Data Encryption Standard

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
email: gianluca.dini@unipi.it
Version: 10/03/2025

1

Data Encryption Standard

- May 15, 1973 (now NIST)
  - National Bureau of Standards (NBS) published a solicitation for cryptosystems in the Federal Register (mildly revolutionary act)
- 1974
  - IBM submitted LUCIFER (n = 64, k = 128)
  - DES was a modification of LUCIFER (n = 64, k = 56, resistant to differential cryptanalysis) under NSA guidance
- March 17, 1975
  - DES was published in the Federal Register

2

Data Encryption Standard

- January 15, 1977 (FIPS PUB 46)
  - (called DEA) considered a standard for "unclassified" applications, after much public discussion
  - Reviewed every 5 years, being January 1994 the most recent review
  - Not a standard since 1998
- 1999 (FIPS PUB 46-3)
  - DES recommended for legacy systems (deprecated)
  - 3DES Recommend (not suitable for performance reasons: DES was 70s tech)
  - DES replaced by AES

↓ brute force was becoming easy

Mar-25 Data Encryption Standard (DES) 3

3

Confusion and diffusion

DES implements Shannon theory: 2 properties

- **Two primitives for strong ciphers** (Shannon 1949)
  - **DIFFUSION** is an encryption operation where the influence of one PT symbol is spread over many CT symbols with the goal of hiding statistical properties of the PT ①
    - A simple diffusion element is ***permutation***
    - DES uses permutations
    - AES uses *MixColumn*
  - **CONFUSION** is an encryption operation where the relationship between key and CT is obscured.
    - A common element to achieve confusion is ***substitution***
    - AES and DES use substitution

Of course I want to maximum invertibility

Mar-25 Data Encryption Standard (DES) 4

4 ① The Monoalph. contained no diffusion, influence of a character only in one char.

Diffusion is a concept in cryptography that ensures that **one plaintext (PT) symbol** (e.g., a bit or a character) influences **many ciphertext (CT) symbols** after encryption. The goal is to **hide patterns** in the plaintext by spreading its influence across the ciphertext.

This makes it difficult for an attacker to extract useful information about the plaintext by analyzing the ciphertext.

Why is Diffusion Important?

Imagine an encryption scheme without diffusion—if one character in the plaintext changed, only **one character in the ciphertext** would change. This would allow an attacker to notice patterns and make guesses about the plaintext structure.

By ensuring that **small changes in plaintext affect multiple parts of the ciphertext**, diffusion makes cryptanalysis (like frequency analysis) much harder.

What is Confusion in Encryption?

Confusion ensures that the **relationship between the encryption key and the ciphertext (CT) is obscured**. This means that even if an attacker has the ciphertext, they should not be able to easily determine anything about the key.

The goal of confusion is to make the encryption process **complex and unpredictable**, so that **changing one bit in the key results in an entirely different ciphertext**.

Why is Confusion Important?

If there was no confusion, an attacker might be able to find patterns that link certain ciphertexts to certain keys, making it easier to break the encryption. A good encryption algorithm should ensure that even a small change in the key **drastically** changes the ciphertext.

For example:

- Without confusion: Changing **one bit** in the key might change only **one bit** in the ciphertext. This is bad because it makes it easier to analyze key patterns.
- With confusion: Changing **one bit** in the key causes the entire ciphertext to change in a complex, unpredictable way.

A good diffusion property

- (INFORMAL) Changing of one bit of PT results on average in the change of half the output bits of the CT, i.e., If PT → PT' ➔ CT → CT' s.t. CT' looks *statistically independent* of CT

The fact that I'm not able to find correlation between pt, ct, k, this is because of confusion

5

Product cipher

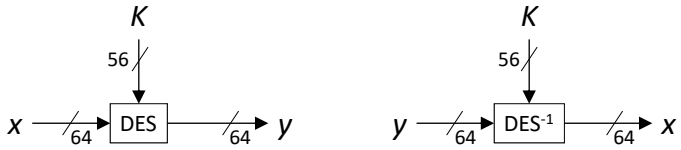
- Confusion only or diffusion only is not secure
  - E.g., shift cipher and Enigma used confusion only
- Confusion and diffusion must be concatenated to build a strong cipher
- Product ciphers are composed of rounds which concatenate confusion and diffusion
  - DES (r = 16)
  - 3DES (r=48)
  - AES-128 (n=10)

each round perform diffusion and confusion

6

Data Encryption Standard (DES)

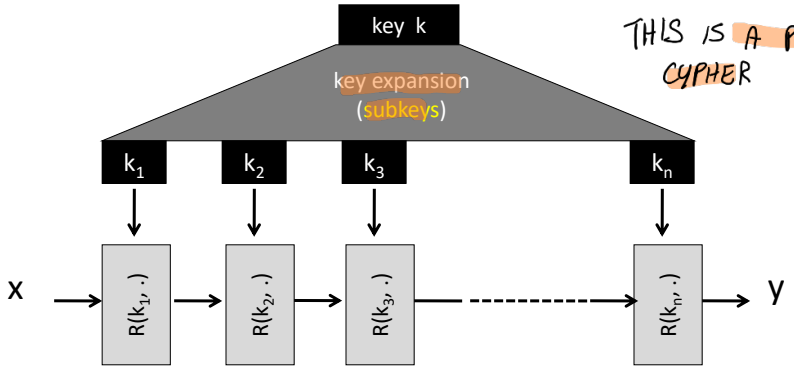
- The 56-bit input key K is specified as a 64-bit key
  - 8 bits (bits 8; 16, …, 64) are used as parity bits
  - The key is 56-bit long



- DES is a product cipher of 16 rounds

7

Block Ciphers Built by Iteration



THIS IS A PRODUCT CYPHER

- $R(k_i, \cdot)$ is the round function
- K_i: subkeys, one per round

every round implements confusion and diffusion

K_i = round subkey

8

Cone is simpl. of round fct. and of key expansion

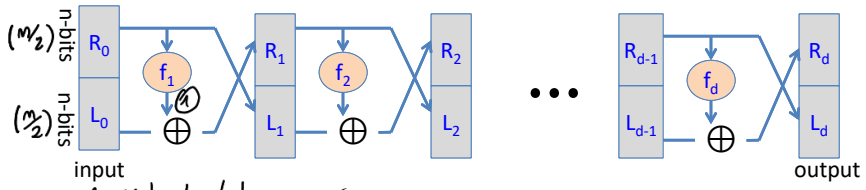
How the round functions are implemented

Feistel Network

Given functions $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$ (should not be linear, otherwise whole system would be linear)

Goal: build invertible function $F: \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$

d = 16 rounds



plaintext divided in 2 pieces

Last round

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f_i(R_{i-1}) \end{cases}$$

① Pseudo random bits

f_1 takes corresponding round key: f introduces confusion and diffusion

$f_1, f_2 \ldots f_d$ can be ANY function, but the whole function has to be invertible

Mar-25 Data Encryption Standard (DES) 9

9

Round f-function

- Function f realizes diffusion and confusion
- Function f can be considered as a pseudorandom generator with two inputs:
  1. Right half of the input R_{i-1}
  2. The round subkey k_i (not shown in the picture)

Mar-25 Data Encryption Standard (DES) 10

10

$$\begin{cases} L_{i} = R_{i-1} \\ R_{i} = L_{i-1} \oplus f(R_{i-1}) \end{cases} \Rightarrow \begin{array}{l} R_{i-1} = L_{i} \\ L_{i-1} = R_{i} \oplus f(L_{i}) \end{array}$$

Feistel net is invertible

Theorem: for *any* $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$, Feistel network F: $\{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is invertible

Proof: *construct inverse*

Given

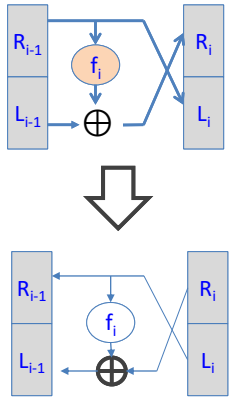
$R_i = L_{i-1} \oplus f_i(R_{i-1})$

$L_i = R_{i-1}$

then

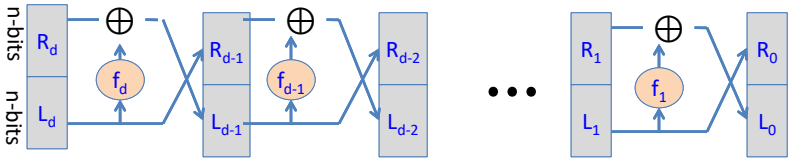
$R_{i-1} = L_i$

$L_{i-1} = R_i \oplus f_i(R_{i-1}) = R_i \oplus f_i(L_i)$



11

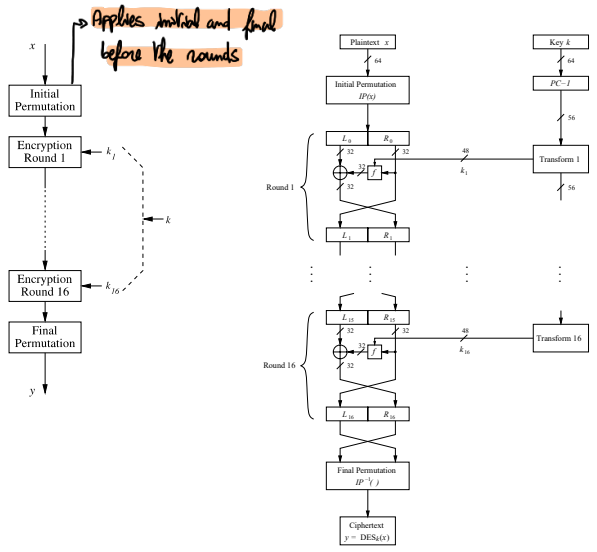
Decryption circuit



- Inversion is basically the same circuit, with $f_1, \ldots, f_d$ applied in reverse order
- FN is a general method for building invertible functions (block ciphers) from arbitrary functions f.

12

The internal structure of DES



13

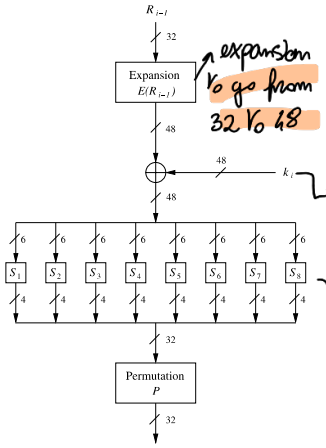
Initial and final permutation

- IP and IP^{-1}
  - Very fast hw implementation
  - Don't increase DES security
  - Their rationale is not known

14

The *f*-function

Block is 64 bits. f works on half the block, 32 bits.



expansion to go from 32 to 48

- Expansion box E increases diffusion
- S-boxes provide confusion
- Permutation P increases diffusion

→ XORed with the round subkey (56 bit key → get 16 rounds keys for 48 bits)

→ split in groups of 6 bits.
Each group is input of substitution boxes: takes 6 bits and return 4 bits.

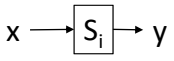
15

S-box

- Provide confusion
  - Core of the DES cryptographic strength
  - The motivations behind S-box were never motivated
- Lookup table: $\{0, 1\}^6 \rightarrow \{0, 1\}^4$ They are basically LUT
  - Larger tables would be better but 4-by-6 tables were close to the maximum size for ICs in the 70s First tech limitation
- The only non-linear element of the system
  - $S(a \oplus b) \neq S(a) \oplus S(b)$
    - If S_i's were linear then DES could be described by a linear system where key bits are the unknowns ➔ easily solved (KPA)

16

S-boxes



8 S-boxes

$x = b_1 b_2 b_3 b_4 b_5 b_6$

Row → $b_1 b_6$ (outer bits)
Column → $b_2 b_3 b_4 b_5$ (inner bits)

row	column number															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
S_1																
[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_2																
[0]	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
[1]	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
[2]	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
[3]	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																
[0]	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
[1]	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
[2]	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
[3]	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																
[0]	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
[1]	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
[2]	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
[3]	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																
[0]	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
[1]	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
[2]	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
[3]	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																
[0]	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
[1]	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
[2]	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
[3]	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																
[0]	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
[1]	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
[2]	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
[3]	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																
[0]	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
[1]	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
[2]	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
[3]	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

17

S-box S_5

S_i: $\{0,1\}^6 \longrightarrow \{0,1\}^4$

$S_5(011011) \rightarrow 1001$

S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

18

The *f*-function - criteria

- The f-function is the core of DES security
- The f-function must be strongly non-linear
- Design criteria
  - Strict avalanche criterion
  - Bit independence criterion: if I commute 1 bit in the input the output bits should appear to change independently

19

S-box

- Design criteria
  - Notation
    - Let in_i denote the i-th input of s-box S
    - Let out_j denote the j-th output of s-box S
  - Strict avalanche criterion
    - If in_i of S is commuted, then out_j commutes with probability 0.5, for all i, j — For the S-box
  - Bit independence criterion
    - If in_i of S is commuted, then out_j and out_k commute independently, for all i, j, and k

In the end those are deterministic functions

20

Avalanche effect

- (Intuition) A "small" change in the plaintext or the key (e.g., 1 bit) must produce a "meaningful" change in the ciphertext
- DES
  - Every bit at the end of the 5-th round depends on every plaintext bit and key bit
  - The ciphertexts corresponding to two plaintexts differing on a single bit differ on average for 32 bit (same key)
  - The ciphertexts corresponding to two keys differing on a single bit differ on average for 32 bit (same plaintext)

21

S-box – Design criteria (refined)

1. Each S-box has six input bits and four output bits.
2. No single output bit should be too close to a linear combination of the input bits.
3. If the lowest and the highest bits of the input are fixed and the four middle bits are varied, each of the possible 4-bit output values must occur exactly once.
4. If two inputs to an S-box differ in exactly one bit, their outputs must differ in at least two bits. [%]

22

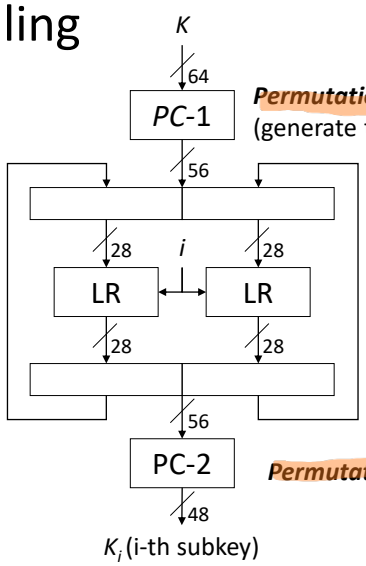
S-box – Design criteria (refined)

4. If two inputs to an S-box differ in the two middle bits, their outputs must differ in at least two bits.
5. If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must be different.
6. For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.
7. A collision (zero output difference) at the 32-bit output of the eight S-boxes is only possible for three adjacent S-boxes.

23

Key scheduling

- PC1 e PC2 guarantee that, at each round, a different subset of bits is extracted
- Each bit of the key participates to 14 rounds on average



Permutation and compression
(generate the first sub-key)

Left rotation
(*1 bit per round = 1, 2, 9, 16; 2 bit otherwise*)

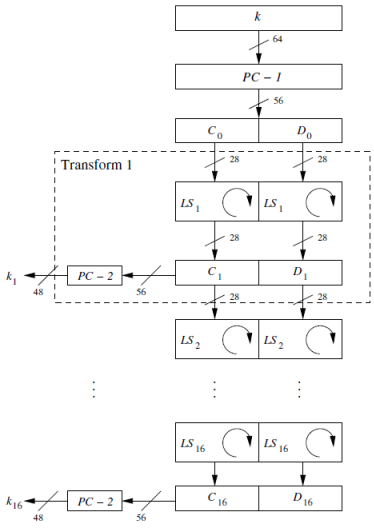
Permutation and compression

24

Summary of the Process

1. **Take a 64-bit key** → Remove 8 parity bits → Get **56-bit key**.
2. **Apply PC-1** → Rearrange into **two 28-bit halves (C and D)**.
3. **For each round (1-16)**:
    - **Left shift C and D**.
    - **Apply PC-2** to select **48 bits** → This is the **round key** for that round.
4. **Each round key is different** due to the shifting process.

Key scheduling: encryption



25

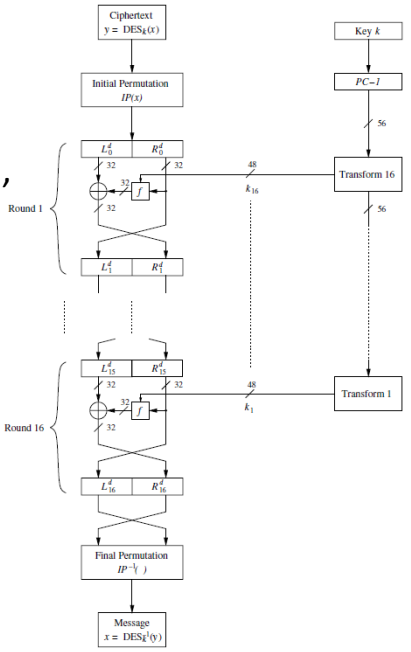
Facts on key schedule

- The key schedule is a method to realize 16 permutations systematically
  - The key schedule is easy to implement in HW
  - The key schedule is such that each of the 56 key bits is used in different rounds
  - Each key bit is used in approximately 14 of the 16 rounds
- Every round key is a selection of 48 permuted bits of the input key
- Total number of rotations: $4 + 12 \times 2 = 28$
  - $C_0 = C_{16}$, $D_0 = D_{16}$ (fundamental for decryption)

26

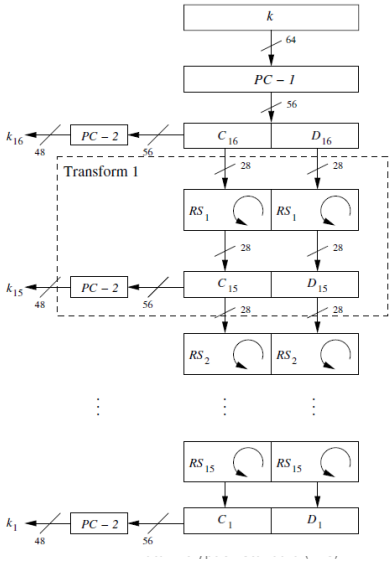
Decryption

- Compared to encryption, only key scheduling is reversed



27

Key scheduling: decryption



28

Decrypting: scheduling keys in reverse order: so in encryption you have $K \to K_1 \ldots K_{16}$ in dec, $K = K_{16} \ldots K_1$

Decryption

- Given k it is easy to reverse the key schedule
  - k_{16} = PC-2(C_{16}, D_{16}) = PC-2(C_0, D_0) = PC-2(PC-1(k))
  - k_{15} = PC-2(C_{15}, D_{15}) = PC-2(RS2(C_{16}), RS2(D_{16})) = PC-2(RS2(C_0), RS2(D_0))
  - …
- Reverse encryption round-by-round
  - Decryption round 1 reverses encryption round 16
  - Decryption round 2 reverses encryption round 15
  - …

29

Decryption

- The input of the 1st decryption round is equal to the output of the last encryption
  - (L^d_0, R^d_0) = IP(Y) = IP(IP^{-1}(R_{16}, L_{16})) = R_{16}, L_{16}
  - Thus $L^d_0 = R_{16}$ and $R^d_0 = L_{16} = R_{15}$
- The first decryption reverses the last encryption
  - L1d = $R_0^d = L_{16} = R_{15}$
  - $R_1^d = L_0^d \oplus f(R_0^d, k_{16}) = R_{16} \oplus f(L_{16}, k_{16}) = [L_{15} \oplus f(R_{15}, k_{16})] \oplus f(R_{15}, k_{16})] = L_{15}$
  - Iteratively
    - $L_i^d = R_{16-i}$
    - $R_i^d = L_{16-i}$
    - where i = 0, 1, 2,… 16

30

Decryption

- After the last decryption round
  - $L_{16}^{d} = R_0$
  - $R_{16}^{d} = L_0$
- Finally,
  - $IP^{-1}(R^{d}_{16}, L^{d}_{16}) = IP^{-1}(L_0, R_0) = IP^{-1}(IP(x)) = x$

31

DES in practice

- DES can be efficiently implemented either in hardware or in software
- Arithmetic operations are
  - exclusive-or
  - E, S-boxes, IP, IP^{-1}, key scheduling can be done in constant time by table-lookup (sw) or by hard-wiring them into a circuit

32

Anyway a lot of bit oriented permutations, shifting etc.

DES in practice [↓]

- One very important DES application is in banking transactions
  - DES is used to encrypt PINs and account transactions carried out at ATM
  - DES is also used in government organizations and for inter-bank transactions

used to be

33

Empirical properties of DES

Empirically, DES fulfills these requirements:

- Each CT bits depends on all key bits and PT bits
- There are no evident statistical relationships between CT and PT
- The change of one bit in the PT (CT) causes the change of every bit in the CT (PT) with 0.5 probability

More or less properties we require for every block cypher

34

DES is not a group

- Experiments gave «overwhelming evidence» that DES was not a group
  - **Practical intuition.** DES provides 2^{56} ($< 10^{17}$) permutations of the $2^{64}!$ possible ones ($> 10^{10^{20}}$) so 2DES would provide a mapping that is not provided by DES with high probability
- In 1992 Campben and Wiener *proved* that DES is not a group (No proof before)
  - K.W. Campbel, M. J. Wiener. DES is not a group. Crypto '92.

35

Security of DES

- Exhaustive key search or brute force attack
- Analytical attacks There are 2 analytical attacks
  - Differential Cryptanalysis, Eli Biham and Adi Shamir, 1990
  - Linear Cryptanalysis, Mitsuru Matsui, 1993
    - Effectiveness of these attacks depend on S-boxes
    - Applicable to any block cipher
  - Not practical for DES
    - Require a large number of (CT, PT)s Very large
    - Collecting and storing (PT, CT)s requires large amount of time and memory
    - Attacks recover just one key ➔ key refresh is an effective countermeasure

found 2 vulnerabilities in algorithm ↑ to attack w/o brute force or Exhaustive key search.

Feasible only if S-boxes and are not well constructed. Can be applied to block ciphers in general.

36

Strength of DES

Here is see complexity of best known attacks

attack method	data complexity(***)		storage complexity	processing complexity
exhaustive precomputation	Dictionary attack	1	2^{56}	1 (table lookup)
exhaustive search	brute force 1 pair (except false +)	—	negligible	2^{55} Average
linear cryptanalysis(*)	2^{43} (85%)*	—	for texts	2^{43}
	2^{38} (10%)	—	for texts	2^{50}
differential cryptanalys(**)	—	2^{47}	for texts	2^{47}
	2^{55}	—	for texts	2^{55}

simple

* 85% to find key

(*) Mitsuru Matsui, 1993
(**) Eli Biham and Adi Shamir, 1990
(***) First column: known-plaintex; second column: chosen-plaintext

37 When NSA scrutinised DES, one S-Box was required to change and key was shorter. When Diff. Cryptoanalysis was discovered it was clear that the modification was to make algorithm to make it stronger. But possibly they reduced size of the key to brute force

DES challenge (1981)

p = "The unknown messages is: XXX …"

c1 c2 c3

- Find $k \in \{0,1\}^{56}$ s.t. $c_i = DES(k, p_i)$, i = 1, 2, 3
  - 1997: Internet search – 3 months
  - 1998: EFF machine (Deep Crack) – 3 days (250K$)
  - 1999: combined search – 22 hours
  - 2006: COPACABANA (120 FPGAs) – 7 days (10K$)
- 56-bit ciphers should not be used

Brute force attack

- In 1977, Diffie & Hellman hypothesized a $ 20 mln dedicated parallel computer able to try 10^6 key per second find a key in 10 hours
- Currently, customary technology allows us to try 10^9 keys per second
- Currently, supercomputer can try 10^{13} keys per second

39

Performance of DES

- Software implementation
  - Desktop ÷ smart cards
  - Bit permutation (E, P, IP) are inefficient in sw
  - S-box moderately efficient in sw
  - Optimization through precomputation
  - Throughput: 100 Megabit/s

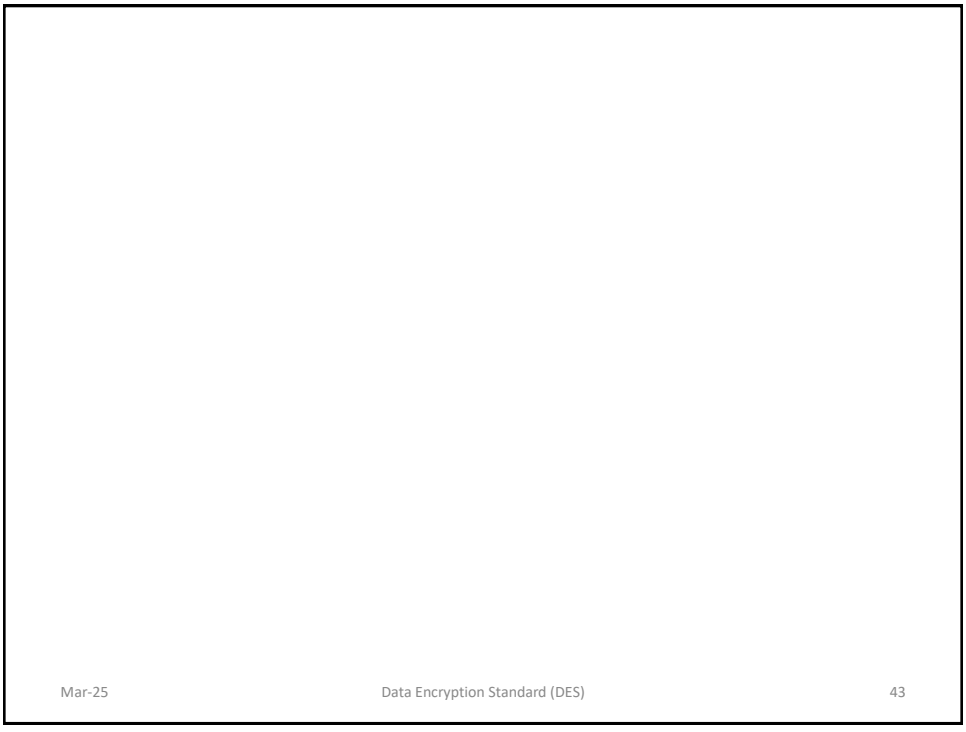
40

Performance of DES

- Hardware implementation
  - Bit permutation are efficient in hw
  - S-box efficiently implemented in Boolean logic (on average a box requires 100 gates)
  - DES requires less than 3000 gates (fit RFIDs)
  - Optimizations: pipelining, FPGA, ASICS
  - Throughput: 100 Gigabit/s

DES alternatives and variants

- 3DES (Triple encryption)
- DESX (Key whitening)
- AES
  - k = 128, 256, 512; n = 128
  - Finalists: Mars, RC6, Serpent, Twofish
    - Efficient especially in SW
    - Mars, Serpent and Twofish are royalty-free
- PRESENT
  - Lightweight encryption, i.e., low complexity, especially in HW
  - Applications RFID tags and pervasive applications



43