CSCI 454/554 Computer and Network Security

Topic 3.1 Secret Key Cryptography – Algorithms

Outline

- Introductory Remarks
- Feistel Cipher
- DES
- AES

Introduction

Secret Keys or Secret Algorithms ?

WILLIAM & MARY

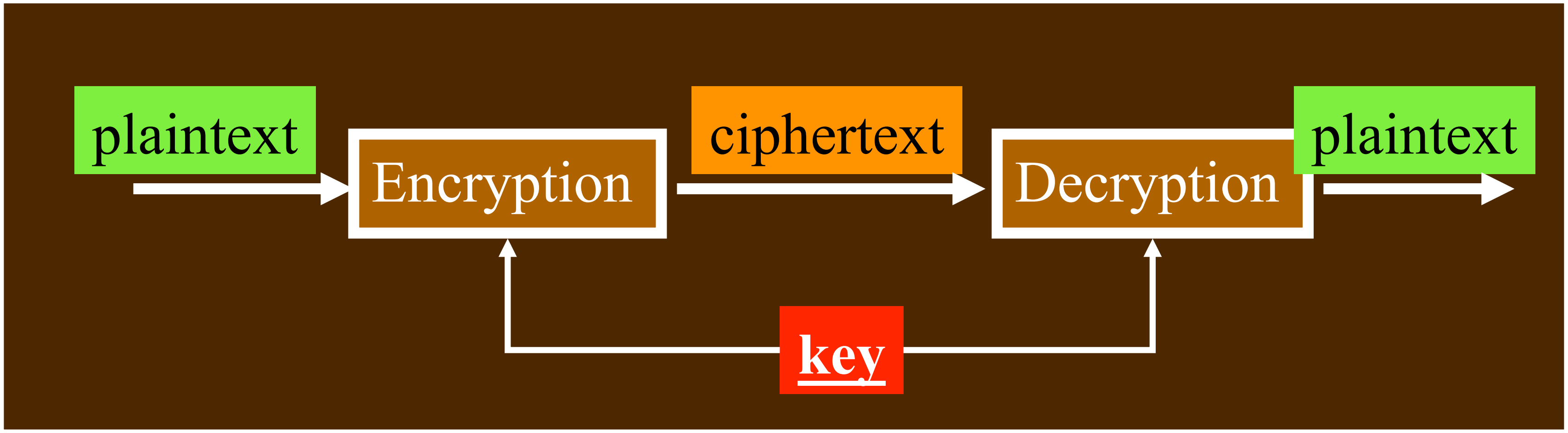
- "Security by obscurity"
  - "hide" the details of the algorithms
  - drawback: hard to keep secret if cipher is used widely, or implementation can be reverse engineered
- Alternative: publish the algorithms
  - fewer vulnerabilities will result if many smart people try and fail to break the cipher
  - security of the cipher depends on the secrecy of the keys, instead

Secrets? (Cont'd)

WILLIAM & MARY

- Commercial world relies upon standardized, public algorithms, and secret keys
- Government tends to also rely on secret algorithms

Secret Key Cryptography



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

Applications of Secret Key Crypto

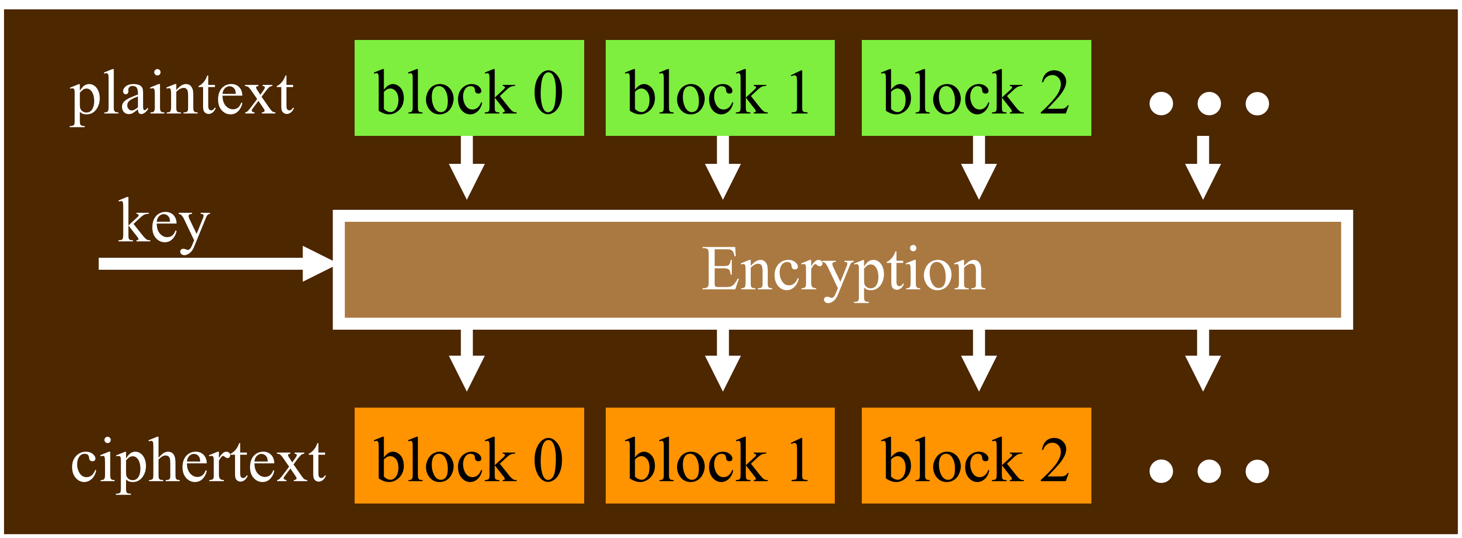
- Communicating securely over an insecure channel
  - Alice encrypts using shared key
  - Bob decrypts result using same shared key
- Secure storage on insecure media
  - Bob encrypts data before storage
  - Bob decrypts data on retrieval using the same key

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 ⊕ 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

WILLIAM & MARY

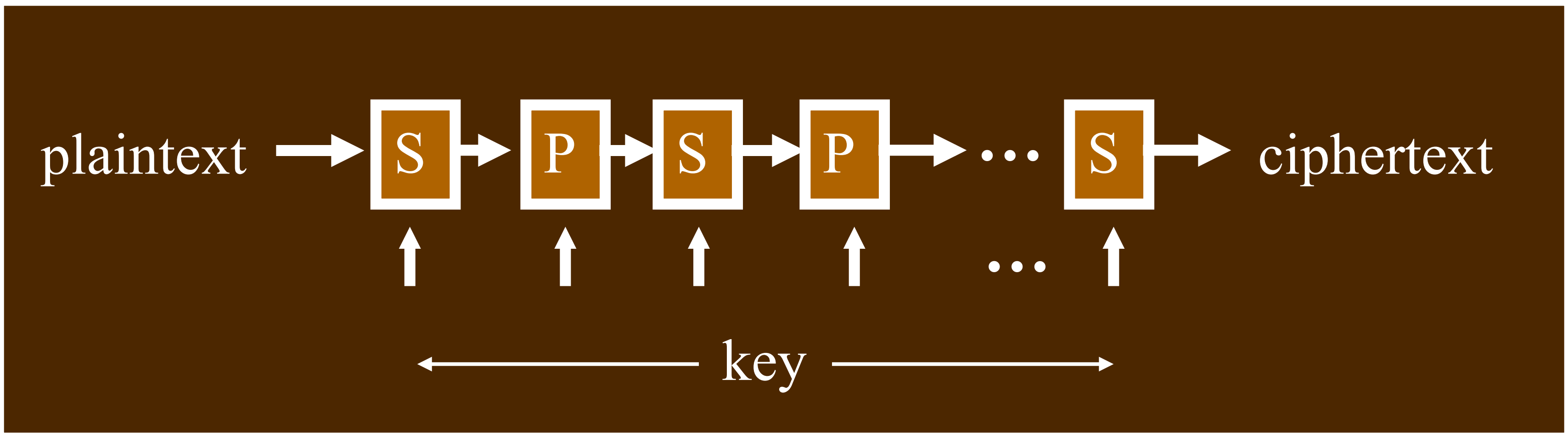
- Confusion:
  - Make the relationship between the <plaintext, key> input and the <ciphertext> output as complex (non-linear) as possible
- Diffusion:
  - Spread the influence of each input bit across many output bits

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→…??
- Confusion is mainly accomplished by substitutions
- Diffusion is mainly accomplished by permutations
- Example ciphers: DES, AES

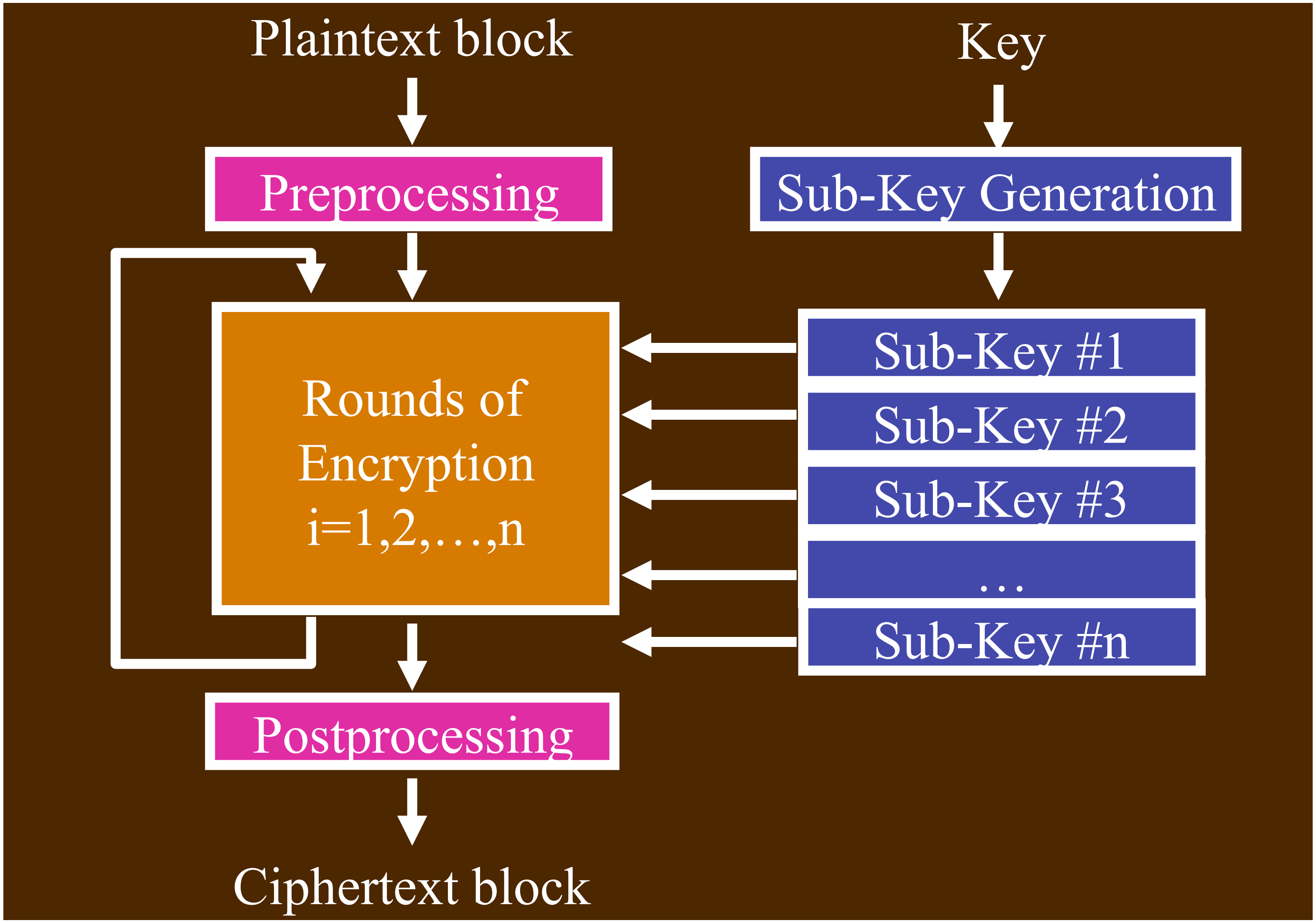
Secret Key… (Cont'd)

- 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



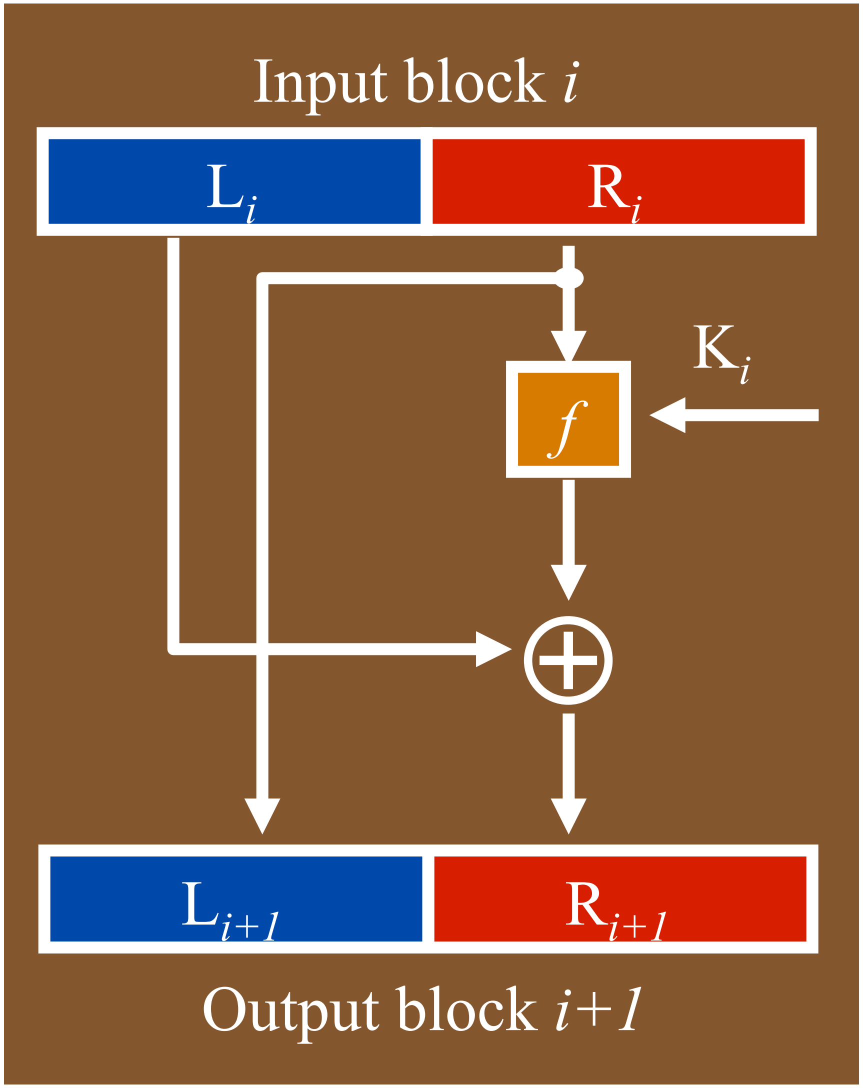
Feistel Ciphers

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

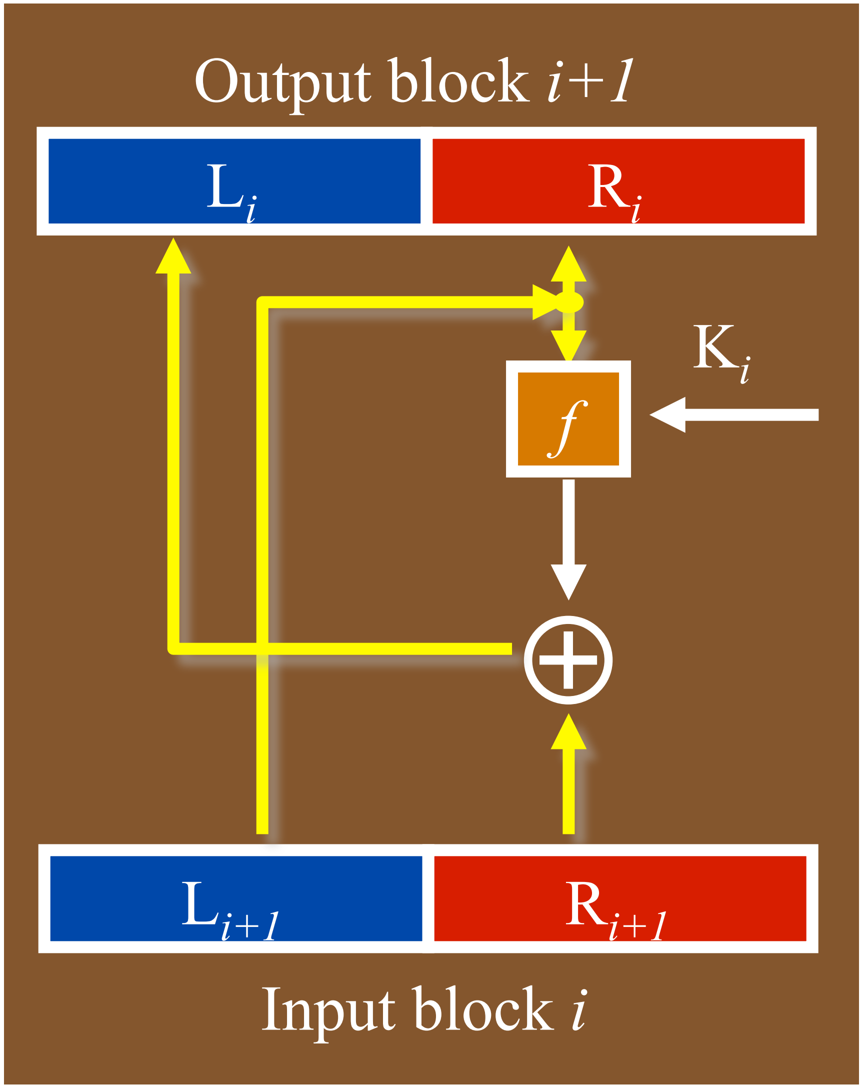
One "Round" of Feistel Encryption

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}

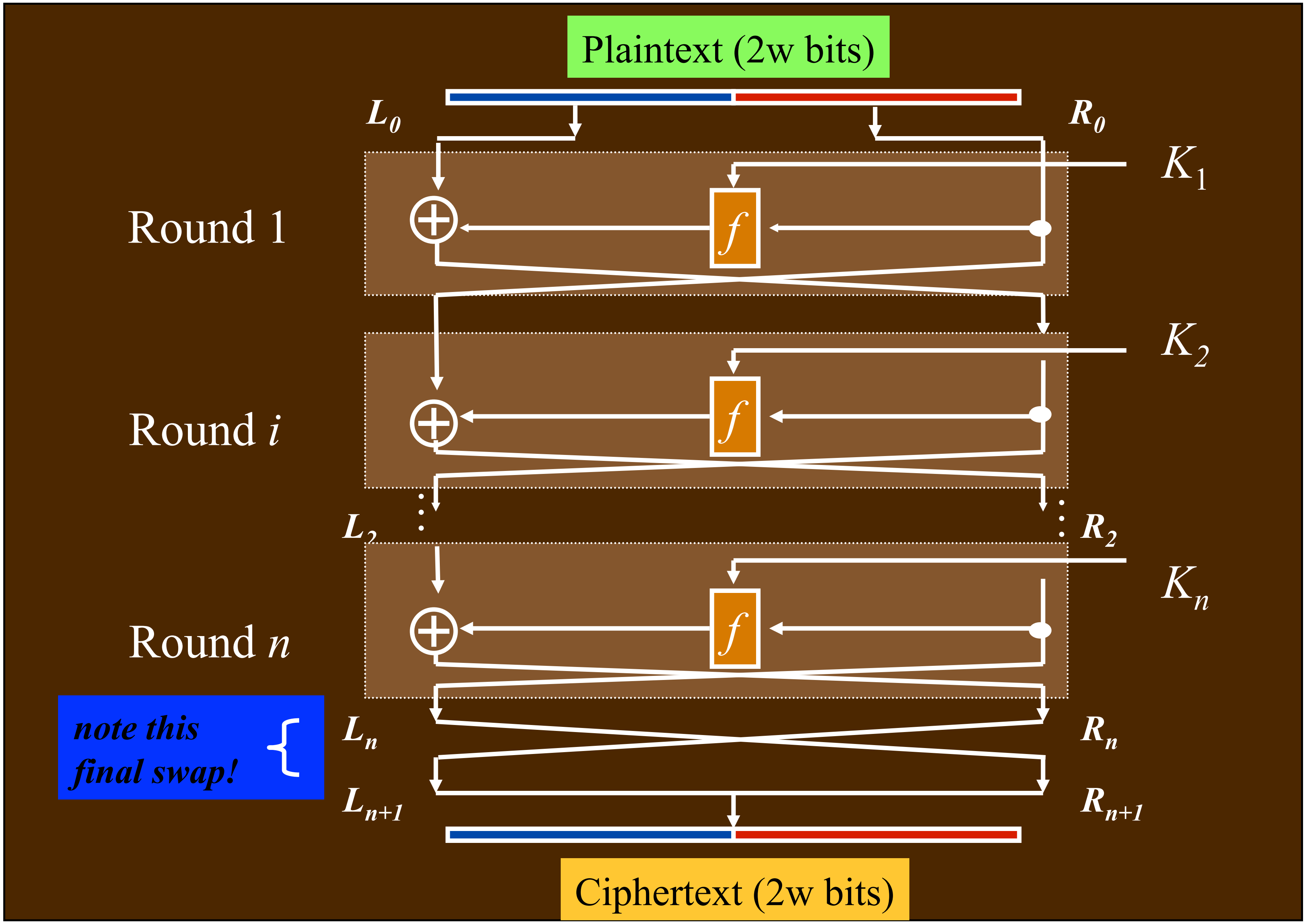


One "Round" of Feistel Decryption

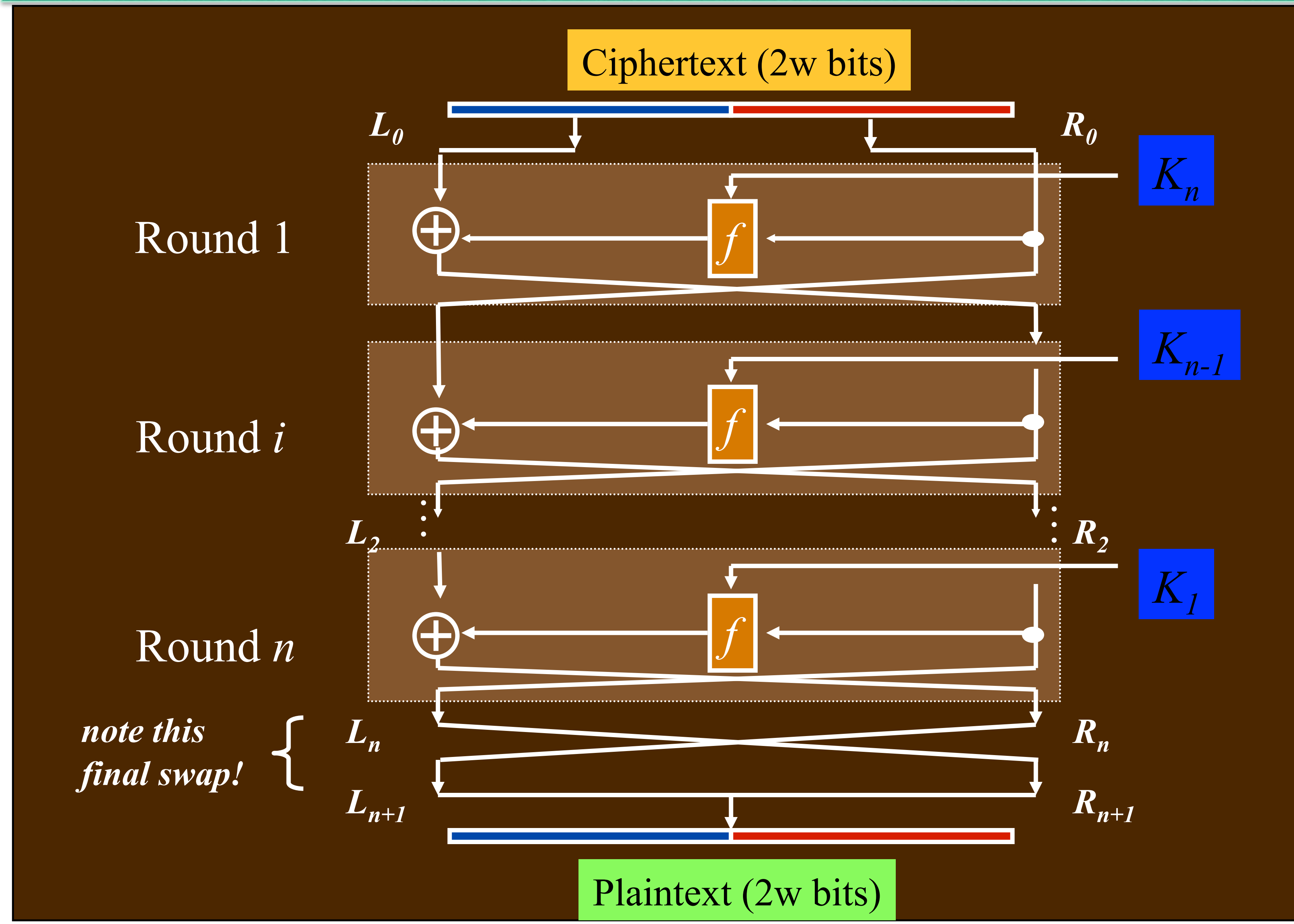
- Just reverse the arrows!



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 the 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)

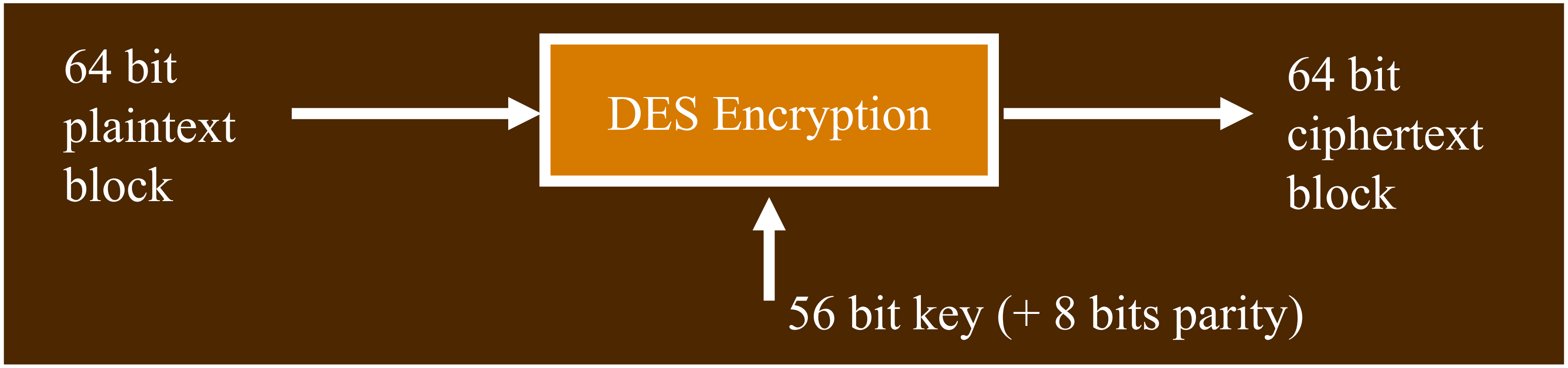
- Standardized in 1976 by NBS (now NIST)
  - proposed by IBM,
  - Feistel cipher
- Criteria (official)
  - provide high level of security
  - security must reside in key, not algorithm
  - not patented
  - must be exportable
  - efficient to implement in hardware

DES… (Cont’d)

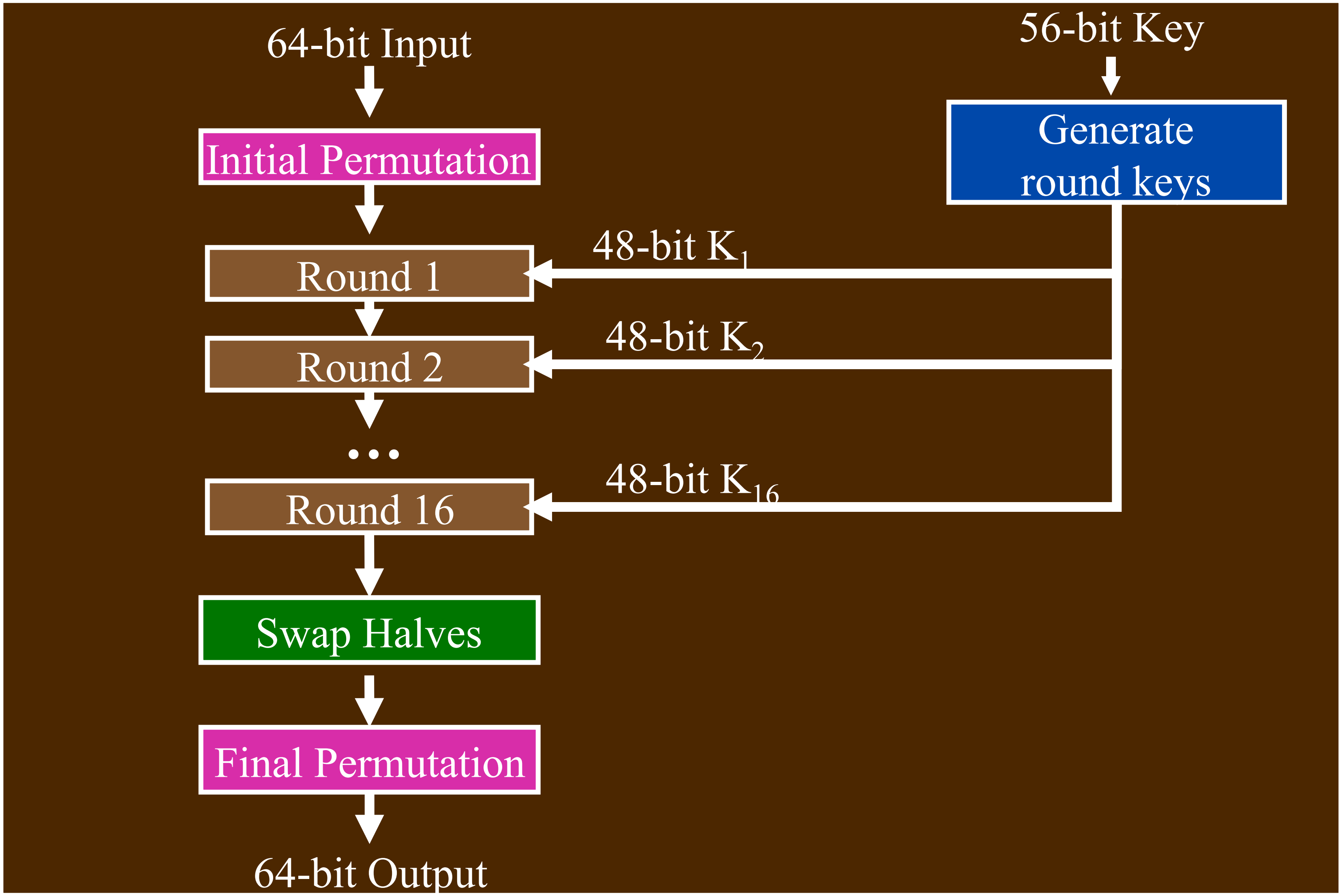
- Criteria (unofficial)
  - must be slow to execute in software
  - must be breakable by NSA :-)

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



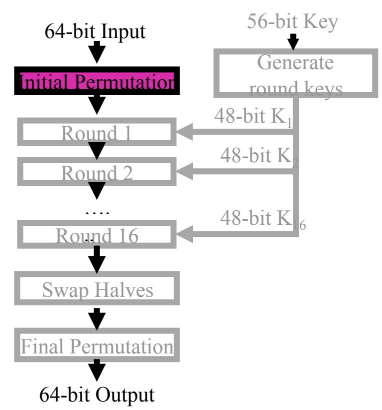
DES Top Level View



Initial and Final Permutations

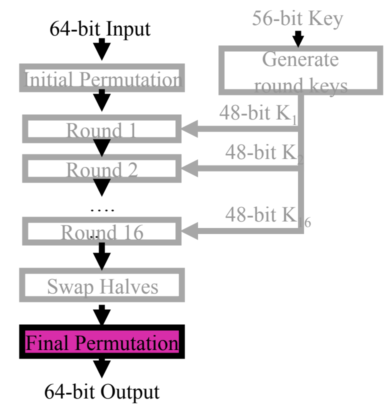
- Initial permutation given below
  - input bit #58→output bit #1, input bit #50→ output bit #2, …

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

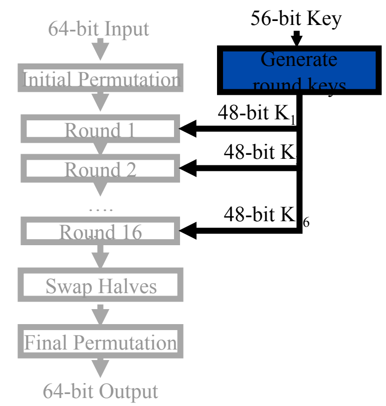
Key Generation: First Permutation

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

7 columns

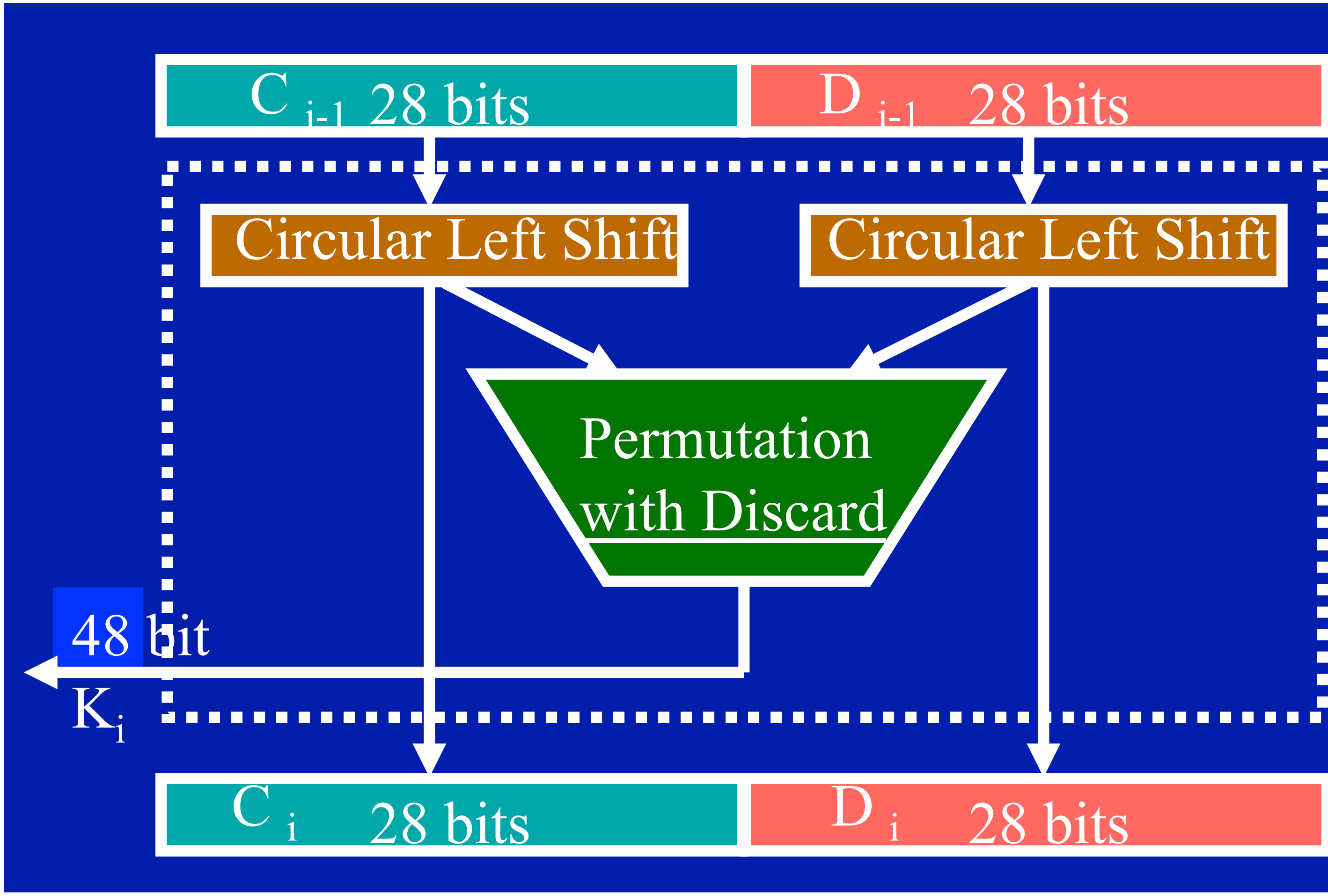
8 rows

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



Rounds i = 1,2,9,16:
left circular shift 1 bit

Other rounds:
left circular shift 2 bits

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

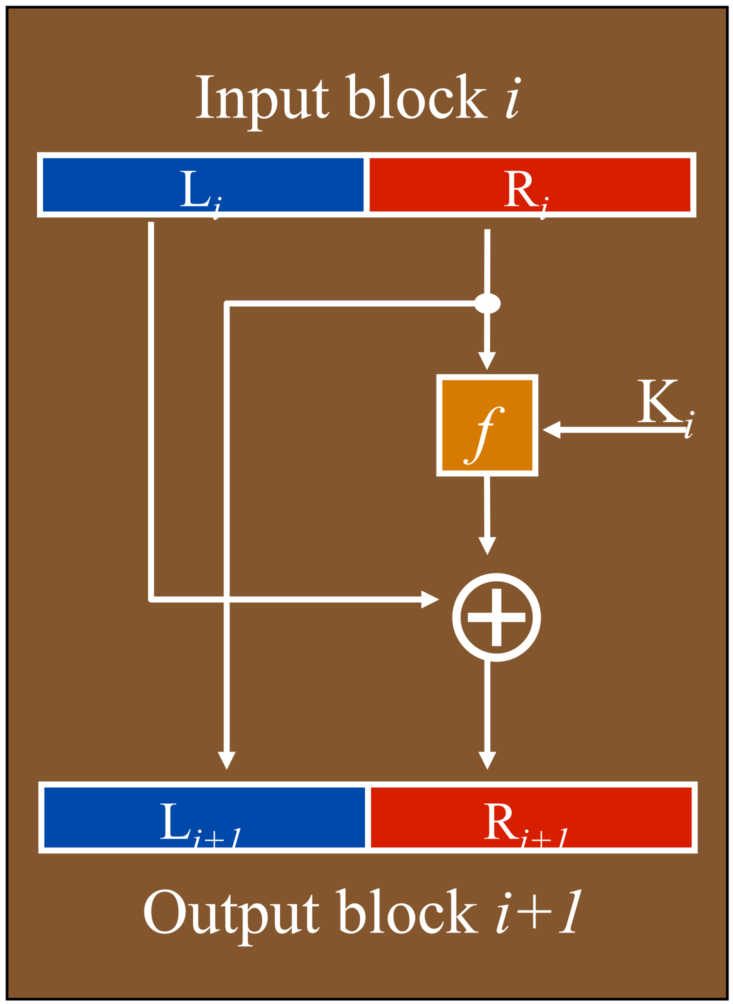
Bits left out:
9,18,22,25

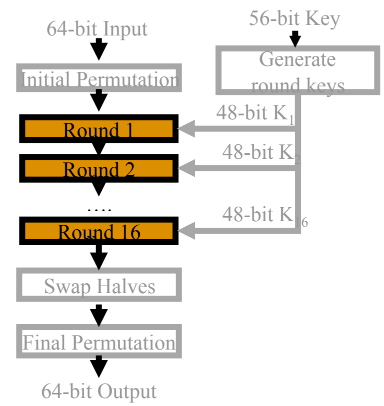
Right half of K_i = permutation of D_i

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

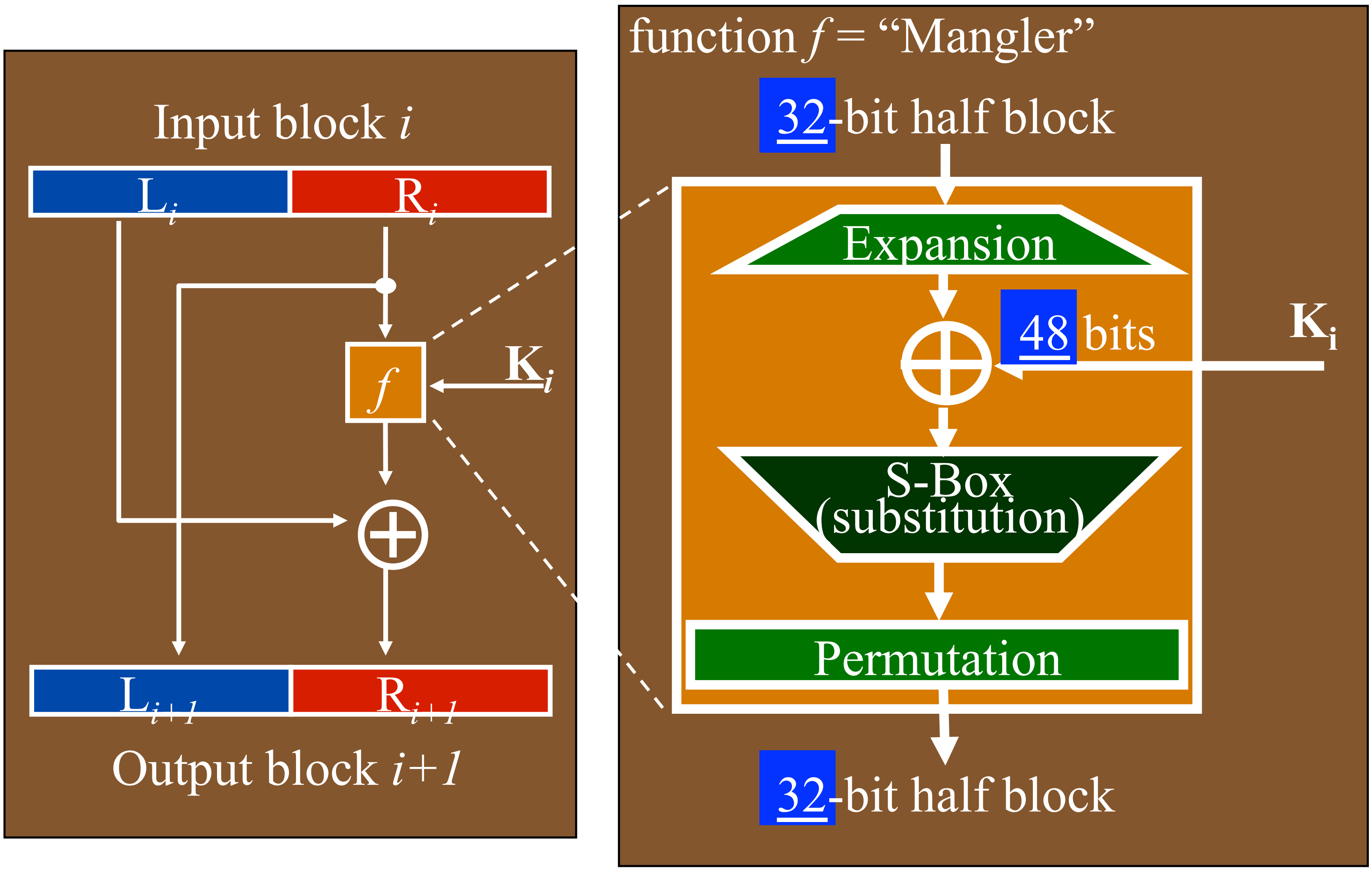
Bits left out:
35,38,43,54

One DES (Feistel) Round





DES Round: *f* (Mangler) Function



f: Expansion Function

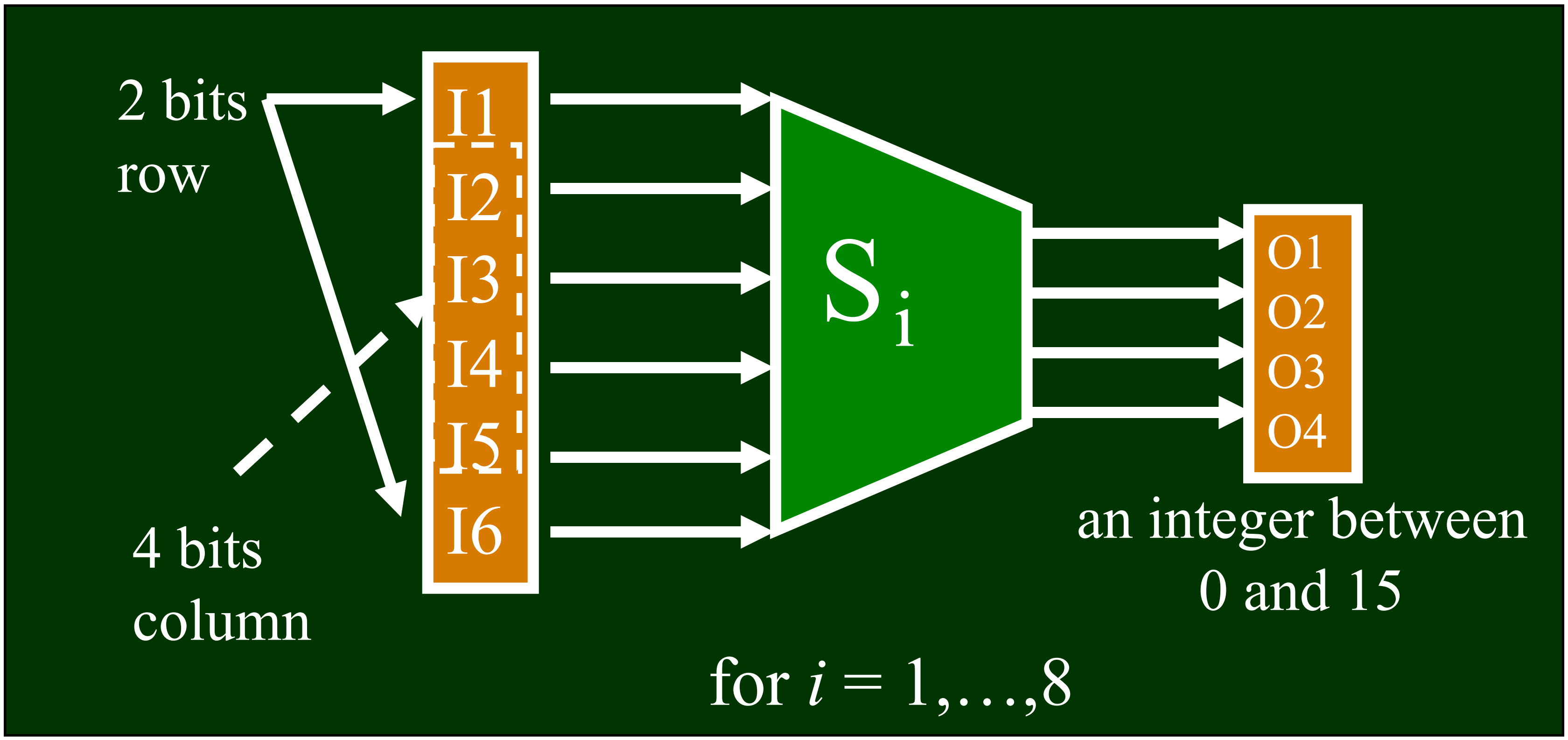
- 32 bits → 48 bits

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-Box (Substitute, Shrink)

- 48 bits ➔ 32 bits
  - 48 bit is broken into eight 6-bit chunks.
  - 6 bits are used to select a 4-bit substitution
  - i.e., for every output, there are four inputs that map to it



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

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

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

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

WILLIAM & MARY

- 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
- *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)

WILLIAM & MARY

- 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
- Ten years ago it took $250,000 to build a machine that could crack DES in a few hours
- Now?

Cryptanalysis of DES

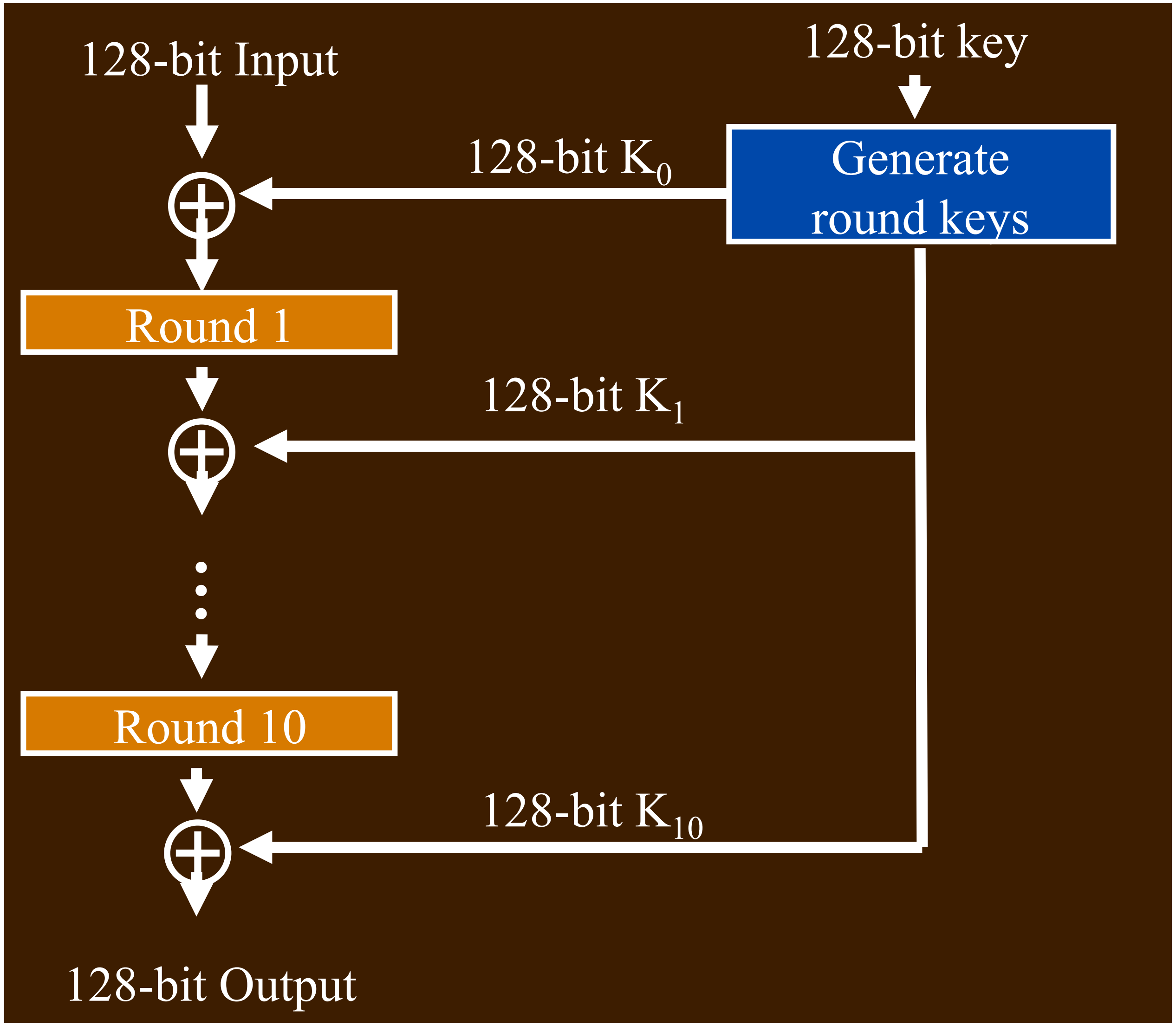
- Differential cryptanalysis exploits differences between encryptions of two different plaintext blocks
  - provides insight into possible key values
  - DES well designed to defeat differential analysis
- Linear cryptanalysis requires known plaintext / ciphertext pairs, analyzes relationships to discover 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 (Advanced Encryption Standard)

Overview

- Selected from an open competition, organized by NSA
  - winner: Rijndael algorithm, standardized as AES
  - A short history: http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html
- 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
- 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

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}

↓

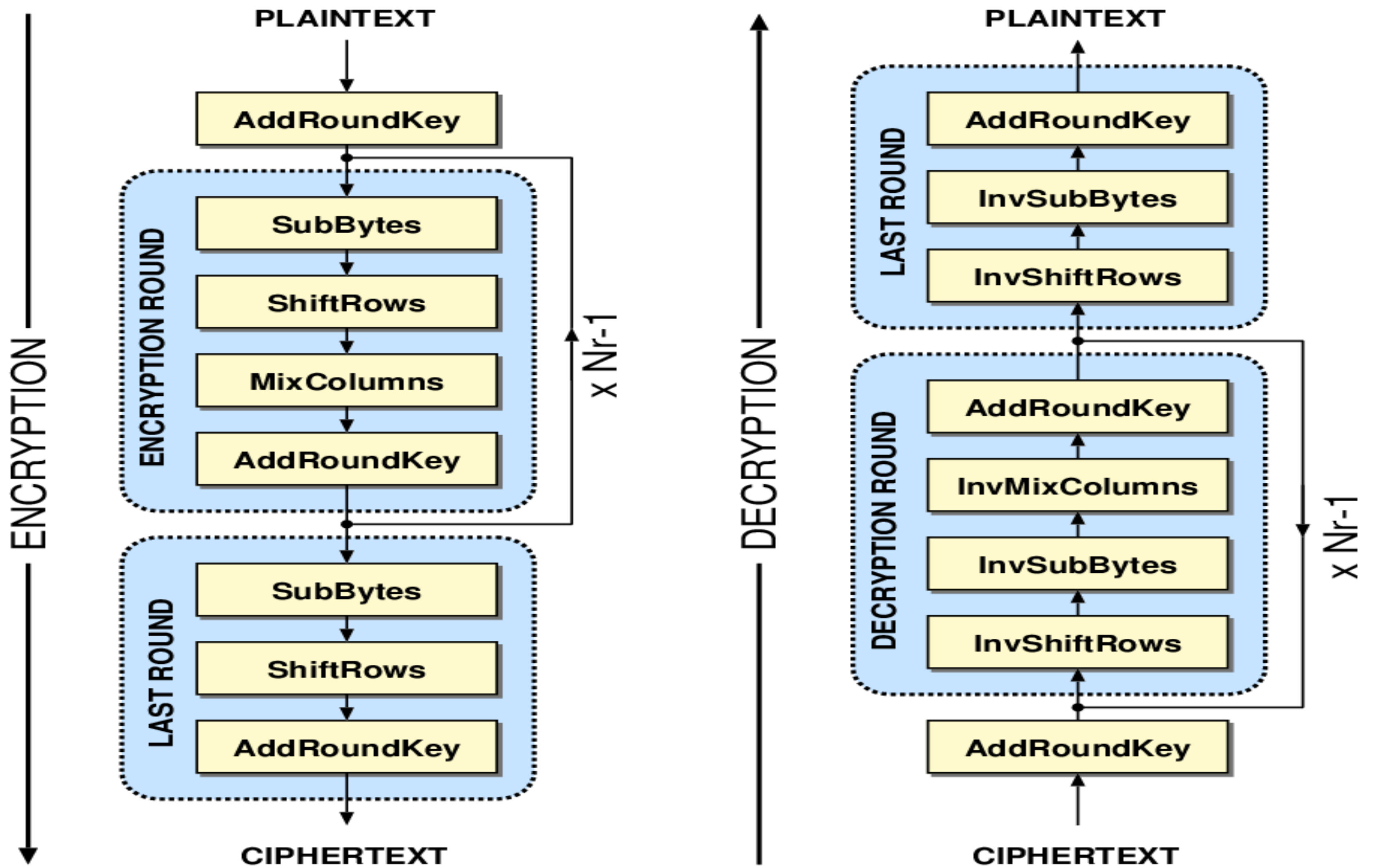
a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

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

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 shifts left 1 column
   - row 2 of the state shifts left 2 columns
   - row 3 of the state shifts left 3 columns
3. Apply MixColumn function to each column of state
   - last round omits this step

AES Encryption/Decryption



Round Step 1. AES S-Box

WILLIAM & MARY

- Each byte of state is replaced by a value from following table
  - eg. byte with value 0x95 is replaced by byte in row 9 column 5, which has value 0x2A

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
	4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-Box (Cont'd)

The S-Box is what makes AES a non-linear cipher

For every value of b there is a unique value for b'

- It is faster to use a substitution table (and easier).

$$
\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} +
\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}
$$

$x = b^{-1}$ in $GF(2^8)$, i.e., x is the inverse of byte b

S-Box Example

- The S-Box is what makes AES a non-linear cipher

State

50	10	D0	81
60	20	4A	93
70	30	E1	A1
00	C0	F7	AF

Sbox(50)	Sbox(10)	Sbox(D0)	Sbox(81)
Sbox(60)	Sbox(20)	Sbox(4A)	Sbox(93)
Sbox(70)	Sbox(30)	Sbox(E1)	Sbox(A1)
Sbox(00)	Sbox(C0)	Sbox(F7)	Sbox(AF)

After SubBytes

53	CA	70	0C
D0	B7	D6	DC
51	04	F8	32
63	BA	68	79

Round Step 2. Rotate (Example)

WILLIAM & MARY

Before Shift Rows

53	CA	70	0C
D0	B7	D6	DC
51	04	F8	32
63	BA	68	79

→

After Shift Rows

53	CA	70	0C
B7	D6	DC	D0
F8	32	51	04
79	63	BA	68

Round Step 3. MixColumn Function

- Applied to each column of the state
- For each column, each byte $a_i \ldots a_{i+3}$ of the column is used to look up four 4-byte intermediate columns $t_i \ldots t_{i+3}$ from a table (next slide)
- The intermediate columns $t_i \ldots t_{i+3}$ are then combined (next slide + 1):
  - rotate vertically so top octet of t_i is in the same row as input octet (a_i)
  - XOR the four rotated columns together

MixColumn… (Cont'd)

- Part of the MixColumn table:

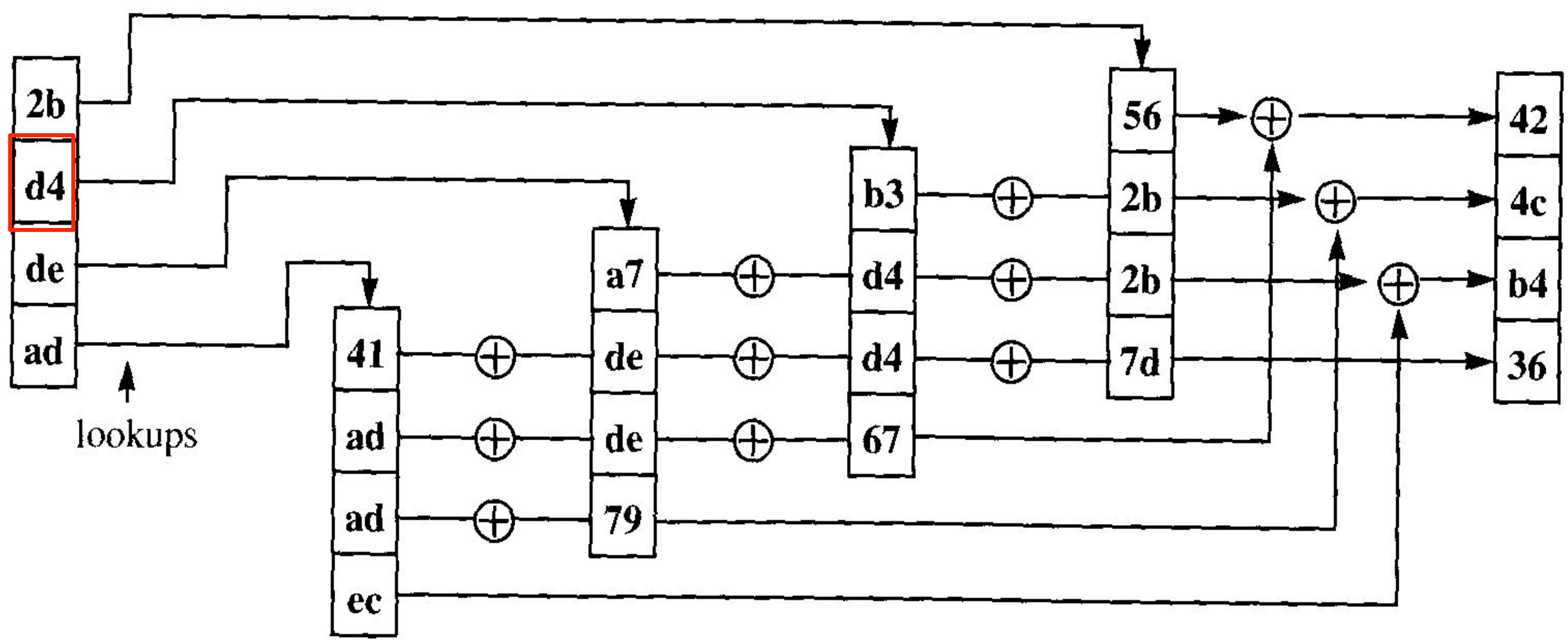
right (low-order) nibble (4 bits)

left (high-order) nibble (4 bits)

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	00	02	04	06	08	0a	0c	0e	10	12	14	16	18	1a	1c	1e
	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
	00	03	06	05	0c	0f	0a	09	18	1b	1e	1d	14	17	12	11
1	20	22	24	26	28	2a	2c	2e	30	32	34	36	38	3a	3c	3e
	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
	30	33	36	35	3c	3f	3a	39	28	2b	2e	2d	24	27	22	21
2	40	42	44	46	48	4a	4c	4e	50	52	54	56	58	5a	5c	5e
	20	21	22	23	24	25	26	27	28	29	2a	2b	2c	2d	2e	2f
	20	21	22	23	24	25	26	27	28	29	2a	2b	2c	2d	2e	2f
	60	63	66	65	6c	6f	6a	69	78	7b	7e	7d	74	77	72	71
	60	62	64	66	68	6a	6c	6e	70	72	74	76	78	7a	7c	7e
d	bb	b9	bf	bd	b3	b1	b7	b5	ab	a9	af	ad	a3	a1	a7	a5
	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	da	db	dc	dd	de	df
	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	da	db	dc	dd	de	df
	6b	68	6d	6e	67	64	61	62	73	70	75	76	7f	7c	79	7a
e	db	d9	df	dd	d3	d1	d7	d5	cb	c9	cf	cd	c3	c1	c7	c5
	e0	e1	e2	e3	e4	e5	e6	e7	e8	e9	ea	eb	ec	ed	ee	ef
	e0	e1	e2	e3	e4	e5	e6	e7	e8	e9	ea	eb	ec	ed	ee	ef
	3b	38	3d	3e	37	34	31	32	23	20	25	26	2f	2c	29	2a
f	fb	f9	ff	fd	f3	f1	f7	f5	eb	e9	ef	ed	e3	e1	e7	e5
	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	fa	fb	fc	fd	fe	ff
	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	fa	fb	fc	fd	fe	ff
	0b	08	0d	0e	07	04	01	02	13	10	15	16	1f	1c	19	1a

MixColumn... (Cont'd)

- Example



Generating Round Keys in AES-128

The key (16 bytes) is arranged in 4 columns of 4 rows, as for the input (plaintext) block

Deriving the round keys makes use of a table of constants:

Round i	Constant c_i
1	0x01
2	0x02
3	0x04
4	0x08
5	0x10
6	0x20
7	0x40
8	0x80
9	0x1b
10	0x36

Removes symmetry and linearity from key expansion

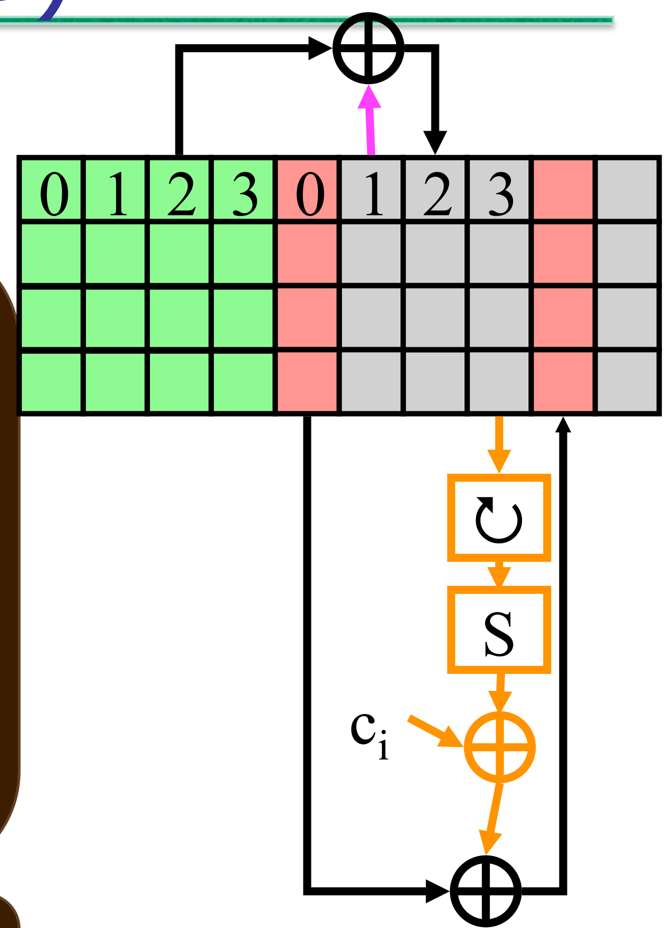
Round Keys… (Cont'd)

For i^{th} round of keys, i = 1..10

for column index j = 0
temp = column 3 of $(i-1)^{th}$ (previous) round
rotate temp upward one byte
S-Box transform each byte of temp
XOR first byte of temp with c_i

for column index j = 1..3
temp = column j-1 of i^{th} (this) round

result = temp XOR j^{th} column of key round i-1



Round Keys… (Cont'd)

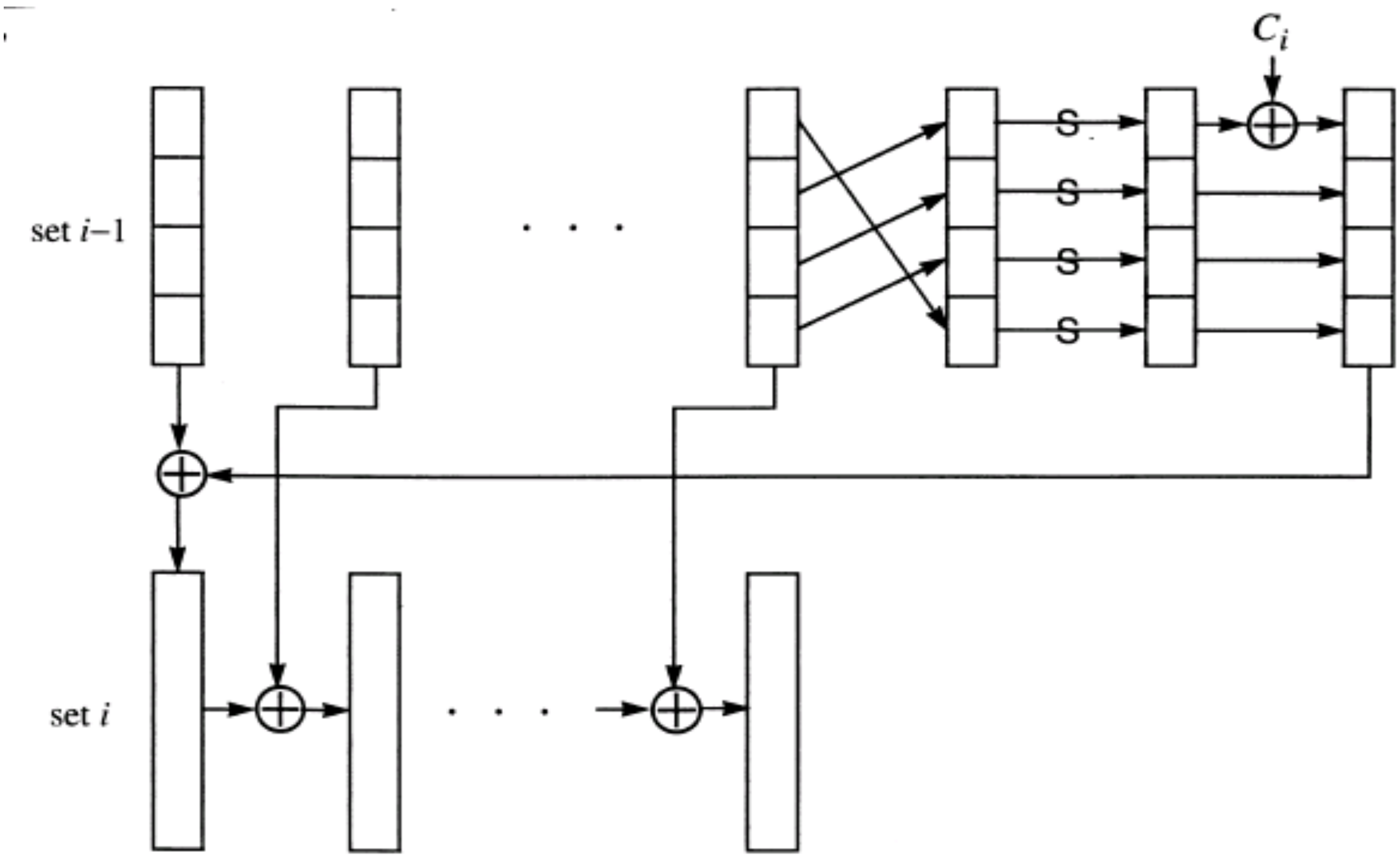


Figure 3-30. Rijndael key expansion, iteration step, $N_k \le 6$

Key Expansion Rationale

- Designed to resist known attacks
- Design criteria include
  - knowing part of the key doesn't make it easy to find entire key
  - key expansion must be invertible, but enough non-linearity to hinder analysis
  - should be fast to compute, simple to describe and analyze
  - key bits should be diffused into the round keys

Mathematics

AES Operates on the binary field $GF(2^8)$

- this can be represented as a polynomial $b(x)$ with binary coefficients $b \in \{0,1\}$:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

Multiplication in $GF(2^8)$ consists of multiplying two polynomials modulo an irreducible polynomial of degree 8

- AES uses the following irreducible polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

AES-128 Decryption (Conceptual)

WILLIAM & MARY

- 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 *(next slide)*
  - inverse of rotation in one direction is rotation in other direction
  - inverse of MixColumn is just the inverse table *(next slide + 1)*

Inverse S-Box

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	52	9	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	8	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	0	8c	bc	d3	0a	f7	e4	58	5	b8	b3	45	6
	7	d0	2c	1e	8f	ca	3f	0f	2	c1	af	bd	3	1	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	7	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	4	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

InvMixColumn

right (low-order) nibble (4 bits)

left (high-order) nibble (4 bits)

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	00	0e	1c	12	38	36	24	2a	70	7e	6c	62	48	46	54	5a
	00	09	12	1b	24	2d	36	3f	48	41	5a	53	6c	65	7e	77
	00	0d	1a	17	34	39	2e	23	68	65	72	7f	5c	51	46	4b
	00	0b	16	1d	2c	27	3a	31	58	53	4e	45	74	7f	62	69
1	e0	ee	fc	f2	d8	d6	c4	ca	90	9e	8c	82	a8	a6	b4	ba
	90	99	82	8b	b4	bd	a6	af	d8	d1	ca	c3	fc	f5	ee	e7
	d0	dd	ca	c7	e4	e9	fe	f3	b8	b5	a2	af	8c	81	96	9b
	b0	bb	a6	ad	9c	97	8a	81	e8	e3	fe	f5	c4	cf	d2	d9
2	db	d5	c7	c9	e3	ed	ff	f1	ab	a5	b7	b9	93	9d	8f	81
	3b	32	29	20	1f	16	0d	04	73	7a	61	68	57	5e	45	4c
	bb	b6	a1	ac	8f	82	95	98	d3	de	c9	c4	e7	ea	fd	f0
	7b	70	6d	66	57	5c	41	4a	23	28	35	3e	0f	04	19	12
d	0c	02	10	1e	34	3a	28	26	7c	72	60	6e	44	4a	58	56
	0a	03	18	11	2e	27	3c	35	42	4b	50	59	66	6f	74	7d
	67	6a	7d	70	53	5e	49	44	0f	02	15	18	3b	36	21	2c
	b1	ba	a7	ac	9d	96	8b	80	e9	e2	ff	f4	c5	ce	d3	d8
e	37	39	2b	25	0f	01	13	1d	47	49	5b	55	7f	71	63	6d
	a1	a8	b3	ba	85	8c	97	9e	e9	e0	fb	f2	cd	c4	df	d6
	0c	01	16	1b	38	35	22	2f	64	69	7e	73	50	5d	4a	47
	7a	71	6c	67	56	5d	40	4b	22	29	34	3f	0e	05	18	13
f	d7	d9	cb	c5	ef	e1	f3	fd	a7	a9	bb	b5	9f	91	83	8d
	31	38	23	2a	15	1c	07	0e	79	70	6b	62	5d	54	4f	46
	dc	d1	c6	cb	e8	e5	f2	ff	b4	b9	ae	a3	80	8d	9a	97
	ca	c1	dc	d7	e6	ed	f0	fb	92	99	84	8f	be	b5	a8	a3

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