Altri cifrari a blocchi

Alfredo De Santis

Dipartimento di Informatica ed Applicazioni
Università di Salerno

ads@dia.unisa.it
http://www.dia.unisa.it/professori/ads

Marzo 2012

Altri cifrari a blocchi

- RC2 [1989]
- IDEA (International Data Encryption Algorithm) [1990]
- Blowfish [1993]
- SAFER (Secure And Fast Encryption Routine)
  - SAFER K-64 [1994], SAFER K-128 [1995]
- RC5 [1994], RC6 [1998]
- Madryga, NDS, FEAL, REDOC, LOKI, Khufu, Knafre, MMB, GOST, ...
- AES

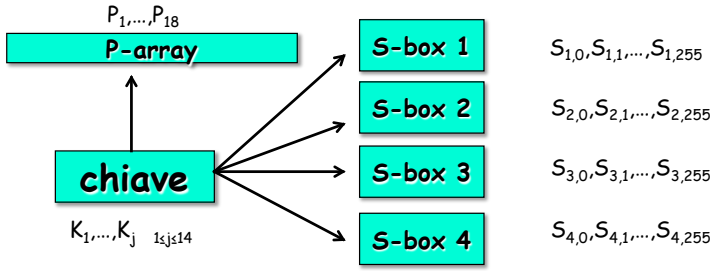
cifrario	bit chiave	bit testo
IDEA	128	64
SAFER K-64	64	64
SAFER K-128	128	64
RC5	<256 byte	32,64,128

Blowfish

- Ideato da Bruce Schneier nel '93
- Cifrario di Feistel
  - Dimensione dei blocchi: 64 bit
  - Numero di round: 16
  - Dimensione della chiave: da 1 a 14 word a 32 bit (da 32 a 448 bit)
- Altre caratteristiche:
  - Veloce (su microprocessori a 32 bit opera a 18 cicli di clock per byte)
  - Compatto (bastano meno di 5Kb di memoria)
  - Semplice da implementare e da analizzare
  - Implementato in numerosi prodotti software
  - Ritenuto più sicuro di DES

Espansione chiave

- Converte una chiave di al più 14 word a 32 bit (K-array) in un array di 18 sottochiavi a 32 bit (P-array)
- Genera 4 S-box 8x32, ognuna con 256 word a 32 bit



Espansione chiave:
Inizializzazione

- Inizializza in sequenza il P-array e le S-box con i valori della parte frazionaria di π
  - P_1=243F6A88
  - P_2=85A308D3
  - …
  - $S_{4,254}$=578FDFE3
  - $S_{4,255}$=3AC372E6
- Esegue lo XOR tra il P-array e il K-array
  - $P_1=P_1 \oplus K_1$
  - …
  - $P_{14}=P_{14} \oplus K_{14}$
  - $P_{15}=P_{15} \oplus K_1$
  - …
  - $P_{18}=P_{18} \oplus K_4$

Le word della chiave sono utilizzate più volte

Espansione chiave:
Inizializzazione

Sia $E_{P,S}[Y]$ la cifratura del blocco Y con il P-array e le S-box. Calcola

- $P_1,P_2=E_{P,S}[0]$
- $P_3,P_4=E_{P,S}[P_1||P_2]$
- …
- $P_{17},P_{18}=E_{P,S}[P_{15}||P_{16}]$
- $S_{1,0},S_{1,1}=E_{P,S}[P_{17}||P_{18}]$
- …
- $S_{4,254},S_{4,255}=E_{P,S}[S_{4,252}||S_{4,253}]$

Blocco di 64 bit tutti zero

Nuovi valori per P-array e S-box

- Sono necessarie 521 cifrature per generare gli array finali P e S
  - 9 per il P-array e 512 per S-box
  - Blowfish non adatto per applicazioni in cui la chiave cambia frequentemente

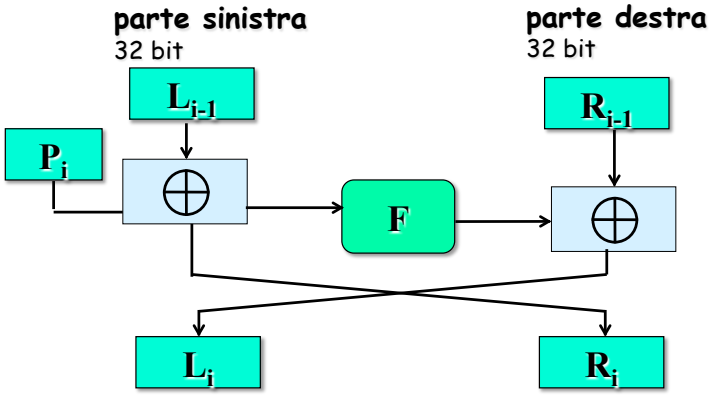
Cifratura

- Blowfish utilizza due operazioni:
  - + per la somma di word (modulo 2^{32})
  - ⊕ per lo XOR
- Testo in chiaro: L_0R_0 (64 bit)
- Testo cifrato: $L_{17}R_{17}$ (64 bit)

i-esima sottochiave

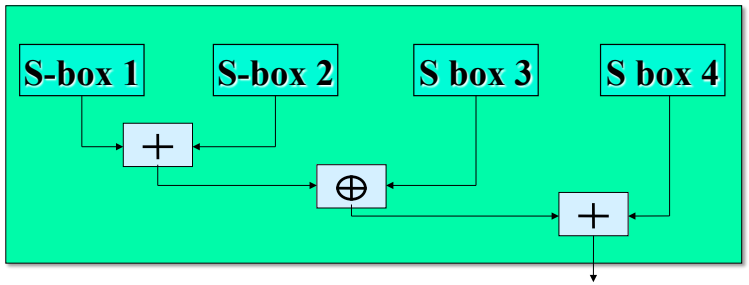
For i=1 to 16 do
$R_i=L_{i-1} \oplus P_i$;
$L_i=F[R_i] \oplus R_{i-1}$;
$L_{17}=R_{16} \oplus P_{18}$;
$R_{17}=L_{16} \oplus P_{17}$;

Struttura del round



La funzione F

- L'input a 32 bit di F viene diviso in 4 byte a,b,c,d:
  $F[a,b,c,d]=(S_{1,a}+S_{2,b}) \oplus S_{3,c})+S_{4,d}$
- Ciascuna fase include somme modulo 2^{32}, XOR e sostituzioni con la S-Box



Decifratura

- Stesso algoritmo, sottochiavi in ordine inverso
  - Testo cifrato: L_0R_0
  - Testo in chiaro: $L_{17}R_{17}$

for i=1 **to** 16 **do**
$R_i = L_{i-1} \oplus P_{19-i}$;
$L_i = F[R_i] \oplus R_{i-1}$;
$L_{17} = R_{16} \oplus P_1$;
$R_{17} = L_{16} \oplus P_2$;

Caratteristiche di Blowfish

- Sia le sottochiavi che le S-box dipendono dalla chiave
  - In DES le S-box sono fissate
- In ogni round le operazioni coinvolgono tutto il blocco
  - In DES, solo la parte destra
- Invulnerabile ad attacchi di forza bruta (se la dimensione della chiave è 14 word):
  - Per testare una sola chiave sono necessarie 522 esecuzioni dell'algoritmo
- Nel 1995, 1000 dollari di premio per la crittoanalisi
  - Vaudenay ha definito attacchi per versioni modificate

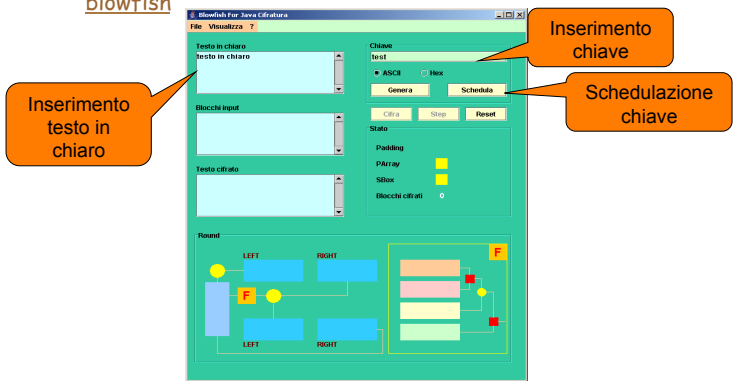
Confronto con altri sistemi

Algoritmo	Cicli di clock per fase	Numero di fasi	Cicli di clock per byte
Blowfish	9	16	18
RC5	12	16	23
DES	18	16	45
IDEA	50	8	50
Triple-DES	18	48	108

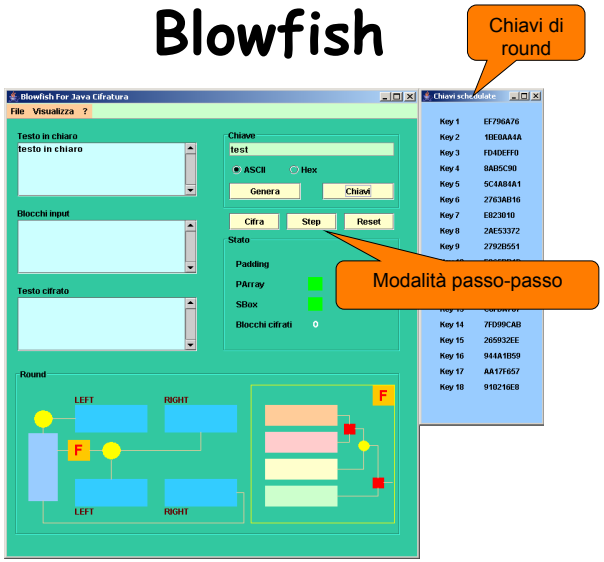
Blowfish

Proviamolo insieme, collegandoci al link

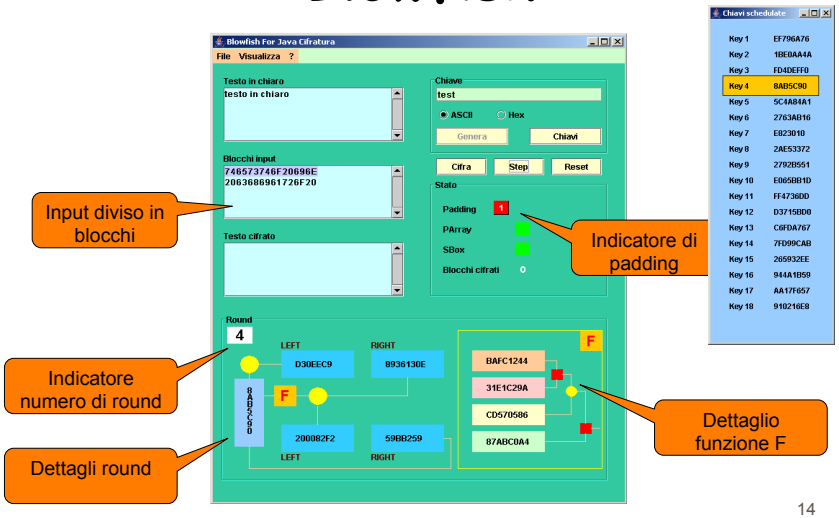
- http://www.dia.unisa.it/professori/masucci/sicurezza0809/blowfish



Blowfish



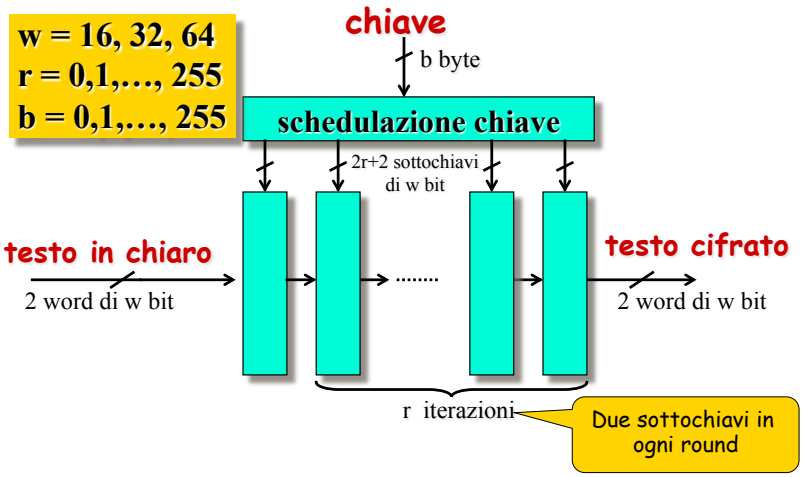
Blowfish



RC5

- Ideato da Ron Rivest nel 1995
- Parametri
  - Dimensione dei blocchi: variabile (32, 64, 128 bit)
  - Numero di round: variabile (da 0 a 255)
  - Dimensione della chiave: variabile (da 0 a 255 byte)
- Altre caratteristiche:
  - E' word-oriented (operazioni eseguite su blocchi della lunghezza della parola macchina)
  - Usa operazioni comuni dei processori e rotazioni data-dependant
  - Usa poca memoria (adatto per smart card e altre device)
  - Semplice da implementare e da analizzare
  - Implementato in numerosi prodotti della RSA Data Security Inc. (BSAFE, JSAFE, S/MAIL)

RC5-w/r/b

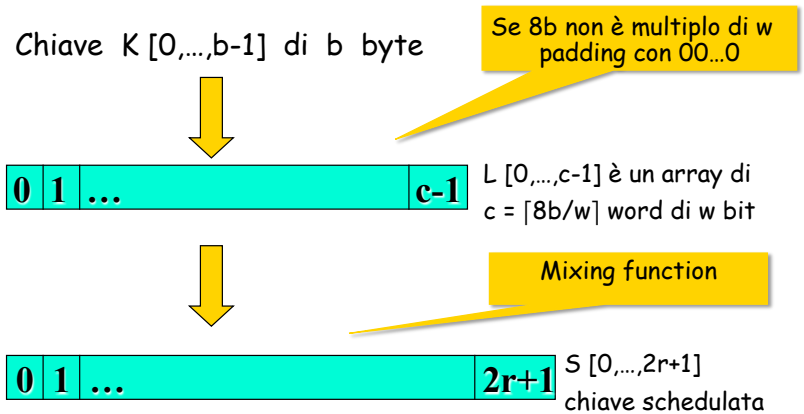


RC5: operazioni

Operazioni su parole di w bit:

- a+b somma modulo 2^w
- a-b sottrazione modulo 2^w
- a⊕b XOR bit a bit
- a « b shift a sinistra di a di un numero di bit dato dai log w bit meno significativi di b
- a » b shift a destra di a di un numero di bit dato dai log w bit meno significativi di b

RC5: schedulazione chiave



RC5

Due differenti architetture

- Little-endian (INTEL)
  - il valore di $a_1a_2a_3a_4$ è $a_1+a_2 2^8+a_3 2^{16}+a_4 2^{24}$
- Big-endian (SPARC)
  - il valore di $a_1a_2a_3a_4$ è $a_4+a_3 2^8+a_2 2^{16}+a_1 2^{24}$

RC5 funziona su macchine con architettura little-endian

RC5: schedulazione chiave

Inizializzazione array S (usa le costanti P_w e Q_w)

Mixing function (aggiornamento array S mediante array L)

```
S[0] = Pw
for  i = 1  to 2r+1  do
     S[i] ← S[i-1]+Qw
X ← Y ← 0
i ← j ← 0
do  3·max(c,2r+2)  times
     X ← S[i] ← (S[i]+X+Y) « 3
     Y ← L[j] ← (L[j]+X+Y) « (X+Y)
     i ← (i+1) mod (2r+2)
     j ← (j+1) mod c
```


Costanti magiche

P_w = espansione binaria a w bit del numero di Nepero

e = 2.71828182459045… (decimale) $P_w = Odd[(e-2)2^w]$

Q_w = espansione binaria a w bit del rapporto aureo

$Q_w = Odd[(\phi-1)2^w]$

$\varphi = (1+\sqrt{5})/2 = 1.61803398874989\ldots.$(decimale)

w	16 bit	32 bit	64 bit
P_w	b7 e1	b7 e1 51 63	b7 e1 51 62 8a ed 2a 6b
Q_w	9E 37	9E 37 79 b9	9E 37 79 b9 7f 4a 7c 15

Odd[x]=intero dispari più vicino a x

RC5: cifratura

Input: testo in chiaro (A,B)

Chiave schedulata: S[0,…,2r+1]

```
A ← A + S[0]
B ← B + S[1]
for  i = 1  to  r  do
     A ← ((A⊕ B) « B) + S[2i]
     B ← ((B ⊕ A) « A) + S[2i+1]
```

Entrambe le metà dei dati aggiornate in ogni round

Output: testo cifrato (A,B)

RC5: decifratura

```
A ← A + S[0]
B ← B + S[1]
for  i = 1  to  r  do
     A ← ((A⊕ B) « B) + S[2i]
     B ← ((B ⊕ A) « A) + S[2i+1]
```

cifratura

```
for  i = r  downto  1  do
     B ← ((B - S[2i+1]) » A) ⊕ A
     A ← ((A - S[2i]) » B ) ⊕ B
B ← B - S[1]
A ← A - S[0]
```

decifratura

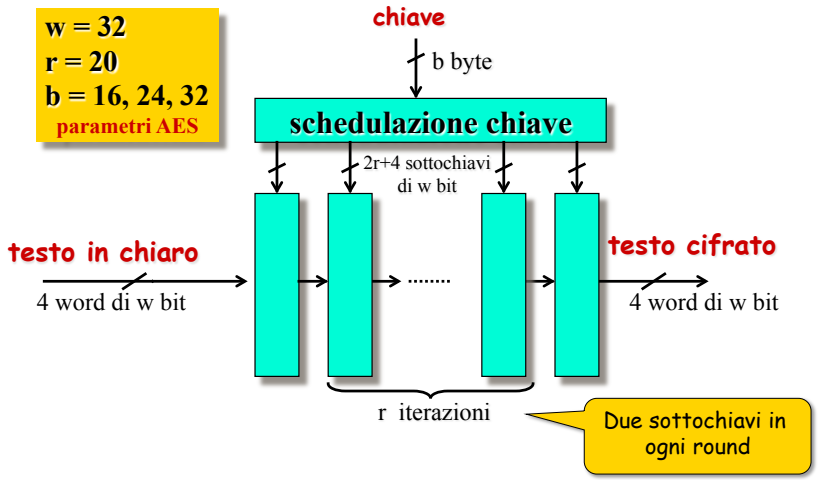
Caratteristiche di RC5

- Le rotazioni data-dependent evitano attacchi di crittoanalisi differenziale e lineare
- In ogni round le operazioni coinvolgono tutto il blocco
  - In DES, solo la parte destra
- RC5-32/16/16 è un buon compromesso tra sicurezza ed efficienza

RC6

- E' un'evoluzione dell'RC5, per andare incontro alle richieste del NIST riguardo le specifiche tecniche dell'AES
- Parametri
  - Dimensione dei blocchi: 128 bit
  - Numero di round: 20
  - Dimensione della chiave: variabile (16, 24 o 32 byte)
- Usa una ulteriore operazione: la moltiplicazione di interi

RC6-w/r/b



RC6

Operazioni su parole di w bit:

- a+b somma modulo 2^w
- a-b sottrazione modulo 2^w
- a⊕b XOR bit a bit
- a • b moltiplicazione modulo 2^w
- a « b shift a sinistra di a di un numero di bit dato dai log w bit meno significativi di b
- a » b shift a destra di a di un numero di bit dato dai log w bit meno significativi di b

RC6: schedulazione chiave

L [0,…,c-1] è un array di c = ⌈8b/w⌉ word di w bit

```
L [0,…,c-1] = chiave con padding di 0 se necessario
S[0] = P_w
for  i ← 1  to 2r+3  do
     S[i] ← S[i-1]+Q_w
A ← B  ← 0
i ← j ← 0
do 3·max(c,2r+4) times
     A ← S[i] ← (S[i]+A+B) « 3
     B ← L[j] ← (L[j]+A+B) « (A+B)
     i ← (i+1) mod (2r+4)
     j ← (j+1) mod c
```


RC6: cifratura

Input: testo in chiaro (A,B,C,D)
Chiave schedulata: S[0,…,2r+3]

```
B ← B + S[0]
D ← D + S[1]
for  i ← 1  to  r  do
     t ← (B · (2B+1)) «  log w
     u ← (D · (2D+1)) «  log w
     A ← ((A⊕ t) « u) + S[2i]
     C ← ((C ⊕ u) « t) + S[2i+1]
     (A,B,C,D) ← (B,C,D,A)
A ← A + S[2r+2]
C ← C + S[2r+3]
```

Output: testo cifrato (A,B,C,D)

Caratteristiche dei moderni cifrari a blocchi

- Lunghezza della chiave variabile
- Lunghezza del blocco variabile
- Numero di round variabile
- Uso di diversi operatori aritmetici e/o Booleani
- Uso di rotazioni data-dependant
- Uso di S-box key-dependant
- Operazioni sull'intero blocco

Bibliografia

- Cryptography and Network Security by W. Stallings (2003)
  - cap. 6
- Tesina di Sicurezza su reti
  - Cifrari a blocchi

Domande?