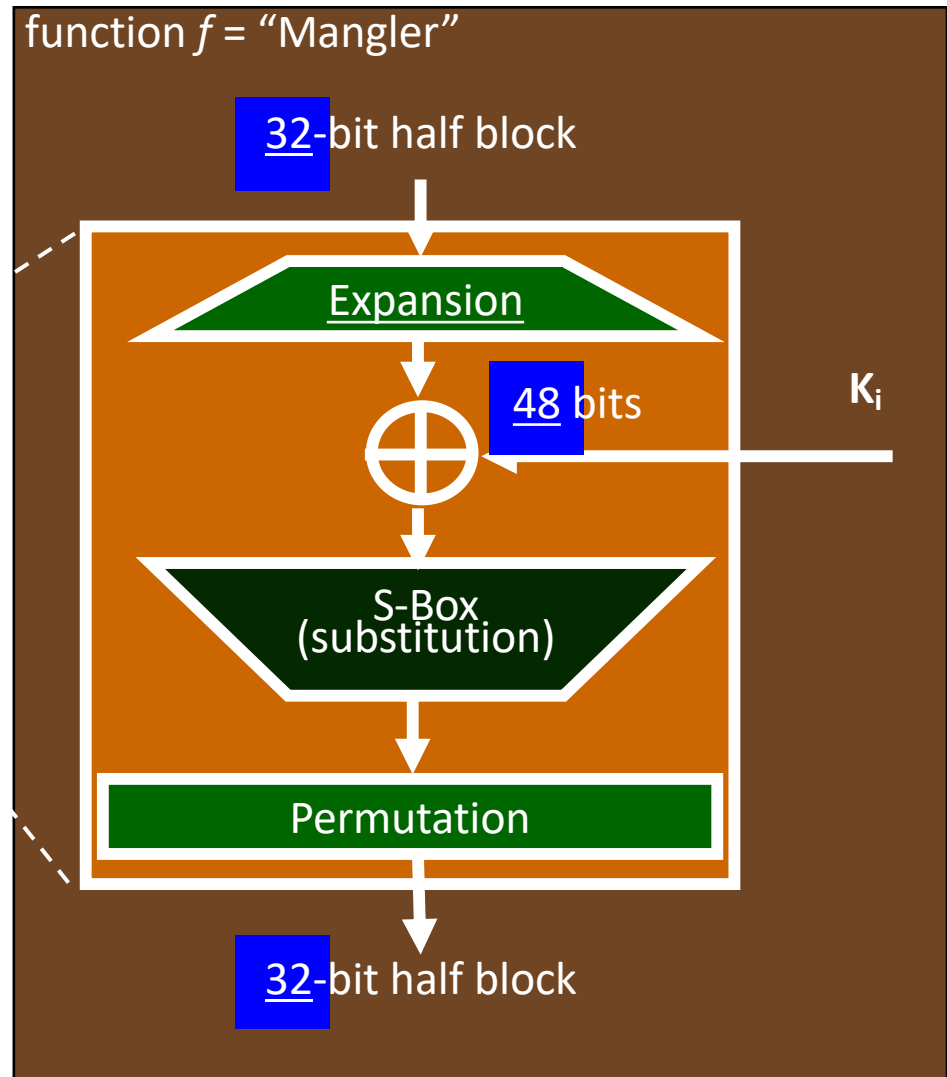
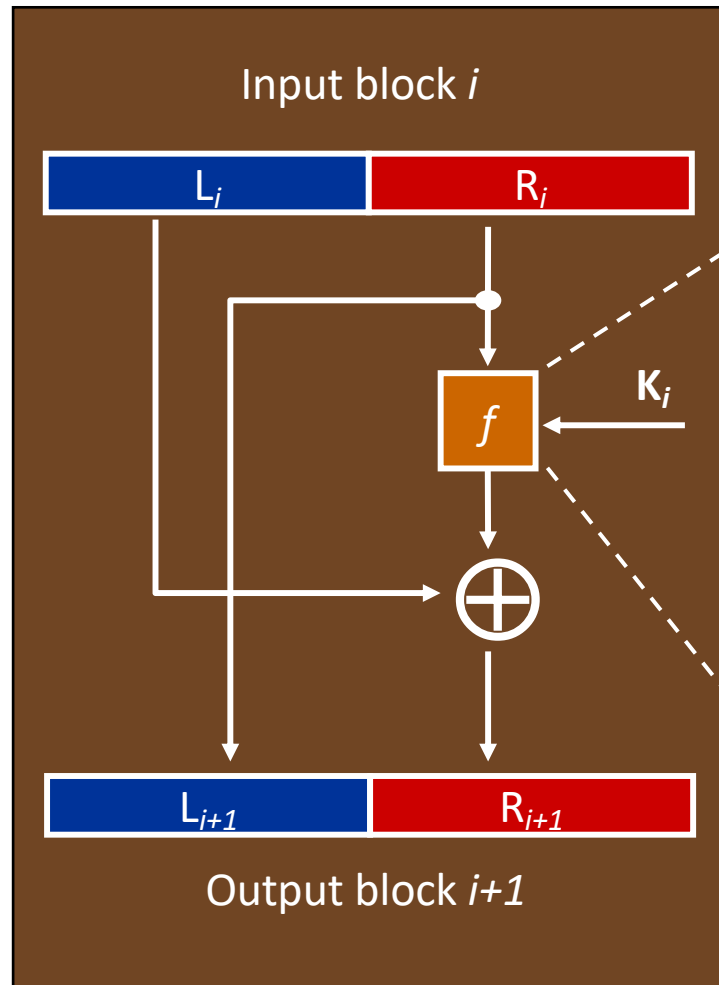
f : Expansion Function

- 32 bits \rightarrow 48 bits

these bits are repeated

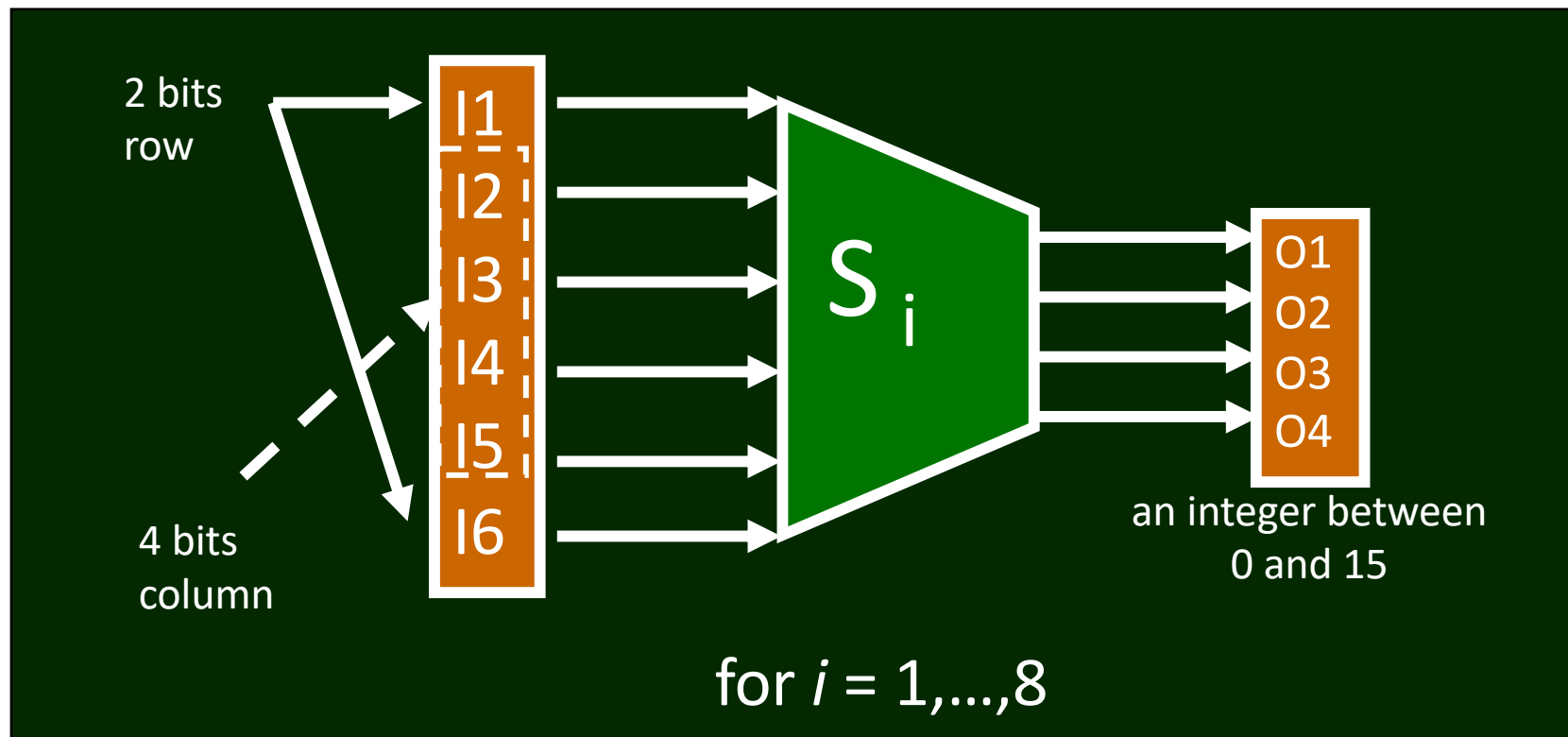
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

DES Round: f (Mangler) Function



f : **S-Box** (Substitute, Shrink)

- 48 bits \rightarrow 32 bits
 - 6 bits are used to select a 4-bit substitution
 - Pre-defined lookup table for every 6 bits of input



$f: S_1$ (Substitution)

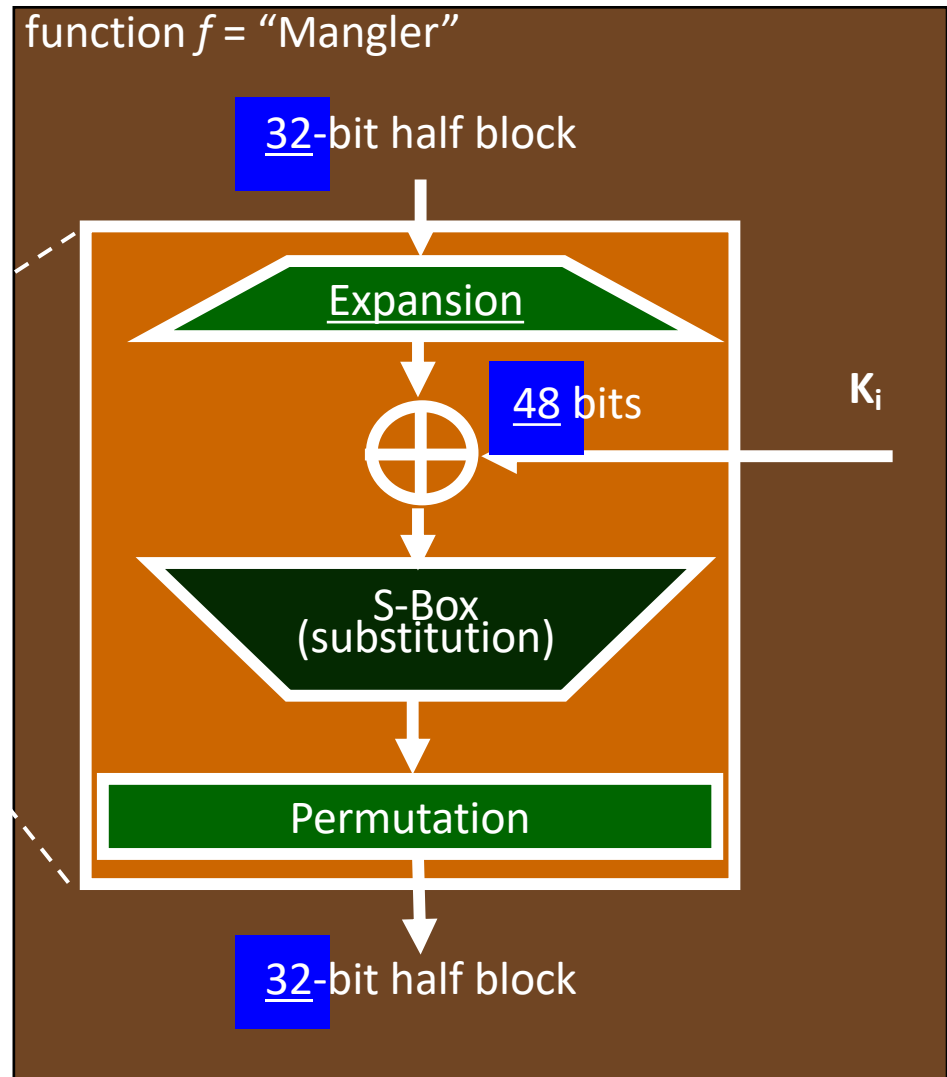
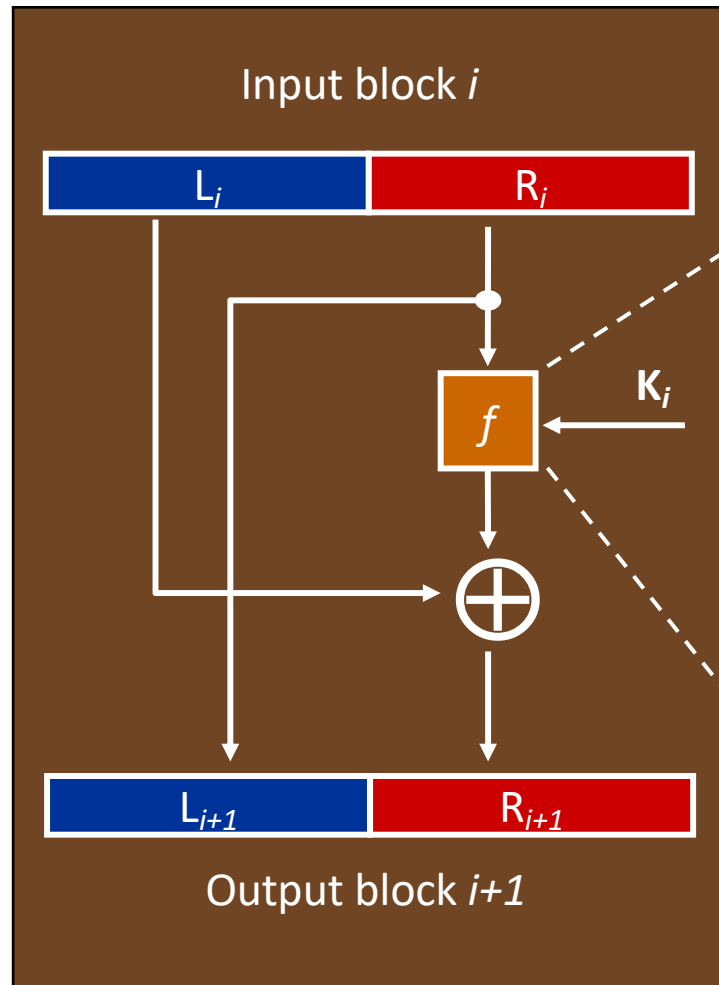
Each row and column contain different numbers

12/13/14/15 →		0	1	2	3	4	5	6	...	F
11/16 →	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

for $S_2..S_8$ (and rest of S_1), see the textbook

DES Round: f (Mangler) Function



f : Permutation

- 32bits \rightarrow 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

DES Analysis

Good Design?

- “We don’t know if
 - the particular details were well-chosen for strength,
 - whether someone flipped coins to construct the S-boxes,
 - or whether the details were chosen to have a weakness that could be exploited by the designers.”

Issues for Block Ciphers

- Number of rounds should be large enough to make advanced attacks as expensive as exhaustive search for the key

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 .5 if any input bit is changed
- 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:
0x**0**0000000000000000 and
0x**8**0000000000000000
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”: 4 keys with property
 $K\{K\{m\}\} = m$
- What’s bad about that?
- These are keys which, after the first key permutation, are:
 - 28 0’s followed by 28 0’s
 - 28 0’s followed by 28 1’s
 - 28 1’s followed by 28 0’s
 - 28 1’s followed by 28 1’s

More Keys to Avoid!

- “Semi-weak keys”: pairs of keys with the property
$$K_1\{K_2\{m\}\} = m$$
- What’s bad about that?
- 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

DES Key Size

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

Cryptanalysis of DES

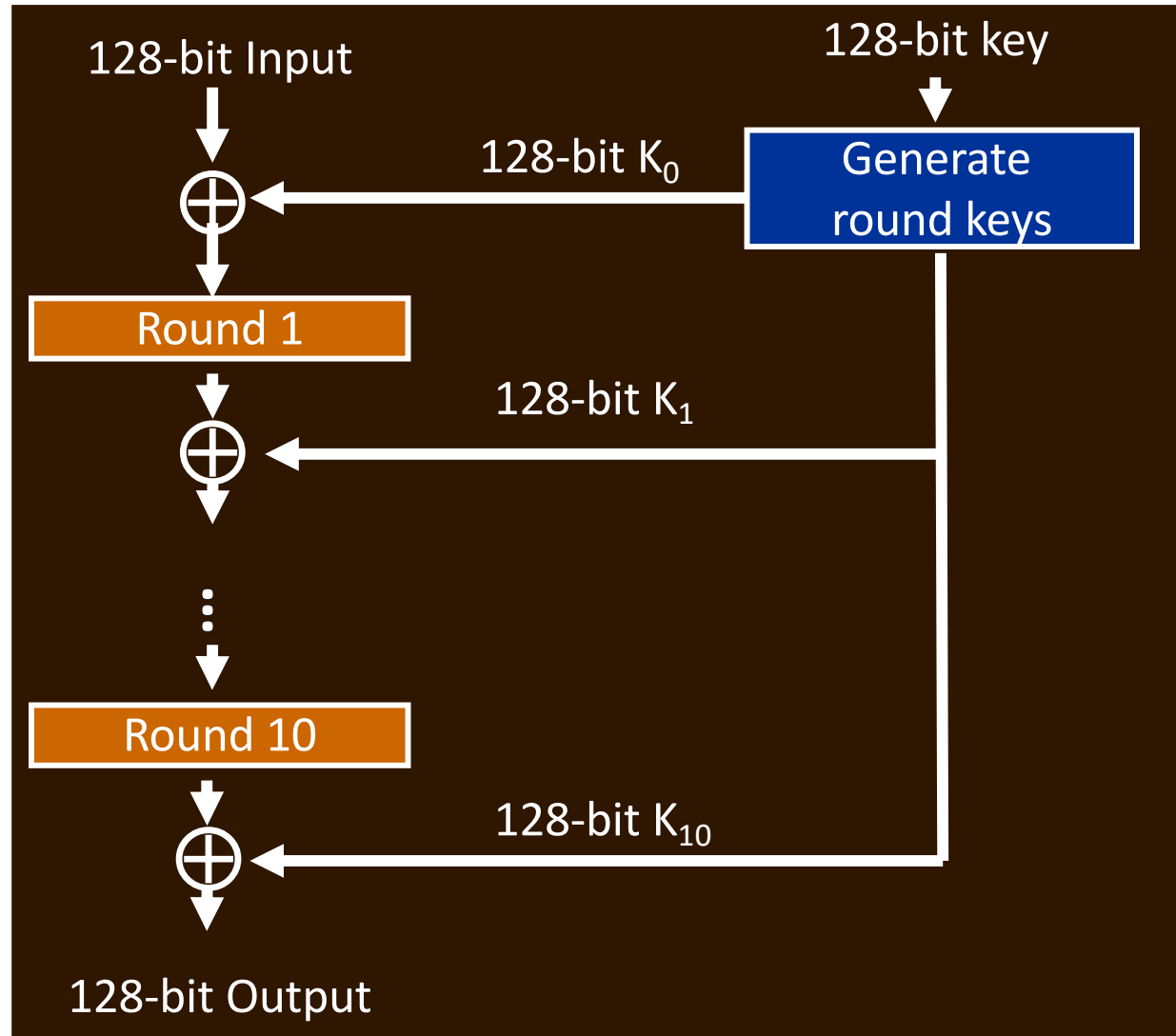
- **Differential cryptanalysis** exploits differences between encryptions of two different plaintext blocks
 - provides insight into possible key values
 - DES well designed to defeat differential analysis
- **Linear cryptanalysis** requires known plaintext / ciphertext pairs, analyzes relationships to discover key value
 - for DES, requires analyzing $O(2^{47})$ pairs
- No attacks on DES so far are significantly better than brute force attacks, for comparable cost

AES

Overview

- Selected from an **open** competition, organized by NSA
 - winner: Rijndael (pronounced “Rhine-dall”) algorithm, standardized as AES
- Some similarities to DES (rounds, round keys, alternate permutation+substitution)
 - but **not** a Feistel cipher
- **Block size = 128 bits**
- Key sizes = 128, 192, or 256 (**Variable key lengths**)
- Main criteria: secure, well justified, fast

AES-128 Overview



- Q1: What happens in each round?
- Q2: How are round keys generated?

AES-128 State

- Each plaintext block of 16 **bytes** is arranged as 4 columns of 4 bytes each



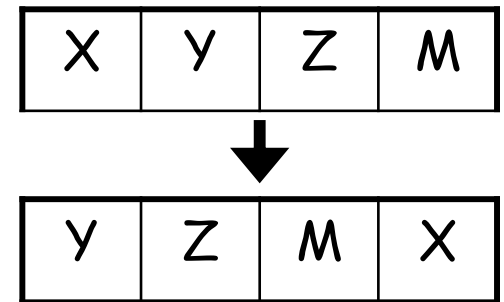
(Padding necessary for messages not a multiple of 16 bytes)

High-level View: One AES-128 Round

1. Apply **S-box** function to **each byte** of the state (i.e., 16 substitutions)

2. Rotate...

- (row 0 of state is unchanged)
- row 1 of the state left 1 column
- row 2 of the state left 2 columns
- row 3 of the state left 3 columns



3. Apply **MixColumn** function to **each column** of state

- last round omits this step

AES-128 Decryption (Conceptual)

- Run cipher in reverse, with inverse of each operation replacing the encryption operations
- Inverse operations:
 - XOR is its own inverse
 - inverse of S-box is just the inverse table
 - inverse of rotation in one direction is rotation in other direction
 - inverse of MixColumn is just the inverse table

AES Decryption (Actual)

- Run cipher in **forward** direction, except...
 - use inverse operations
 - apply round keys in reverse order
 - apply InvMixColumn to round keys K1..K9
- Decryption takes more memory and cycles encryption
 - can only partially reuse hardware for encryption

AES Assessment

- Speed: about **16 clock cycles/byte** on modern 32-bit CPUs
 - 200 MByte/s on a PC, no special hardware!
- No known successful attacks on full AES
 - best attacks work on 7–9 rounds (out of 10–14 rounds)
- Clean design
- For brute force attacks, AES-128 will take **4×10^{21} X** (= 2^{72}) 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

- S-Box & MixColumns are designed to frustrate Linear Analysis

Side Channel Attacks: based on peculiarities of the **implementation** of the cipher

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

Power Attacks: measures power to do operations

- changing one bit requires considerably less power than changing many bits in a byte

Summary

- Secret key crypto is (a) good quality, (b) faster to compute than public key crypto, and (c) the most widely used crypto
- DES strong enough for non-critical applications, but triple-DES is better
- AES even better (stronger and much faster), has versions with 128-, 192-, and 256-bit keys
- Secret key crypto requires “out-of-band”, bilateral key negotiation/agreement