Block Ciphers and the Data Encryption Standard (DES)

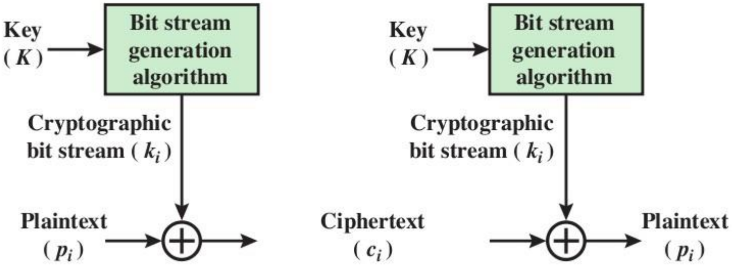
Dr. Xiaochen Yuan

2021/2022

Contents

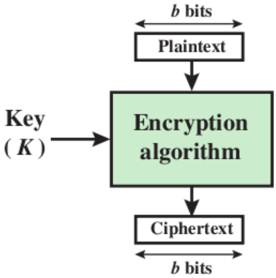
Stream Ciphers

- Encrypts a digital data stream one bit or one byte at a time
- **One time pad** is example; but practical limitations
- Typical approach for stream cipher:
  - Key (K) used as input to bit-stream generator algorithm
  - Algorithm generates cryptographic bit stream (k_i) used to encrypt plaintext
  - Users share a key; use it to generate keystream



Block Ciphers

- Encrypt a block of plaintext as a whole to produce same sized ciphertext
- Typical block sizes are **64** or **128** bits
- Modes of operation used to apply block ciphers to larger plaintexts



Reversible and Irreversible Mappings

- n-bit block cipher takes n bit plaintext and produces n bit ciphertext
- 2^n possible different plaintext blocks
- Encryption must be **reversible** (decryption possible)
- Each plaintext block must produce **unique** ciphertext block
- Total transformations is $2^n!$

Reversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

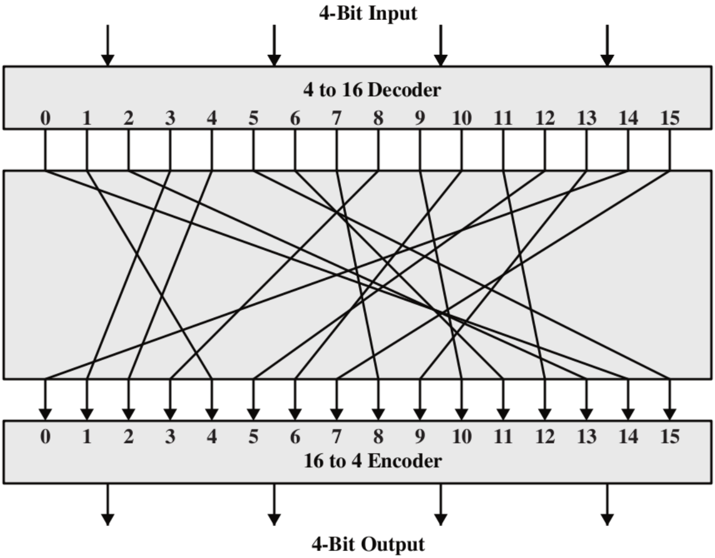
Plaintext	Ciphertext
00	11
01	10
10	01
11	01

Ideal Block Cipher

- n-bit input maps to 2^n possible input states
- Substitution used to produce 2^n output states
- Output states map to n-bit output
- Ideal block cipher allows **maximum** number of possible encryption mappings from plaintext block

- Problems with ideal block cipher:
  - Small block size: equivalent to classical substitution cipher; cryptanalysis based on statistical characteristics feasible
  - Large block size: key must be very large; performance/implementation problems

General Block Substitution



Encryption/Decryption Tables

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

- Most general form of block cipher.
- Can be used to define any reversible mapping between plaintext and ciphertext.
- Feistel refers to it as: ***ideal block cipher*** --- it allows for the maximum number of possible encryption mappings from the plaintext block.

Feistel Structure for Block Ciphers

- Feistel proposed applying two or more simple ciphers in sequence so final result is cryptographically stronger than component ciphers
- n-bit block length; k-bit key length; 2^k transformations
- Feistel cipher alternates: substitutions, transpositions (permutations)
- Applies concepts of diffusion and confusion
- Applied in many ciphers today

- Approach:
  - Plaintext split into halves
  - Subkeys (or round keys) generated from key
  - Round function, F, applied to right half
  - Apply substitution on left half using XOR
  - Apply permutation: interchange to halves

Diffusion and Confusion

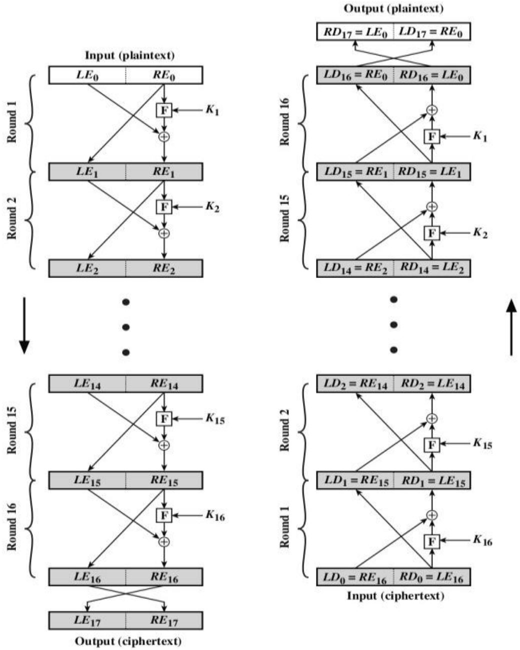
Diffusion

- ***Statistical nature*** of plaintext is reduced in ciphertext
- E.g. A plaintext letter affects the value of many ciphertext letters
- How: **repeatedly** apply permutation (transposition) to data, and then apply function

Confusion

- Make relationship between ciphertext and key as **complex** as possible
- Even if attacker can find some statistical characteristics of ciphertext, still hard to find key
- How: apply complex (non-linear) substitution algorithm

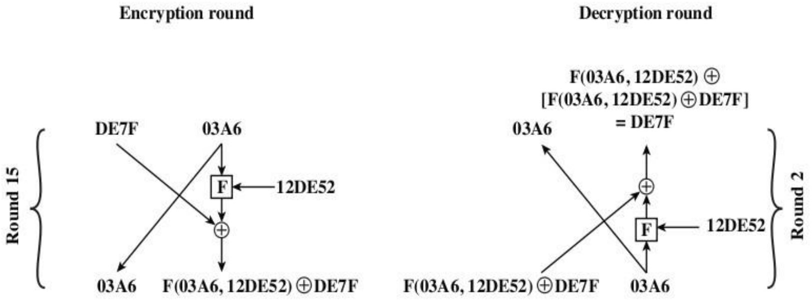
Feistel Encryption and Decryption



Using the Feistel Structure

- Exact implementation depends on various design features
  - **Block size**, e.g. 64, 128 bits: larger values leads to more diffusion
  - **Key size**, e.g. 128 bits: larger values leads to more confusion, resistance against brute force
  - **Number of rounds**, e.g. 16 rounds
  - **Subkey generation algorithm**: should be complex
  - **Round function** F: should be complex
- Other factors include fast encryption in software and ease of analysis
- Trade-off: **security** vs **performance**

Feistel Example



Contents

Data Encryption Standard

- **Symmetric** block cipher
  - 56-bit key, 64-bit input block, 64-bit output block
- One of most used encryption systems in world
  - Developed in 1977 by NBS/NIST
  - Designed by IBM (Lucifer) with input from NSA
  - Principles used in other ciphers, e.g. 3DES, IDEA
- Simplified DES (S-DES)
  - Cipher using principles of DES
  - Developed for education (not real world use)

Contents

Simplified DES

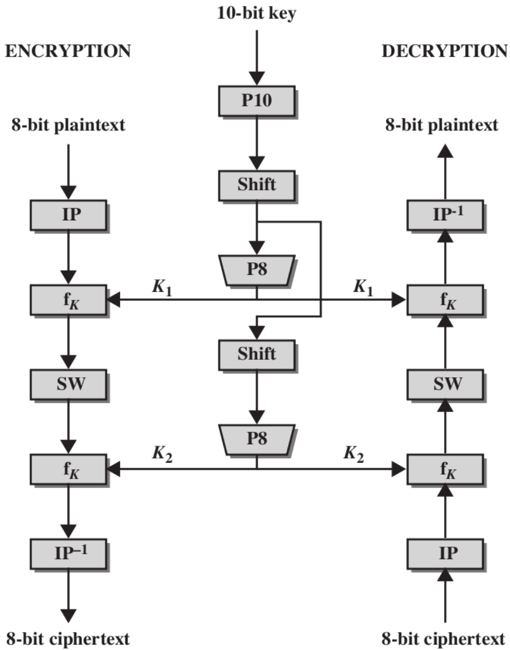
- Input (plaintext) block: 8-bits
- Output (ciphertext) block: 8-bits
- Key: 10-bits
- Rounds: 2
- Round keys generated using **permutations** and **left shifts**

- **Encryption**: initial permutation, round function, switch halves
- **Decryption**: Same as encryption, except round keys used in opposite order

S-DES Algorithm

Initial Permutation

switch halves



S-DES Operations

- **P10** (permutate)

Input :	1	2	3	4	5	6	7	8	9	10
Output:	3	5	2	7	4	10	1	9	8	6

- **P8** (select and permutate)

Input :	1	2	3	4	5	6	7	8	9	10
Output:	6	3	7	4	8	5	10	9		

- **P4** (permutate)

Input:	1	2	3	4
Output:	2	4	3	1

S-DES Operations

- EP (expand and permutate)

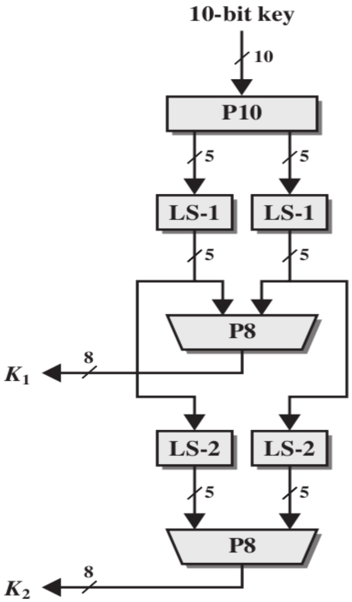
  Input : 1 2 3 4
  Output: 4 1 2 3 2 3 4 1

- IP (initial permutation)

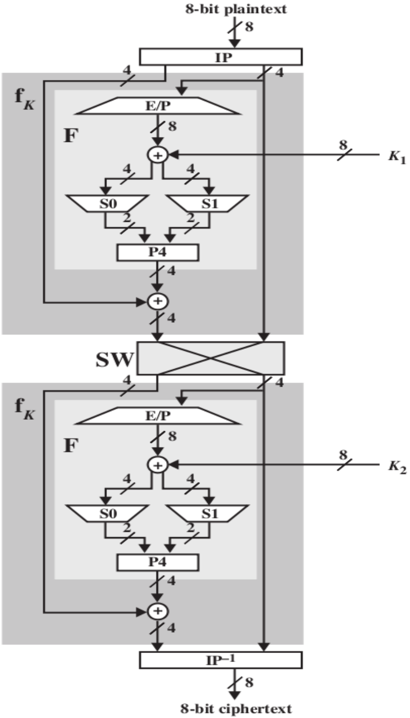
  Input : 1 2 3 4 5 6 7 8
  Output: 2 6 3 1 4 8 5 7

- IP^{-1} (inverse of IP)
- LS-1 (left shift 1 position)
- LS-2 (left shift 2 positions)

S-DES Key Generation



S-DES Encryption Details



S-DES S-Boxes

- S-DES (and DES) perform substitutions using **S-Boxes**
- S-Box considered as a matrix: input used to select row/column; selected element is output
- 4-bit input: $bit_1, bit_2, bit_3, bit_4$
- $bit_1 bit_4$ specifies row (0, 1, 2 or 3 in decimal)
- $bit_2 bit_3$ specifies column
- 2-bit output

$$S0 = \begin{matrix} 01 & 00 & 11 & 10 \\ 11 & 10 & 01 & 00 \\ 00 & 10 & 01 & 11 \\ 11 & 01 & 11 & 10 \end{matrix} \qquad S1 = \begin{matrix} 00 & 01 & 10 & 11 \\ 10 & 00 & 01 & 11 \\ 11 & 00 & 01 & 00 \\ 10 & 01 & 00 & 11 \end{matrix}$$

S-DES Example

- Plaintext: `01110010`
- Key: `1010000010`
- Ciphertext: `01110111`

S-DES Summary

- Educational encryption algorithm
- S-DES expressed as functions:

$$ciphertext = IP^{-1}(f_{K_2}(SW(f_{K_1}(IP(plaintext)))))$$

$$plaintext = IP^{-1}(f_{K_1}(SW(f_{K_2}(IP(ciphertext)))))$$

- Security of S-DES:
  - 10-bit key, 1024 keys: brute force easy
  - If know plaintext and corresponding ciphertext, can we determine key? Very hard

Contents

Comparing DES and S-DES

S-DES

- 8-bit blocks
- 10-bit key: 2 x 8-bit round keys
- IP: 8-bits
- F operates on 4 bits
- 2 S-Boxes
- 2 rounds

DES

- 64-bit blocks
- 56-bit key: 16 x 48-bit round keys
- IP: 64 bits
- F operates on 32 bits
- 8 S-Boxes
- 16 rounds

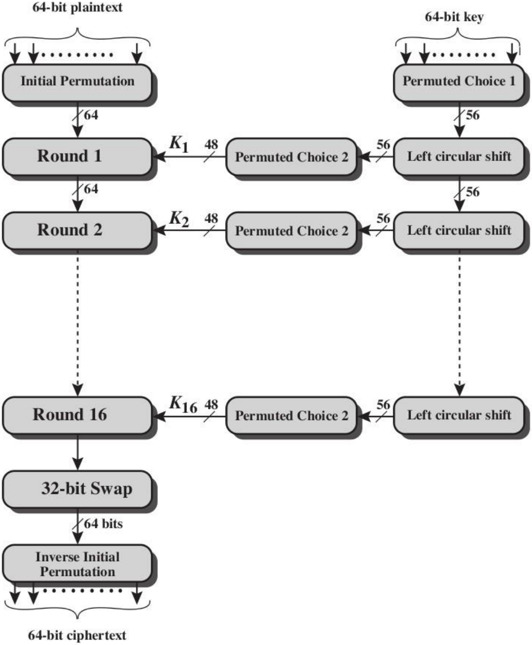
S-DES encryption:

$$ciphertext = IP^{-1}(f_{K_2}(SW(f_{K_1}(IP(plaintext)))))$$

DES encryption:

$$ciphertext = IP^{-1}(f_{K_{16}}(SW(f_{K_{15}}(SW(\ldots(f_{K_1}(IP(plaintext)))$$

General DES Encryption Algorithm



Permutation Tables for DES

(a) Initial Permutation (IP)

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

(b) Inverse Initial Permutation (IP^{-1})

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	41	9	49	17	57	25

Permutation Tables for DES

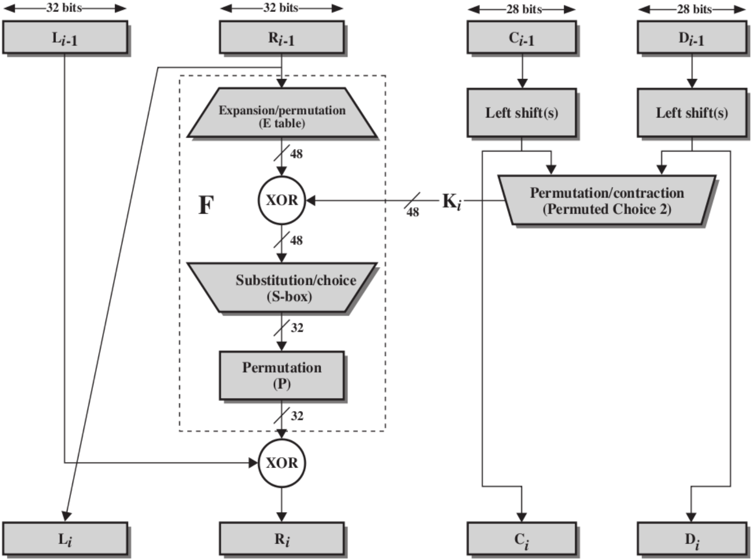
(c) Expansion Permutation (E)

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

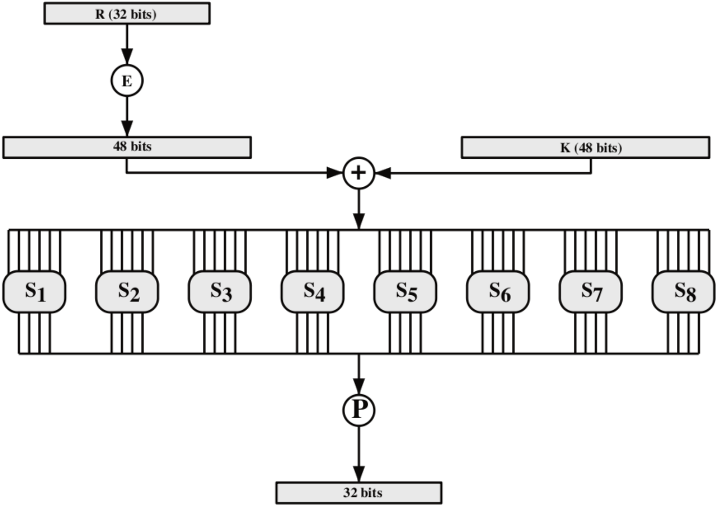
(d) Permutation Function (P)

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

Single Round of DES Algorithm

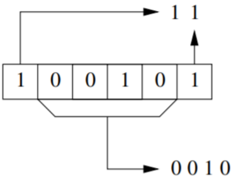


Calculation of F(R,K)



Example of S-BOX

- 6-bit input: $bit_1, bit_2, bit_3, bit_4, bit_5, bit_6$
- $bit_1 bit_6$ specifies row (0, 1, 2 or 3 in decimal)
- $bit_2 \sim bit5$ specifies column
- 4-bit output



Example:

Input: 100101

S-box 1

Output: S1(100101)=8=1000

Definition of DES S-Boxes

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Definition of DES S-Boxes

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

DES Key Schedule Calculation

(a) Input Key

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

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Example

Plaintext: **P** = 0123456789ABCDEF (hexadecimal)
P = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 (64 bits)

Key: **K** = 133457799BBCDFF1 (hexadecimal)
K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001 (64 bits)

C = 1000 0101 1110 1000 0001 0011 0101 0100 0000 1111 0000 1010 1011 0100 0000 0101
Ciphertext: C = 85E813540F0AB405 (hexadecimal)

DES Example

STEP 1: Generate 16 subkeys, 48 bits / subkey

PC-1 permutation using *DES KEY CALCULATION-(a)*:

K+=1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Split into left & right halves:

C_0 = 1111000 0110011 0010101 0101111
D_0 = 0101010 1011001 1001111 0001111

Left Shift using *DES KEY CALCULATION-(d)*:

C_1 = 1110000110011001010101011111
D_1 = 1010101011001100111100011110
C_2 = 1100001100110010101010111111
D_2 = 0101010110011001111000111101
C_3 = 0000110011001010101011111111
D_3 = 0101011001100111100011110101
C_4 = 0011001100101010101111111100
D_4 = 0101100110011110001111010101
C_5 = 1100110010101010111111110000
D_5 = 0110011001111000111101010101
C_6 = 0011001010101011111111000011
D_6 = 1001100111100011110101010101

DES Example

C_7 = 1100101010101111111100001100
D_7 = 0110011110001111010101010110
C_8 = 0010101010111111110000110011
D_8 = 1001111000111101010101011001
C_9 = 0101010101111111100001100110
D_9 = 0011110001111010101010110011
C_{10} = 0101010111111110000110011001
D_{10} = 1111000111101010101011001100
C_{11} = 0101011111111000011001100101
D_{11} = 1100011110101010101100110011
C_{12} = 0101111111100001100110010101
D_{12} = 0001111010101010110011001111
C_{13} = 0111111110000110011001010101
D_{13} = 0111101010101011001100111100
C_{14} = 1111111000011001100101010101
D_{14} = 1110101010101100110011110001
C_{15} = 1111100001100110010101010111
D_{15} = 1010101010110011001111000111
C_{16} = 1111000011001100101010101111
D_{16} = 0101010101100110011110001111

DES Example

PC-2 permutation using *DES KEY CALCULATION-(c)*:

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
K_3 = 010101 011111 110010 001010 010000 101100 111110 011001
K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
K_8 = 111101 111000 101000 111010 110000 010011 101111 111011
K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111
K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

DES Example

STEP 2: DES Encoding

IP using *PERMUTATION TABLES FOR DES-(a)*:

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Split into left & right halves:

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

1^{st} round:

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$R_1 = L_0$ XOR $f(R_0, K_1)$

***f* function calculation:**

- **EP using *PERMUTATION TABLES FOR DES-(c)*:**

E(R_0) = 011110 100001 010101 010101 011110 100001 010101 010101

- **XOR calculation**

K_1 XOR E(R_0) = 011000 010001 011110 111010 100001 100110 010100 100111

- **S-boxes S = $S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$**

S = 0101 1100 1000 0010 1011 0101 1001 0111

- **Permutation using *PERMUTATION TABLES FOR DES-(d)*:**

f = P(S) = 0010 0011 0100 1010 1010 1001 1011 1011

DES Example

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
$f(R_0,K_1)$ = 0010 0011 0100 1010 1010 1001 1011 1011

$R_1 = L_0$ XOR $f(R_0,K_1)$ = 1110 1111 0100 1010 0110 0101 0100 0100

Nth round:

$L_n = R_{n-1}$
$R_n = L_{n-1}$ XOR $f(R_{n-1},K_n)$
$= L_{n-1}$ XOR P(S (K_n XOR E(R_{n-1})))

.

.

.

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100
R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101
$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

Inverse IP using *PERMUTATION TABLES FOR DES-(b)*:

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

C = 85E813540F0AB405

Contents

The Avalanche Effect

Aim: small change in key (or plaintext) produces large change in ciphertext

- Avalanche effect is present in DES (good for security)

- Following examples show the number of bits that change in output when two different inputs are used, differing by 1 bit
  - Plaintext 1: 02468aceeca86420
    Plaintext 2: 12468aceeca86420
    Ciphertext difference: 32 bits

    Hexadecimal

    the fourth bit is changed

  - Key 1: 0f1571c947d9e859
    Key 2: 1f1571c947d9e859
    Ciphertext difference: 30

Avalanche Effect in DES: Change in Plaintext

Round		δ	Round		δ
	02468aceeca86420 12468aceeca86420	1	9	c11bfc09887fbc6c 99f911532eed7d94	32
1	3cf03c0fbad22845 3cf03c0fbad32845	1	10	887fbc6c600f7e8b 2eed7d94d0f23094	34
2	bad2284599e9b723 bad3284539a9b7a3	5	11	600f7e8bf596506e d0f23094455da9c4	37
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18	12	f596506e738538b8 455da9c47f6e3cf3	31
4	0bae3b9e42415649 171cb8b3ccaca55e	34	13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
5	4241564918b3fa41 ccaca55ed16c3653	37	14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
6	18b3fa419616fe23 d16c3653cf402c68	33	15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
7	9616fe2367117cf2 cf402c682b2cefbc	32	16	75e8fd8f25896490 1ce2e6dc365e5f59	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33	IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Avalanche Effect in DES: Change in Key

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

Key Size

- Although **64 bit** initial key, only **56 bits** used in encryption (other 8 for parity check)
- $2^{56} = 7.2 \times 10^{16}$
  - 1977: estimated cost $US20m to build machine to break in 10 hours
  - 1998: EFF built machine for $US250k to break in 3 days
  - Today: 56 bits considered too short to withstand brute force attack
- **3DES** uses **128-bit** keys

Attacks on DES

Timing Attacks

- Information gained about key/plaintext by observing *how long implementation takes* to decrypt
- No known useful attacks on DES

Differential Cryptanalysis

- Observe how pairs of plaintext blocks evolve
- Break DES in 2^{47} encryptions (compared to 2^{55}); but require 2^{47} chosen plaintexts

Linear Cryptanalysis

- Find linear approximations of the transformations
- Break DES using 2^{43} known plaintexts

DES Algorithm Design

DES was designed in private; questions about the motivation of the design

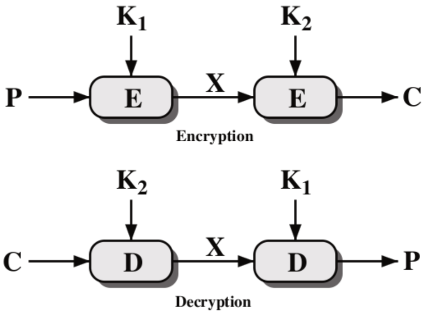
- **S-Boxes** provide non-linearity
  - important part of DES, generally considered to be secure
- **S-Boxes** provide increased confusion
- **Permutation** P chosen to increase diffusion

Contents

Multiple Encryption with DES

- DES is vulnerable to **brute force attack**
- Alternative block cipher that makes use of DES software/equipment/knowledge
  - encrypt multiple times with different keys
- Options:
  1. **Double DES**: not much better than single DES
  2. **Triple DES (3DES) with 2 keys**: brute force 2^{112}
  3. **Triple DES with 3 keys**: brute force 2^{168}

Double Encryption

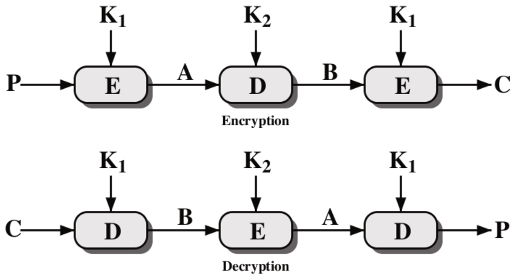


- For DES, 2×56-bit keys, meaning 112-bit key length
- Requires 2^{112} operations for brute force?
  - Meet-in-the-middle attack makes it easier

Meet-in-the-Middle Attack

- **Double DES** Encryption: $C = E(K_2, E(K_1, P))$
- Say $X = E(K_1, P) = D(K_2, C)$
- Attacker knows two plaintext, ciphertext pairs (P_a, C_a) and (P_b, C_b)
  1. Encrypt P_a using all 2^{56} values of K_1 to get multiple values of X
  2. Store results in table and sort by X
  3. Decrypt C_a using all 2^{56} values of K_2
  4. As each decryption result produced, check against table
  5. If match, check current K_1, K_2 on C_b. If P_b obtained, then accept the keys
- With two known plaintext, ciphertext pairs, probability of successful attack is almost 1
- Encrypt/decrypt operations required: 2^{56} (twice as many as single DES)

Triple Encryption



- 2 keys, 112 bits
- 3 keys, 168 bits
- Why E-D-E? To be compatible with single DES:

$$C = \mathrm{E}(K_1, \mathrm{D}(K_1, \mathrm{E}(K_1, P))) = \mathrm{E}(K_1, P)$$

Advanced Encryption Standard (AES)

- NIST called for proposals for new standard in 1997
  - Aims: security, efficient software/hardware implementations, low memory requirements, parallel processing
  - Candidate algorithms from around the world
  - Rijndael chosen, standard called AES created in 2001
- AES:
  - Block size: 128 bits (others possible)
  - Key size: 128, 192, 256 bits
  - Rounds: 10, 12, 14 (depending on key)
  - Operations: XOR with round key, substitutions using S-Boxes, mixing using Galois Field arithmetic
- Widely used in file encryption, network communications
- Generally considered secure
- Refer to "Lec 03 Appendix - Advanced Encryption Standard" for detailed procedures of AES

Other Symmetric Encryption Algorithms

- Blowfish (Schneier, 1993): 64 bit blocks/32–448 bit keys; Feistel structure
- Twofish (Schneier et al, 1998): 128/128, 192, 256; Feistel structure
- Serpent (Anderson et al, 1998): 128/128, 192, 256; Substitution-permutation network
- Camellia (Mitsubishi/NTT, 2000): 128/128, 192, 256; Feistel structure
- IDEA (Lai and Massey, 1991): 64/128
- CAST-128 (Adams and Tavares, 1996): 64/40–128; Feistel structure
- CAST-256 (Adams and Tavares, 1998): 128/up to 256; Feistel structure
- RC5 (Rivest, 1994): 32, 64 or 128/up to 2040; Feistel-like structure
- RC6 (Rivest et al, 1998): 128/128, 192, 256; Feistel structure

Cryptanalysis on Block Ciphers

Cipher	Method	Key space	Required resources: Time	Memory	Known data
DES	Brute force	2^{56}	2^{56}	-	-
3DES	MITM	2^{168}	2^{111}	2^{56}	2^{2}
3DES	Lucks	2^{168}	2^{113}	2^{88}	2^{32}
AES 128	Biclique	2^{128}	$2^{126.1}$	2^{8}	2^{88}
AES 256	Biclique	2^{256}	$2^{254.4}$	2^{8}	2^{40}

- Known data: chosen pairs of (plaintext, ciphertext)
- MITM: Meet-in-the-middle
- Lucks: S. Lucks, Attacking Triple Encryption, in *Fast Software Encryption*, Springer, 1998
- Biclique: Bogdanov, Khovratovich and Rechberger, Biclique Cryptanalysis of the Full AES, in *ASIACRYPT2011*, Springer, 2011