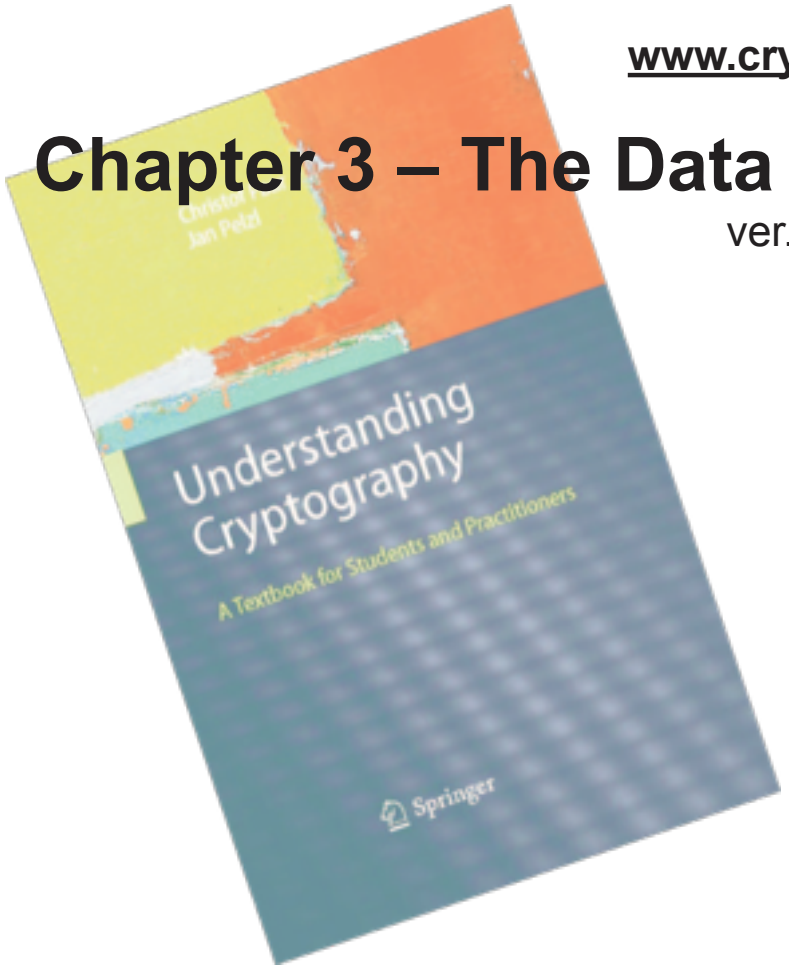
Understanding Cryptography – A Textbook for Students and Practitioners

by Christof Paar and Jan Pelzl



www.crypto-textbook.com

Chapter 3 – The Data Encryption Standard (DES)

ver. Nov 26, 2010

These slides were prepared by Markus Kasper, Christof Paar and Jan Pelzl
and modified by Sam Bowne

Some legal stuff (sorry): Terms of Use

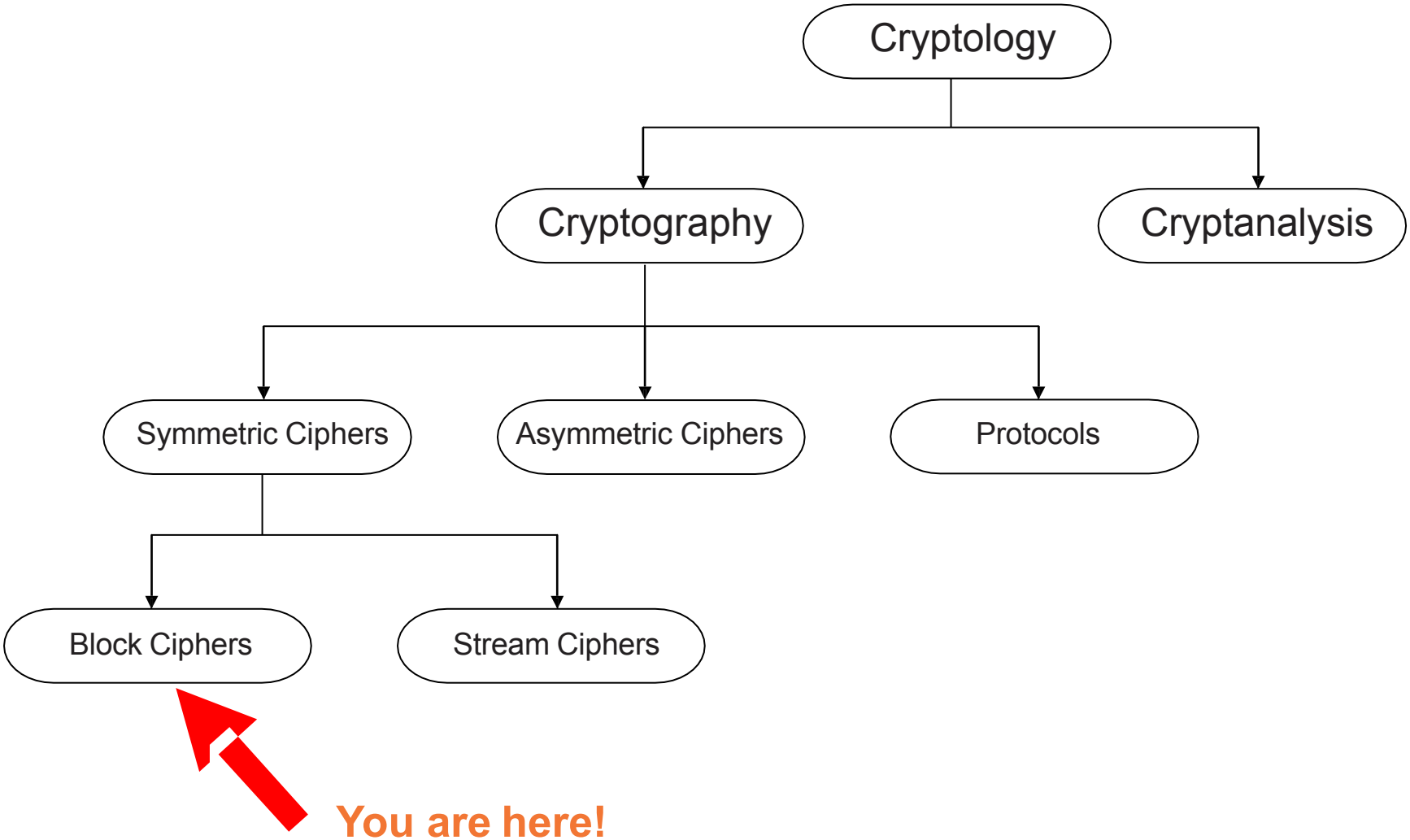
- The slides can used free of charge. All copyrights for the slides remain with the authors.
- The title of the accompanying book “Understanding Cryptography” by Springer and the author’s names must remain on each slide.
- If the slides are modified, appropriate credits to the book authors and the book title must remain within the slides.
- It is not permitted to reproduce parts or all of the slides in printed form whatsoever without written consent by the authors.

Contents of this Chapter

- Design Process of DES
- Confusion and Diffusion
- Feistel Networks, S-boxes, Key Schedule
- Security Analysis of DES
- Alternatives to DES

Design Process of DES

Classification of DES in the Field of Cryptology



Political Climate

- Before 1972 cryptography was kept secret, reserved for military use
- In 1972 US National Bureau of Standards (now NIST) requested proposals for a cipher algorithm to be the standard for banking & commerce
- IBM proposed **Lucifer** cipher, with 128-bit key
- NBS secretly asked NSA to to test its security
- NSA made it stronger so it could resist **differential cryptanalysis**
- And weaker by shortening the key to 56 bits

Differential Cryptanalysis

- A portion of the cipher algorithm is approximated as series of XOR operations
- An XOR-based algorithm is found that is more likely to be correct than pure chance (50%)
- Encrypting a large number of chosen plaintexts reveals information about the key
- The number of encryptions required is approximately $1 / (p-.50)^2$
- So if the approximation is 60% correct, you need 100 encryptions to deduce the key
- Link Ch 3b

DES Facts

- Data Encryption Standard (DES) encrypts **blocks of size 64 bits**
- **Standardized 1977** by the **NBS** (now **NIST**)
- Most popular **block cipher** for most of the last 30 years.
- By far best studied **symmetric algorithm**.
- Nowadays considered **insecure** due to **small key length of 56 bits**
- **But: 3DES is a very secure cipher**, still widely used today.
- Replaced by the *Advanced Encryption Standard* (**AES**) in 2000

Confusion and Diffusion

Confusion and Diffusion

Claude Shannon: There are two primitive operations with which strong encryption algorithms can be built:

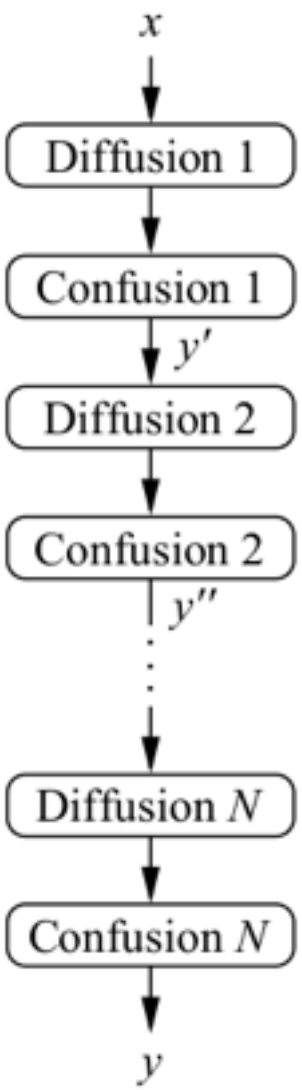
1. **Confusion:** An encryption operation where the **relationship between key and ciphertext is obscured**.
   Today, a common element for achieving confusion is **substitution**, which is found in both AES and DES.

2. **Diffusion:** An encryption operation where the **influence of one plaintext symbol is spread over many ciphertext symbols** with the goal of hiding statistical properties of the plaintext.
   A simple diffusion element is the **bit permutation**, which is frequently used within DES.

Confusion and Diffusion

- Confusion-only ciphers
  - **Caesar cipher**, **Enigma**, **Substitution cipher**
  - Not secure
- Diffusion-only ciphers are also insecure
- **Product ciphers**
  - Use several rounds of confusion and diffusion to make better ciphers

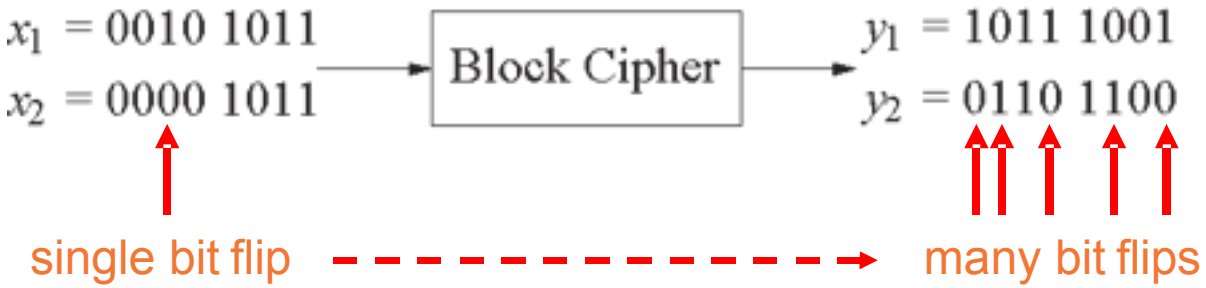
Product Ciphers



Most of today's block ciphers are *product ciphers* as they consist of rounds which are applied repeatedly to the data.

Can reach excellent diffusion: **changing of one bit of plaintext results** *on average* in the **change of half the output bits**.

Example:



AES in Python

- Changing one bit of the key changes many bits throughout the ciphertext

```
>>> from Crypto.Cipher import AES
>>> key = "BBBBBBBBBBBBBBBB"
>>> cipher = AES.new(key)
>>> cipher.encrypt("A secret message").encode("hex")
'181bbc3d48a9c6e9eba7476cab11e3aa'
>>>
>>> key = "BBBBBBBBBBBBBBBC"
>>> cipher = AES.new(key)
>>> cipher.encrypt("A secret message").encode("hex")
'0a7a6805eee7eb1864a1b63936a7275a'
>>>
```

AES Keys in Python

- "B" is **01000010**
- "C" is **01000011** *Changed only the last bit*

```
>>> bin(ord("B"))
'0b1000010'
>>> bin(ord("C"))
'0b1000011'
```

DES in Python

- Changing one bit of the key changes many bits throughout the ciphertext
- Last bit of each byte is **parity bit** and is ignored (link Ch 3a)

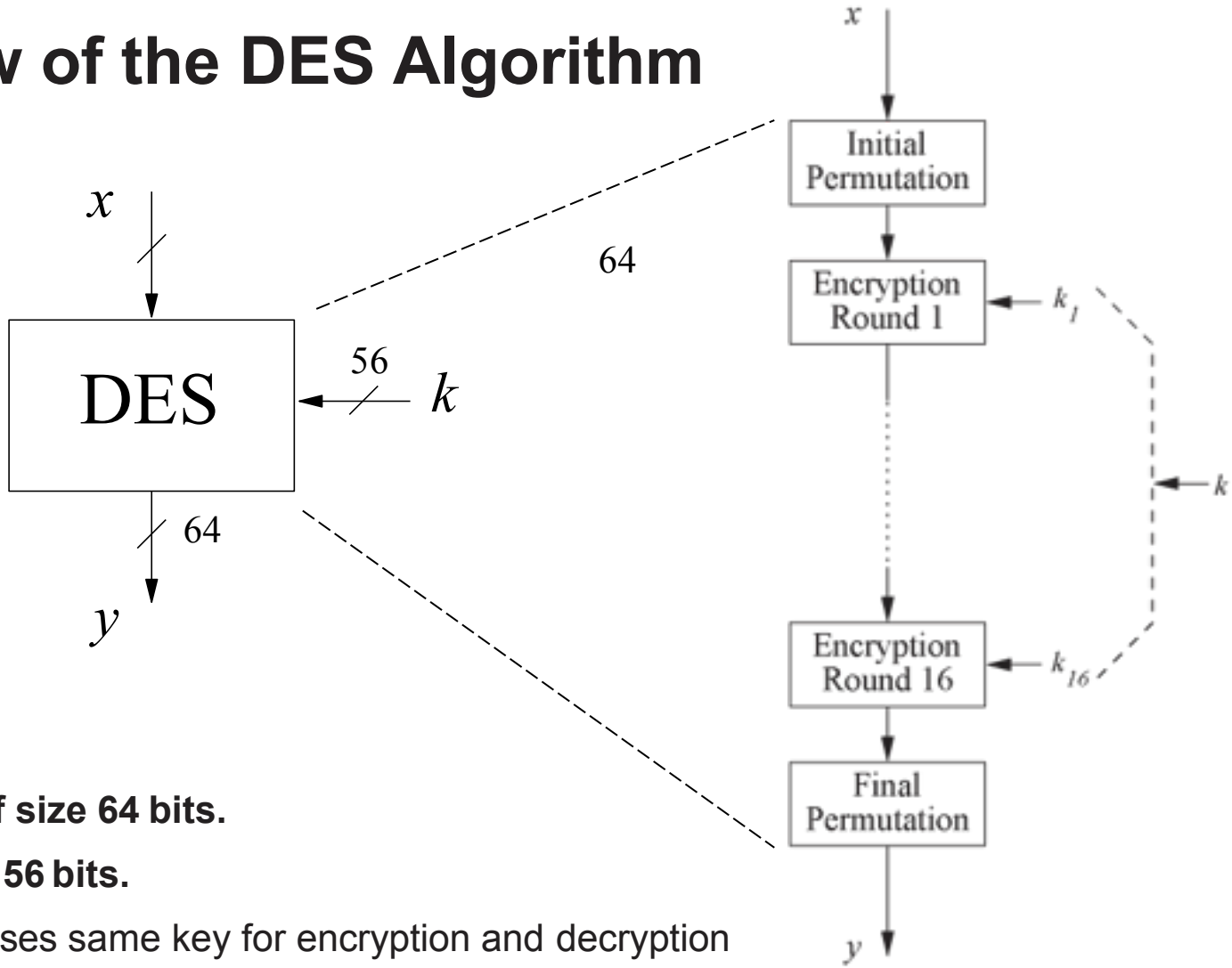
```
>>> from Crypto.Cipher import DES
>>> key = "BBBBBBBB"
>>> cipher = DES.new(key)
>>> cipher.encrypt("A secret").encode("hex")
'0a4a66a2cf5b0244'
>>>
>>> key = "BBBBBBBC"
>>> cipher = DES.new(key)
>>> cipher.encrypt("A secret").encode("hex")
'0a4a66a2cf5b0244'
>>>
>>> key = "BBBBBBBD"
>>> cipher = DES.new(key)
>>> cipher.encrypt("A secret").encode("hex")
'be52c52606850742'
>>>
```

DES Keys in Python

- "B" is **01000010**
- "C" is **01000011** *Changed only the parity bit*
- "D" is **01000100** *Changed two non-parity bits*

```
>>> bin(ord("B"))
'0b1000010'
>>> bin(ord("C"))
'0b1000011'
>>> bin(ord("D"))
'0b1000100'
>>>
```

Overview of the DES Algorithm



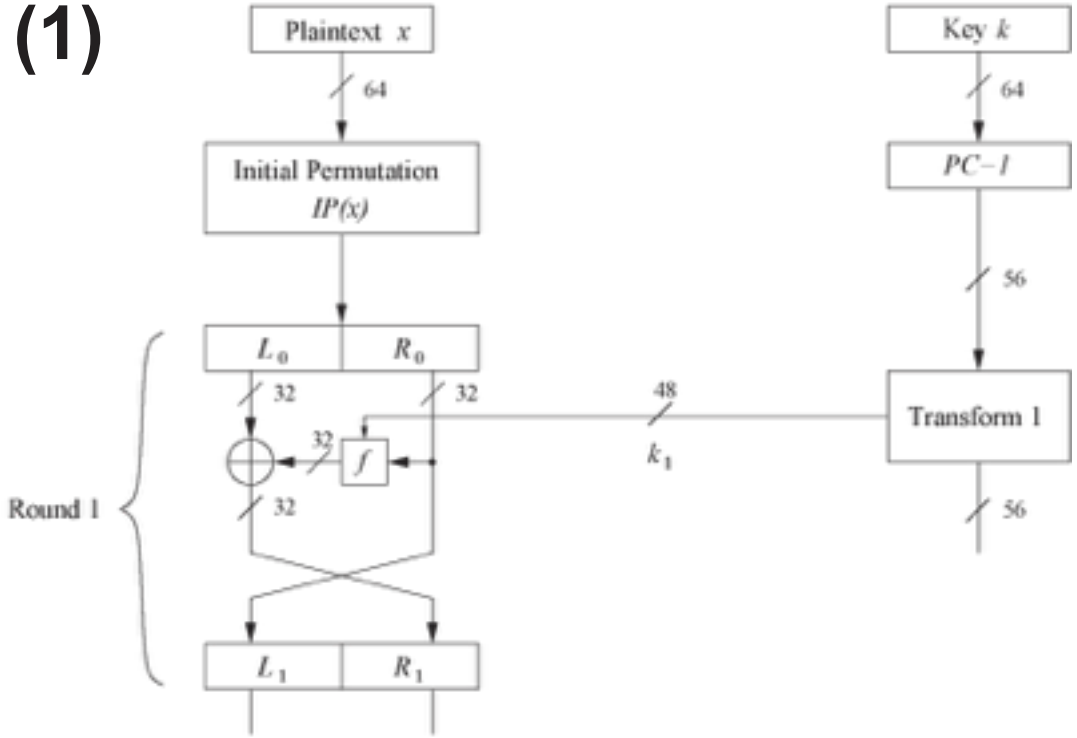
- **Encrypts blocks of size 64 bits.**
- **Uses a key of size 56 bits.**
- Symmetric cipher: uses same key for encryption and decryption
- Uses 16 rounds which all perform the identical operation
- Different subkey in each round derived from main key


Kahoot!

Feistel Networks
S-Boxes
Key Schedule

The DES Feistel Network (1)

- DES structure is a *Feistel network*
- Advantage: encryption and decryption differ only in keyschedule



- Bitwise initial permutation, then 16 rounds
  1. Plaintext is split into 32-bit halves L_i and R_i
  2. R_i is fed into the function f, the output of which is then XORed with L_i
  3. Left and right half are swapped
- Rounds can be expressed as:

$$L_i = R_{i-1},$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$$

Feistel Network

- Only encrypts the left half (L) of the input in each round
- Right half (R) is used with the key to encrypt L
- R is copied to the next round unchanged
- But the halves are switched in the next round, so R gets encrypted there

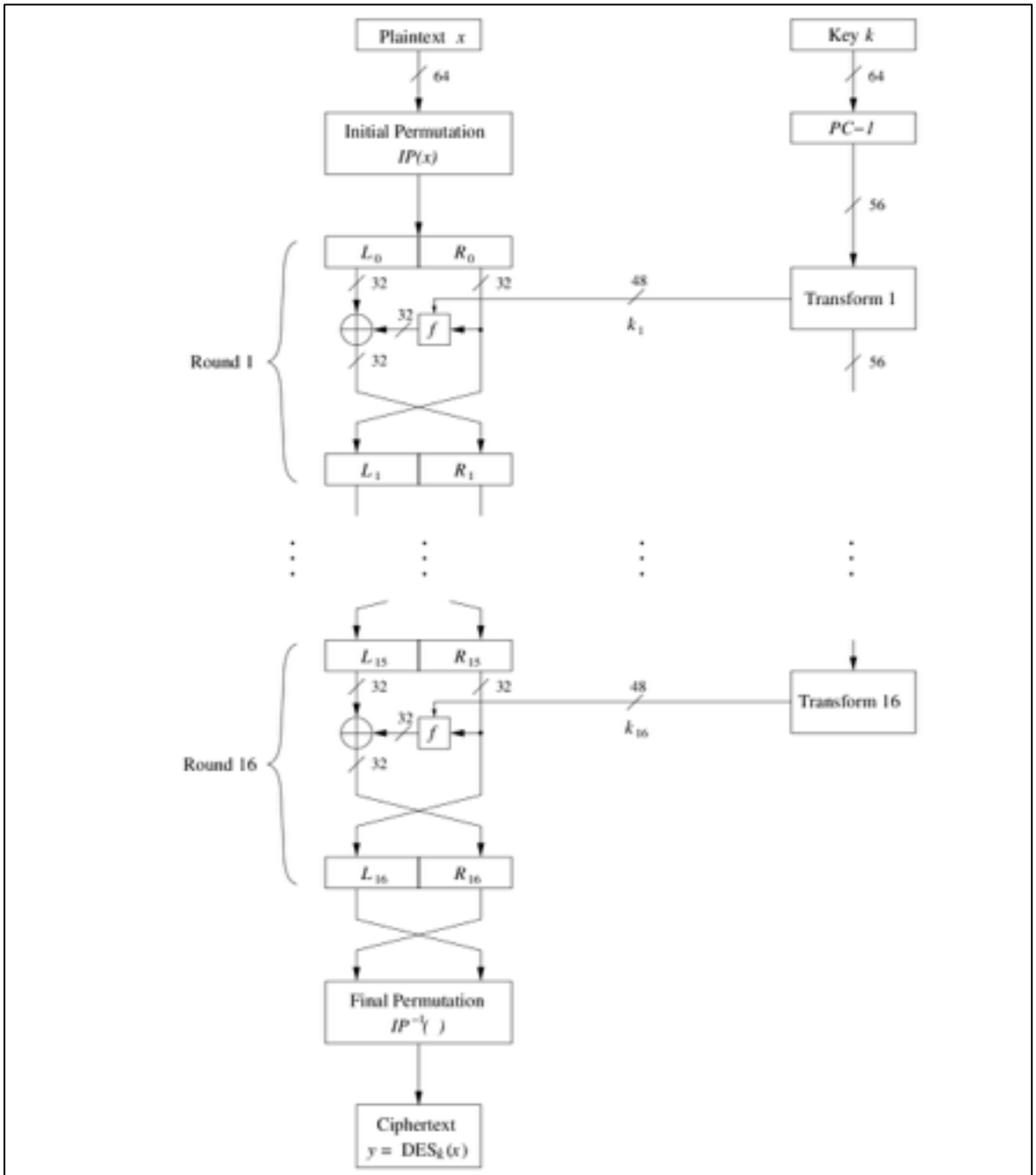
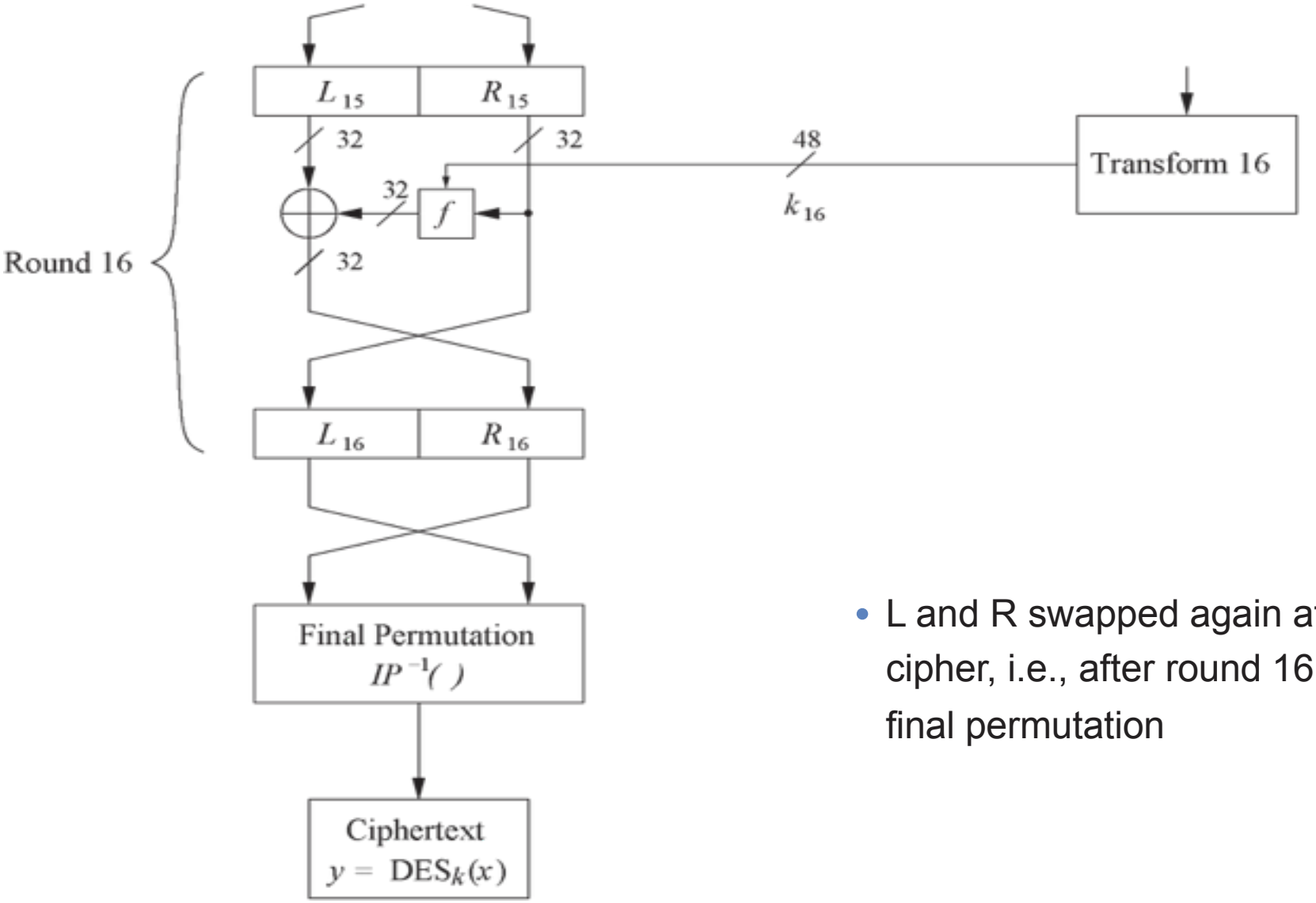


Fig. 3.5 The Feistel structure of DES

The DES Feistel Network (2)



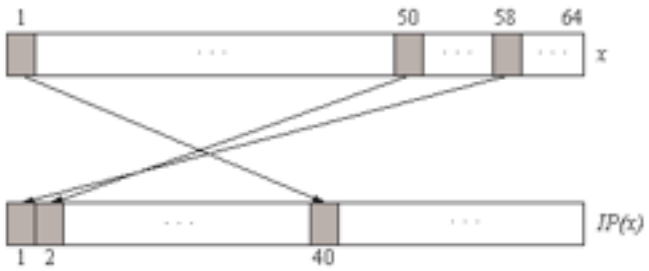
- L and R swapped again at the end of the cipher, i.e., after round 16 followed by a final permutation

Initial and Final Permutation

- Bitwise Permutations.
- Inverse operations.
- Described by tables *IP* and IP^{-1}.

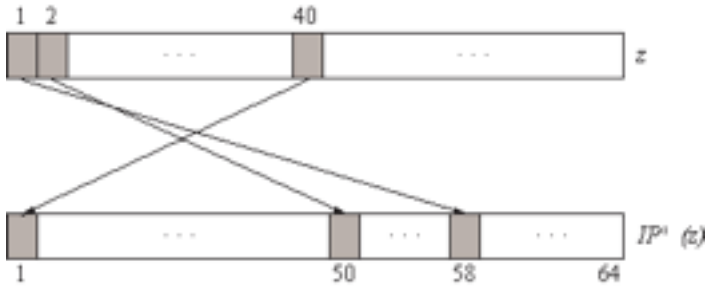
Initial Permutation

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



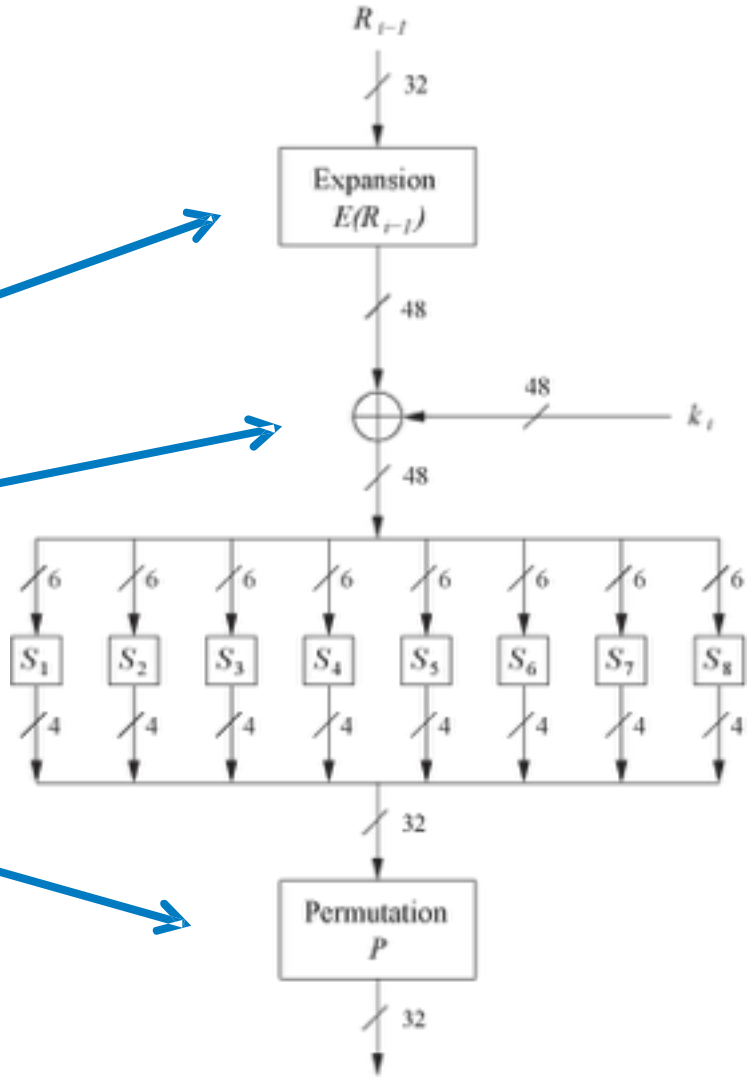
Final Permutation

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



The f-Function

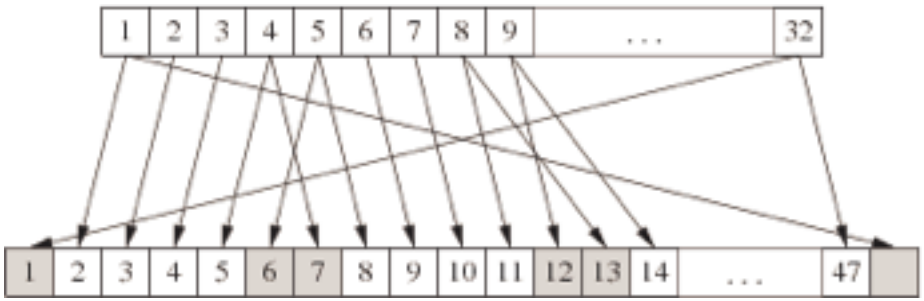
- **main operation of DES**
- *f*-Function inputs:
  R_{i-1} and round key k_i
- **4 Steps**:
  1. Expansion *E*
  2. XOR with round key
  3. S-box substitution
  4. Permutation

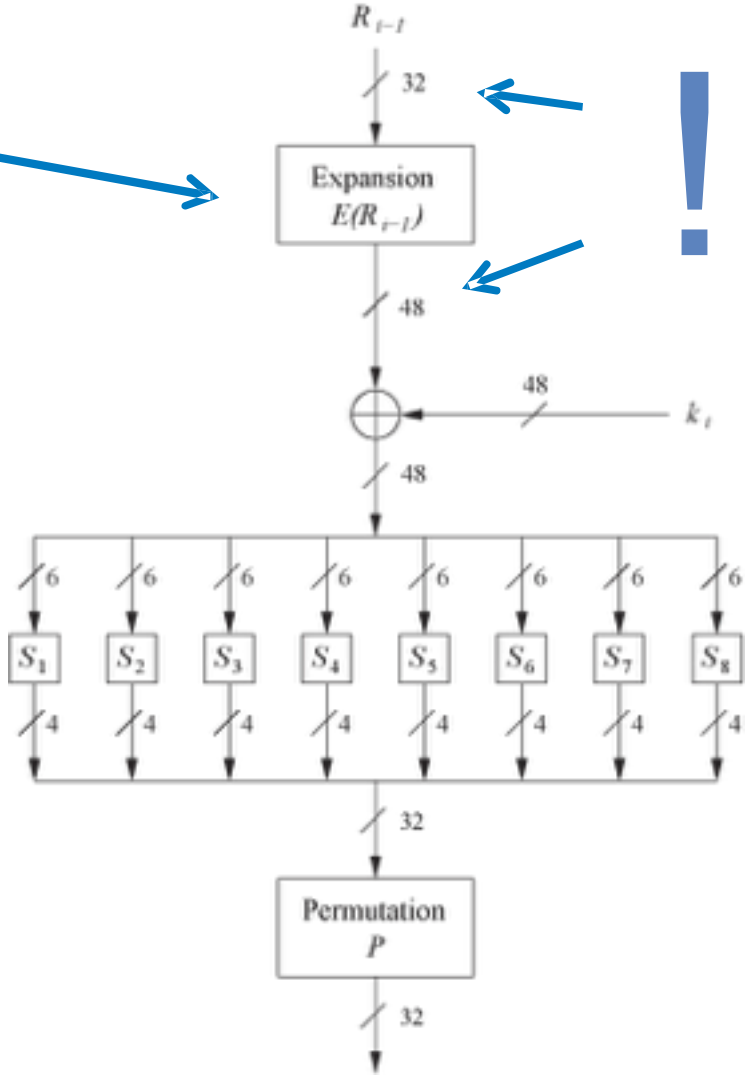


The Expansion Function E

1. **Expansion *E***
- **main purpose:**
  increases diffusion

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

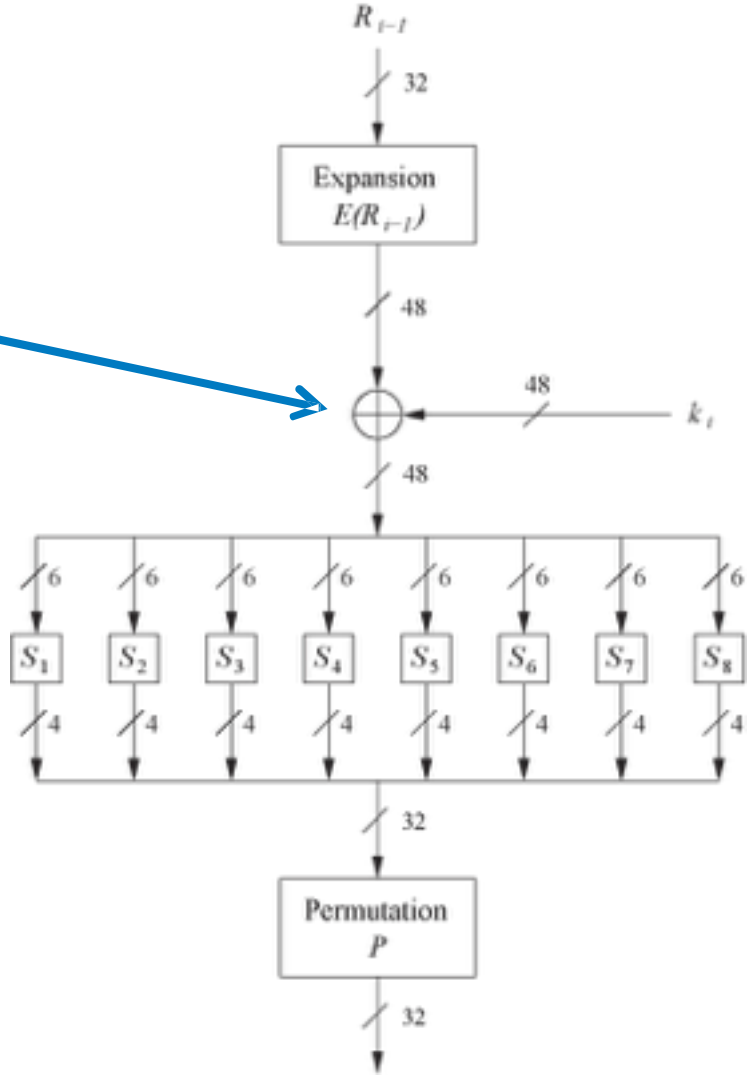




Add Round Key

2. XOR Round Key

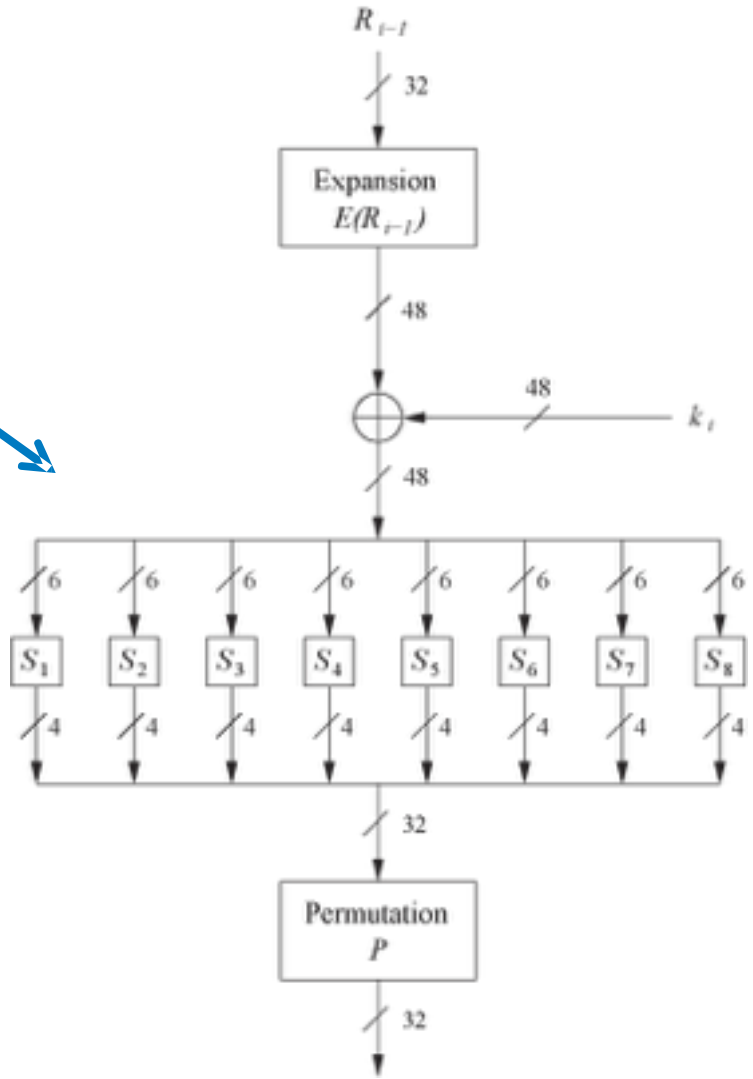
- **Bitwise XOR of the round key and the output of the expansion function *E***
- **Round keys are derived from the main key in the DES keyschedule (in a few slides)**

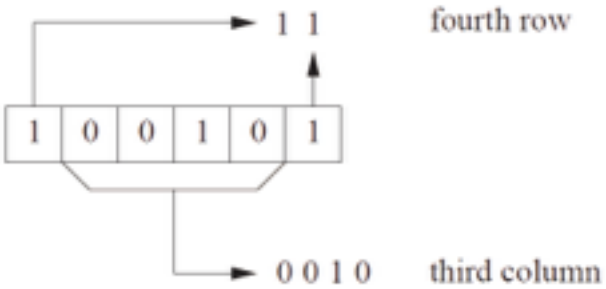


The DES S-Boxes

3. S-Box substitution

- Eight substitution tables.
- 6 bits of input, 4 bits of output.
- Non-linear and resistant to differential cryptanalysis.
- Crucial element for DES security!
- Find all S-Box tables and S-Box design criteria in *Understanding Cryptography* Chapter 3.





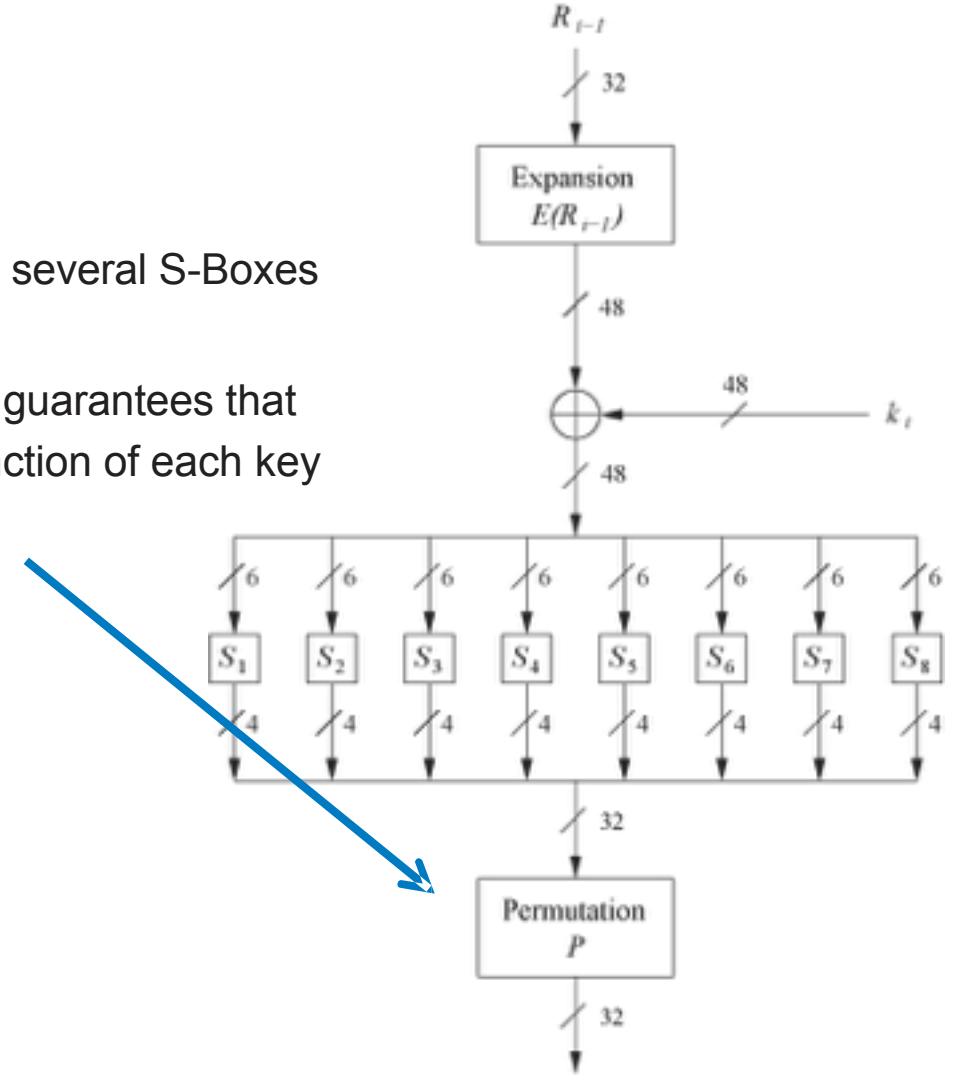
S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

The Permutation P

4. Permutation P

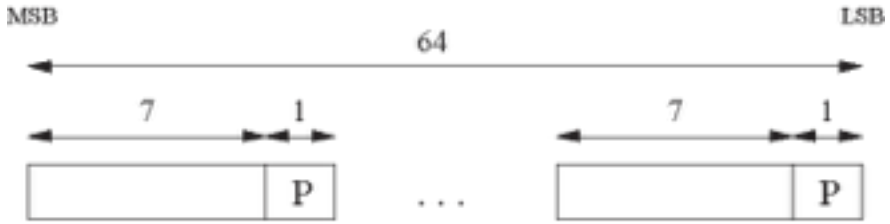
- Bitwise permutation.
- Introduces diffusion.
- Output bits of one S-Box effect several S-Boxes in next round
- Diffusion by E, S-Boxes and P guarantees that after Round 5 every bit is a function of each key bit and each plaintext bit.

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



Key Schedule (1)

- Derives 16 round keys (or *subkeys*) k_i of 48 bits each from the original 56 bit key.
- The input key size of the DES is 64 bit: **56 bit key** and 8 bit parity:



P = parity bit

- **Parity bits are removed** in a first **permuted choice** *PC-1*:
(note that the bits 8, 16, 24, 32, 40, 48, 56 and 64 are not used at all)

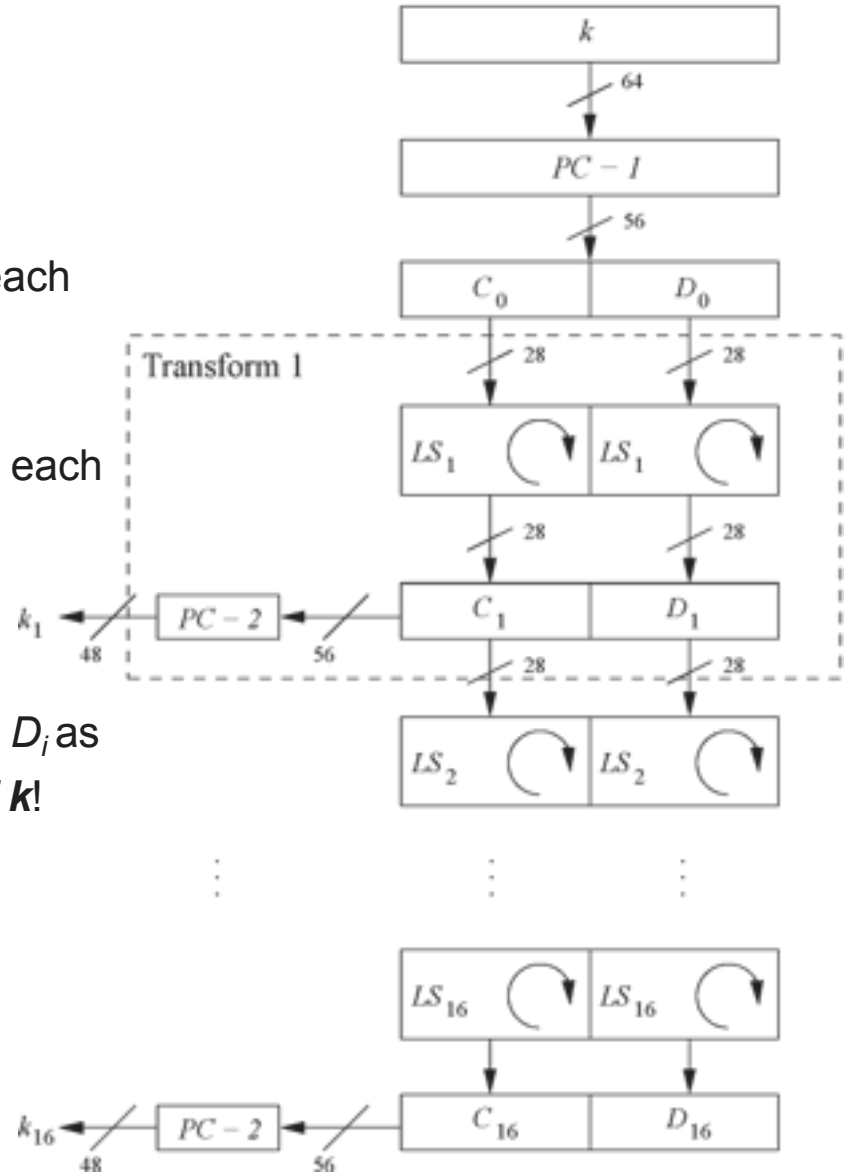
PC – 1							
57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4

Key Schedule (2)

- Split key into 28-bit halves C_0 and D_0.
- In **rounds *i* = 1, 2, 9 ,16,** the two halves are each rotated left by **one bit**.
- In **all other rounds** where the two halves are each rotated left by **two bits**.
- *In each round i permuted choice* ***PC-2*** selects a permuted subset of 48 bits of C_i and D_i as round key k_i, i.e. **each k_i is a permutation of *k***!

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

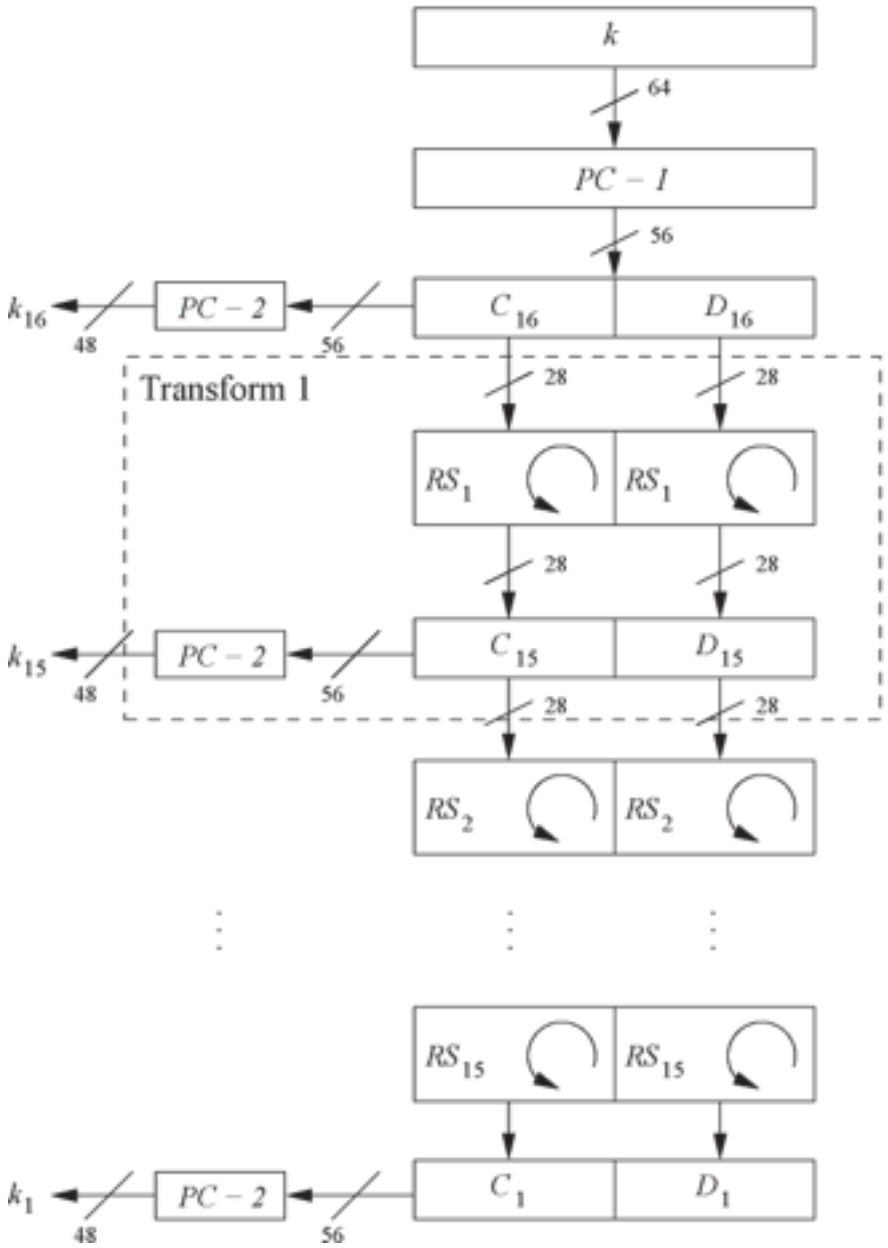
- **Note:** The total number of rotations:
  4 x 1 + 12 x 2 = 28 $\Rightarrow D_0 = D_{16}$ and $C_0 = C_{16}$!



Decryption

- In **Feistel ciphers** only the keyschedule has to be modified for decryption.
- Generate the same 16 round keys in reverse order.
  (for a detailed discussion on why this works see *Understanding Cryptography* Chapter 3)

- **Reversed key schedule:**
  As $D_0=D_{16}$ and $C_0=C_{16}$ the first round key can be generated by applying *PC-2* right after *PC-1* (no rotation here!).
  All other rotations of *C* and *D* can be reversed to reproduce the other round keys resulting in:
  - No rotation in round 1.
  - One bit rotation **to the right** in rounds 2, 9 and 16.
  - Two bit rotations **to the right** in all other rounds.



Security Analysis of DES

Security of DES

- **After proposal of DES two major criticisms arose:**
  1. Key space is too small (2^{56} keys)
  2. S-box design criteria have been kept secret: Are there any hidden analytical attacks (*backdoors*), only known to the NSA?

- **Analytical Attacks:** DES is highly resistent to both *differential* and *linear cryptanalysis*, which have been published years later than the DES. This means IBM and NSA had been aware of these attacks for 15 years!
  So far there is no known analytical attack which breaks DES in realistic scenarios.

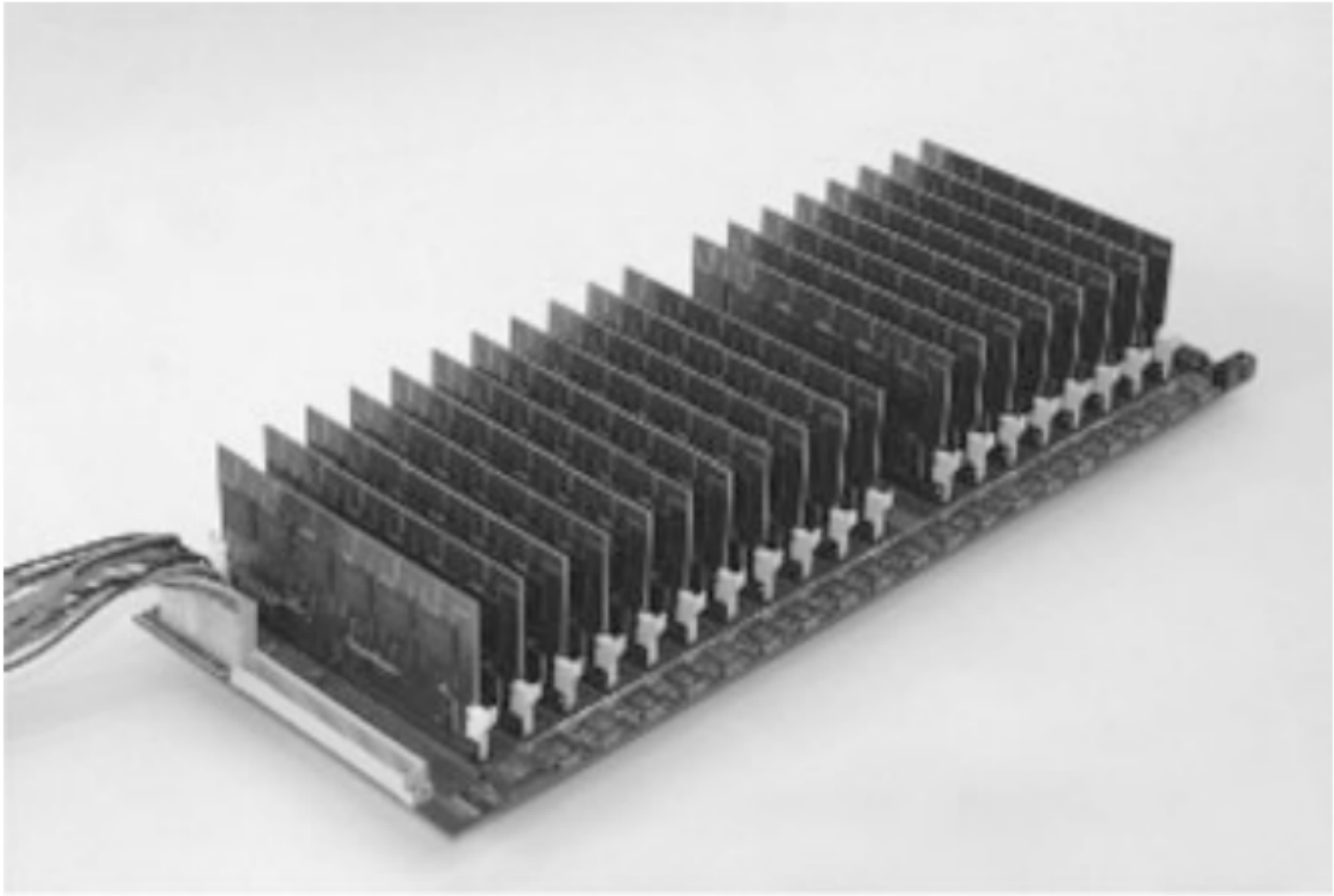
- **Exhaustive key search:** For a given pair of plaintext-ciphertext (x, y) test all 2^{56} keys until the condition $DES_k^{-1}(x)=y$ is fulfilled.
  $\Rightarrow$ Relatively easy given today's computer technology!

History of Attacks on DES

Year	Proposed/ implemented DES Attack
1977	Diffie & Hellman, (under-)estimate the costs of a key search machine
1990	Biham & Shamir propose differential cryptanalysis (2^{47} chosen ciphertexts)
1993	Mike Wiener proposes design of a very efficient key search machine: Average search requires 36h. Costs: $1.000.000
1993	Matsui proposes linear cryptanalysis (2^{43} chosen ciphertexts)
Jun. 1997	DES Challenge I broken, 4.5 months of distributed search
Feb. 1998	DES Challenge II--1 broken, 39 days (distributed search)
Jul. 1998	DES Challenge II--2 broken, key search machine *Deep Crack* built by the Electronic Frontier Foundation (EFF): 1800 ASICs with 24 search engines each, Costs: $250 000, 15 days average search time (required 56h for the Challenge)
Jan. 1999	DES Challenge III broken in 22 h 15 m (distributed search assisted by *Deep Crack*)
2006-2008	Reconfigurable key search machine *COPACOBANA* developed at the Universities in Bochum and Kiel (Germany), uses 120 FPGAs to break DES in 6.4 days (avg.) at a cost of $10 000.

Deep Crack

COPACABANA

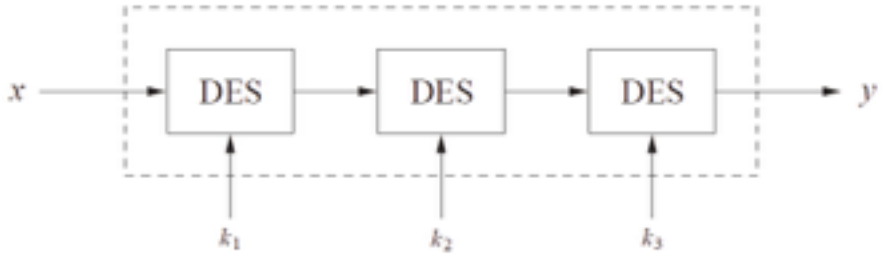


Alternatives to DES

Triple DES – 3DES

- Triple encryption using DES is often used in practice to extend the effective key length of DES to 112. For more info on multiple encryption and effective key lengths see Chapter 5 of *Understanding Cryptography.*

$$y = DES_{k_3}(DES_{k_2}(DES_{k_1}(x)))$$



- Alternative version of *3DES:* $y = DES_{k_3}(DES^{-1}_{k_2}(DES_{k_1}(x)))$.

  Advantage: choosing k_1=k_2=k_3 performs single DES encryption.
- No practical attack known today.
- Used in many legacy applications, i.e., in banking systems.

Key Whitening

- Two additional 64-bit keys are used
- XORed with the plaintext and ciphertext, respectively
- Makes DES much more resistant to brute-force attacks
- One version of this is called **DESX**

$$y = DES_{k,k_1,k_2}(x) = DES_k(x \oplus k_1) \oplus k_2$$

Alternatives to DES

Algorithm	I/O Bit	key lengths	remarks
AES / Rijndael	128	128/192/256	DES "replacement", worldwide used standard
Triple DES	64	112 (effective)	conservative choice
Mars	128	128/192/256	AES finalist
RC6	128	128/192/256	AES finalist
Serpent	128	128/192/256	AES finalist
Twofish	128	128/192/256	AES finalist
IDEA	64	128	patented

Lessons Learned

- DES was the dominant symmetric encryption algorithm from the mid-1970s to the mid-1990s. Since 56-bit keys are no longer secure, the Advanced Encryption Standard (AES) was created.
- Standard DES with 56-bit key length can be broken relatively easily nowadays through an exhaustive key search.
- DES is quite robust against known analytical attacks: In practice it is very difficult to break the cipher with differential or linear cryptanalysis.
- By encrypting with DES three times in a row, triple DES (3DES) is created, against which no practical attack is currently known.
- The “default” symmetric cipher is nowadays often AES. In addition, the other four AES finalist ciphers all seem very secure and efficient.


Kahoot!