CIS 6930/4930 Computer and Network Security

Project requirements

Project Requirement

- Form a team of 3 people to complete the course project.
- The project has 100pts + 20pts (extra credit)
- Report requirement:
  – at least 5 pages, at most 1.5 line spacing,
  – time New Roman, 11pt.
- Report deadline: April 30th

Project Goal

- Train yourself to think about security
- Learn to **collaborate** with others
- Learn to find useful resources
- Learn to write clear and well-organized technical reports.

Assigned Project

- Secure Instant Point-to-Point (P2P) Messaging
- In this project, you need to design a secure instant messaging tool for Alice and Bob (like Gtalk, skype or ICQ chat). The system supports the following functions
  - Alice and Bob can use the tool to send instant messages to each other.

Assigned Project (Cont'd)

- Project description (Cont'd)
  - Alice and Bob share the same password (or passphrase), they must use the password to set up the tool to correctly encrypt and decrypt messages shared between each other.
  - Each message during Internet transmission must be encrypted using a 56-bit key.

Computer Language

- You can use any computer language (Java, C++, Python) and leverage any existing open-source software, tools, or commands (e.g., md5sum, sha1sum) to design the system.

Design Issues

- With a 56-bit key, what cipher you should use?
- DO NOT directly use the password as the key, how can you generate the same key between Alice and Bob to encrypt messages?
- What will be used for padding?

Design Issues (Cont'd)

- A graphical user interface (GUI) is preferred.
  - Display ciphertext and plaintext
- How should Alice and Bob set up an initial connection and also maintain the connection with each other on the Internet?
  - You may refer to socket/network programming in a particular computer language

Extra Credits

- Design a key management mechanism to periodically update the key used between Alice and Bob. Justify why the design can enhance security.

Research Paper

- In lieu of the assignment project, you can also write a research paper.
- If you plan to write a research paper, you need to submit a proposal before Feb 20th.
- Your research paper is due on April 30th.

Research paper (Cont'd)

- Identify a security-related problem
- Analyze the problem, propose solution and verity its effectiveness
- It may be a new problem, i.e., nobody has considered before
- Or it may be a problem already addressed by others. But you have a better solution
- Validation: show your approach is workable or better
  - Theoretical analysis + experimental verification

Suggested Topics

- Behavior biometrics
- Location tracking
- Attacks against automobile systems
- Security of Bitcoins
- Trust management in social networks
- Detection of fake Wi-Fi spots
- Privacy-preserving for the cloud service
- Security in the smart grid
- Threats to the implantable medical devices
- Big data security
- Other topics you are interested
  - PhD students are encouraged to connect the project with your research topic

Places for good references

- DBLP
  - http://www.informatik.uni-trier.de/~ley/db/
- Citeseer
  - http://citeseer.ist.psu.edu
- ACM Portal
  - http://dl.acm.org/
- IEEE Xplore
  - http://ieeexplore.ieee.org/Xplore/guesthome.jsp

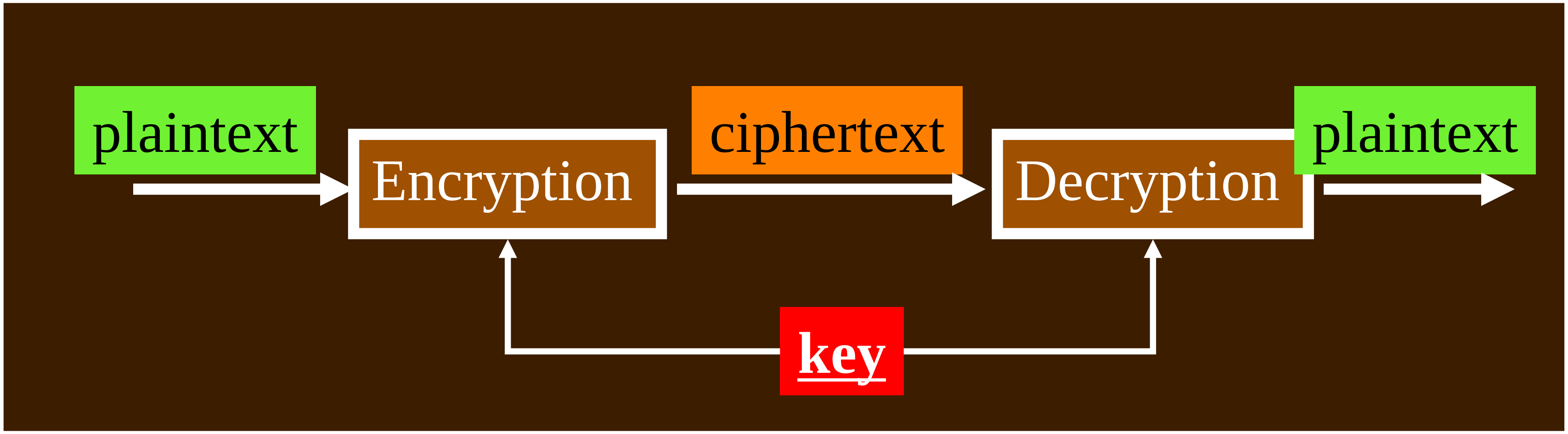
Places for good references (Cont'd)

- Major security conferences
  - IEEE symposium on Security and Privacy (Oakland conference, IEEE S&P)
  - ACM Conference on Computer and communications security (CCS)
  - USENIX Security symposium
  - Network and Distributed System Symposium (NDSS)
  - Annual International Cryptology Conference (crypto)
  - Eurocrypto Conference (enrocrypto)
  - Top database and networking conferences
    - SIGMOD, VLDB, ICDE,WWW,…
    - SIGCOMM, INFOCOMM,…

CIS 6930/4930 Computer and Network Security

Topic 3.1 Secret Key Cryptography – 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 securely
- Also known as *symmetric key* cryptography, or *shared key* cryptography

Applications of Secret Key Crypto

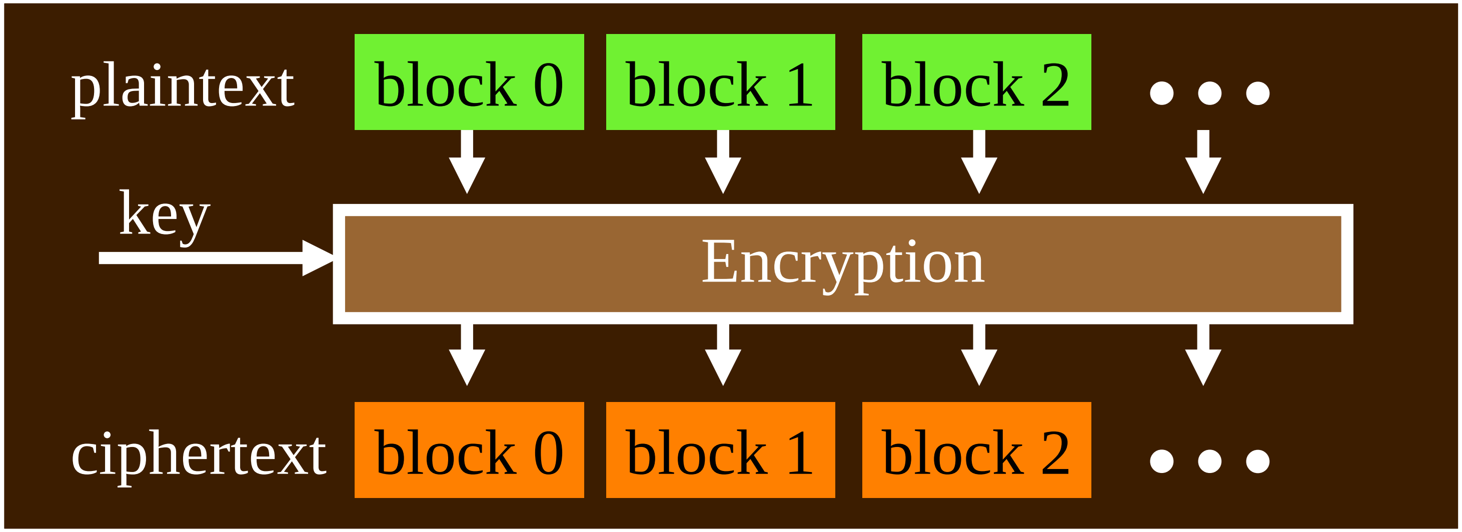
- Communicating securely over an insecure channel
  - Alice encrypts using shared key
  - Bob decrypts result using same shared key
- *Authentication*
  - Bob can verify if a message is generated by Alice

Applications… (Cont'd)

- *Message integrity*
  - Alice computes a *message integrity code* (MIC) from the message using the shared key
  - Bob decrypts the MIC on receipt, and verifies that it agrees with message contents

Generic Block Encryption

- Converts one input plaintext block of fixed size *n* bits to an output ciphertext block also of *n* bits



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
$X \mid Y$	Concatenation of X and Y
$K\{m\}$	Message *m* encrypted with secret key K

Two Principles for Cipher Design

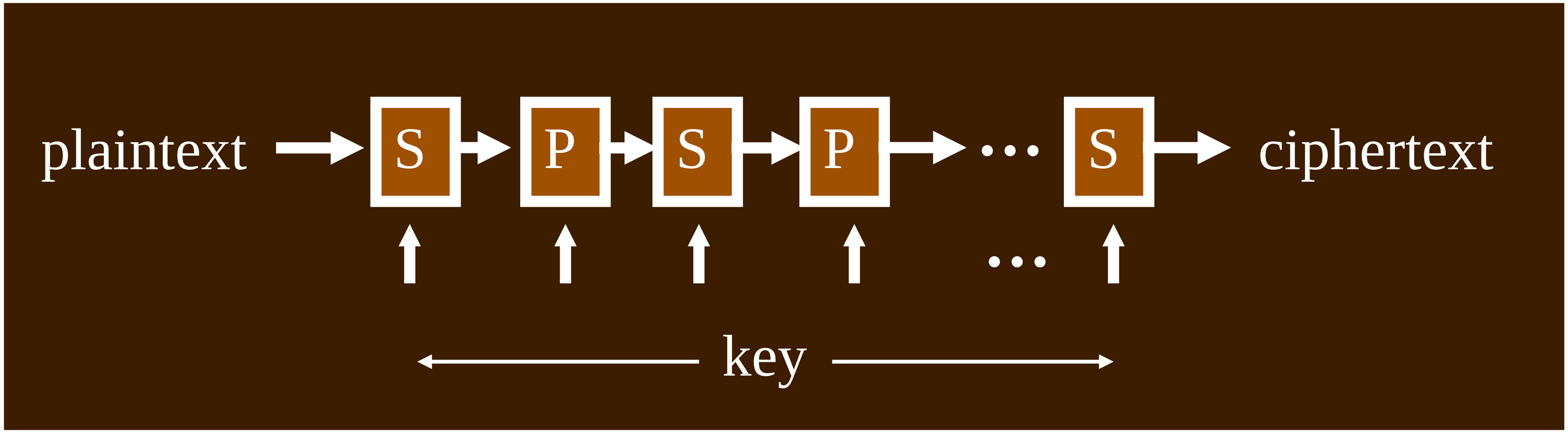
- Confusion:
  - Make the relationship between the <plaintext, key> input and the <ciphertext> output as complex (non-linear) as possible
- Diffusion:
  - Spread the influence of each input bit across many output bits

Exploiting the Principles

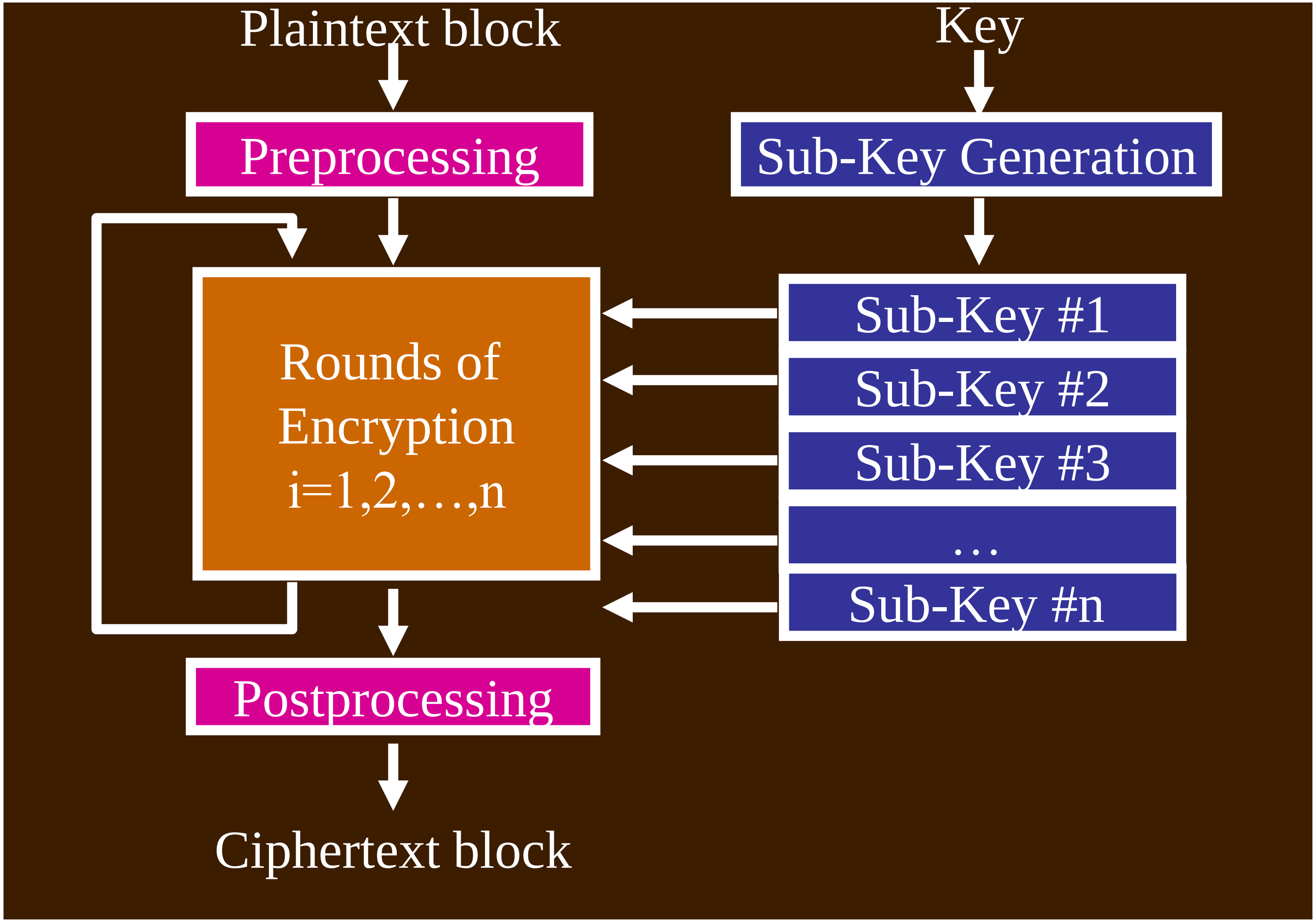
- Idea: use multiple, alternating permutations and substitutions, e.g.,
  - S→P→S→P→S→…
  - P→S→P→S→P→…
- Do they have to alternate? e.g.…
  - S→S→S→P→P→P→S→S→…?
  - Consecutive Ps or Ss do not improve security
- Confusion is mainly accomplished by substitutions
- Diffusion is mainly accomplished by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



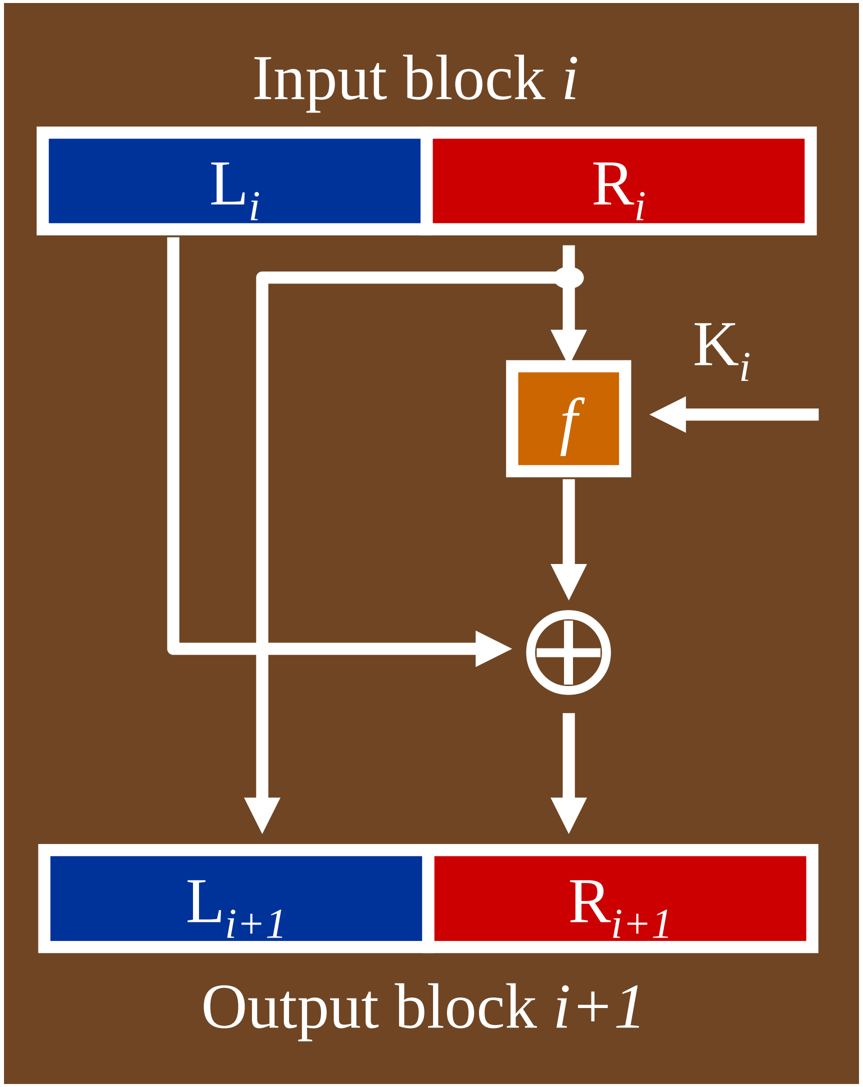
Feistel Ciphers

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

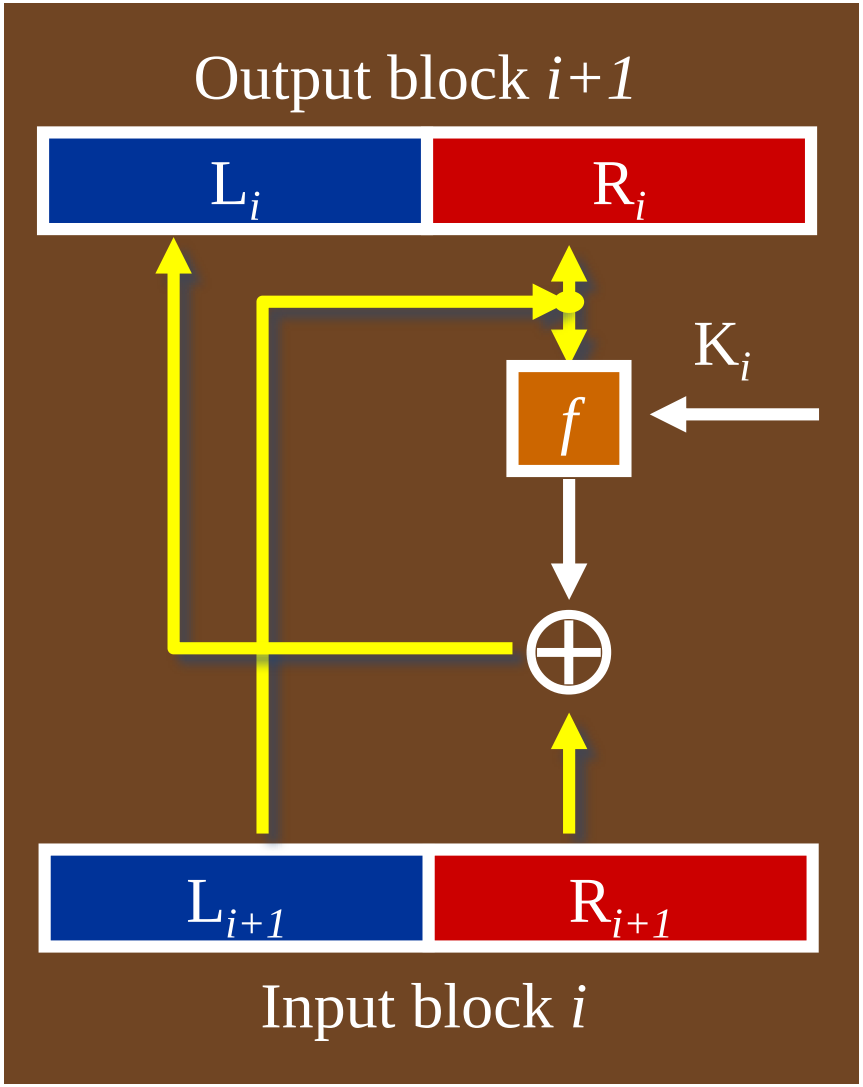
One “Round” of Feistel Encryption

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

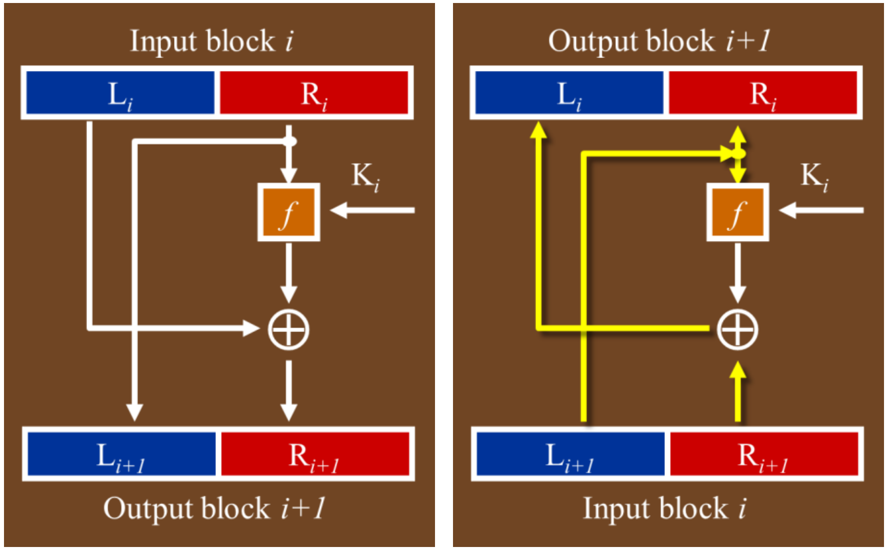
- Just reverse the arrows!
- Why?



Feistel Cipher: Decryption (cont'd)

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

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



Parameters of a Feistel Cipher

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

Summary

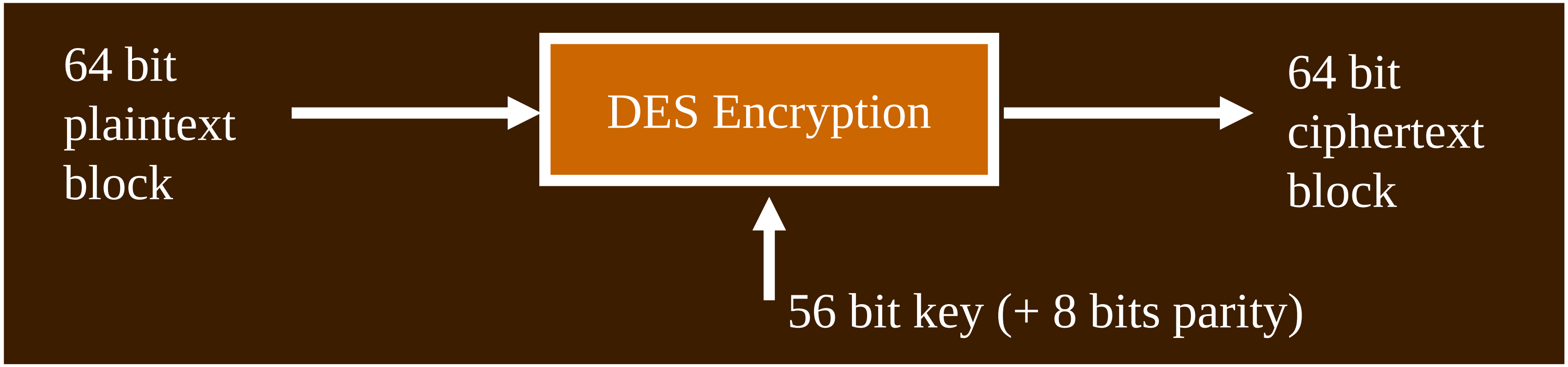
- Decryption is same as encryption, only reversing the order in which round keys are applied
  - Reversability of Feistel cipher derives from reversability of xor $\oplus$
- Function *f* can be anything
  - Hopefully something easy to compute
  - There is no need to invert *f*

DES (Data Encryption Standard)

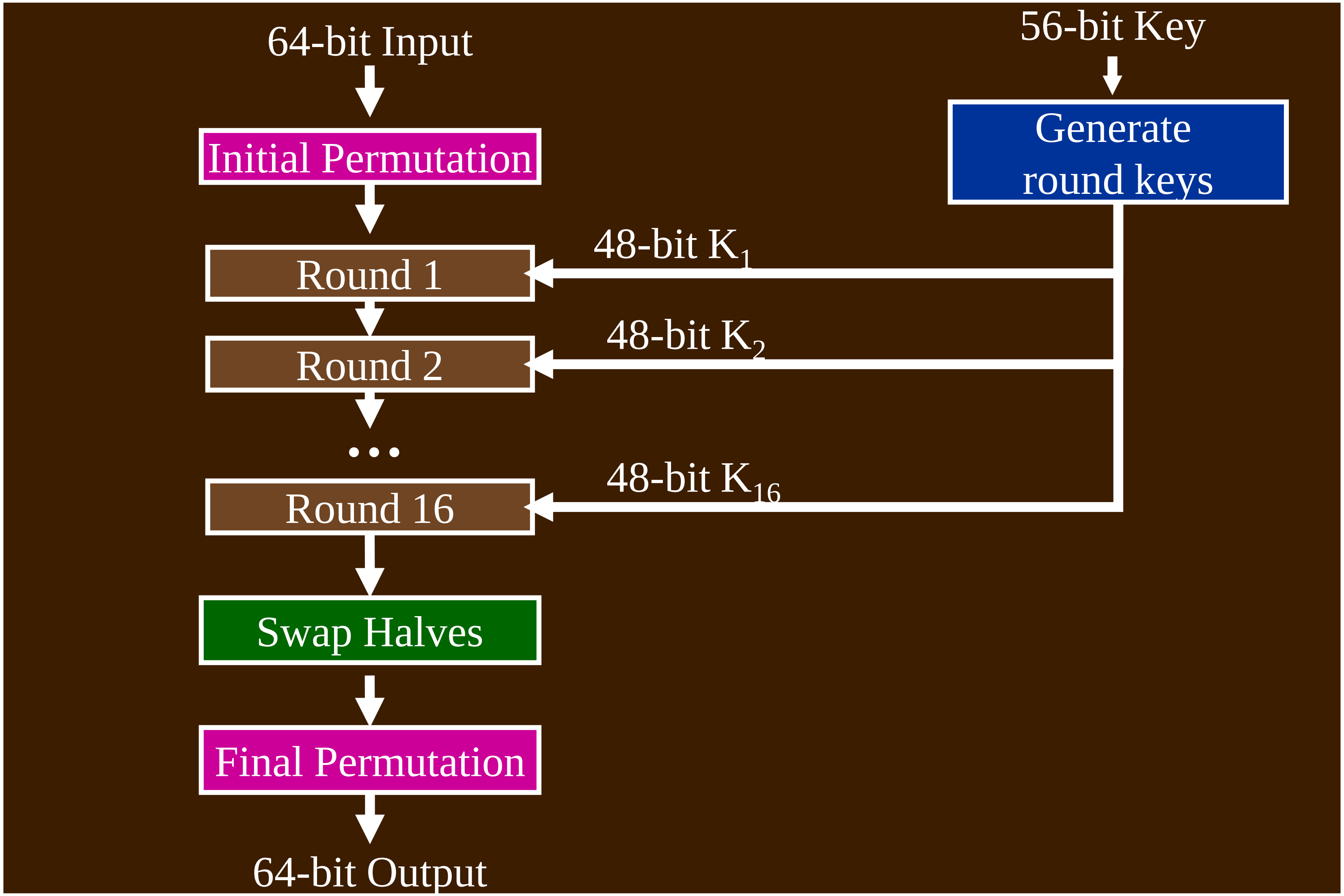
- Standardized in 1976 by NBS
  - proposed by IBM,
  - Feistel cipher
- Criteria (official)
  - provide high level of security
  - security must reside in key, not algorithm
  - not patented
  - efficient to implement in hardware
  - must be slow to execute in software

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

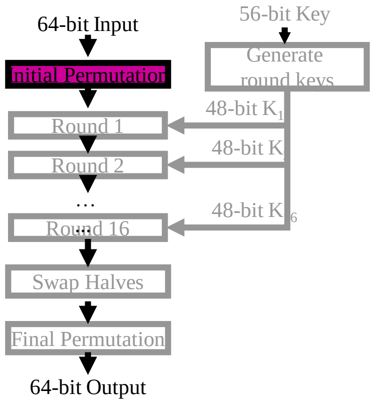


DES Top Level View



Initial and Final Permutations

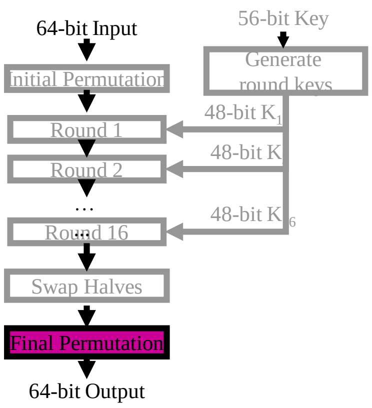
- Initial permutation given below
  – input bit #58→output bit #1, input bit #50→ output bit #2, …



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
  - …

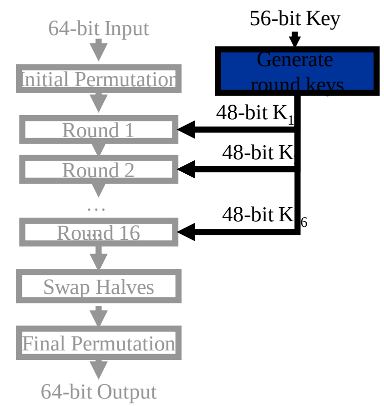


Initial… (Cont'd)

- 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



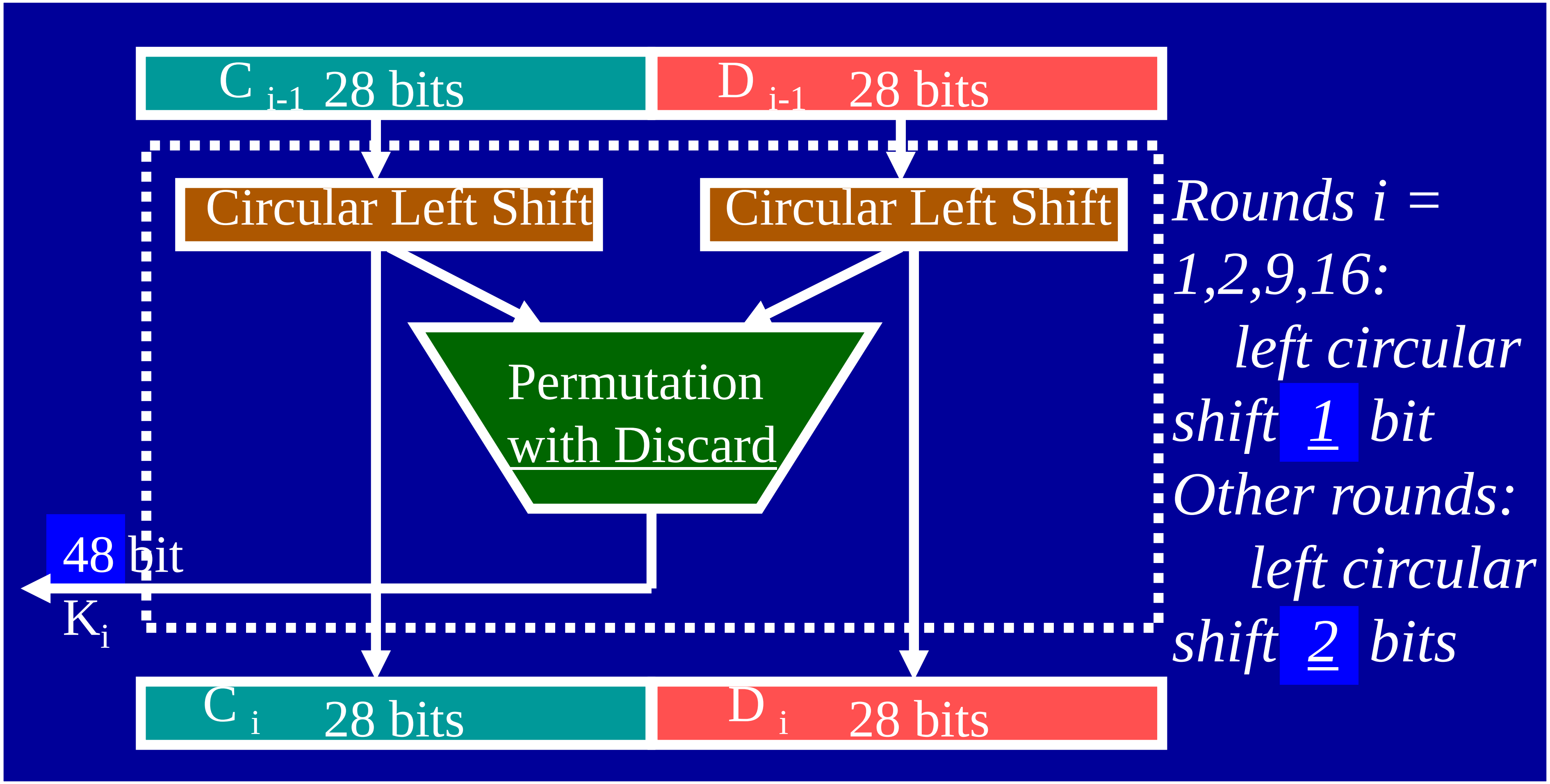
7 columns

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

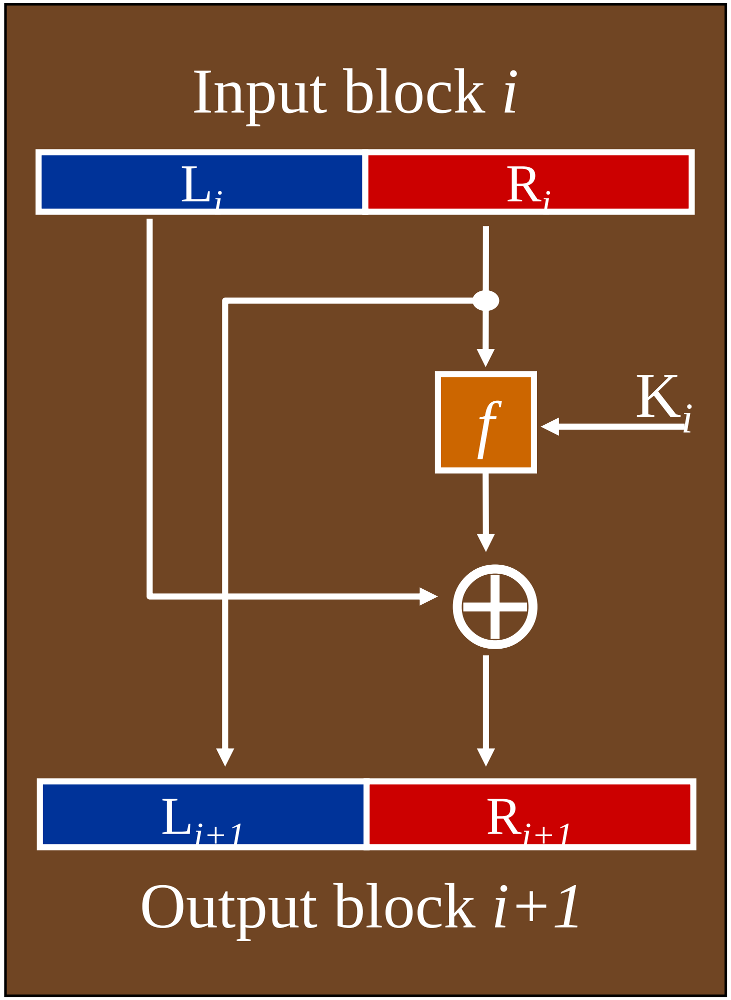
Bits dicarded: 9,18,22,25

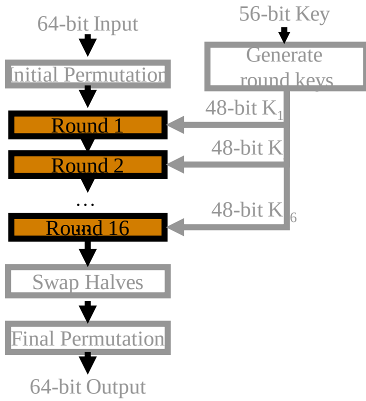
Right half of K_i = permutation of D_i

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

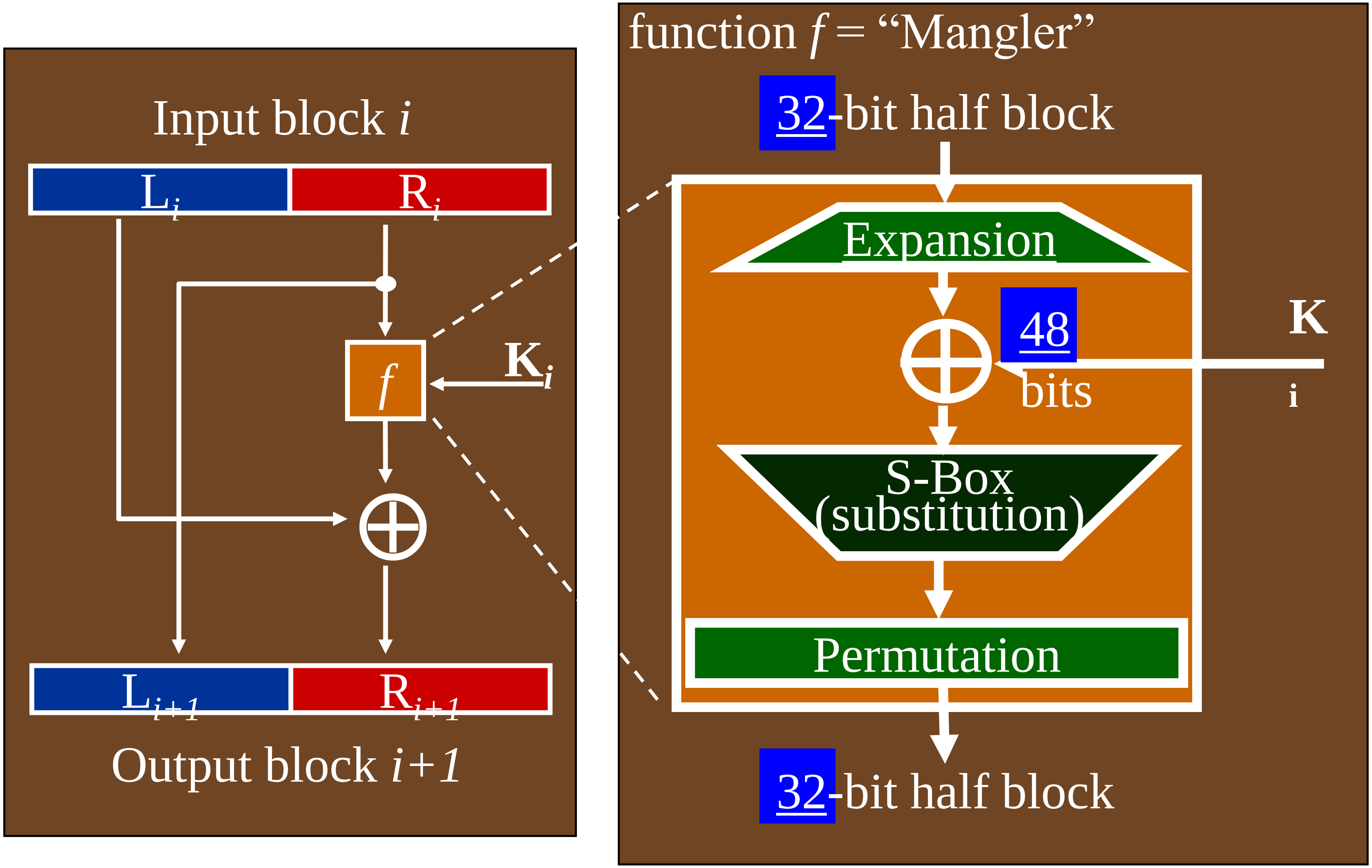
Bits discarded: 35,38,43,54

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