Block Ciphers

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
gianluca.dini@unipi.it
Version: 2021-03-15

Block Ciphers

GENERAL CONCEPTS

USATI IN CLIENT-SERVER APPLICATION => BROWSER

POSSIBILI PROBLEMI: 1) COSA SUCCEDE SE ... ARRIVANO ... PIU' BIT DI ...

I DETTAGLI IMPLEMENTATIVI DI E() E D() SONO NOTI

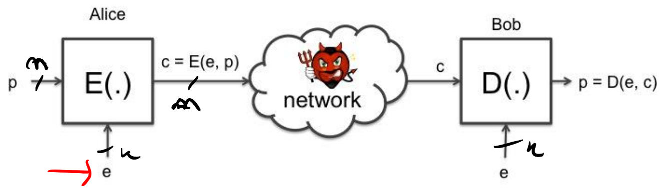
Block cipher

E e D SONO PERMUTAZIONI

ES. AES -> 128

- Block ciphers break up the plaintext in blocks of fixed length *n* bits and encrypt one block at time

CRIPTO UN BLOCCO DI BIT ALLA VOLTA



CHIAVI

CAMBIA DA ALGORITMO AD ALGORITMO, NON PUO' CAMBIARE DURANTE L'ES...

- E_k: $\{0,1\}^n \rightarrow \{0,1\}^n$ D_k: $\{0,1\}^n \rightarrow \{0,1\}^n$
- E is a keyed permutation: $E(k, m) = E_k(m)$
- $E_K(\cdot)$ is a permutation

Permutation

- E_k is a permutation
  - E_K is efficiently computable → TEMPO POLINOMIALE
  - Ek is bijective
    - Surjective (or onto)
    - Injective (or one-to-one)
  - E_k^{-1} is efficiently computable

Examples

UNIVERSITÀ DI PISA

- Block ciphers
  - DES n = 64 bits, k = 56 bits
  - 3DES n = 64 bits, k = 168 bits
  - AES n = 128 bits k = 128, 192, 256 bits

NON POSSO ESSERE LEVATI

- Performance (AMD Opteron, 2.2 GHz)
  - RC4 126 MB/s
  - Salsa20/12 643 MB/s
  - Sosemanuk 727 MB/s
  - 3DES 13 MB/s
  - AES-128 109 MB/s → STANDARD

Random permutations

TUTTI I POSSIBILI OUTPUT

||

TUTTI I POSSIBILI INPUT

2^N INPUT

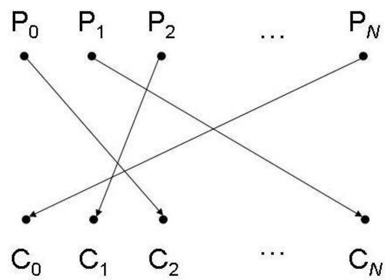
00 --- 00

00 --- 01

11 --- 11

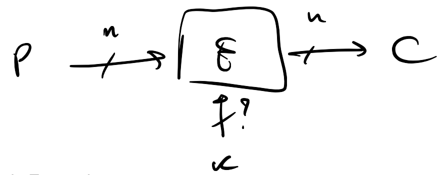
TUTTE LE POSSIBILI PERMUTAZIONI DI N BIT

$N = 2^n - 1$



A random permutation π

- Let Perm_n be the set of all permutations $\pi: \{0,1\}^n \rightarrow \{0,1\}^n$
- $|\text{Perm}_n| = 2^n!$ = NUMERO DI POSSIBILI PERMUTAZIONI CON UN BLOCCO DI n BIT
- A true random cipher
  - implements all the permutations in Perm_n
  - uniformly selects a permutation $\pi \in \text{Perm}_n$ at random



True Random Cipher

- A True random cipher is perfect → SE CONDO LA DEFINIZIONE DI SHAN
- A true random cipher implements all possible Random permutations ($2^n!$)
  - Need a uniform random key for each permutation (naming) → RICONOSCERE UNA PERMUTAZIONE DALL'ALTRA
    - key size := $\log_2(2n!) \approx (n - 1.44)\, 2^n$ = NUMERO DI BIT CHE SERVONO PER RAPPRE LE CERANI_
      - Exponential in the block size!
      - The block size cannot be small in order to avoid a dictionary attack
- A true random cipher cannot be implemented

AL CRESCERE DEL BLOCCO LA CHIAVE CRESCE ESPONENZIALMENTE! NON POSSO USARE CHIAVI ALURE.

SERVE UNA CHIAVE PER PERMUTAZIONE

$2^n!$

$k = n \cdot 2^n$, n: BLOCK SIZE, k = KEY SIZE

DES $n = 64$ BITS $\Rightarrow k = 64 \cdot 2^{64}$

AES $n = 128$ BITS $\Rightarrow k = 128 \cdot 2^{128}$

NUMERI ENORMI NON REALISTICO

PERCHÉ NON USARE $n = 4$? $k = 4 \times 2^4 = 4 \times 16 = 64$ BITS FATTIBILE!

MA TOTALMENTE INSICURA

↓

DICTIONARY ATTACK

KNOWN - PLAINTEXT ATTACK

L'ATTACCANTE HA A DISPOSIZIONE COPPIE DI TESTO CIFRATO E NON CIFRATO

CT	PT
C_1	P_1
C_2	P_2
$\vdots$	$\vdots$
C_n	P_n

2^n ; $2n$

VALIDO PER UNA CERTA K_

DIMENSIONE DEL DIZIONARIO MEMORIA:

$$2^n \times (2 \cdot n) = n\, 2^{n+1}$$

MODO PIU' EFFICIENTE

PT

0, 1, ⋮, $2^n - 1$

P_i

MEMORIA = $2^n \times$

$O(2^n)$

C

COME POSSO PREVENIRE QUESTO TIPO DI ATTACCO?

DEVO RENDERE IL DIZIONARIO NON MEMORIZZABILE IN MEMORIA_

$n = 32 \Rightarrow 32 \times 2^{32} = \underbrace{2^5 \times 2^{32}}$ bits

$\frac{2^5 \times 2^{32}}{2^3} = \underline{16\ \text{GByte}}$

TROPPO POCO

MOLTO FATTIBILE

$$\boxed{\text{BLOCK SIZE} \geq 64 \text{ BITS}}$$

ALMENO!

A

$C = E_k(P)$ → $P =$

Pseudorandom permutations

UNIVERSITÀ DI PISA

- Consider a *family of permutations* parametrized by $k \in K = \{0, 1\}^k$, $E_k: \{0,1\}^n \rightarrow \{0,1\}^n$
- A E_k is a pseudorandom permutation (PRP) if it is indistinguishable from a uniform random permutation by a limited adversary
- $|\{E_k\}| = 2^k << |Perm_n|$, with $|k| = k$

  $2^n!$
- A block cipher is a practical instantiation of a PRP

Practical block cipher

- In practice, the encryption function corresponding to a randomly chosen key should appear as a randomly chosen permutation to a limited adversary

x_i

$\pi(x_i)$ or $E(k, x_i)$?

SICURO SE NON RIESCE A DISTINGUERLI

$\pi \leftarrow \text{Perms}(\{0,1\}^n)$

UNA DELLE POSSIBILI PERM

$k \leftarrow K$, E_k

CASO UNA PERMUTAZIONE PSEUDO-RANDOM

- Oracle access
  - adversary cannot look into the box

UNICA COSA CHE PUO' FARE L'AVVERSARIO E' INVIARE UN INPUT E OSSERVARE IL RISULTATO

Let's neglect how our block ciphers (DES, 3DES, AES,…) are implemented inside. Assume they are secure according to the above notion.

Exhaustive key search

- The attack
  - Given a pair (pt, ct), check whether ct == E_{ki}(pt), i = 0, 1, …, $2^k - 1$
    - Known-plaintext attack
    - Time complexity: $O(2^k)$
- False positives
  - Do you expect that just one key k maps pt into ct?
  - How many keys (false positives) do we expect to map pt into ct?
  - How do you discriminate the good one?

Exhaustive key search

- False positives
  - Do you expect that just one key k maps pt into ct?
  - How many keys (false positives) do we expect to map pt into ct?
  - How do you discriminate the good one?

uniform random variable

$$pt \xrightarrow[n]{} \boxed{E_k} \xrightarrow[n]{} ct$$

Brute force attack (pt, ct) nota

$\forall k,\ ct == E_k(pt)?$

Data k, quale è la Pr$[E_k(pt) == ct] =? = \frac{1}{2^n}$
probabilità

Expected number of keys che cifrano pt in ct?

$$\boxed{2^k \times \frac{1}{2^n}} = 2^{k-n}$$

DES $n = 64$ bit, $k = 56$ bit

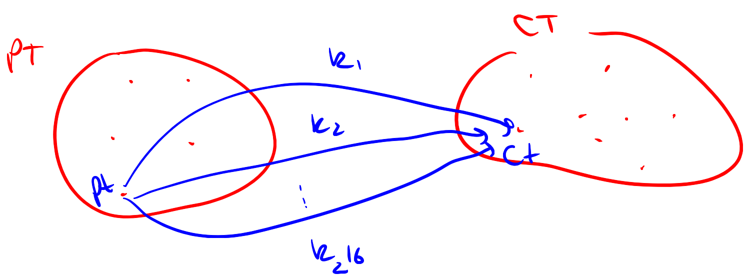
keys che cifrano pt in ct = … $= 2^{-8} = \frac{1}{256}$

Perché … 1

È sufficiente una coppia (pt, ct) …
fare un known-plaintext attac…

SKIPJACK $n = 64$ bit, $k = 80$ bit

$\#\text{keys} = 2^{80-64} = 2^{16}$



Per fare un attacco brute-force ho bisogno di 2 o più coppie (p…

False positives

- Problem: Given (ct, pt) s.t. ct = E_{k*}(pt) for a given k*, determine the number of keys that map pt into ct
- Solution.
  - Given a certain key k, P(k) = Pr[E_{k*}(pt) == ct] = $1/2^n$
  - The *expected* number of keys that map pt into ct is $2^k \times 1/2^n = 2^{k-n}$

False positives

- Example 1 – DES with n = 64 and k = 56
  - On average 2^{-8} keys map pt into ct
  - One pair (pt, ct) is sufficient for an exhaustive key search
- Example 2 – Skipjack with n = 64 and k = 80
  - On average 2^{16} keys map pt into ct
  - Two or more plaintext-ciphertext pairs are necessary for an exhaustive key search

False positives

- Consider now t pairs (pt_i, ct_i), i = 1, 2,…, t
  - Given k*, $Pr[E_{k^*}(pt_i) = ct_i$, for all i = 1, 2,…, t] = $1/2^{tn}$
  - Expected number of keys that map pt_i into ct_i, for all i = 1, 2, …, t, is $2^k/2^{tn} = 2^{k-tn}$
- Example 3 – Skypjack with k = 80, n = 64, t = 2
  - The expected number of keys is = $2^{80 - 2\times 64} = 2^{-48}$
  - Two pairs are sufficient for an exhaustive key search

False positives

- THEOREM
  – Given a block cipher with a key lenght of *k* bits and a block size of *n* bits, as well as *t* plaintext-ciphertext pairs, $(pt_1, ct_1), \ldots, (pt_t, ct_t)$, the expected number of false keys which encrypt all plaintexts to the corresponding ciphertexts is 2^{k-tn}
- FACT
  – Two input-output pairs are generally enough for exhaustive key search

See exercise.

For example, let us consider DES, Skipjack and AES for t = 2

- **DES**: n = 64, k = 56, 2^(56 - 128) = 2^(-72)
- **Skipjack**: n = 64, k = 80, 2^(80 – 128) = 2^(-48)
- **AES**: n = 128, k = 128 = 2^(128 - 256) = 2^(-128)

Block ciphers

EXERCISES

Exercise 1 - Exhaustive key search

UNIVERSITÀ DI PISA

- Exhaustive key search is a known-plaintext attack
- However, the adversary can mount a cyphertext-only attack if (s)he has some knowledge on PT

DEVO AVERE UN CERTO NUMERO DI COPPIE (2, 3, 4...)

(pt_1, ct_1)
(pt_2, ct_2)
$\vdots$

METTIAMO CHE L'AVVERSARIO CONOSCE IL TESTO CIFRATO E HA ALCUNE INFORMAZIONI SUL TESTO IN CHIARO.

↓

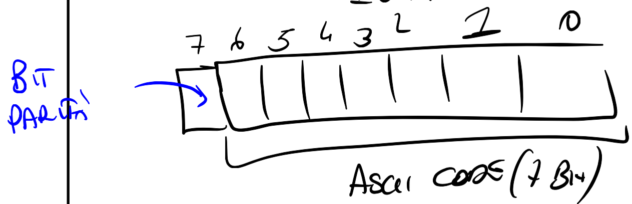
CASO MOLTO COMUNE

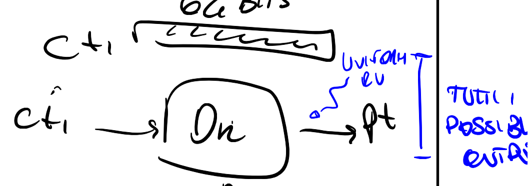
Exercise 1 – exhaustive key search

- Assume DES is used to encrypt 64-bit blocks of 8 ASCII chars, with one bit per char serving as parity bit
- How many CT blocks the adversary needs to remove false positives with a probability smaller than ε?

CONOSCENZA DELL'AVVERSARIO

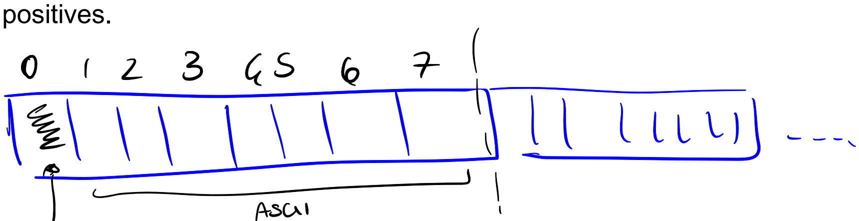
DES m=64 BITS => 8 BYTES = 8 CH





$\forall i \in 0 \ldots 2^k, k_i$
PROVA TUTTE LE CHIAVI

Let us consider a ciphertext block. Trial decryption of one ciphertext block with a given key k yields 8 correct parity bits with a probability $p(1) = 2^{-8}$. Actually, every bit is correct with a probability of 1/2. In the case of t blocks, the probability p(t) of 8t parity bits being correct is $p(t) = 2^{-8t}$. Thus, we can determine t by means of the following condition $2^{-8t} < \varepsilon$. Consequently, the expected number of keys $r = 2^{k-8t}$.
In the DES case, $r = 2^{56-8t}$. For most practical purposes, t = 10 suffices to avoid false positives.



BIT DI PARITA', PROBABILITA' CHE SIA CORRETTO

DATO CHE HO Ct
Ct
⋮
C

$PR(0) = \frac{1}{2}$, CONGIUNTA: $\frac{1}{2} \times \frac{1}{2} \times \cdots \times \frac{1}{2} = \frac{1}{2^8}$

PROBABILITA' $= \frac{1}{2^{8t}}$ PROBABILITA' DI INDOVINARE IL BIT DI PARITA' PER t TESTI CIFRATI –

$$\#KEYS = 2^k \times \frac{1}{2^{8t}} = 2^{k-8t}$$

DES => 2^{56-8t} → $t \geq 10$ => RIDUCO NOTEVOLMENTE IL NUMERO DI FALSI POSITIVI

Exercise 2 - dictionary attack

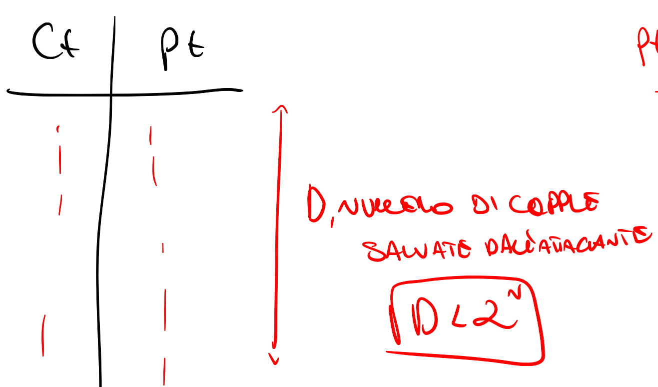
UNIVERSITÀ DI PISA

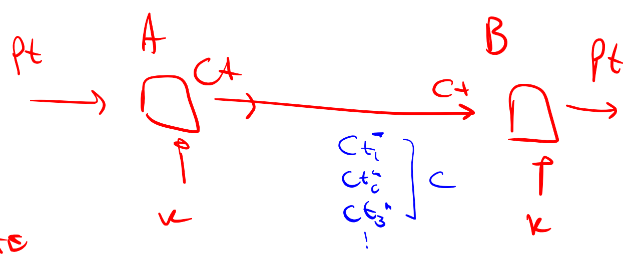
- Consider E with k and n.
- The adversary has collected D pairs (pt_i, ct_i), i = 1,…, D, with D << 2^n
- Now the adversary reads C newly produced cyphertexts ct^*_j, j = 1,…, C.
- Determine the value of C s.t. the Pr[Exists j, j = 1, 2,… C, s.t. c^*_j is in the dictionary] = P

Let Q = 1 − P , where Q is the probability that none of the C newly produced ciphertexts are in the dictionary. Initially, we compute q = Pr[probability q that a ciphertext is *not* in the dictionary]. Probability q = (#favourable cases) / (#possible cases) = $(2^n - D) / 2n = 1 - \alpha$ with $\alpha = D / 2^n$. It follows that α << 1. Then, we compute Q = Pr[none of the C ciphertexts is in the dictionary] = Pr[(ct*1 is not in the dictionary), (ct*2 is not in the dictionary, ...)] = Pr[(ct*1 is not in the dictionary)] × Pr[(ct*2 is not in the dictionary)] × … = $q^C = (1 - \alpha)^C \approx 1 - C \times \alpha$. It follows that $P = C \times \alpha$.

If we specify P = 1, then $C \times \alpha = 1$, which implies that $C \times D/2^n = 1$ and thus $C = 2^n/D$. If $D = 2^{n/2}$ then $C = 2^{n/2}$. Let n = 64, then D = 232 entries, each of $2 \times 8 = 16$ bytes. It follows that the dictionary size in bytes is $2^4 \times 2^{32}$ = 16 Gbytes.

Of course the challenge for the adversary is to collect D pairs. For this reason, it is wise to change the encryption key periodically.





DATO P, TROVARE C;

$Q = 1 - P$

Q = Pr[NESSUN NUOVO Ct PRODOTTO NEL DIZIONARIO

K FISSATA ALL'INTERNO DEL DIZIONARIO

$q = [\text{UN NUOVO Ct NON E' NEL DIZIONARIO}] = \frac{2^N - D}{2^N} = 1 - \frac{D}{2^n} = 1 - \alpha$ dove $\alpha = \frac{D}{2^n} \ll 1$

$Q = \Pr[C Ct_1 \text{ non nel dizionario}), C Ct_2 \text{ non nel dizionario} \ldots] = \prod_{i=1}^{c} \Pr[Ct_i \text{ non nel dizionario}]$

$= \prod_{i=1}^{c} q = q^c = (1-\alpha)^c \simeq 1 - c\alpha + \cancel{\alpha^2 p_2} + \cancel{p_3 \alpha^3} + \ldots + \cancel{p_c \alpha^c}$

$\alpha << 1$, piccolissimo

$Q \simeq 1 - c\alpha \Rightarrow P = 1 - Q = c\alpha \Rightarrow P = c\alpha \Rightarrow P = 1 \longrightarrow c = \frac{1}{\alpha} = \frac{2^N}{D}$

se $D = 2^{m/2} \Rightarrow c = 2^{m/2}$, $m = 64$

$D = 2^{32}$

$C = 2^{32}$

Exercise 3 - Rekeying

Probabilità che almeno uno sia nel dizionario

- An adversary can successfully perform an exhaustive key search in a month.
- Our security policy requires that keys are changed every hour.
- What is the probability P that, in a month, the adversary is able to find any key before it is changed?
  - For simplicity assume that every month is composed of 30 days.
- What if we refresh key every minute? ma non è certo realistico

Let us assume that a month is composed of 30 days and thus of H = 720 hours.

As a key is found in one month by means of a brute force attack, we can reasonably assume that p = Pr[a given key is guessed in a given hour before it is changed] = 1/H = 1.4 × 10−3.

Let q = Pr[no key is guessed in a given hour before it is changed] = 1 – p = 1 – 1/H.

Let P = Pr[probability that, in a month, the adversary guesses at least one key before it is changed]. Furthermore, let Q = 1 – P be the probability of the complementary event, i.e., Pr[the adversary cannot find any key before it is changed in a month] = Pr[(no key is guessed in hour 1), (no key is guessed in hour 2),...] = Pr[no key is found in hour 1] × Pr[no key is found in hour 2] × … × Pr[no key is found in hour H] = $q^H = (1 – p)^H = (1 – 1/H)^H$. Consequently, P = 1 − Q = 1 − $(1 − 1/H)^H$. With H =720, Q ≈ 0.37 and P ≈ 0.63.

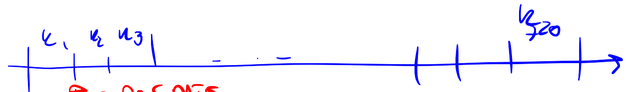
Assume we increase the refresh frequency (e.g., every minute). This means that the value of H increases and becomes H = 720 × 60. When H becomes very large, H → ∞, then Q → 1/e ≈ 0.37, and thus P →(1 − 1/e) ≈ 0.63. This means that increasing the frequency with which the key is changed does not meaningfully improve the value of P.

The only practical advantage of frequently changing the key mainly reside in the fact that a smaller amount of material is encrypted with that key and, therefore, for each key less material is available to a cryptanalyst.

L'avversario può fare un attacco brute-force in un mese.

H = 1 month = 30×24 hours = 720 hr

Quale è la probabilità p che l'avversario indovini la chiave prima che scada?



$p = \frac{1}{720} = \frac{1}{H}$, $q = \Pr[\text{una chiave non è indovinata prima che scada}] = 1 - p = \left(1 - \frac{1}{H}\right)$

$P = \Pr[\text{avversario riesca ad indovinare almeno una chiave prima che scada}]$, $Q = 1 - P = \Pr[\text{l'avversario non indovina nessuna chiave prima che scada}]$

$$= \Pr[(\text{AVVERSARIO NON INDOVINA } k_1), (\text{AVVERSARIO NON INDOVINA } k_2) \ldots ( )] = \prod_{i=1}^{H} \Pr\{\text{AVVERSARIO NON INDOVINA } k_i \text{ PRIMA CHE SCADA}\}$$

$$= \prod_{i=1}^{H} q = q^H = (1-p)^H = \left(1 - \frac{1}{H}\right)^H = Q, \quad H = 720$$

$$P = 1 - Q = 1 - \left(1 - \frac{1}{H}\right)^H = 1\left(1 - \frac{1}{720}\right)^{720}$$

$Q \simeq 0.37$

$\boxed{P \simeq 0.63}$

COSA SUCCEDE SE INVECE DI CAMBIARE LA CHIAVE OGNI 720 LA CAMBIO OGNI n = # minuti in un mese?

$$M = 720 \cdot 60, \quad Q = \left(1 - \frac{1}{M}\right)^M \text{ DOVE } M \text{ AUMENTA!}$$

$$\lim_{M \to \infty} Q = \frac{1}{e} = 0.37, \quad \lim_{M \to \infty} P = 1 - \frac{1}{e} = 0.63$$

NON DECRESCE

Symmetric Encryption

MULTIPLE ENCRYPTION AND KEY WHITENING

DES challenge (1981)

p = "The unknown messages is: XXX …"

c1 c2 c3

- Find $e \in \{0,1\}^{56}$ s.t. $c_i = \mathrm{DES}(e, p_i)$, 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

The **DES Challenges** were a series of brute force attack contests created by RSA Security to highlight the lack of security provided by the Data Encryption Standard.
Three plaintexts were provided. RSA would pay 10.000 dollars to whom who was able to find the key and thus break the system.

The plaintext is provided in order to discard false positives.

COPACABANA (Cost-Optimized Parallel Code-Breaker) – University of Bochum and University of Kiel, Germany

Increasing the Security of Block Ciphers

UNIVERSITÀ DI PISA

- DES is a secure cipher
  - No efficient cryptanalys is known → UNICO ATTACCO POSSIBILE È UNO BRUTEFORCE!
- DES key has become too short
- Can we improve the security of DES? DEVO RENDERLO NON FATTIBILE
- Yes, by means of two techniques
  - Multiple encryption
  - Key whitening

DES non sicuro perché la chiave è troppo corta

È importante che non ci siano attacchi analitici efficienti

Two-times Encryption (2E)

Right key

- $y = 2E((e_L, e_R), m) = E(e_R, E(e_L, x))$ cosa succede se invece di criptare una volta, cripto 2?
  - key size is 2k bits (left key)
  - Brute force attack requires 2^{2k} steps
  - 2E is two times slower than E → introduco sicuramente overhead
- Is it really secure?
- Meet-in-the-middle attack

$$x \longrightarrow \boxed{E(e_L, \cdot)} \longrightarrow \boxed{E(e_R, \cdot)} \longrightarrow y$$

As to computing the number of false postives, you may consider L-subsequent encryptions as a cipher having a key large Lk-bits.

KNOW - PLAINTEXT ATTACK

Meet-in-the-middle attack

UNIVERSITÀ DI PISA

- Attack Sketch
  1. Build a table T containing $z = E(e_L, x)$ for all possible keys e_L. Keep T sorted according to z.
  2. Check whether $z' = D(e_R, y)$ is contained in the table T, for all possible key e_R.
     1. If z' in contained in T then (e_L, e_R) maps x into y with e_L s.t. $T[e_L] = z'$.

POTREBBE ESSERE UN FALSO POSITIVO ERA CANDIDATA.

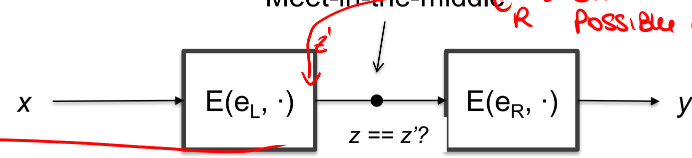
T

ORDINATA PER Z

e_L' CHIAVE CANDIDATA!

z' E' NELLA TABELLA?

SI



e_R' = UNA DELLE POSSIBILI CHIAVI DX

NO, PROVO CON ALTRO e_R'

ATTACCO EFFICIENTE?

A naıve brute-force attack would require us to search through all possible combinations of both keys, i.e., the effective key lengths would be 2^{2k} and an exhaustive key search would require $2^k \times 2^k = 2^{2k}$ encryptions (or decryptions). However, using the meet-in-the-middle attack, the key space is drastically reduced.

The Meet-in-the-middle Attack.

1. Build a table T containing z = E(eL, x) for all possible key left-keys eL and keep T sorted according to eL.
2. For all possible eR, check whether z' = D(eR, y) is contained in the table T. Assume that for a given eR*, exists eL* s.t. T[eL*] = z', the pair (eL*, eR*) maps p into c and thus it is a candidate key.
3. In order to get rid of false positives you may need two or more pairs (x, y).

Meet-in-the-middle attack

UNIVERSITÀ DI PISA

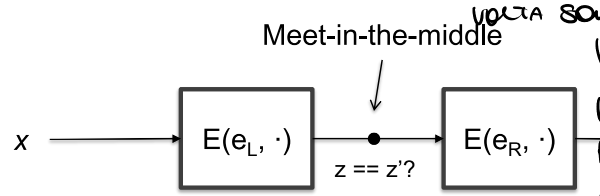
DATA COMPLEXITY → NON RILEVANTE

- Attack complexity
  - Storage complexity → UNA ENTRATA PER OGNI CHIAVE
    - Storage necessary for table T ≈ $O(2^k)$
  - Time complexity
    - Time complexity for step 1 + Time complexity for step 2 = Time for building and sorting the table + Time for searching in a sorted table = $k\,2^k + k\,2^k \approx O(2^k)$ NON PIÙ SICURO DA CIFRARE UNA VOLTA SOLA

TENGO LA TABELLA ORDINATA PER Z COSÌ ALLA RICERCA DI Z' POSSO USARE UN ALGORITMO EFFICIENTE

RICHIEDE SOLO PIÙ STORAGE

→ NON HO VANTAGGI DAL PUNTO DI VISTA DEL TEMPO.



Attack complexity

Data complexity (negligible): a few pairs.

Storage complexity := he storage necessary for T: $O(2^{56})$. Remarkable!

Time complexity := Time complexity for step 1 + Time complexity for step 2 = (Time for building the table + Time to sort) + (Time to perform all the searches in a sorted table) = $(56 \times 2^{56}) + (56 \times 2^{56}) = O(2^{56})$

Two-times DES

UNIVERSITÀ DI PISA

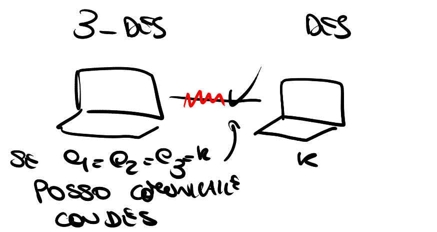
- 2DES → CHIAVE DES
  - Time complexity: 2^{56} (doable nowadays!)
  - Space complexity: 2^{56} (lot of space!)
  - 2DES brings no advantage

As to computing the number of false positives, you may consider L-subsequent encryptions as a cipher having a key large Lk-bits.

Triple DES (3DES) ANCORA USATO.

EEE → VARIANTE, NON HO COSTRUTTO DI DECRIPTAZIONE

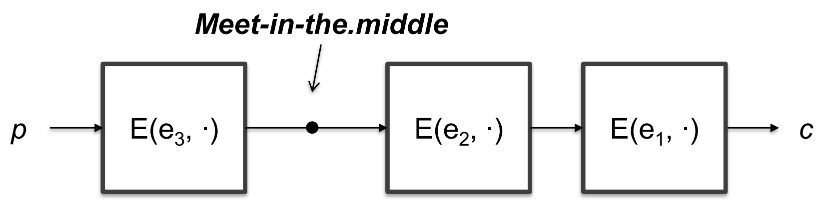
- EDE scheme
  - Standard ANSI X9.17 and ISO 8732
  - $Y = 3E((e_1, e_2, e_3), x) = E(e_1, D(e_2, E(e_3, x)))$ PIÙ USATO EDE PER QUESTO MOTIVO
    - If $e_1 = e_2 = e_3$, 3DES becomes DES
      - backward compatibility
  - Key size = 168-bits
  - 3 times slower than DES
  - Simple attack ≈ 2^{118}



Nowadays anything that is larger than 2^{90} is considered secure.

3DES – meet-in-the-middle attack

- Time = 2^{112} (undoable!)
- Space = 2^{56} (lot of space!)



$2^k \times 2^k = 2^{2k}$ TIME COMPLEXITY

The value for time complexity neglects the time for keeping the table sorted in the first phase and for searching in the table in the second phase.

False positives for multiple encryption

- THEOREM
  - Given there are r subsequent encyptions with a block cipher with a key lenght of k bits and a block size of n bits, as well as t plaintext-ciphertext pairs, $(pt_1, ct_2),\ldots, (pt_t, ct_t)$, the expected number of false keys which encrypt all plaintext to the corresponsig ciphertext is $2^{rk - tn}$

Limitations of 3DES

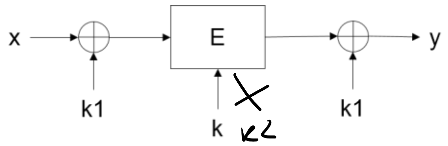
- 3DES resists brute force but COSTRUITO PER BASARSI SU HARDWARE
  - It is not efficient regarding software implementation
  - It has a short block size (64 bit)
    - A drawback if you want to make a hash function from 3DES, for example
  - Key lengths of 256+ are necessary to resist quantum computing attack

BUONO PER CIFRARI, MENO PER IMPLEMENTARE HASH FUNCTIONS

Key whitening

DEVE COMUNQUE ESSERE SICURO AUTONOMAMENTE <->

- Considerations
  - KW is not a “cure” for weak ciphers
- Applications
  - DESX: a variant of DES
  - AES: uses KW internally
- Performance
  - Negliglible overhead w~~.r.t. E (Just two~~ XOR’s!)



Definition 5.3.1 Key whitening for block ciphers
Encryption: $y = e_{k,k_1,k_2}(x) = e_k(x \oplus k_1) \oplus k_2$.
Decryption: $x = e^{-1}_{k,k_1,k_2}(x) = e^{-1}_k(y \oplus k_2) \oplus k_1$

To some extent, Multiple Encryption makes the resulting cipher more resistant to linear and differential cryptoanalysis. In contrast, Key Whitening does not. So KW is not a “cure” for weak ciphers.

Key whitening

- Attacks
  - Brute-force attack
    - Time complexity: 2^{k+2n} encryption ops
  - Meet-in-the-middle:
    - Time complexity 2^{k+n}
    - Storage complexity: 2^n data sets
  - The most efficient attack
    - If the adversary can collect 2^m pt-ct pairs, then time complexity becomes Ridotta 2^{k+n-m}
      - The adversary cannot control m (rekeying)
    - Example: DES (m = 32)
      - Time complexity 2^{88} encryptions (nowadays, out of reach)
      - Storage complexity 2^{32} pairs = 64 GBytes of data (!!!)

Symmetric Encryption

ENCRYPTION MODES

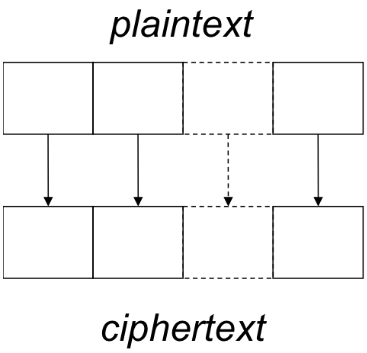
Other encryption modes

- Other encryption modes
  - To build a stream cipher out of a block cipher
    - Cipher Feedback mode (CFB)
    - Output Feedback mode (OFB)
    - Counter mode (CTR)
  - Authenticated encryption
    - Galois Counter mode (GCM, CCM, …)
  - and many others (e.g., CTS, …)
- Block ciphers are very versatile components

Encryption Modes

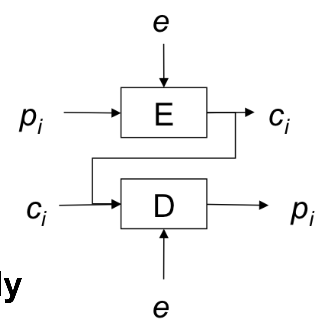
- A block cipher encrypts PT in fixed-size *n*-bit blocks
- When the PT len exceeds n bits, there are several modes to the block cipher
  - Electronic Codebook (ECB)
  - Cipher-block Chaining (CBC)
  - Cipher-feedback (CFB)
  - Output feedback (OFB)

Electronic codebook



$$\forall 1 \le i \le t, c_i \leftarrow E(e, p_i)$$

$$\forall 1 \le i \le t, p_i \leftarrow D(e, c_i)$$



PT blocks are encrypted separately

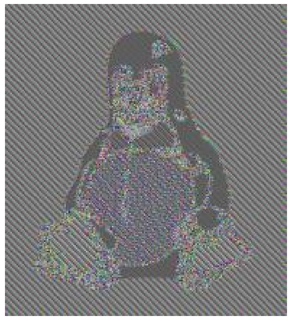
ECB - properties

- **PROS**
  - No block synchronization is required
  - No error propagation
    - *One or more bits in a single CT block affects decipherment of that block only*
  - Can be parallelized
- **CONS**
  - Identical PT results in identical CT
    - *ECB doesn't hide data pattern*
    - *ECB allows traffic analysis*
  - Blocks are encrypted separately
    - *ECB allows block re-ordering and substitution*

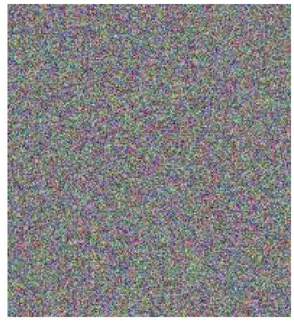
ECB doesn't hide data patterns

UNIVERSITÀ DI PISA

Plaintext



ECB encrypted



Non-ECB encrypted

ECB mode is used to encrypt a bitmap image with large areas of uniform colour. While the colour of each individual pixel is encrypted, the overall image may still be discerned as the pattern of identically coloured pixels in the original remains in the encrypted version.

ECB – block attack

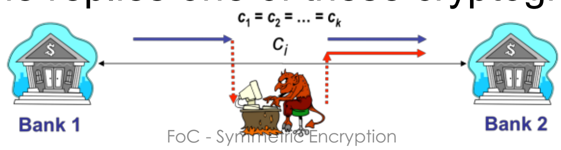
- Bank transaction that transfers a client U’s amount of money D from bank B1 to bank B2
  - Bank B1 debits D to U
  - Bank B1 sends the “credit D to U” message to bank B2
  - Upon receiving the message, Bank B2 credits D to U
- Credit message format
  - Src bank: M (12 byte)
  - Rcv banck: R (12 byte)
  - Client: C (48 byte)
  - Bank account: N (16 byte)
  - Amount of money: D (8 byte)
- Cipher: n = 64 bit; ECB mode

ECB – block attack

- Mr. Lou Cipher is a client of the banks and wants to make a fraud
- Attack aim
  – To replay Bank B1’s message "credit 100$ to Lou Cipher" many times
- Attack strategy
  – Lou Cipher activates multiple transfers of 100$ so that multiple messages "credit 100$ to Lou Cipher" are sent from B1 to B2
  – The adversary identifies at least one of these messages
  – The adversary replies the message several times

ECB – block attack

- Mr. Lou Cipher performs *k* equal transfers
  - credit 100$ to Lou Cipher → c1
  - credit 100$ to Lou Cipher → c2
  - ...
  - credit 100$ to Lou Cipher → ck
- Then, he searches “his own” CT in the network
  - k equal CTs!
- Finally he replies one of these cryptograms



The number *k* is large enough to allow the adversary to identify the cryptograms corresponding to its transfers with high probability.

ECB – block attack

- An 8-byte timestamp field T (block #1) is added to the message to prevent replay attacks

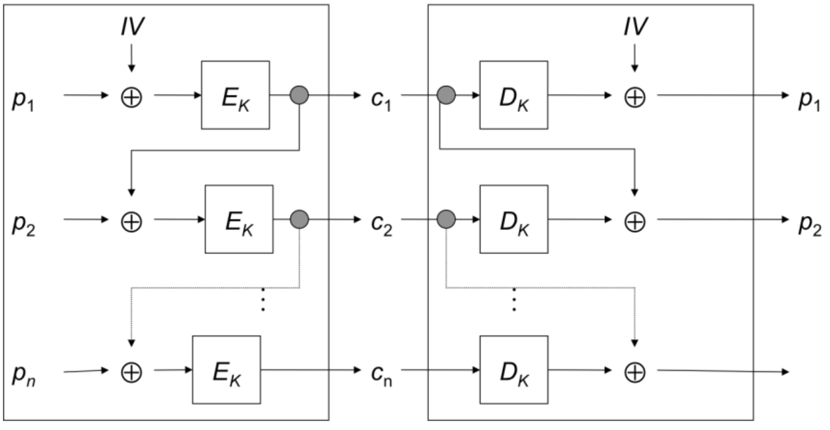
block no.	1	2	3	4	5	6	7	8	9	10	11	12	13
	T	**M**		**R**				**C**			**N**		**D**

- However, Mr Lou Cipher can
  - Identify “his own” CT by inspecting blocks #2-#13
  - Intercept any “fresh” CT
  - Substitute block #1 of “his own” CT with block #1 of the intercepted “fresh” block
  - Replay the resulting CT

Cipher block chaining (CBC)

$$c_0 \leftarrow IV . \forall 1 \le i \le t, c_i \leftarrow E_k(p_i \oplus c_{i-1})$$

$$c_0 \leftarrow IV . \forall 1 \le i \le t, p_i \leftarrow c_{i-1} \oplus D_k(c_i)$$



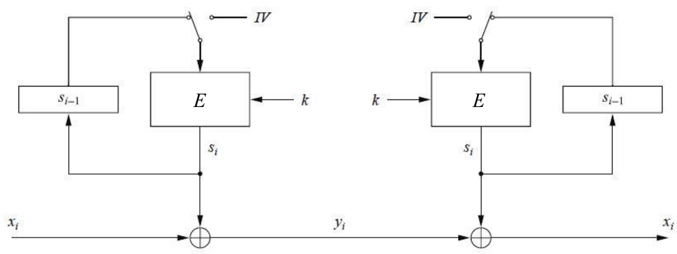
CBC - properties

- Chaining dependencies: c_i depends on p_i and all preceding PT blocks
- Encryption is *randomized* by using IV
  - CBC is *non deterministic*
    - Identical ciphertext results from the same PT under the same key and IV
  - IV is a *nonce*
- CT-block reordering affects decryption
- IV can be sent in the clear but its integrity must be guaranteed
- CBC suffers from Error propagation
  - Bit errors in c_i affect decryption of c_i and c_{i+1} (*error propagation*)
  - CBC is self-synchronizing (*error recovery*)
  - CBC does not tolerate “lost” bits (*framing errors*)

CBC – block attack

- If Bank A chooses a random IV for each wire transfer the attack will not work because LC
- However if LC substitutes blocks 5 – 10 and 13, bank B would decrypt *account number* and *deposit amount* to random numbers => *this is highly undesirable*
- Encryption itself is not sufficient, we need to protect integrity
  - We need additional mechanisms: MDC, MAC, dig sig

Output Feedback Mode (OFB)



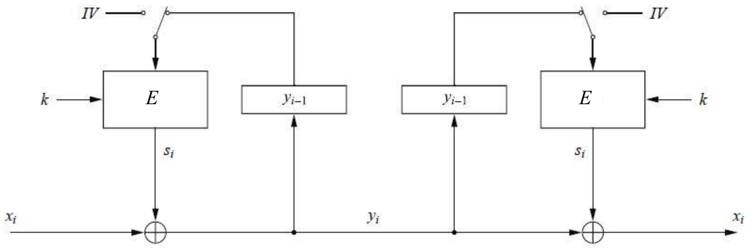
Let $e()$ be a block cipher of block size b; let x_i, y_i and s_i be bit strings of length b; and IV be a nonce of length b.
Encryption (first block): $s_1 = e_k(IV)$ *and* $y_1 = s_1 \oplus x_1$
Encryption (general block): $s_i = e_k(s_{i-1})$ *and* $y_i = s_i \oplus x_i, \quad i \geq 2$
Decryption (first block): $s_1 = e_k(IV)$ *and* $x_1 = s_1 \oplus y_1$
Decryption (general block): $s_i = e_k(s_{i-1})$ *and* $x_i = s_i \oplus y_i, \quad i \geq 2$

Output Feedback Mode (OFB)

- OFB builds a stream cipher out of a block cipher
- The key stream is generated block-wise
- OFB is a synchronous stream cipher
- The receiver does not use decryption (E^{-1})
- IV should be a nonce and make OFB non-deterministic
- Since OFB is synchronous, pre-computation of key stream blocks is possible

Cipher Feedback Mode (CFB)

UNIVERSITÀ DI PISA

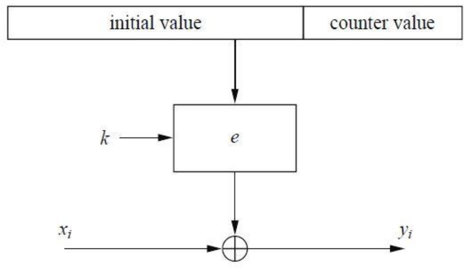


Definition 5.1.4 Cipher feedback mode (CFB)
Let $e()$ be a block cipher of block size b; let x_i and y_i be bit strings of length b; and IV be a nonce of length b.
Encryption (first block): $y_1 = e_k(IV) \oplus x_1$
Encryption (general block): $y_i = e_k(y_{i-1}) \oplus x_i, \quad i \geq 2$
Decryption (first block): $x_1 = e_k(IV) \oplus y_1$
Decryption (general block): $x_i = e_k(y_{i-1}) \oplus y_i, \quad i \geq 2$

Cipher Feedback Mode (CFB)

- OFB builds a stream cipher out of a block cipher
- CFB is an asynchronous stream cipher as the key stream is also a function of the CT
- Key stream is generated block-wise
- IV is a nonce and makes CFB nondeterministic

Counter Mode (CTR)



Definition 5.1.5 Counter mode (CTR)
Let $e()$ be a block cipher of block size b, and let x_i and y_i be bit strings of length b. The concatenation of the initialization value IV and the counter CTR_i is denoted by $(IV||CTR_i)$ and is a bit string of length b.
***Encryption**: $y_i = e_k(IV||CTR_i) \oplus x_i, \quad i \geq 1$*
***Decryption**: $x_i = e_k(IV||CTR_i) \oplus y_i, \quad i \geq 1$*

Counter Mode (CTR)

- CTR prevents 2TP
- CTR can be parallelized
- IV || CTR_i does not have to be kept secret
  – It can be transmitted together with CT_i
- Counter can be a regular counter or a more complex functions, e.g., LFSR

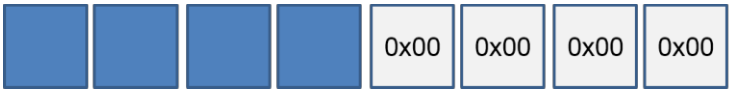
2TP

If we use the same input twice, then CT1 = PT1 xor S and CT2 = PT2 xor S. In case of KPA against one pt, e.g., against (CT1, PT1), then the adversary can determine S and then decrypt also the other pt.

Consider e = AES (b = 128 bit). Then, we can choose |IV| = 96 bit and |counter| = 32. Therefore, we can encrypt 2^{32} different pt's, each one being 128 bit (16 byte) long. Therefore, we can encrypt $2^{32} \times 2^4$ bytes = 64 Gbytes in total.

The need for a padding scheme

Naïve (wrong) solution: Pad the message with zeroes to the right, without ambiguous boundaries



Problem: What if the message was a NULL-terminated string?

At the receiving side: Was it a NULL-terminated string or a 7-bytes pt?

The PKCS#7 padding scheme

UNIVERSITÀ DI PISA

- Padding is necessary when PT len is not a block multiple

Block

If PT len is NOT a block multiple
Padding bytes ← #bytes to complete a block

H	E	L	L	O	3	3	3

If PT is a block multiple
Padding = block
Each padding byte ← 8

8	8	8	8	8	8	8	8

Padding give rise to **ciphertext expansion**

Exercise**:**

Let us suppose that you decrypt a CT and get the last block as […, 13, 06, 05].

Is this possible? Can CT be a valid ciphertext?

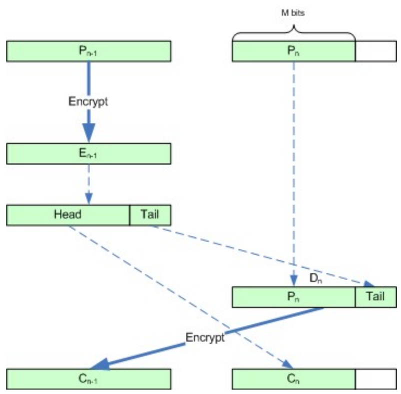
The answer is NO.

Ciphertext Stealing (CTS)

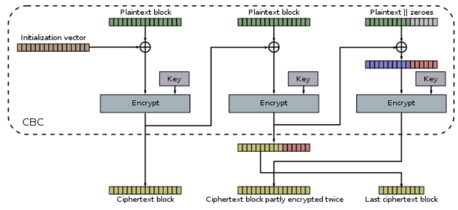
- CTS allows encrypting PT that is not evenly divisible into blocks without resulting in any ciphertext expansion
- sizeof(CT) = sizeof(PT)
- CTS operates on the last two blocks
  - A portion of the 2nd-last CT block is stolen to pad the last PT block

Ciphertext stealing (CTS)

ECB mode
(from Wikipedia)



CBC mode
(from Wikipedia)



Look at Wikipedia for a detailed treatment

To implement CTS encryption or decryption for data of unknown length, the implementation must delay processing (and buffer) the two most recent blocks of data, so that they can be properly processed at the end of the data stream.