Lecture 4
Data Encryption Standard (DES)

Block Ciphers

- Map n-bit plaintext blocks to n-bit ciphertext blocks (n = block length).
- For n-bit plaintext and ciphertext blocks and a fixed key, the encryption function is a bijection;
- $E : P_n \times K \rightarrow C_n$ s.t. for all key $k \in K$, $E(x, k)$ is an invertible mapping, written $E_k(x)$.
- The inverse mapping is the decryption function, $y = D_k(x)$ denotes the decryption of plaintext x under k.

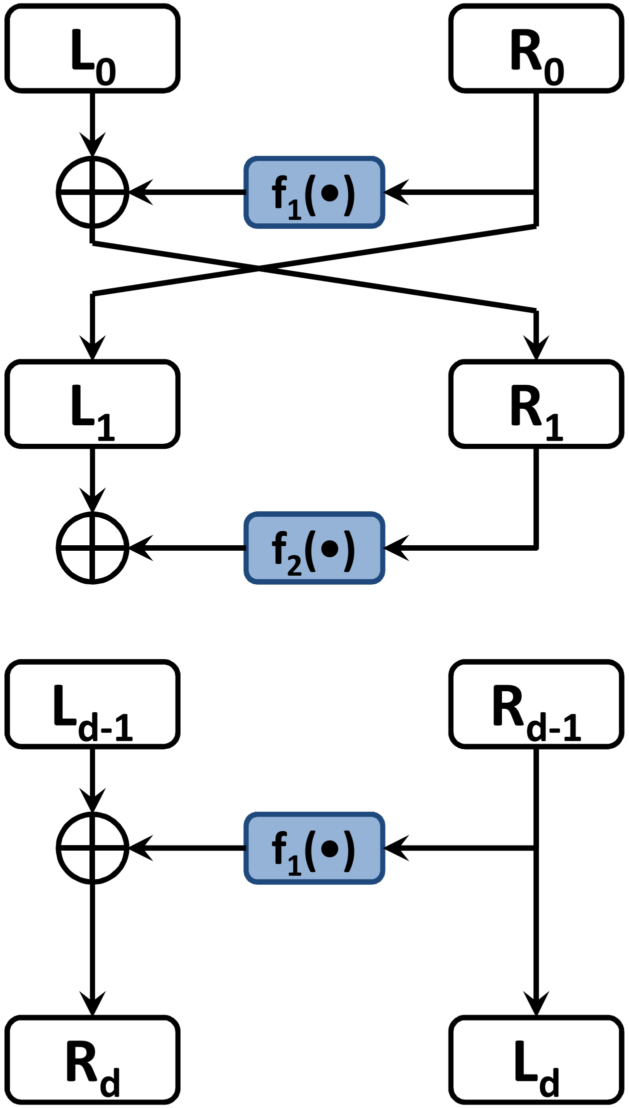
Block Ciphers Features

- Block size: in general *larger* block sizes mean *greater* security.
- Key size: *larger* key size means *greater* security (larger key space).
- Number of rounds: multiple rounds offer increasing security.
- Encryption modes: define how messages larger than the block size are encrypted, *very important* for the security of the encrypted message.

Feistel Network

- Several block ciphers are based on the structure proposed by *Feistel* in 1973
- A *Feistel Network* is fully specified given
  - the *block size*: n = 2w
  - *number of rounds*: d
  - d *round functions* $f_1, \ldots, f_d: \{0,1\}^w \rightarrow \{0,1\}^w$
- Used in DES, IDEA, RC5 (Rivest's Cipher n. 5), and many other block ciphers.
- <u>Not</u> used in AES

Feistel Network



- **Encryption**:
  - $L_1 = R_0 \quad R_1 = L_0 \oplus f_1(R_0)$
  - $L_2 = R_1 \quad R_2 = L_1 \oplus f_2(R_1)$

    ...
  - $L_d = R_{d-1} \quad R_d = L_{d-1} \oplus f_d(R_{d-1})$
- **Decryption**:
  - $R_{d-1} = L_d \quad L_{d-1} = R_d \oplus f_d(L_d)$

    ...
  - $R_0 = L_1; \quad L_0 = R_1 \oplus f_1(L_1)$

A Word About NIST and Standards

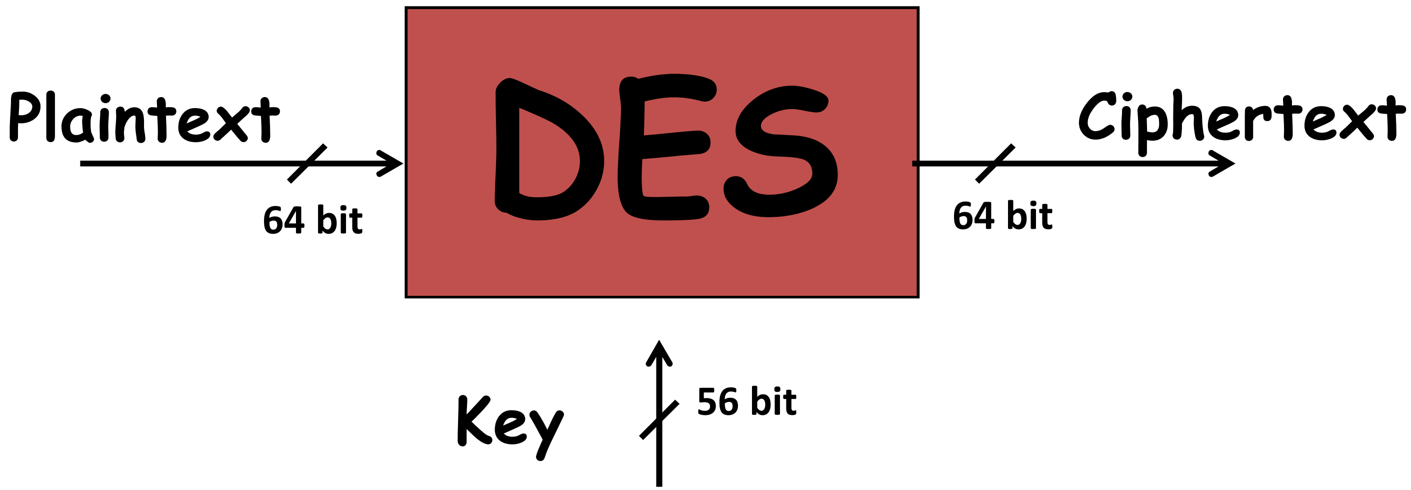
- “Founded in 1901 NIST, the *National Institute of Standards and Technology*, (former NBS) is a non-regulatory federal agency within the U.S. Commerce Department’s Technology Administration.
- NIST‘s mission is to develop and promote measurement, standards, and technology to enhance productivity, facilitate trade, and improve the quality of life.”
- Cryptographic Standards & Applications.
- Federal Information Processing Standards (FIPS): define security standards

History of Data Encryption Standard (DES)

- 1967: Feistel at IBM
  - Lucifer: block size 128; key size 128 bit
- 1972: NBS asks for an encryption standard
- 1975: IBM developed DES (modification of Lucifer)
  - block size 64 bits; key size 56 bits
- 1975: NSA suggests modifications
- 1977: NBS adopts DES as encryption standard in (FIPS 46-1, 46-2).
- 2001: NIST adopts Rijndael as replacement to DES.

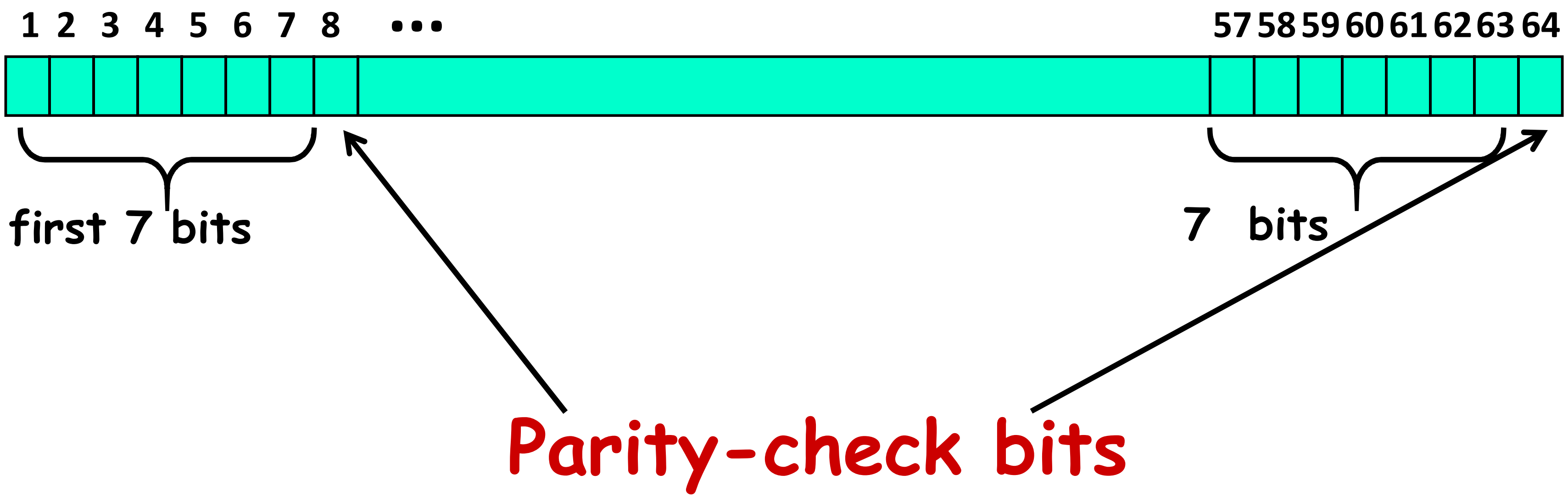
DES Features

- Features:
  - Block size = 64 bits
  - Key size = 56 bits (in reality, 64 bits, but 8 are used as parity-check bits for error control, see next slide)
  - Number of rounds = 16
  - 16 intermediary keys, each 48 bits



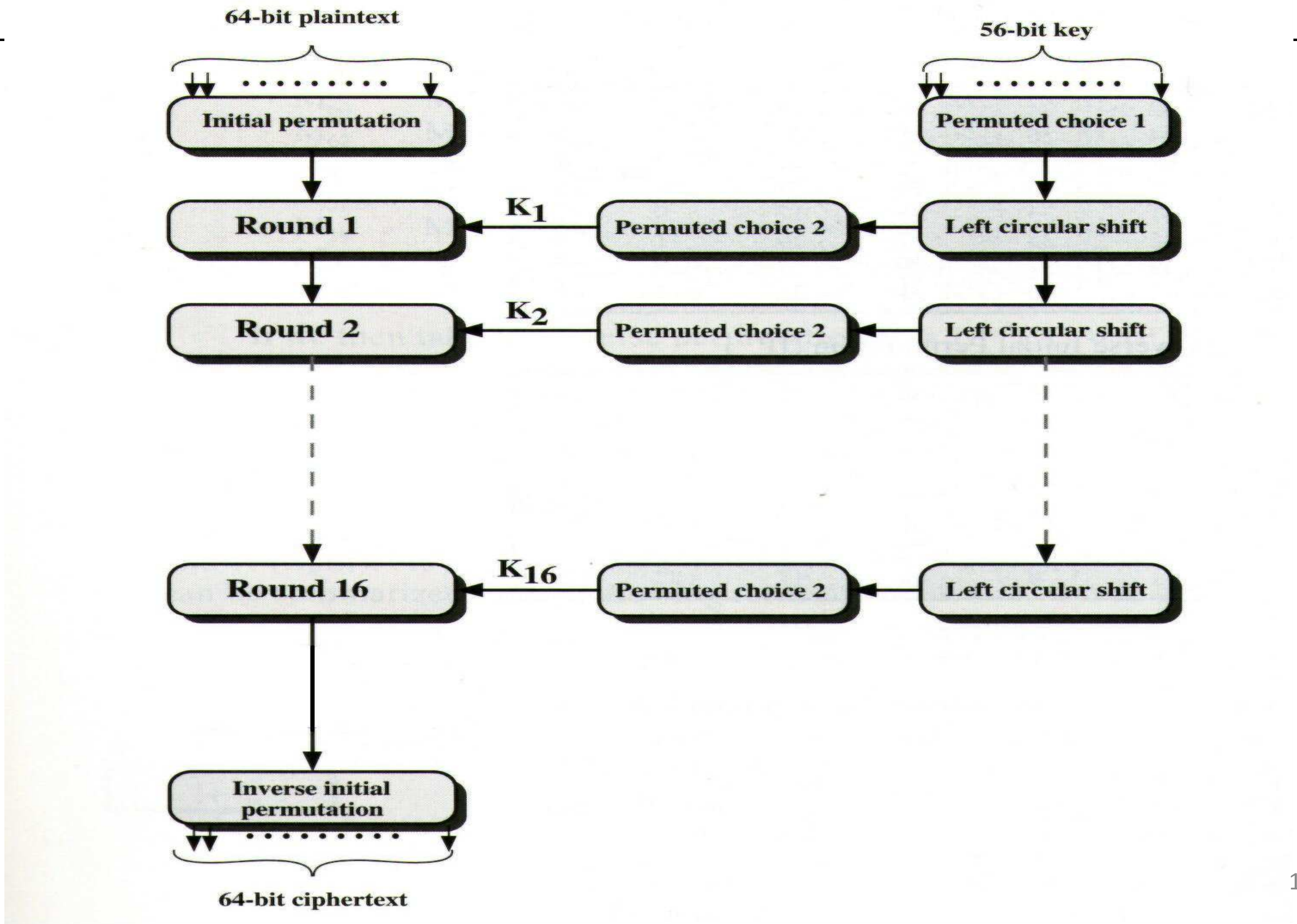
Key length in DES

- In the DES specification, the key length is 64 bit:
- 8 bytes; in each byte, the 8th bit is a parity-check bit



Each parity-check bit is the XOR of the previous 7 bits

DES Rounds

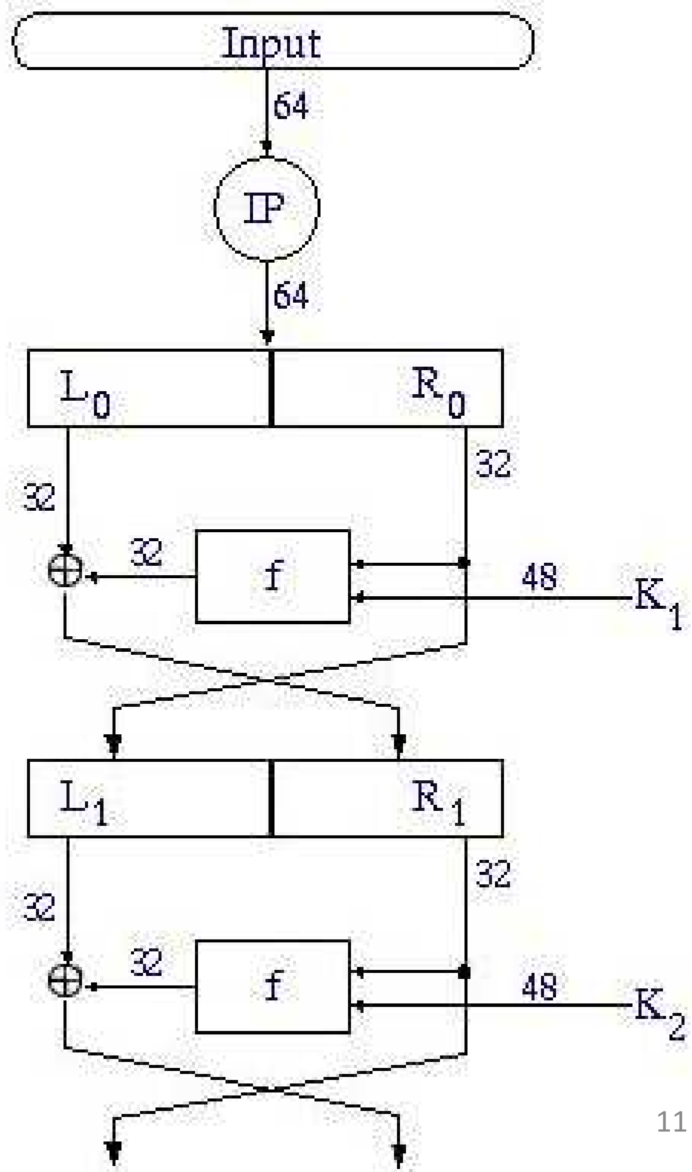


64-bit plaintext
56-bit key
Initial permutation
Permuted choice 1
Round 1
K1
Permuted choice 2
Left circular shift
Round 2
K2
Permuted choice 2
Left circular shift
Round 16
K16
Permuted choice 2
Left circular shift
Inverse initial permutation
64-bit ciphertext

Details

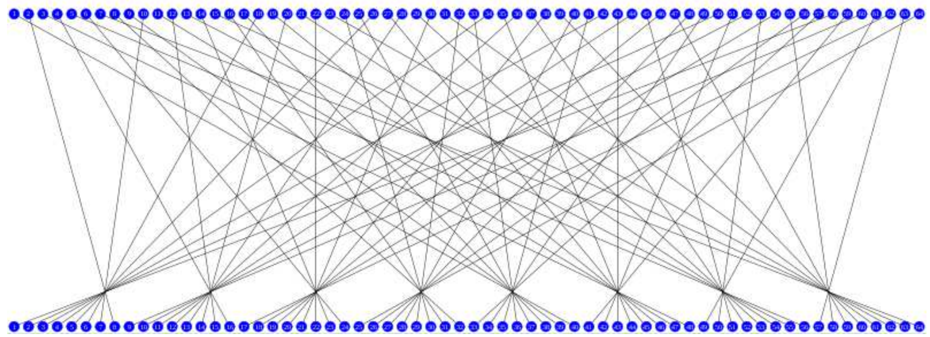
- $IP(x) = L_0R_0$
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- $y = IP^{-1}(R_{16}L_{16})$

Note: IP means Initial Permutation



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

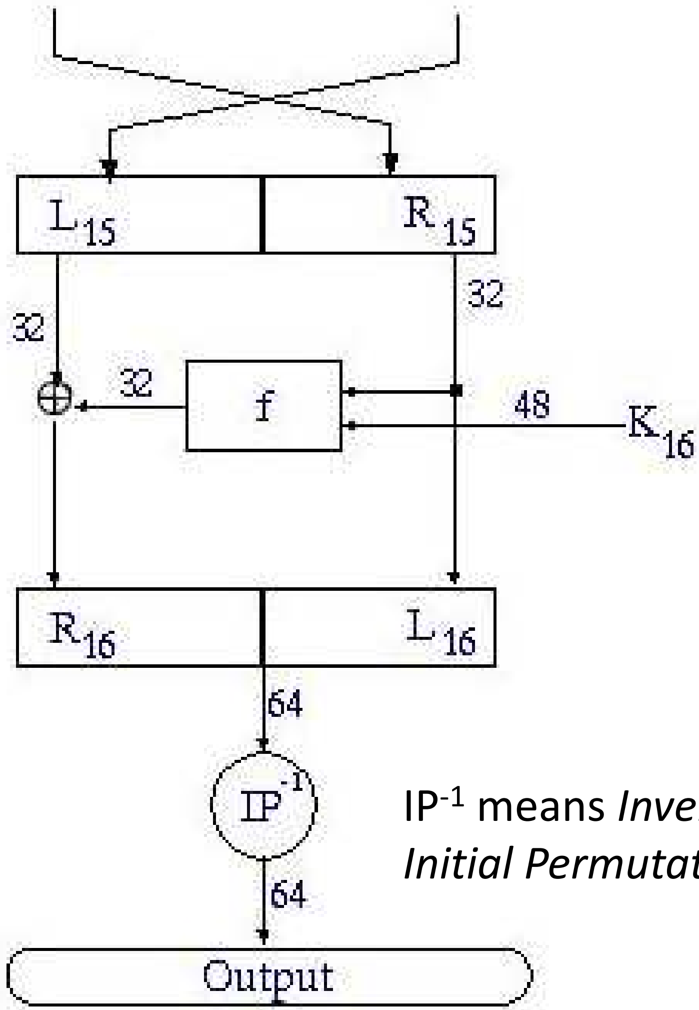


- This table specifies the input permutation on a 64-bit block.
- The meaning is as follows:
  - the first bit of the <u>output</u> is taken from the 58th bit of the <u>input</u>; the second bit from the 50th bit, and so on, with the last bit of the output taken from the 7th bit of the input.
- This information is presented as a table for ease of presentation:
  - it is a vector, not a matrix.

DES Rounds

- $IP(x) = L_0R_0$
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$
- $y = IP^{-1}(R_{16}L_{16})$

- Note that, as usual:
  - $R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
  - $L_{16} = R_{15}$
- … but they are *switched* in the pre-output

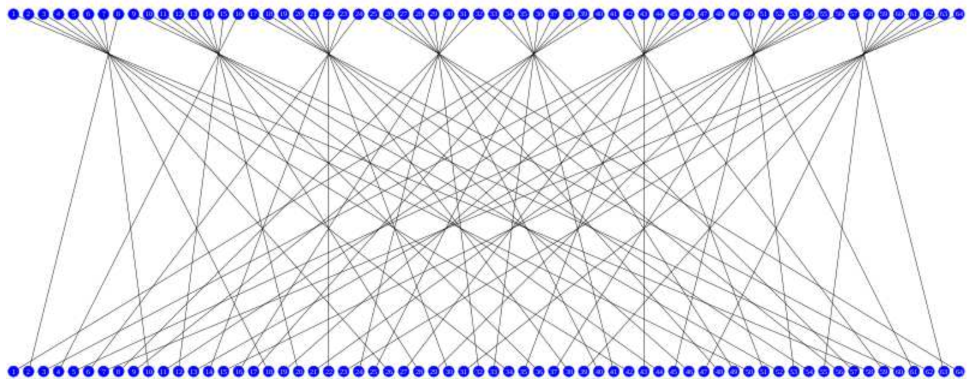


y

IP^{-1} means *Inverse Initial Permutation*

Final Permutation (IP^{-1})

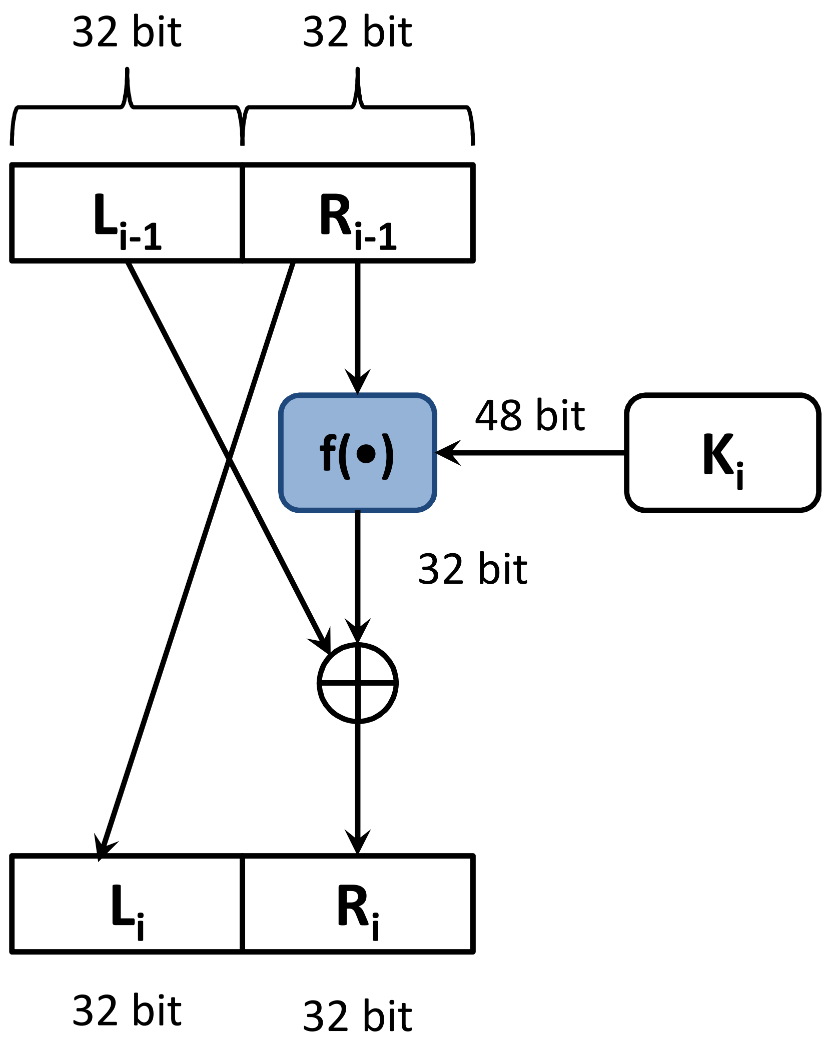
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



- The final permutation is the *inverse* of the initial permutation; the table is interpreted similarly.
  - That is, the output of the *Final Permutation* has bit 40 of the preoutput block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preoutput block is the last bit of the output.

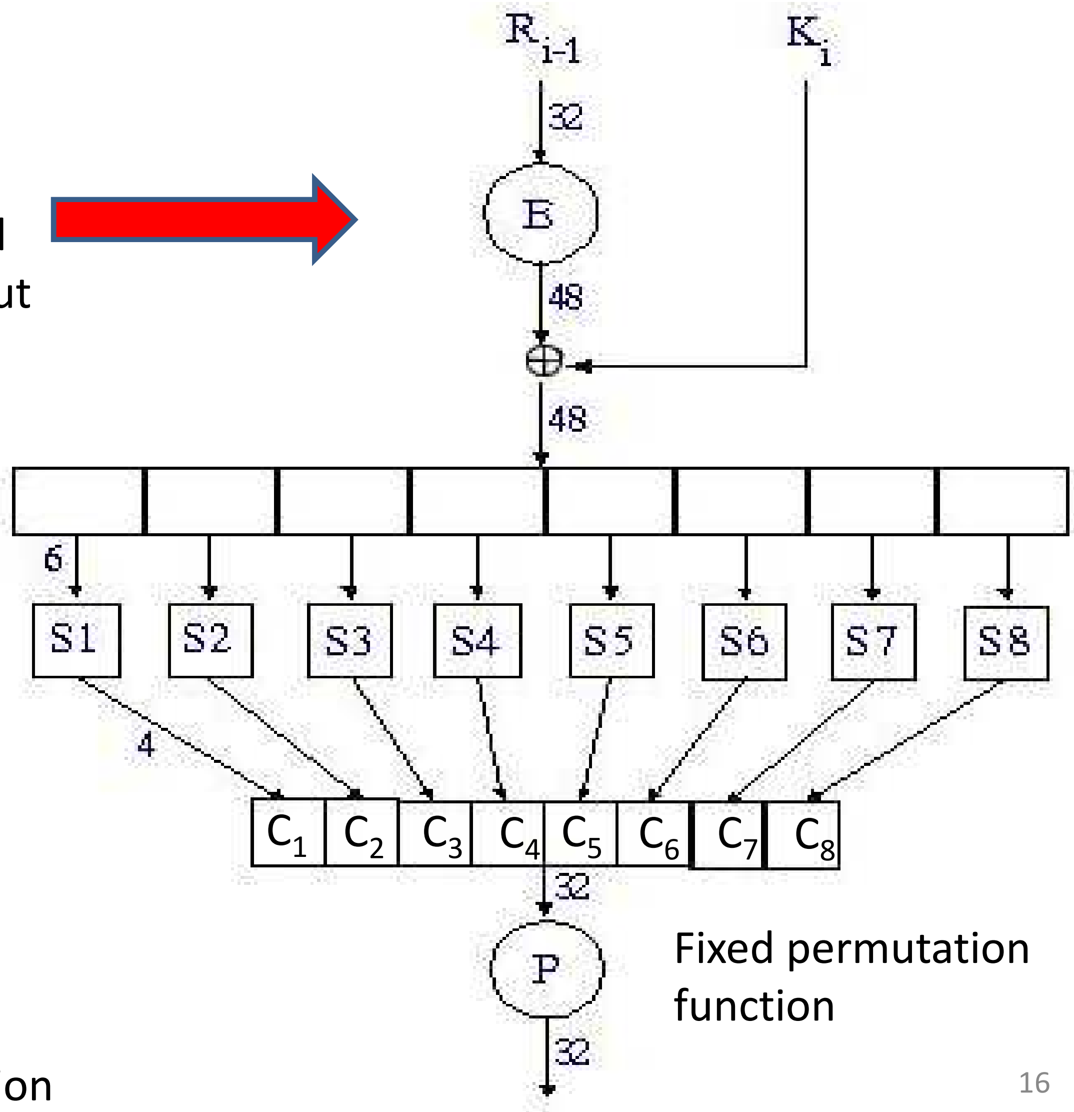
DES Round i

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



DES “f(•)” Function

E is an *expansion function* which takes a block of 32 bits as input and produces a block of 48 bits as output



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

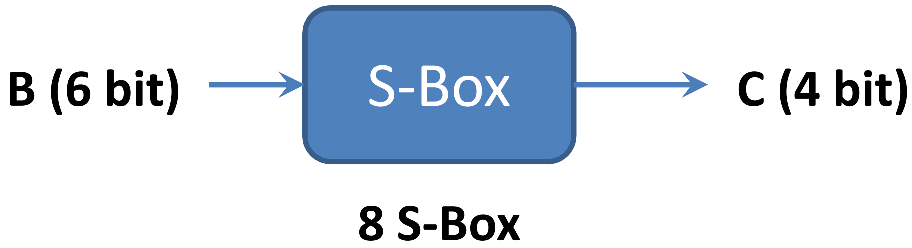
Fixed permutation function

16 bits appear twice, in the expansion

S-boxes

- S-boxes are the only ***non-linear*** elements in DES design

Each of the unique selection functions **S_1,S_2,…,S_8**, takes a 6-bit block as input and yields a 4-bit block as output



- S = matrix 4x16, values from 0 to 15
- B (6 bit long) = $b_1b_2b_3b_4b_5b_6$
  - b_1b_6 → r = row of the matrix (2 bits: 0,1,2,3)
  - $b_2b_3b_4b_5$ → c = column of the matrix (4 bits:0,1,…15)
- C (4 bit long) = Binary representation of S(r, c)

Example (S1)

Row # S_1 1 2 3 … **7** 15 Column #

Row #																
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	**7**	5	11	3	14	10	0	6	13

S(i, j) <16, can be represented with 4 bits

Example: B =101111

$b_1 b_6$ = 11 = row 3

$b_2 b_3 b_4 b_5$= 0111 = column 7

→ C=7=<u>0111</u>

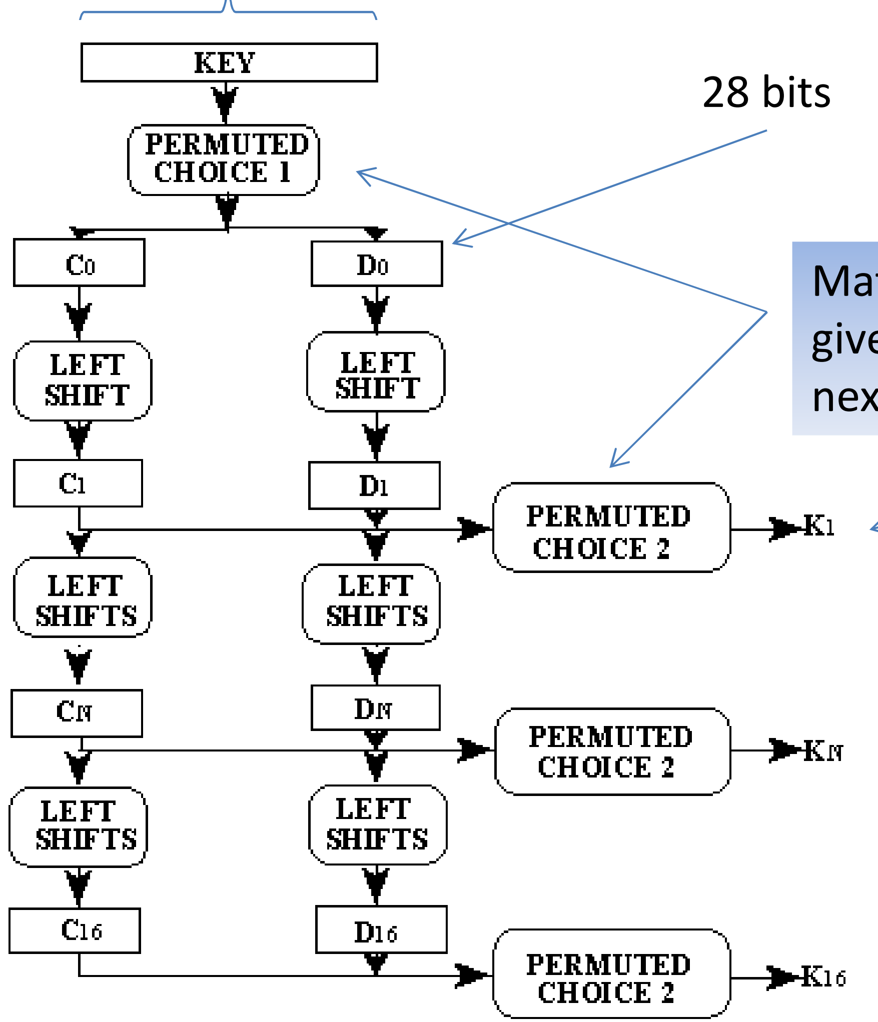
Another example: B=011011, C=?

DES Key Generation ($K_1 – K_{16}$)

64 bit key (including parity-check bits)

28 bits

28 bits



Matrix PC-1 and PC-2 are given by the standard (see next slide)

48 bits

$C_i=LS_i(C_{i-1})$
$D_i=LS_i(D_{i-1})$
$K_i=PC\text{-}2(C_iD_i)$

LS=Left Shift
-shift *one* position
 if i=1,2,9 or 16
-shift *two* positions
 otherwise

DES Permuted Choice 1 and 2 (PC-1, PC-2)

Parity-check bits (namely, bits 8,16, 4,32,40,48,56,64) are not chosen, they do not appear in **PC-1**

Left						
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
Right						
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

PC-2 selects the 48-bit subkey for each round from the 56-bit key-schedule state

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

DES Weak Keys

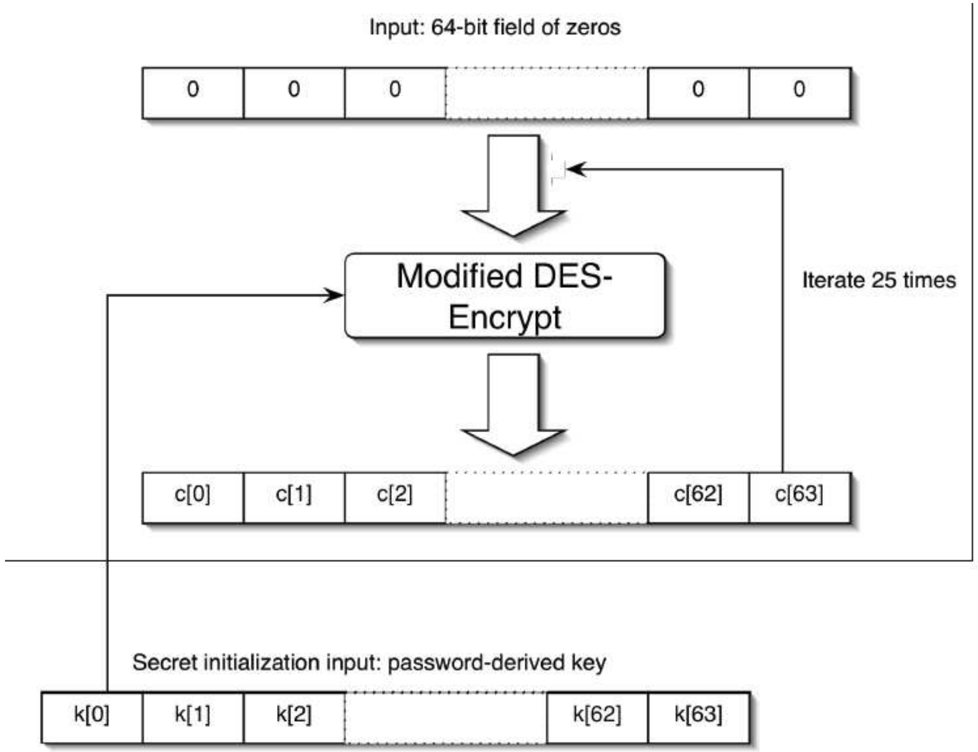
- DES uses 16 48-bits keys generated from a master 56-bit key (64 bits if we consider also parity bits)
- Weak keys: keys make the same sub-key to be generated in more than one round.
- Result: reduce cipher complexity
- Weak keys can be avoided at key generation.
- DES has 4 weak keys
  - 01010101 01010101
  - FEFEFEFE FEFEFEFE
  - E0E0E0E0 F1F1F1F1
  - 1F1F1F1F 0E0E0E0E

DES Decryption

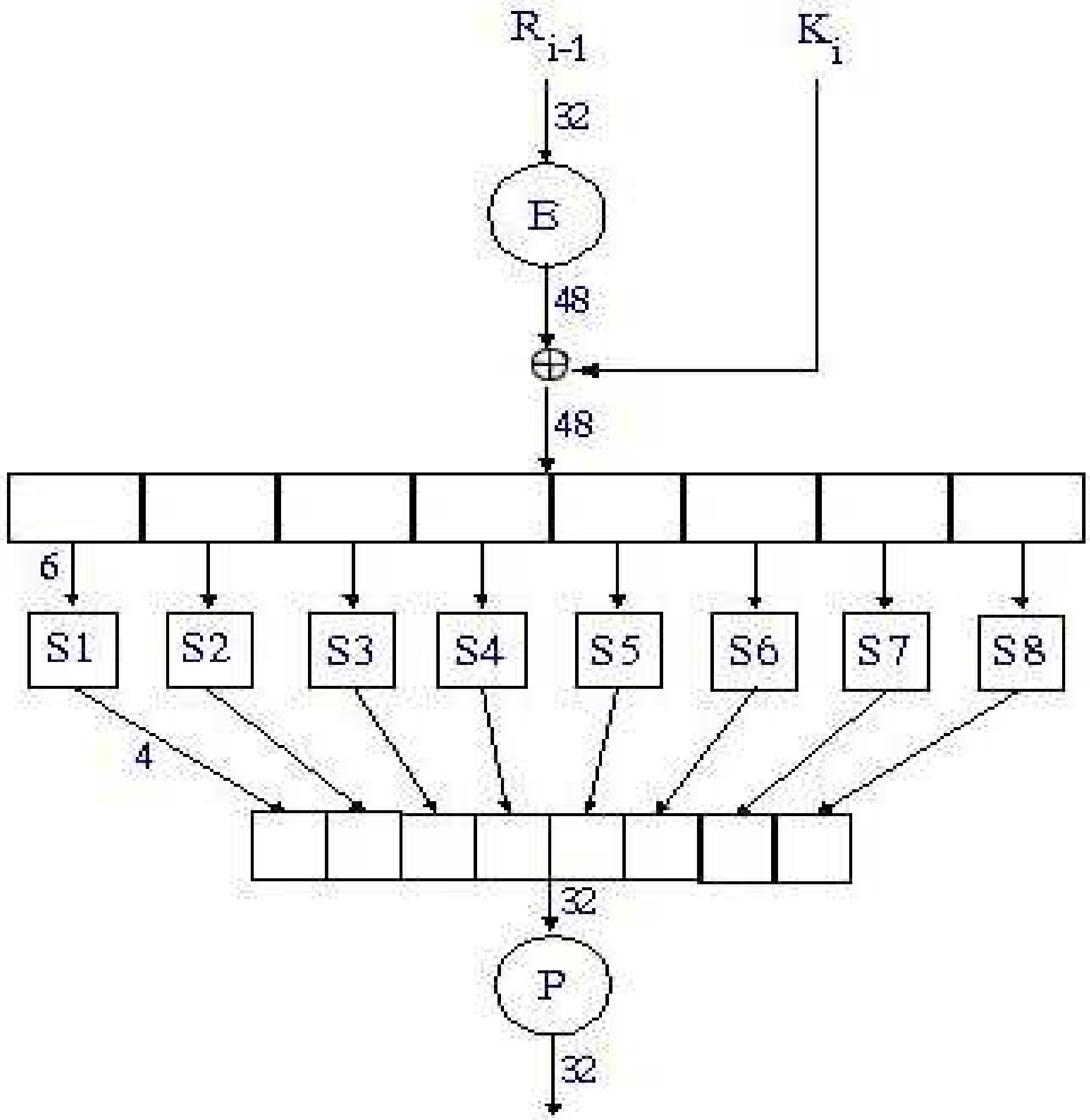
- Decryption uses the same algorithm as encryption, except that the subkeys K_1, K_2, …K_{16} are applied in reversed order

Unix crypt

- Password encryption function of Unix systems
- Password used as DES key (truncated to 8 characters, each coerced down to 7 bits 8*7= 56 bits DES key)
- An all-zeros block in encrypted always with the same key …
- … and so on for 25 DES rounds
- Salt (12 bits, two-character string) used to address dictionary attacks.
  - This string is used to perturb the algorithm in one of 4096 different ways.



DES “f(•)” Function



Salt

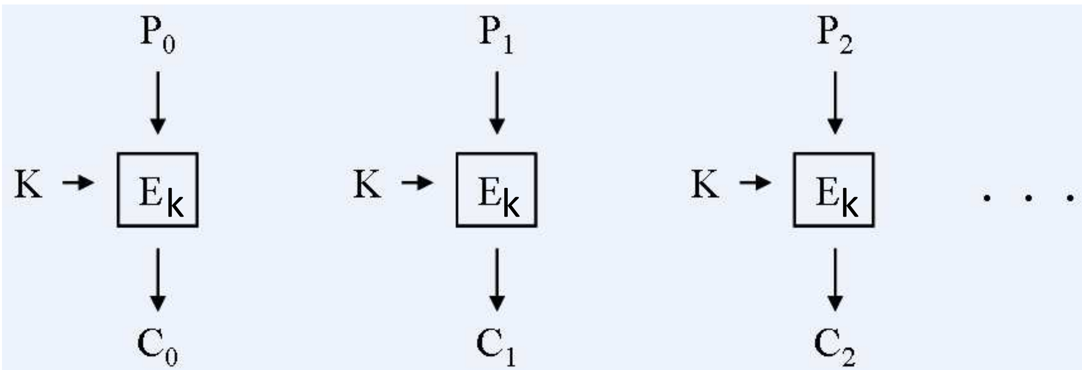
- 12-bit Salt is chosen randomly, stored with the password
- Salt creates 4096 different DES functionings: if the ith bit of the salt is set (non-zero), then the bits i and i+24 of the output of the expansion function are swapped.
- Result: same password will have different encryptions in the password file
- Dictionary attack is still possible!

Block Cipher Encryption Modes: ECB

- Message is broken into independent blocks of *block_size* bits;
- **Electronic Code Book** (**ECB**): each block encrypted separately.
- **Encryption**: $C_i = E_k(P_i)$
- **Decryption**: $P_i = D_k(C_i)$

E_k = DES encryption function

D_k = DES decryption function



Properties of ECB

- Deterministic: the same data block gets encrypted the same way; this reveals patterns of data when a data block repeats.

- Malleable: reordering ciphertext results in reordered plaintext.

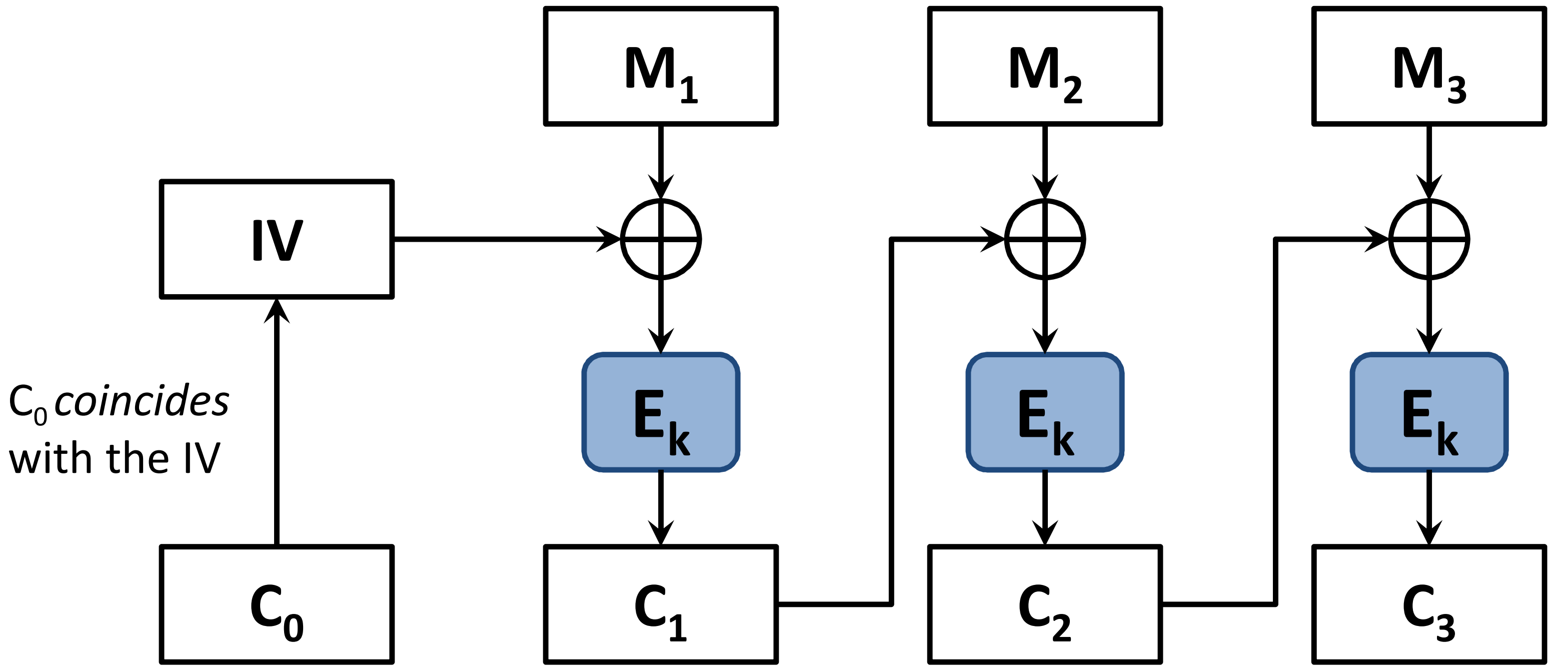
- Errors in one ciphertext block do not propagate.
- **Usage**: not recommended to encrypt more than one block of data.

DES Encryption Modes: CBC

- **Cipher Block Chaining** (**<u>CBC</u>**): *next* input depends upon *previous* output
- **Encryption**: $C_i = E_k(M_i \oplus C_{i-1})$, with C_0=IV
- **Decryption**: $M_i = C_{i-1} \oplus D_k(C_i)$, with C_0=IV

E_k = DES encryption function

D_k = DES decryption function



Properties of CBC

- Randomized encryption: repeated text gets mapped to different encrypted data.
  - can be proven to be "secure" assuming that the block cipher has desirable properties and that <u>*random IV's*</u> are used

- A ciphertext block depends on all preceding plaintext blocks; reorder affects decryption

- Errors in one block propagate to two blocks
  - one bit error in C_j affects all bits in M_j and one bit in M_{j+1}

- Sequential encryption, cannot use parallel hardware

<u>Usage:</u> chooses random IV and protects the integrity of IV

Observation:

if $\mathbf{C_i = C_j}$ then $E_k(M_i \oplus C_{i-1}) = E_k(M_j \oplus C_{j-1})$;

thus $\mathbf{M_i \oplus C_{i-1} = M_j \oplus C_{j-1}}$

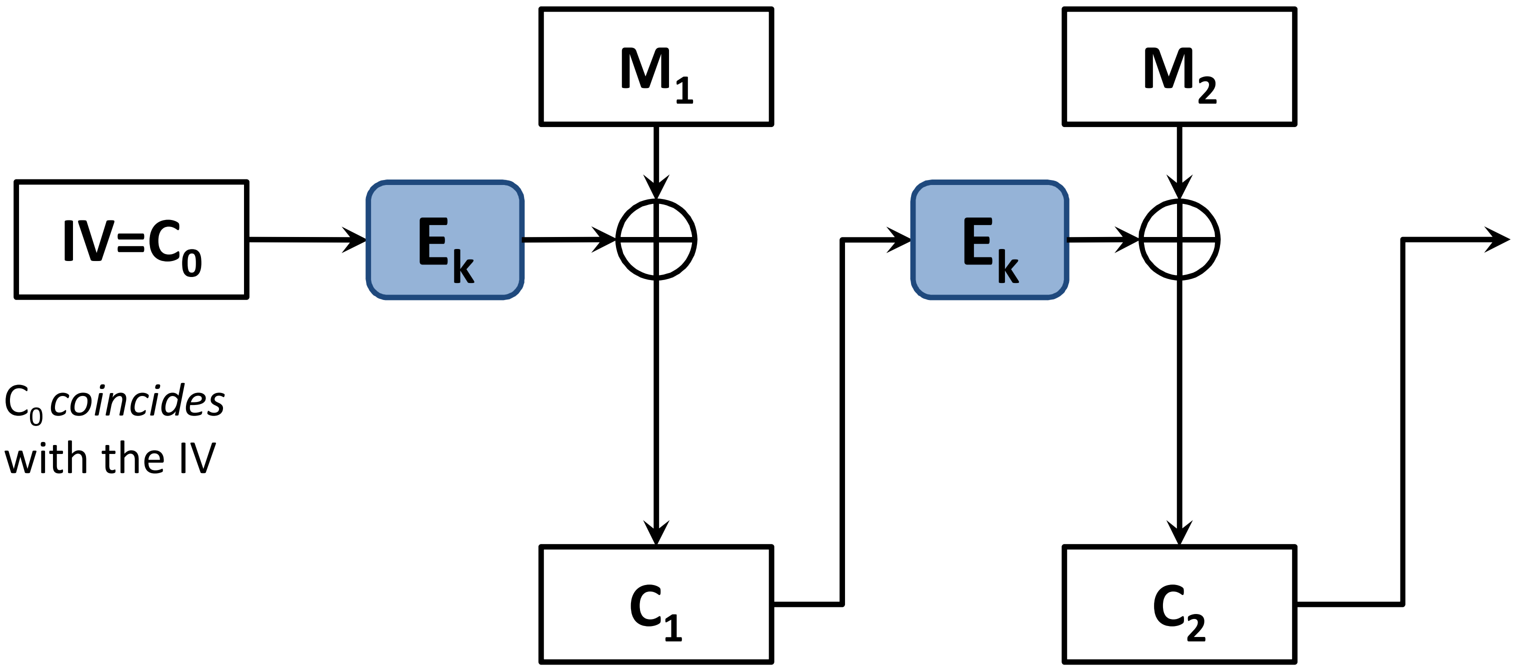
thus $\mathbf{M_i \oplus M_j = C_{i-1} \oplus C_{j-1}}$

Use DES to construct Stream Ciphers

- **Cipher Feedback** (**CFB**)
- **Output Feedback** (**OFB**)
- **Counter Mode** (**CTR**)
- Common properties:
  - uses only the encryption function E_k of the cipher both for encryption and for decryption
  - malleable: possible to make predictable bit changes

Encryption Modes: CFB

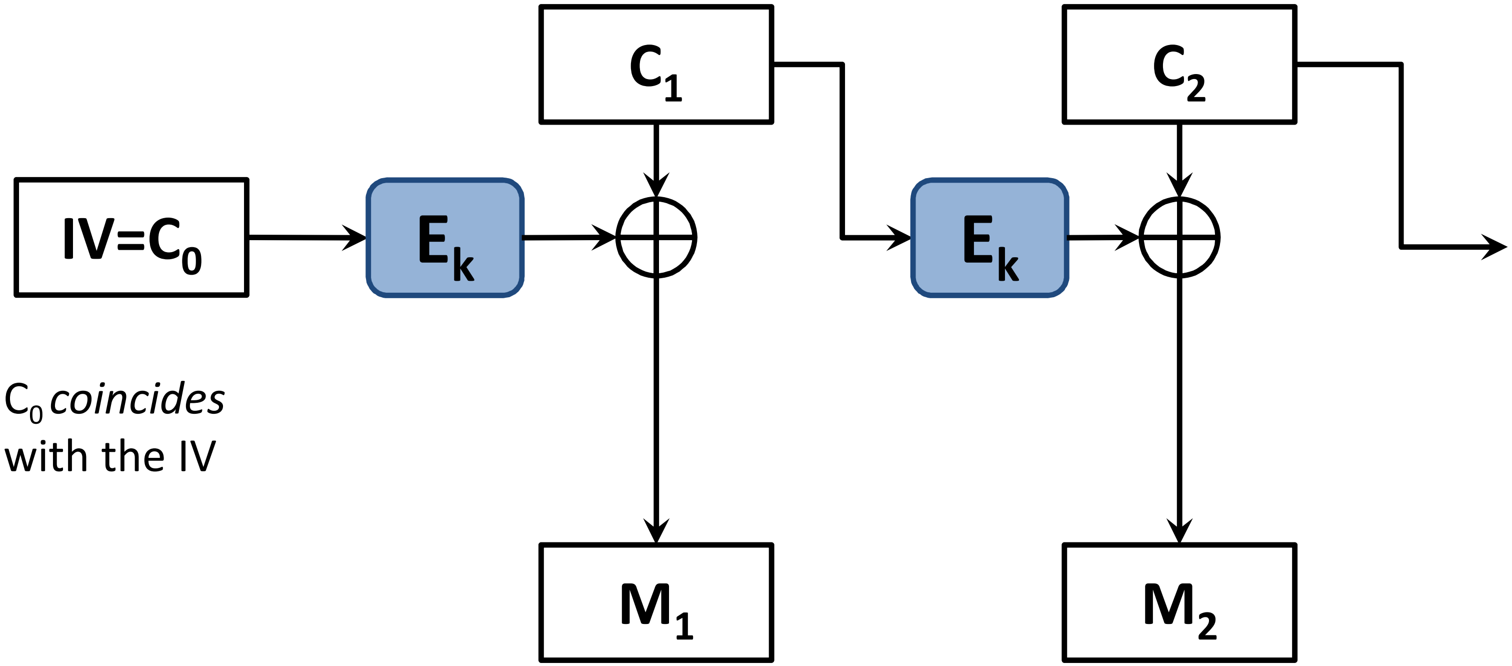
- **Cipher Feedback (CFB)**: the message is XORed with the feedback of encrypting the previous block
- **<u>Encryption</u>**: $C_i = M_i \oplus E_k(C_{i-1})$, with C_0=IV



C_0 *coincides* with the IV

Encryption Modes: CFB

- **Decryption**: $M_i = C_i \oplus E_k(C_{i-1})$, with C_0=IV
- The *same* encryption function E_k is used here also for decryption



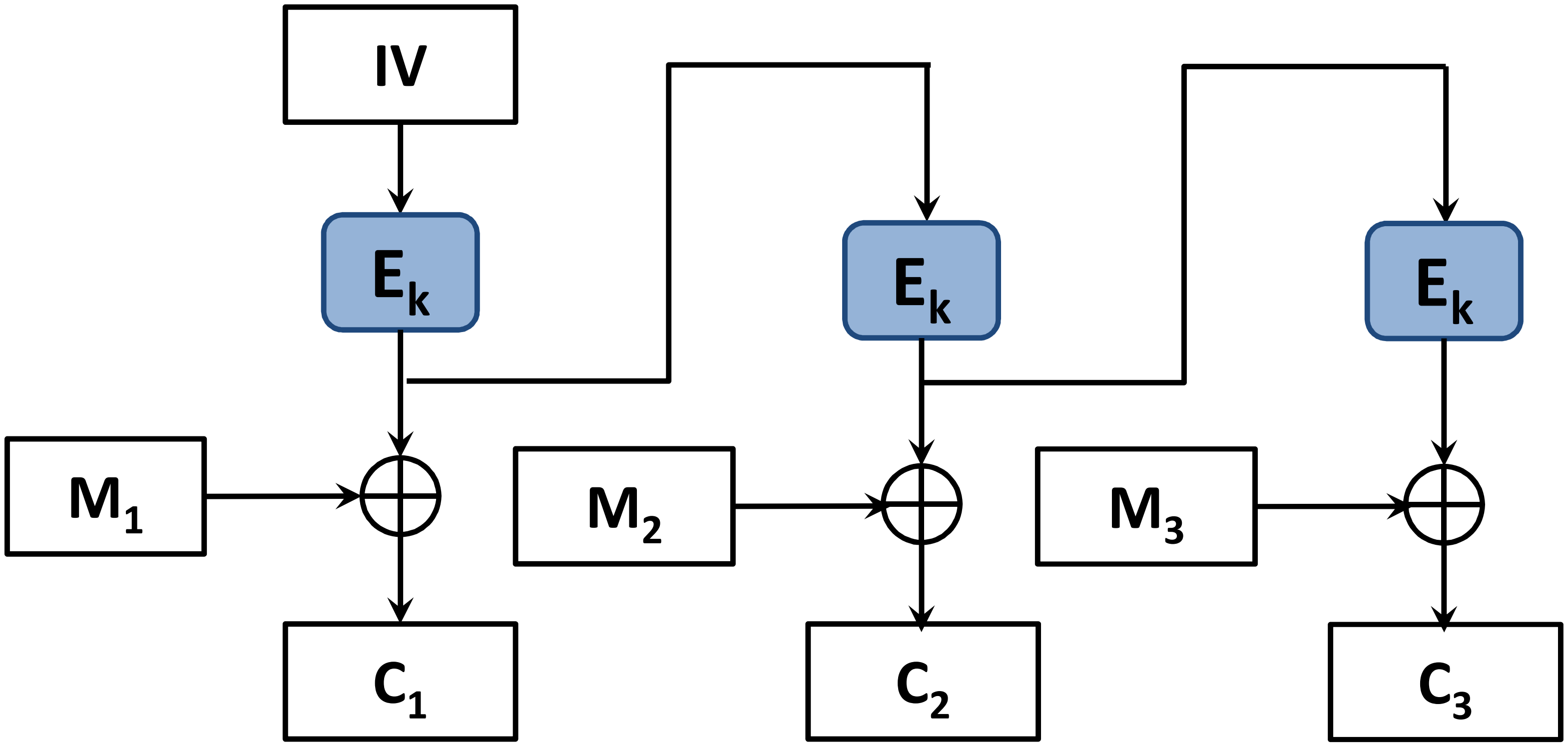
C_0 *coincides* with the IV

Properties of CFB

-  Randomized encryption
-  A ciphertext block depends on all preceding plaintext blocks; reorder affects decryption
-  Errors propagate for several blocks after the error, but the mode is self-synchronizing (like CBC).
-  Decreased throughput.
  – Can vary the number of bits feed back, trading off throughput for ease of use
-  Sequential encryption

Encryption Modes: OFB

- **Output Feedback (OFB)**:
  - constructs a Pseudo Random Number Generator using DES E_k function



Properties of OFB

- Randomized encryption
- Sequential encryption, but pre-processing possible
- Error propagation limited
- Subject to limitations of stream ciphers

Encryption Modes: CTR

- **Counter Mode** (**CTR**): Another way to construct PRNG using DES
  - **Encryption**: $C_i = M_i \oplus E_k[\text{nonce} + i]$ (counter: i)
  - nonce= number used only once
    (equivalent to an IV=Initialization Vector)
  - **Decryption**: $M_i = C_i \oplus E_k[\text{nonce} + i]$
  - Sender and receiver share: nonce (does *not* need to be secret) and the secret key *k*.

Properties of CTR

- *Software and hardware efficiency*: different blocks can be encrypted in parallel.
- *Preprocessing*: the encryption part can be done offline and when the message is known, just do the XOR.
- *Random access*: decryption of a block can be done in random order, very useful for hard-disk encryption.
- *Messages of arbitrary length*: ciphertext is the same length with the plaintext (i.e., no IV).

Cryptanalysis of DES

DES Weak Keys

- DES has 4 weak keys (64-bit)
  - 01010101 01010101
  - FEFEFEFE FEFEFEFE
  - E0E0E0E0 F1F1F1F1
  - 1F1F1F1F 0E0E0E0E
- Using weak keys, the outcome of the Permuted Choice 1 (PC1) in the DES key schedule leads to round keys (K_1---K_{16}) being either *all zeros*, *all ones* or *alternating zero-one* patterns.

- Since all the subkeys are identical, and DES is a Feistel network, the encryption function becomes self-inverting; that is, encrypting twice with a weak key K produces the original plaintext.
  - $E_K(E_K(x))=x$ for all x, i.e., the encryption and the decryption are the same
- Weak keys should be avoided at key generation.

DES semi-weak keys

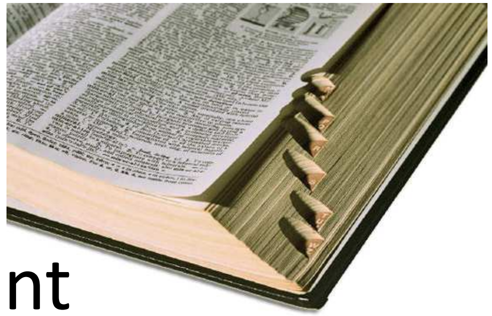
- DES has also *semi-weak keys,* which only produce two different subkeys, each used eight times in the algorithm
- We can refer to them as K_1 and K_2
- They have the property that $E_{K1}(E_{K2}(x))=x$
- There are six pairs of DES semi-weak keys

- Note that *weak* and *semi-weak* keys are not considered "fatal flaws" of DES. There are 2^{56} (7.21×10^{16}) possible keys for DES, of which *only* four are weak and twelve are semi-weak …

Cryptanalysis of DES

- **Brute Force:**
- Known-Plaintext Attack (the cryptanalyst knows one or several pairs of ciphertext and the corresponding plaintext.)
- Try all 2^{56} possible keys
- DES challenges: a series of brute force attack contests created by *RSA Security*
- msg="The secret message is: xxxxxxxx"
  - First challenge in 1997 (thousands of volunteers connected by Internet) : solved in 96 days (3 months). Message was "The secret message is: Many hands make light work."
  - 1998 EFF (Electronic Frontier Foundation, non-profit organization) machine (costs $250K): 3 days
  - 1999 (distributed.net and Deep Crack, combined): 22 hours and 15 minutes (Message was *"See you in Rome (second AES Conference, March 22-23, 1999)")*

Cryptanalysis of DES



- **Dictionary attack:**
- Each plaintext may result in 2^{64} different ciphertexts, but there are only 2^{56} possible different key values.
- Encrypt the known plaintext with all possible keys.
- Keep a *look up table* of size 2^{56}
- Given a Plaintext/Ciphertext pair (P,C), look up C in the table

Double DES

- DES uses a 56-bit key, this raised concerns about brute force attacks.
- One proposed solution:

  double DES.

- Apply DES twice using two keys, K1 and K2.
  - Encryption: **$C = E_{K2} [E_{K1} [P]]$**
  - Decryption: **$P = D_{K2} [D_{K1} [C]]$**
- This leads to a 2x56=112 bit key, so it is more secure than DES. **Is it?**

Meet-in-the-Middle Attack

- To improve the security of a *block cipher*, one might get the (naive) idea to simply use two independent keys to encrypt the data twice.
- **$C = E_{K2} [E_{K1} [P]]$**
- Naively, one might think that this would *square* the security of the double-encryption scheme.
- In fact, an exhaustive search of all possible *combinations* of keys would take 2^{2n} attempts (if each key K1, K2 is n bits long), compared to the 2^n attempts required for searching a single key.

Meet-in-the-Middle Attack

- Assume the attacker knows a set of Plaintext (P) and Ciphertext (C). That is,

  $C = E_{K2} [E_{K1} [P]]$

  where E is the encryption function (cipher), and K_1 and K_2 are the two keys.

1) The attacker can first compute $E_K(P)$ for all possible keys K and *store* the results in memory (in a lookup table).
2) Afterwards he can decrypt the ciphertext by computing $D_K(C)$ for each K.

- Any matches between these two resulting sets are likely to reveal the correct keys. (To speed up the comparison, the $E_K(P)$ set is stored in an in-memory *lookup table*, then each $D_K(C)$ can be matched against the values in the lookup table to find the candidate keys.)
- Once the matches are discovered, they can be verified with a second test-set of Plaintext and Ciphertext.
- If the key-size is n, this attack uses only 2^{n+1} (for Double DES, $2^{56+1}=2^{57}$) encryptions/decryptions (and $O(2^n)$ memory space) in contrast to the naive attack, which needs 2^{2n} encryptions/decryptions (but only $O(1)$ space).

Time-Memory tradeoff

Triple DES (Triple Data Encryption Algorithm, TDEA)

- Use three different keys
  - Encrypt: $\mathbf{C = E_{K3} [\ D_{K2} [\ E_{K1} [P]\ ]\ ]}$
  - Decrypt: $\mathbf{P = D_{K1} [\ E_{K2} [\ D_{K3} [C]\ ]\ ]}$

- The standard specifies three keying options:

1) Keying option 1: All three keys are independent.
2) Keying option 2: K_1 and K_2 are independent, and $K_3 = K_1$.
3) Keying option 3: All three keys are identical, i.e. $K_1 = K_2 = K_3$.

- Using keying option 1: the key space is 56 x 3 = 168 bits
- No known practical attack against it.
- Many protocols/applications use 3DES (example PGP)
  - The electronic payment industry uses Triple DES and continues to develop and promulgate standards based upon it (e.g. EMV, Europay-Visa-Mastercard).

Triple DES (Triple Data Encryption Algorithm, TDEA)

- Question: if we use three completely different keys $K_1 \neq K_2 \neq K_3$ …
  - Encrypt: **$C = E_{K3} [D_{K2} [E_{K1} [P]]]$**
  - Decrypt: **$P = D_{K1} [E_{K2} [D_{K3} [C]]]$**

- **… will the effective strength be that of 56x3= 168 bits?**

- Keying option 2 provides less security than option 1, with 2 × 56 = 112 key bits. However, this option is *stronger* than double DES (with K_1 and K_2), because it protects against meet-in-the-middle attacks.
  - Note that this option is susceptible to certain chosen-plaintext or known-plaintext attacks, and thus it is designated by NIST to have only 80 bits of *real* security

- Keying option 3 is equivalent to DES, with only 56 key bits. This option provides *backward compatibility* with DES.

Differential Cryptanalysis (Biham-Shamir)

- **Main idea:**
- This is a *chosen plaintext attack*, assumes than an attacker knows (Plaintext, Ciphertext) pairs
- Diff. Cryptanalysis involves comparing the XOR of 2 plaintexts to the XOR of the 2 corresponding ciphertexts
- *Difference* $\Delta_P = P_1 \oplus P_2$, $\Delta_C = C_1 \oplus C_2$
- Distribution of Δ_C's given Δ_P may reveal information about the key (certain key bits)
- After finding several bits, use brute-force for the rest of the bits to find the key.

Differential Cryptanalysis of DES

- Surprisingly … DES was resistant to differential cryptanalysis.
- At the time DES was designed, the authors *already* knew about differential cryptanalysis. S-boxes were designed to resist differential cryptanalysis.
- Against 8-round DES, such attack requires 2^{38} known plaintext-ciphertext pairs (a couple of minutes on a small PC).
- Against 16-round DES, attack requires 2^{47} chosen plaintexts.
- **Differential cryptanalys is not effective against DES in practice.**

Linear Cryptanalysis of DES

- Another attack described in 1993 by M. Matsui
- Instead of looking for isolated points at which a block cipher behaves like something simpler, it involves trying to ***create a simpler approximation to the block cipher*** as a whole.
- It is an attack that can be applied to an iterated cipher.

Linear Cryptanalysis of DES

- M. Matsui showed (1993/1994) that DES can be broken:
  - 8 rounds: 2^{21} known plaintext
  - 16 rounds: 2^{43} known plaintext, 40 days to generate the pairs (plaintext, ciphertext) and 10 days to find the key
- **The attack has no practical implication, requires too many pairs.**
- Exhaustive search remains the most effective attack.

DES Strength Against Various Attacks

Attack Method	Known	Chosen	Storage Complexity	Processing Complexity
Exhaustive precomputation	**-**	**1**	**2^{56}**	**1**
Exhaustive search	**1**	**-**	**Negligible**	**2^{55}**
Linear cryptanalysis	**2^{43}** **2^{38}**	**-** **-**	**For texts**	**2^{43}** **2^{50}**
Differential cryptanalysis	**-** **2^{55}**	**2^{47}** -	**For texts**	**2^{47}** **2^{55}**

The weakest point of DES remains the size of the key (56 bits)!

How to Improve Block Ciphers

- Variable key length
- Mixed operators: use more than one arithmetic and/or Boolean; this can provide non-linearity
- Data dependent rotation
- Key-dependent S-boxes
- Lengthy key schedule algorithm
- Variable plaintext/ciphertext block length
- Variable number of rounds
- Operation on both data halves each round
- Variable *f()* function (varies from round to round)
- Key-dependent rotation