Stream cipher

Alfredo De Santis

Dipartimento di Informatica ed Applicazioni
Università di Salerno

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

Marzo 2012

Cifrari simmetrici

I cifrari simmetrici possono essere:

- **Cifrari a blocchi**:
  - trasformazione di grandi blocchi del testo in chiaro
- **Stream Cipher**:
  - trasformazione, dipendente dal tempo, di singoli caratteri del testo in chiaro

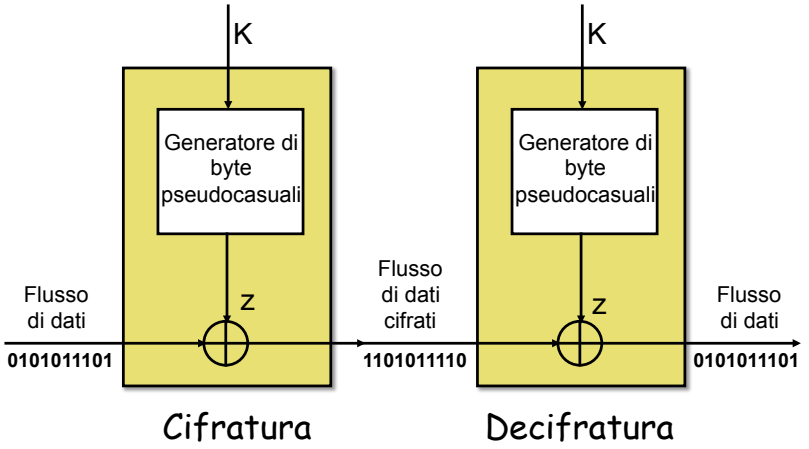
Stream cipher

- Cifra il messaggio un byte (o bit) alla volta
- Utilizza una sequenza (keystream) pseudo-casuale generata a partire dalla chiave (e dal messaggio)
- Cifra il messaggio mediante la keystream

Testo in chiaro	$M_0\ M_1\ M_2\ M_3\ \ldots\ M_i\ \ldots$
Keystream	$z_0\ z_1\ z_2\ z_3\ \ldots\ z_i\ \ldots$
Testo cifrato	$C_0\ C_1\ C_2\ C_3\ \ldots\ C_i\ \ldots$

Su alfabeto binario: $C_i = M_i$ XOR z_i

Stream cipher



Stream cipher

Molto più veloci dei cifrari a blocchi
- Poche linee di codice
- Operazioni semplici

Per complicare la crittoanalisi
- keystream con lungo periodo
  (più tardi inizia a ripetersi, meglio è)
- keystream con le stesse caratteristiche di una sequenza casuale. Ad esempio:
  - Dovrebbe avere lo stesso numero di 0 e di 1
  - Se vista come sequenza di byte, ciascuno dei 256 valori possibili dovrebbe apparire lo stesso numero di volte

Stream cipher

Descriveremo:
- Cifrario autokey
- LSFR (Linear Feedback Shift Register)
- RC4

Cifrario Autokey

Testo in chiaro	lettere 0, 1, …, 25
Keystream	$z_0 = K$, $z_i = M_{i-1}$ per $i=1,2,\ldots$
Testo cifrato	$C_i = M_i + z_i \mod 26$

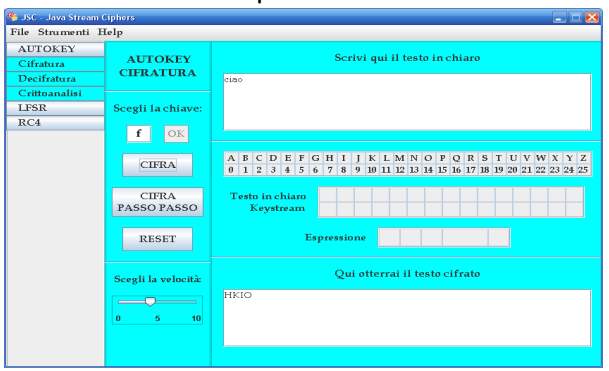
Esempio con chiave k=5

testo in chiaro: CIAO (2 8 0 14)
keystream: 5 2 8 0
cifrato: 7 10 8 14 → HKIO

chiave

Cifrario Autokey

Proviamolo insieme, collegandoci al link
- http://www.dia.unisa.it/professori/masucci/sicurezza0809/streamciphers/FrameIniziale.html



Cifrario Autokey

Testo in chiaro lettere 0,1,… , 25

Keystream $z_0 = K, \quad z_i = M_{i-1}$ per i=1,2,…

Testo cifrato $C_i = M_i + z_i \mod 26$

Quanto è sicuro?

Cifrario Autokey
known ciphertext attack

Testo in chiaro $M_0\ M_1\ M_2\ M_3\ \ldots\ M_i\ \ldots$

Keystream $K\ M_0\ M_1\ M_2\ \ldots\ M_{i-1}\ \ldots$

Testo cifrato $C_0\ C_1\ C_2\ C_3\ \ldots\ C_i\ \ldots$

$C_0 = M_0 + K \mod 26$

$C_i = M_i + M_{i-1} \mod 26, \quad i=1,2,\ldots$

Cifrario Autokey
known ciphertext attack

Testo in chiaro $M_0\ M_1\ M_2\ M_3\ \ldots\ M_i\ \ldots$

Keystream $K\ M_0\ M_1\ M_2\ \ldots\ M_{i-1}\ \ldots$

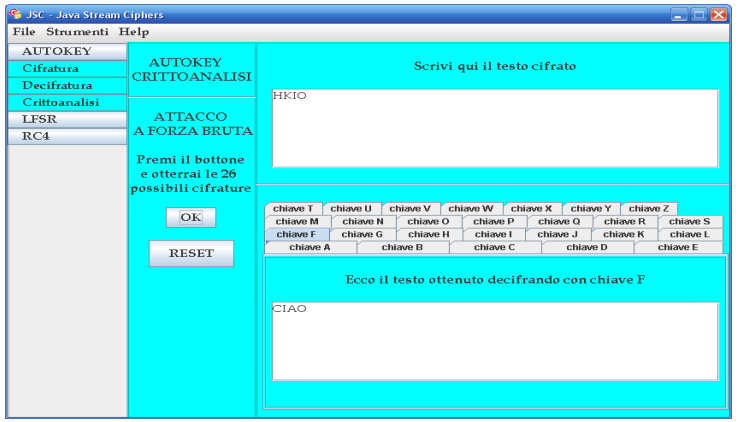
Testo cifrato $C_0\ C_1\ C_2\ C_3\ \ldots\ C_i\ \ldots$

$C_0 = M_0 + K \mod 26$

$C_i = M_i + M_{i-1} \mod 26$

Solo 26 chiavi!

Cifrario Autokey
known ciphertext attack



Linear Feedback Shift Register

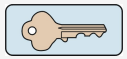
$$z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j} \mod 2 \qquad i = 0,1,2,\ldots$$

- Ricorrenza lineare di grado m
- Coefficienti $c_0\ c_1 \ldots c_{m-1}$ (costanti binarie predeterminate)
- Inizializzazione: $z_0 = k_0 \ldots , z_{m-1} = k_{m-1}$
- Implementazione hardware efficiente

Linear Feedback Shift Register

Fissati m coefficienti $c_0\ c_1 \ldots c_{m-1}$

$$z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j} \mod 2 \qquad i = 0,1,2,\ldots$$



Chiave: i valori di inizializzazione $k_0\ k_1 \ldots k_{m-1}$ e i coefficienti $c_0\ c_1 \ldots c_{m-1}$

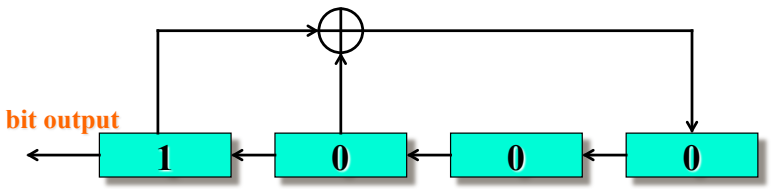
Testo in chiaro	M_0	M_1	M_2	M_3	M_4	…
Keystream	z_0	z_1	z_2	z_3	z_4	…
Testo cifrato	Y_0	Y_1	Y_2	Y_3	Y_4	…

Esempio $Y_i = M_i \oplus z_i$

Linear Feedback Shift Register

m=4

$z_{i+4} = z_i + z_{i+1} \mod 2$ i=0,1,2,…

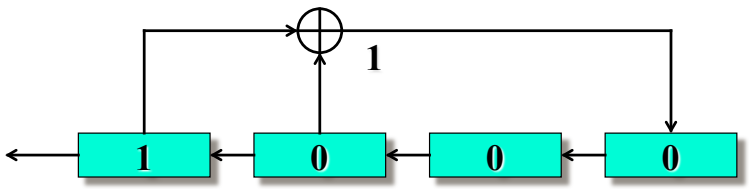


Inizializzazione: $z_0 = 1\ \ z_1 = 0\ \ z_2 = 0\ \ z_3 = 0$

Keystream: 1000100110

Linear Feedback Shift Register

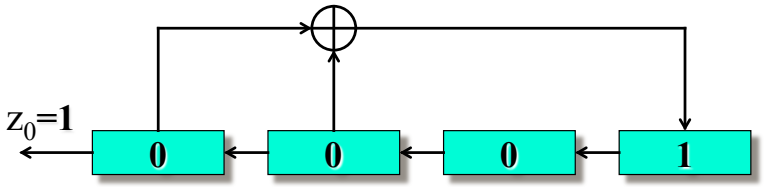
$z_{i+4} = z_i + z_{i+1} \mod 2$ i=0,1,2,…



Inizializzazione: $z_0 = 1\ \ z_1 = 0\ \ z_2 = 0\ \ z_3 = 0$

Linear Feedback Shift Register

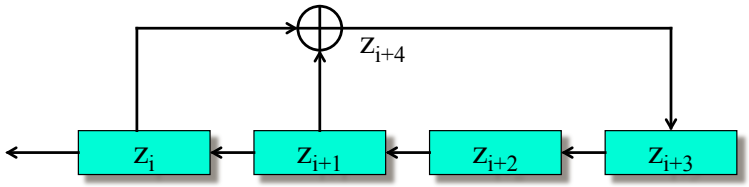
$z_{i+4} = z_i + z_{i+1} \mod 2 \qquad i=0,1,2,\ldots$



Inizializzazione: $z_0 = 1,\ z_1 = 0,\ z_2 = 0,\ z_3 = 0$

Linear Feedback Shift Register

$z_{i+4} = z_i + z_{i+1} \mod 2 \qquad i=0,1,2,\ldots$

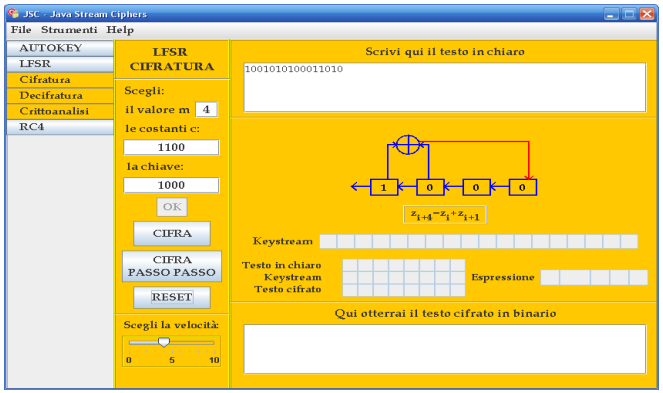


Inizializzazione: $z_0 = 1,\ z_1 = 0,\ z_2 = 0,\ z_3 = 0$

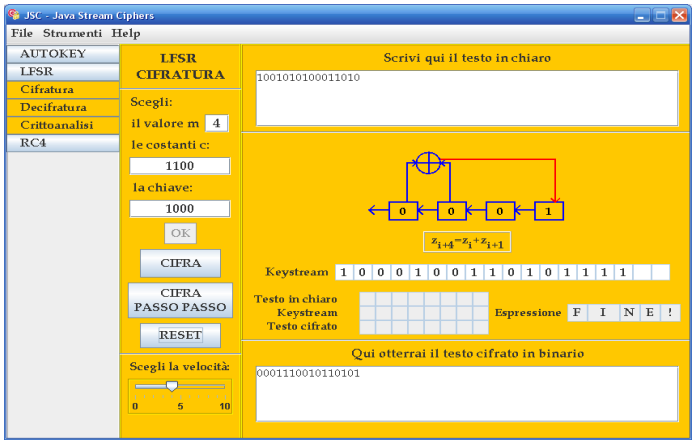
Keystream di periodo 15: **100010011010111…**

Linear Feedback Shift Register

http://www.dia.unisa.it/professori/masucci/sicurezza0809/streamciphers/FrameIniziale.html



Linear Feedback Shift Register



LFSR: Crittoanalisi

Known Plaintext Attack

-  conosce
  - testo in chiaro: $M_0 \ldots M_n$
  - testo cifrato: $Y_0 \ldots Y_n$
- ...e può calcolare $z_0 \ldots z_n$ $(z_i = M_i \oplus Y_i)$
- Se $n \geq 2m$ e conosce anche m, può computare i coefficienti $c_0, c_1, \ldots, c_{m-1}$ e quindi l'intera keystream

Tutte le operazioni sono lineari!

LFSR: Crittoanalisi

$$\left[z_{m+1}, z_{m+2}, \ldots, z_{2m}\right] = \left[c_0, c_1, \ldots, c_{m-1}\right] \begin{bmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \cdots & \cdots & \cdots & \cdots \\ z_m & z_{m+1} & & z_{2m-1} \end{bmatrix}$$

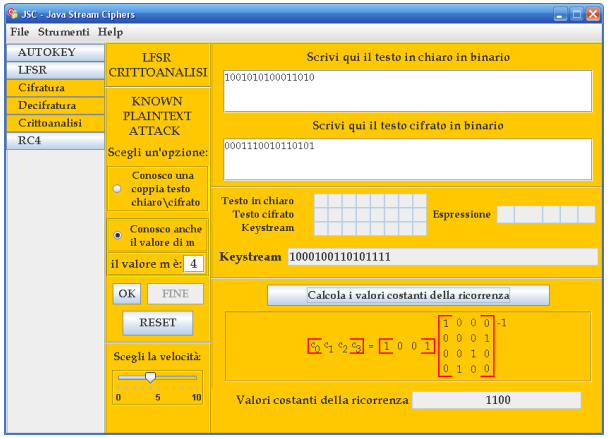
m equazioni lineari in m incognite $c_0, \ldots c_{m-1}$

LFSR: Crittoanalisi

$$\left[z_{m+1}, z_{m+2}, \ldots, z_{2m}\right] = \left[c_0, c_1, \ldots, c_{m-1}\right] \begin{bmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \cdots & \cdots & \cdots & \cdots \\ z_m & z_{m+1} & & z_{2m-1} \end{bmatrix}^{-1}$$

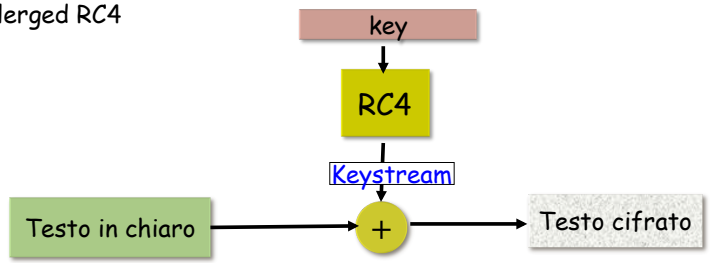
La matrice inversa esiste se il determinante è diverso da zero

LFSR: Crittoanalisi



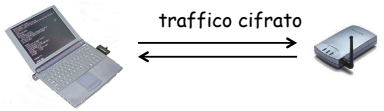
RC4

- Ideato da Ron Rivest nel 1987
- RC acronimo: "Rivest Cipher", "Ron's Code"
- Tenuto secreto, comparve in forma anonima nel 1994
  - mailing-list CypherPunks e newsgroup sci.crypt
  - Allerged RC4



RC4

- Caratteristiche:
  - Chiave: lunghezza variabile (da 1 a 256 byte)
  - Semplice da implementare e da analizzare
  - Genera una keystream con periodo maggiore di 10^{100}
  - Usa operazioni orientate ai byte
- Implementato in numerosi prodotti
  - SSL/TLS per la comunicazione sicura sul Web
  - IEEE 802.11 wireless LAN: WEP

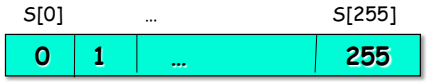


RC4: la chiave

- K[0]…K[h-1] vettore della chiave, di h byte
  - $1 \leq h \leq 256$

RC4

- Usa un vettore di byte S contenente una permutazione degli interi da 0 a 255
  - S[0]…S[255]
  - Inizializzazione S[i]=i, i=0,…,255



for i=0 **to** 255 **do**
S[i]=i

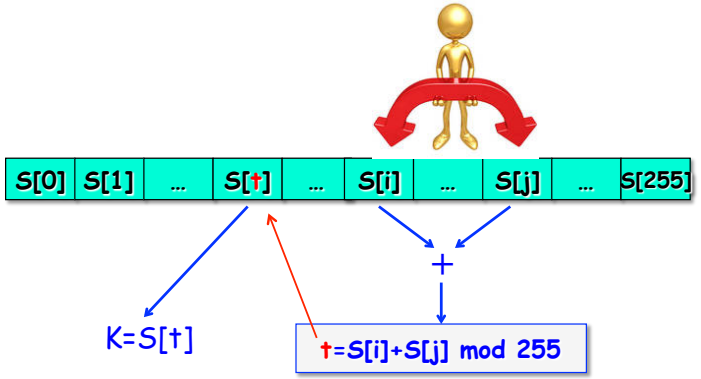
RC4: schedulazione chiave

- K è usata per aggiornare il vettore S
  - Produce una permutazione, scambiando ciascun S[i] con un S[j], dipendente da S[i] e K[i mod h]

```
j=0
for i=0 to 255 do
    j=(j+S[i]+K[i mod h]) mod 256;
    Swap(S[i],S[j]);
```

RC4: la keystream

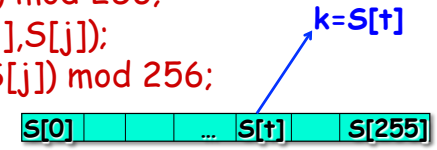
- Il vettore S è usato per generare la keystream



RC4: la keystream

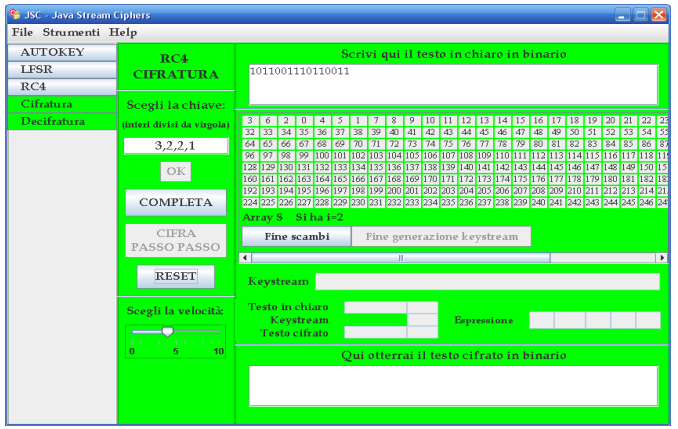
- Il vettore S è usato per generare la keystream

```
i,j=0
while (true)
    i=i+1 mod 256;
    j=(j+S[i]) mod 256;
    Swap(S[i],S[j]);
    t=(S[i]+S[j]) mod 256;
    k=S[t]
```

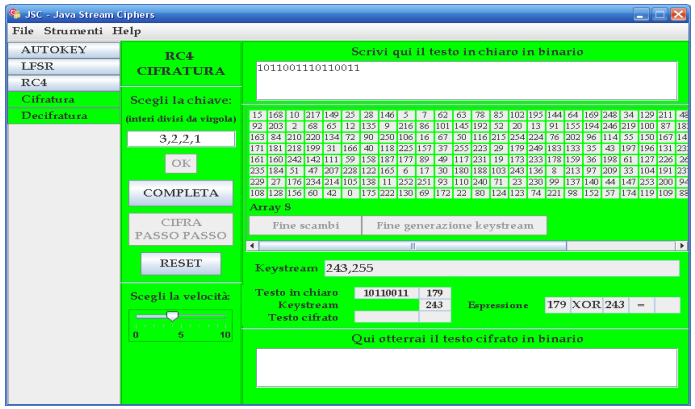


- Effettua lo XOR del byte k con il prossimo byte del testo in chiaro

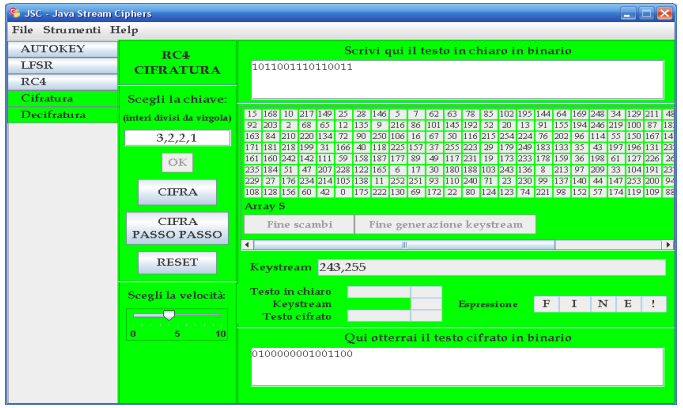
RC4



RC4

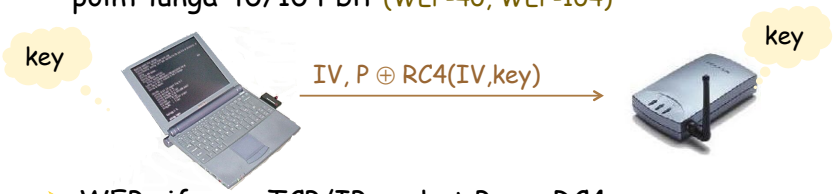


RC4

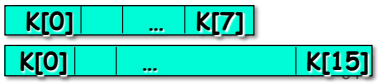


Wired Equivalent Privacy (WEP/WEP2)

- IEEE 802.11 wireless LAN
- Protocollo per garantire confidenzialità
- Usa una chiave key condivisa tra utenti e access point lunga 40/104 bit (WEP-40, WEP-104)

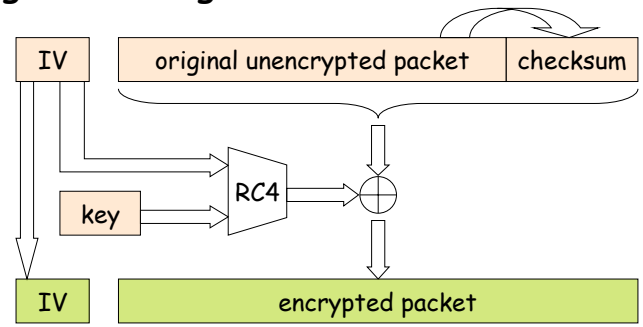


- WEP cifra un TCP/IP packet P con RC4
- IV "Initialization Vector" di 24 bit
- Chiave K di RC4: 8/16 byte



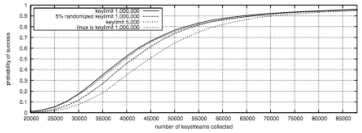
Wired Equivalent Privacy (WEP/WEP2)

Maggiori dettagli:



WEP: attacchi

- Fluhrer, Mantin, Shamir (2001)
  - Problema nella generazione della chiave in input a RC4
  - Il problema non dipende da RC4
  - Proposte modifiche a WEP per rendere vano l'attacco
- E. Tews, R. Weinmann, A. Pyshkin, Breaking 104-bit WEP in under a minute (2007)
  - 40.000 packet, probabilità successo 50%
  - 85.000 packet, probabilità successo 95%



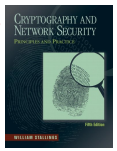
WEP: implementazioni attacchi

Aircrack-ng

- network software suite
  - detector
  - packet sniffer
  - WEP e WPA cracker (implementazione attacco PTW)
  - tool di analisi
- incluso in BackTrack (rinominata backtrack-ng)

Bibliografia

- Cryptography and Network Security by W. Stallings, 2010
  - cap. 6 (RC4)



- Cryptography: Theory and Practice by D. Stinson (1995)
  - cap. 1 (autokey, LFSR)

- Tesina di Sicurezza su reti
  - Crittografia classica a.a. 1999-2000

Domande?