cefriel
Consorzio per la Formazione e la Ricerca in Ingegneria dell'Informazione
Politecnico di Milano

POLITECNICO DI MILANO

RC4

Davide Cerri
CEFRIEL - Politecnico di Milano
cerri@cefriel.it
http://www.cefriel.it/~cerri/

RC4

- **RC4** è un **cifrario a flusso** progettato da Ron Rivest (la "R" di RSA) nel 1987.
- Era un segreto commerciale della RSA Security, ma nel 1994 è stato inviato in maniera anonima a una mailing list su Internet, e da allora è stato ampiamente analizzato.
- È utilizzato, tra l'altro, nel protocollo **TLS/SSL** (Transport Layer Security / Secure Sockets Layer) per la trasmissione sicura di pagine web e nel protocollo **WEP** (Wired Equivalent Privacy) all'interno dello standard IEEE 802.11b.
- È **semplice e veloce**.

Cifratura a flusso

- RC4 è una funzione che, a partire da una **chiave** (lunga da 1 a 256 ottetti), genera una sequenza pseudocasuale (**keystream**) utilizzata per cifrare e decifrare (mediante **XOR**) un flusso dati.

P	**10001111**	*cifratura*
	⊕	
RC4(K)	**00101011**	
C	**10100100**	*decifratura*
	⊕	
RC4(K)	**00101011**	
P	**10001111**	

Descrizione generale

- RC4 mantiene come **informazione di stato**:
  - un vettore di 256 ottetti: *S*[256];
  - due contatori: *i* e *j*.
- La chiave *K* (lunga da 1 a 256 ottetti) viene utilizzata **solo in fase di inizializzazione** dello stato.
- Il vettore *S* contiene in ogni istante una **permutazione** dei valori da 0 a 255.
- Ad ogni passo viene generato un ottetto di keystream prendendo uno degli elementi di *S*, che vengono ogni volta permutati.

Inizializzazione

- Inizializzazione dello stato:
  - in *S* si inseriscono i valori da 0 a 255: *S*[*n*] = *n*;
  - in un altro vettore temporaneo *T* (di 256 ottetti) si inserisce la chiave *K* (ripetendola se più corta);
  - i contatori *i* e *j* si inizializzano a zero;
  - si percorre *S* scambiando l'elemento corrente (*i*-esimo) con un altro determinato usando la chiave:

```
for i = 0 to 255
    j = (j + S[i] + T[i]) mod 256
    scambia (S[i], S[j])
```

  - si reinizializzano *i* e *j* a zero e si scarta *T* (la chiave non viene più utilizzata).

Generazione del keystream

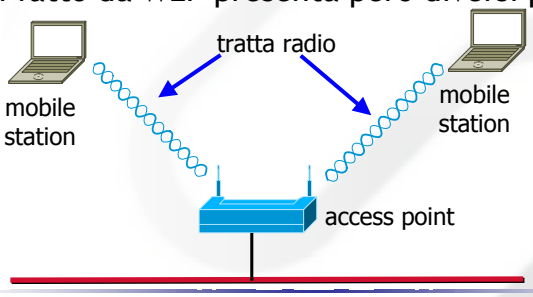
- Si percorre il vettore *S*, scambiando l'elemento corrente (*i*-esimo) con un altro determinato dallo stato corrente di *S* e *j*.
- Per generare un ottetto *k* del keystream, dallo stato corrente (*S*, *i*, *j*):

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
scambia (S[i], S[j])
t = (S[i] + S[j]) mod 256
k = S[t]
```

- Considerando *S*, *i*, *j*, RC4 può trovarsi in ben $256! \times 256^2$ (circa 2^{1700}, o $5{,}62 \times 10^{511}$) stati.

Uso di RC4: WEP

- RC4 è un algoritmo **ben noto e molto utilizzato**, e se usato correttamente **è considerato sicuro**.
- RC4 è utilizzato tra l'altro all'interno dello standard IEEE 802.11b per la cifratura delle trame sulla tratta radio (**WEP** - Wired Equivalent Privacy), l'utilizzo di RC4 fatto da WEP presenta però diversi problemi.



WEP: funzionamento (1)

- Le stazioni **condividono una chiave segreta** K (40 bit, oppure 104 bit nel cosiddetto "WEP a 128 bit") **con l'access point** (può essere la stessa chiave per tutte le stazioni).
- Viene calcolata una **checksum** (CRC) sul payload (attenzione: un CRC non è un MAC!).
- Il payload e il CRC vengono **cifrati con RC4**, utilizzando come chiave la concatenazione di K e di un vettore di inizializzazione (IV) di 24 bit. IV è scelto dal trasmettitore e inserito **in chiaro** nel pacchetto.
- Il ricevitore estrae IV, decifra il payload e la checksum, controlla la checksum.

WEP: funzionamento (2)

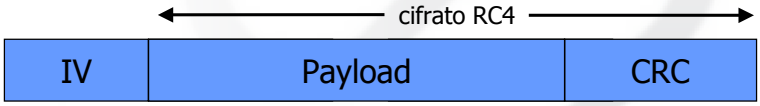
- Sia M il payload, si calcola la checksum $\mathrm{CRC}(M)$, il plaintext è $P = M \,||\, \mathrm{CRC}(M)$, si concatena IV con K e si calcola:

$$C = P \oplus \mathrm{RC4}(IV \,||\, K)$$

- Il ricevitore estrae IV e calcola:

$$P' = C \oplus \mathrm{RC4}(IV \,||\, K)$$

- Il ricevitore estrae da P' il messaggio e la checksum, se la checksum calcolata localmente coincide con quella ricevuta accetta il pacchetto.



WEP: riuso del keystream (1)

- La presenza di IV serve a **cambiare il keystream**.
- Ecco che cosa accade se due messaggi sono cifrati con lo stesso keystream:

$$C_1 = P_1 \oplus \mathrm{RC4}(IV \,||\, K) \qquad C_2 = P_2 \oplus \mathrm{RC4}(IV \,||\, K)$$
$$C_1 \oplus C_2 = P_1 \oplus \mathrm{RC4}(IV \,||\, K) \oplus P_2 \oplus \mathrm{RC4}(IV \,||\, K) =$$
$$= P_1 \oplus P_2$$

- Facendo lo XOR di due messaggi cifrati si ottiene **lo XOR dei due messaggi in chiaro**!
  - Se si conosce uno dei due messaggi si ottiene l'altro.
  - Se si hanno molti messaggi cifrati con lo stesso keystream le cose si fanno più semplici.
- In un cifrario a flusso **non si deve riutilizzare il keystream**!

WEP: riuso del keystream (2)

- La presenza di IV serve a evitare i problemi visti, ma...
  - gli IV possibili sono 2^{24} = 16.777.216 (**pochi**!);
  - il numero degli IV è **indipendente dalla lunghezza della chiave** K (IV è comunque di 24 bit);
  - IV è **trasmesso in chiaro**, quindi è facile vedere se due messaggi hanno lo stesso IV.
- In più:
  - K cambia molto raramente (richiede riconfigurazione manuale) e spesso è comune a tutte le stazioni;
  - molte schede 802.11b usano come IV un contatore, inizializzato a 0 quando vengono attivate.

WEP: dizionari di attacco

- Se l'attaccante ottiene il messaggio in chiaro P corrispondente a un certo messaggio cifrato C ottiene anche il relativo keystream:
  $P \oplus C = P \oplus P \oplus \mathrm{RC4}(IV \,||\, K) = \mathrm{RC4}(IV \,||\, K)$
- L'attaccante può costruire un **dizionario di attacco** con tutti i keystream, con cui decifrare qualunque messaggio.
- La cosa è un po' lunga, ma è importante notare che:
  - la dimensione del dizionario è di 2^{24} keystream, **indipendente** dalla dimensione di K;
  - se le schede 802.11b ripartono ogni volta da 0 con IV, gli IV bassi saranno quelli più usati.

WEP: controllo di integrità

- WEP utilizza per il controllo di integrità un CRC, ma CRC è pensato per rivelare **errori casuali**, non per resistere ad **attacchi intenzionali** (non è un MAC!).
- Il CRC è **lineare** rispetto allo XOR, ovvero:
  $\mathrm{CRC}(X \oplus Y) = \mathrm{CRC}(X) \oplus \mathrm{CRC}(Y)$
- Sfruttando questa proprietà, combinata al fatto che la cifratura RC4 è effettuata mediante XOR, è possibile **alterare un messaggio alterando corrispondentemente anche la checksum**!
  - Dato un messaggio cifrato C (corrispondente a un messaggio in chiaro non noto M), è possibile creare un nuovo C' che venga decifrato in M' (con checksum corretta), dove $M' = M \oplus \Delta$ e Δ è scelto arbitrariamente.

WEP: alterazione (dimostrazione)

- Dimostrazione:

$$C = \mathrm{RC4}(IV \,||\, K) \oplus (M \,||\, \mathrm{CRC}(M))$$

$$\begin{aligned} C' &= C \oplus (\Delta \,||\, \mathrm{CRC}(\Delta)) \\ &= \mathrm{RC4}(IV \,||\, K) \oplus (M \,||\, \mathrm{CRC}(M)) \oplus (\Delta \,||\, \mathrm{CRC}(\Delta)) \\ &= \mathrm{RC4}(IV \,||\, K) \oplus ((M \oplus \Delta) \,||\, (\mathrm{CRC}(M) \oplus \mathrm{CRC}(\Delta)) \\ &= \mathrm{RC4}(IV \,||\, K) \oplus ((M \oplus \Delta) \,||\, \mathrm{CRC}(M \oplus \Delta)) \\ &= \mathrm{RC4}(IV \,||\, K) \oplus (M' \,||\, \mathrm{CRC}(M')) \end{aligned}$$

- Quindi il pacchetto **può essere alterato** eludendo il controllo di integrità.
  - È sufficiente fare lo XOR tra il pacchetto originale e il pacchetto "differenza", e la checksum si modificherà in modo da **risultare comunque corretta**.

WEP: debolezze di RC4

- Gli attacchi visti finora sono indipendenti da eventuali debolezze di RC4.
- In un articolo dell'agosto 2001 sono state però pubblicate delle **debolezze di RC4** dalle quali WEP, per il modo in cui genera e usa le chiavi, è affetto (mentre TLS/SSL ad esempio non lo è).
  - si sfrutta il fatto che alcuni IV danno luogo a **chiavi deboli** (e gli IV viaggiano in chiaro...);
  - si sfrutta il fatto che si conosce l'inizio (primo byte) dei pacchetti in chiaro;
  - avendo a disposizione circa 5 milioni di pacchetti, si è solitamente in grado di **ottenere la chiave segreta**!
  - l'attacco è **totalmente passivo**!

WEP: riassunto debolezze

- WEP, pur utilizzando un algoritmo crittografico noto e studiato (RC4) lo fa in **modo non corretto**:
  - chiavi corte e "simili"
    (la parte maggiore - K - è fissa);
  - limitazione degli IV
    (i valori possibili sono troppo pochi);
  - riutilizzo keystream.
- Progettare un sistema sicuro è **difficile**, anche se si utilizzano algoritmi crittografici noti e studiati...