Data Encryption Standard

Gianluca Dini
Dept. of Ingegneriadell'Informazione
University of Pisa
email: gianluca.dini@unipi.it
Version: 2021-03-16

By far, DES is the best studied symmetric algorithm, its design principles have inspired many current ciphers.

Data Encryption Standard

- On May 15, 1973, National Bureau of Standards (NBS) published a solicitation for cryptosystems in the Federal Register (mildly revolutionary act)
- On 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
- DES was published in the Federal Register of March 17, 1975

Data Encryption Standard

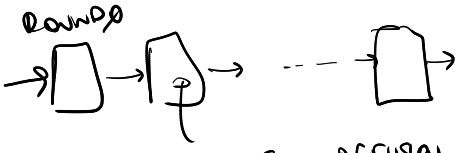
- DES was considered a standard for “unclassified” applications on January 15, 1977 after much public discussion (FIPS PUB 46)
- DES has been reviewed every 5 years
- The most recent review was January 1994
- It is not a standard since 1998. In 1999 DES was replaced by AES

Confusion and diffusion

- Two primitives for strong ciphers (Shannon)
  - **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
  - **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
    - A simple diffusion element is the bit permutation (DES)
    - AES uses the more advanced MixColumn

Confusion and diffusion

- Confusion only or diffusion only is not secure
  - 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

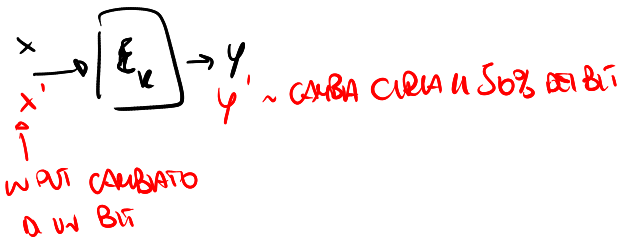


NUMERO DI ROUNDS NON PUÒ ESSERE TROPPO PICCOLO PER NON RENDERNE TROPPO CURANDO

CONFUSION + DIFFUSION, PER ASSICURARE AD OGNI ROUND.

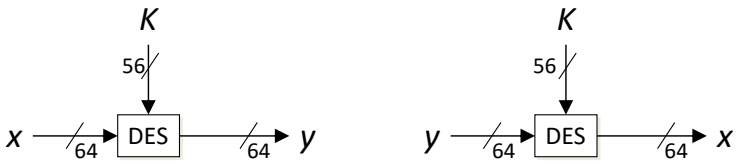
A good diffusion property (an intuition)

- Changing of one bit of PT results on average in the change of half the output bits of the CT, i.e., If PT $\rightarrow$ PT' $\Rightarrow$ CT $\rightarrow$ CT' s.t. CT' looks statistically independent of CT



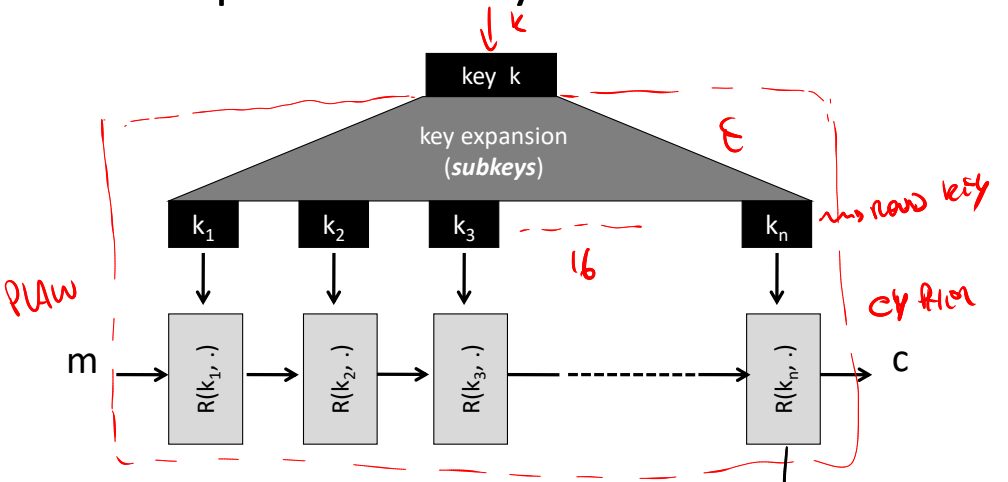
Data Encryption Standard (DES)

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



The 2^{56} keys implement (at most) 2^{56} of the 2^{64}! possible permutations on 64-bit blocks.

Block Ciphers Built by Iteration



R(k,·) is called a **round function**

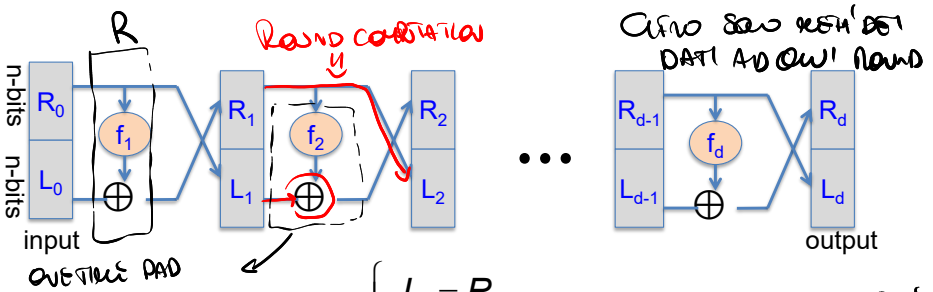
DES (r = 16) , 3DES (r=48), AES-128 (n=10)

To specify the block cipher, you need to specify: the key expansion process and the round function. Many round functions give PRPs after sufficiently many iterations. The challenge is to design a round function that quickly becomes a PRP.

Feistel Network

Given functions $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

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



In symbols:

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

mar-21 DES

The structure in the slide is called a Feistel network. It can lead to very strong ciphers if carefully designed. Feistel networks are used in many, but certainly not in all, modern block ciphers. (In fact, AES is not a Feistel cipher.) In addition to its potential cryptographic strength, one advantage of Feistel networks is that encryption and decryption are almost the same operation. Decryption requires only a reversed key schedule, which is an advantage in software and hardware implementations.

It is crucial to note that the Feistel structure really only encrypts (decrypts) half of the input bits per each round, namely the left half of the input. The right half is copied to the next round unchanged. In particular, the right half is not encrypted with the f function. In order to get a better understanding of the working of Feistel cipher, the following interpretation is helpful: Think of the f function as a pseudorandom generator with the two input parameters R_{i-1} and k_i. The output of the pseudorandom generator is then used to encrypt the left half L_{i-1} with an XOR operation. As we saw, if the output of the f function is not predictable for an attacker, this results in a strong encryption method.

Actually $f_i()$ takes the sub-key k_i as input parameter. It is not shown in the next slides for the sake of simplicity.

The two aforementioned basic properties of ciphers, i.e., confusion and diffusion, are realized within the f -function. In order to thwart advanced analytical attacks, the f -function must be designed extremely carefully. Once the f -function has been designed securely, the security of a Feistel cipher increases with the number of key bits used and the number of rounds.

Round function

- Round *f*-function
  - Realizes ~~substitution~~ diffusion and confusion
  - Can be considered as a pseudorandom generator with two inputs
    - Right half of the input R_{i-1}
    - The round subkey k_i

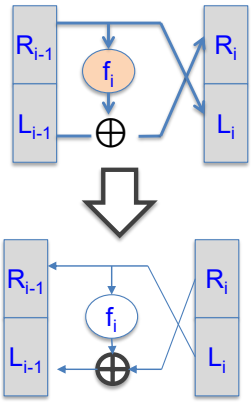
Feistel net is invertible

Theorem: for all $f_1, \ldots, f_d$: $\{0,1\}^n \longrightarrow \{0,1\}^n$

Feistel network F: $\{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ is invertible

Proof: *construct inverse*

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



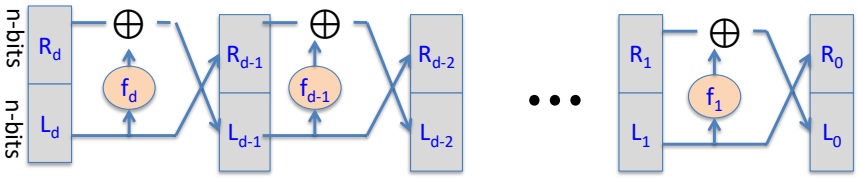
mar-21 DES 11

An advantage of the Feistel network is that encryption and decryption are "almost" the same operation.

The Feistel structure of each round bijectively maps a block of 64 input bits to 64 output bits (i.e., every possible input is mapped uniquely to exactly one output, and vice versa). *This mapping remains bijective for some arbitrary function f , i.e., even if the embedded function f is not bijective itself*. In the case of DES, the function f is in fact a surjective (many-to-one) mapping. It uses nonlinear building blocks and maps 32 input bits to 32 output bits using a 48-bit round key k_i, with $1 \le i \le 16$.

Decryption circuit

DATO CHE f è INVERTIBILE PER DESCRIPTARE BASTA APPLICARLA AL CONTRARIO



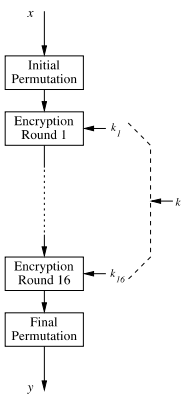
- Inversion is basically the same circuit, with $f_1, \ldots, f_d$ applied in reverse order
- FN is a general method for building invertible functions (block ciphers) from arbitrary functions f.

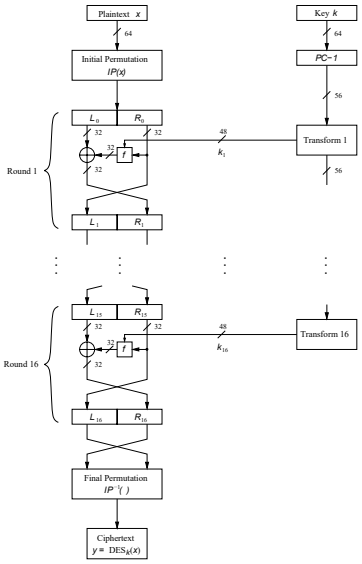
Each round of the Feistel network bijectively maps a block of 64 input bit to 64 output bits.

This mapping is bijective regardless even though f is not. In the case of DES, f is surjective (many to one). Actually it is non linear and maps 32 input bits into 32 output bits using a 48-bit round key.

It is function f which introduces confusion and diffusion.

The internal structure of DES

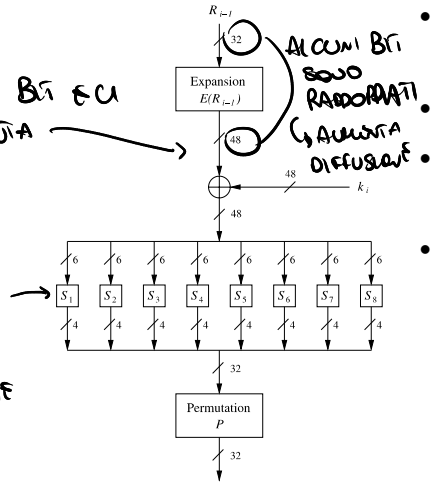




Initial and final permutation

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

The *f*-function



- Expansion box E increases diffusion
- S-boxes provide confusion
- Permutation P increases diffusion
- Avalanche effect
  - Diffusion caused by E, S and P guarantees that every bit at the end of the 5-th round depends on every plaintext bit and key bit

S-box

- Provide confusion
- Lookup table: $\{0, 1\}^6 \rightarrow \{0, 1\}^4$
  - Larger tables would be better but 4-by-6 tables were close to the maximum size for ICs in the 70s
- Core of the DES cryptographic strength
- 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 %

S(

S-box

- The motivations behind S-box were never motivated
- Design criteria

1. Each S-box has six input bits and four output bits.
    1. No single output bit should be too close to a linear combination of the input bits.
    2. 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.
    3. If two inputs to an S-box differ in exactly one bit, their outputs must differ in at least two bits.

S(

S-box

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.

S(

S-box

Reti combinatorie → operazioni logiche

vettori

x → S_i → y

Row

$x = b_1 b_2 b_3 b_4 b_5 b_6$

Column

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

S-box

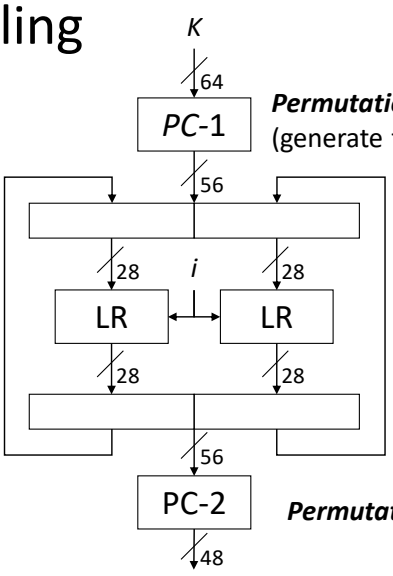
$S_i: \{0,1\}^6 \longrightarrow \{0,1\}^4$

$S_5(011011) \rightarrow 1001$

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

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

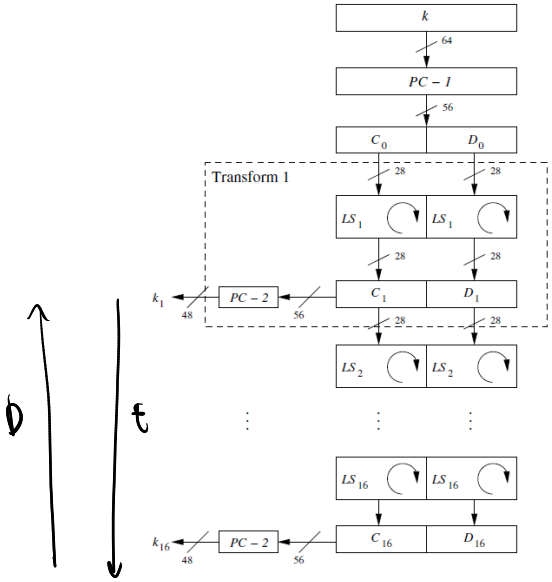


Permutation and compression
(generate the first sub-key)

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

Permutation and compression

Key scheduling: encryption

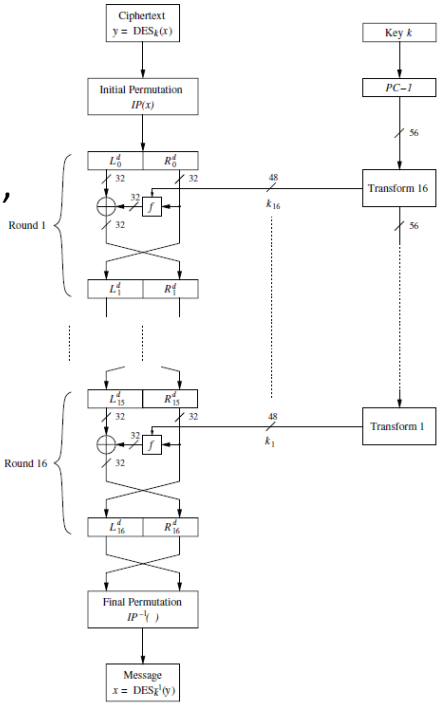


Facts on key schedule

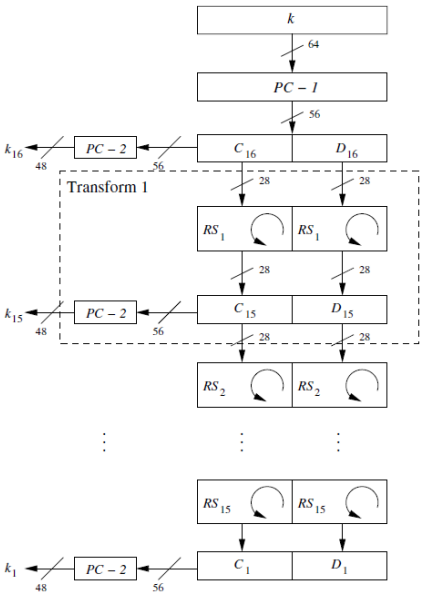
- The key schedule is a method to realize 16 permutations systematically
  - The key schedule is easy to implement in HW
  - The key schedule is such that each of the 56 key bits is used in different rounds
  - Each key bit is used in approximately 14 of the 16 rounds
- Every round key is a selection 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)

Decryption

- Compared to encryption, only key scheduling is reversed



Key scheduling: decryption



Decryption

Skip

- 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 rounf 2 reverses encryption round 15
  - …

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

Decryption

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

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

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

Empirical properties of DES

→ NON CI SONO PROVE CHE SIA SICURO MA HA RETTO FUORA, NON CI SONO ALGORITMI EFFICIENTI.

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

Security of DES

- Exhaustive key search or brute force attack
- 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 PTs
    - Collecting and storing (PT, CT)s requires large amount of time and memory
    - Attacks recover just one key (key refresh)

Strength of DES

attack method	data complexity(***)		storage complexity	processing complexity
exhaustive precomputation	—	1	2^{56}	1 (table lookup)
exhaustive search	1	—	negligible	2^{55}
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}

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

THE STRENGTH OF DES

- The complexity of the best attacks currently known against DES is given in Table 7.7.
- Percentages indicate success rate for specified attack parameters.
- The *processing complexity* column provides only an estimate of the expected cost (operation costs differ across the various attacks); for exhaustive search, the cost is in DES operations.
- Regarding storage complexity, both linear and differential cryptanalysis require only negligible storage in the sense that known or chosen texts can be processed individually and discarded, but in a practical attack, storage for accumulated texts would be required if ciphertext was acquired prior to commencing the attack.

PRACTICALITY OF ATTACK MODELS

- To be meaningful, attack comparisons based on different models must appropriately weigh the feasibility of extracting (acquiring) enormous amounts of chosen (known) plaintexts, which is considerably more difficult to arrange than a comparable number of computing cycles on an adversary's own ma-chine.
- Exhaustive search with one known plaintext-ciphertext pair and 2^{55} DES operations is significantly more feasible in practice (e.g., using highly parallelized custom hardware) than linear cryptanalysis (LC) requiring 2^{43} known pairs.
- While exhaustive search, linear, and differential cryptanalysis allow recovery of a DES key and, therefore, the entire plaintext, the dictionary attacks, which become feasible once about 2^{32} ciphertexts are available, may be more efficient if the goal is to recover only part of the text.

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
- 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

non molto efficiente!

DES alternatives and variants

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