

# Modulo 2: Stream Ciphers, RC4 e la Debacle di WEP

## Sommario

In questo modulo analizziamo la classe di algoritmi nota come **Stream Ciphers**. Partendo dalla teoria, vedremo come questi tentino di approssimare il One-Time Pad in modo pratico. Successivamente, analizzeremo un caso di studio fondamentale nella storia della sicurezza informatica: il protocollo **WEP (Wired Equivalent Privacy)**, un esempio perfetto di come un cifrario "decente" (RC4) possa essere implementato in modo catastrofico.

## Indice

<b>1 Stream Ciphers: Approssimare la Perfezione</b>	<b>1</b>
1.1 Il Concetto di "Keystream" . . . . .	1
1.2 Lo Stato (State) e la Generazione . . . . .	2
1.3 Il Ruolo Cruciale del Nonce (IV) . . . . .	2
<b>2 L'Algoritmo RC4</b>	<b>2</b>
2.1 Struttura di RC4 . . . . .	3
2.2 Debolezze Crittografiche . . . . .	3
<b>3 Case Study: Il Disastro di WEP (Wired Equivalent Privacy)</b>	<b>3</b>
3.1 Problema 1: Gestione dell'IV (Confidenzialità) . . . . .	3
3.2 Problema 2: Autenticazione (Challenge-Response Fallita) . . . . .	4
3.3 Problema 3: Integrità e Malleabilità (Linearità) . . . . .	5
<b>4 Sintesi delle Lezioni Apprese da WEP</b>	<b>6</b>

---

## 1 Stream Ciphers: Approssimare la Perfezione

Come visto nel modulo precedente, il One-Time Pad (OTP) offre sicurezza perfetta ma richiede una chiave lunga quanto il messaggio. Gli **Stream Ciphers** (cifrari a flusso) nascono per risolvere questo problema pratico.

### 1.1 Il Concetto di "Keystream"

L'idea base è utilizzare una chiave segreta corta ( $K$ ) per generare pseudo-casualmente una sequenza di bit arbitrariamente lunga, detta Keystream ( $S$  o  $k'$ ).

Il funzionamento è identico all'OTP, ma la chiave randomica è sostituita dal keystream pseudo-randomico:

- **Encryption:**  $C_i = P_i \oplus S_i$

- **Decryption:**  $P_i = C_i \oplus S_i$

Dove  $S_i$  è l'i-esimo bit (o byte) generato da un **PRNG** (Pseudo-Random Number Generator) inizializzato con la chiave  $K$ .

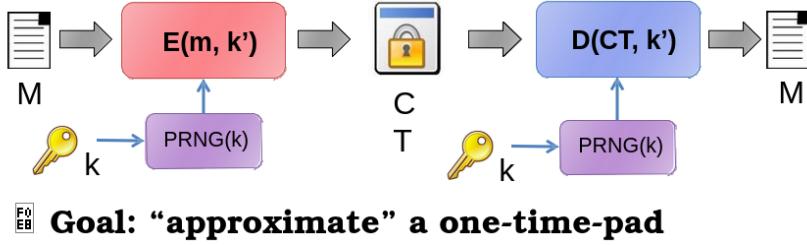


Figura 1: Schema generale di uno Stream Cipher

## 1.2 Lo Stato (State) e la Generazione

A differenza dei cifrari a blocchi (come AES), gli stream ciphers operano sui dati bit per bit o byte per byte. Formalmente, uno stream cipher è una macchina a stati:

1. Ha uno **Stato Interno** segreto che evolve nel tempo.
2. Una funzione di aggiornamento (*Update*) modifica lo stato ad ogni passo.
3. Una funzione di output (*Generate*) estrae bit dallo stato per formare il keystream.

## 1.3 Il Ruolo Cruciale del Nonce (IV)

Poiché il PRNG è deterministico, la stessa chiave  $K$  produrrà sempre lo stesso keystream. Se cifriamo due messaggi diversi con la stessa chiave, otteniamo:

$$C_1 = P_1 \oplus S$$

$$C_2 = P_2 \oplus S$$

Un attaccante può calcolare  $C_1 \oplus C_2 = P_1 \oplus P_2$ , rompendo la cifratura (problema del *Two-Time Pad*).

Per evitare ciò, gli stream ciphers moderni prendono in input non solo la chiave  $K$ , ma anche un valore pubblico chiamato Nonce o **IV** (Initialization Vector).

$$S = \text{PRNG}(K, IV)$$

Cambiare l'IV per ogni messaggio garantisce un keystream completamente diverso, preservando la sicurezza semantica anche riutilizzando la chiave a lungo termine.

## 2 L’Algoritmo RC4

**RC4** (Rivest Cipher 4) è lo stream cipher più famoso della storia, utilizzato in SSL/TLS e, sfortunatamente, in WEP.

## 2.1 Struttura di RC4

RC4 è semplice ed elegante, ottimizzato per implementazioni software (a differenza dei registri a scorrimento LFSR usati nell'hardware).

- **Stato:** Una permutazione  $S$  di tutti i 256 byte possibili ( $S[0]...S[255]$ ).
- **KSA (Key Scheduling Algorithm):** Inizializza la permutazione  $S$  mescolandola in base alla chiave  $K$ .
- **PRGA (Pseudo-Random Generation Algorithm):** Genera il keystream scambiando elementi nell'array  $S$  e sommando indici modulo 256.

Formalmente:

$$\begin{aligned} \text{ENC}(\text{Key}, \text{MSG}) &= \text{MSG} \oplus \text{RC4}(\text{IV}, \text{Key}) \\ \text{Keystream} &= \text{RC4}(\text{IV}, \text{Key}) \end{aligned}$$

## 2.2 Debolezze Crittografiche

Sebbene considerato sicuro negli anni '90, RC4 ha mostrato gravi difetti statistici:

1. **Biases (Distorsioni):** I primi byte del keystream non sono perfettamente casuali. Alcuni valori appaiono con probabilità leggermente superiore ad altri.
2. **Attacco Fluhrer, Mantin, Shamir (FMS):** Hanno dimostrato che specifici "IV deboli" fanno trapelare informazioni sulla chiave segreta nei primi byte del keystream.

## 3 Case Study: Il Disastro di WEP (Wired Equivalent Privacy)

WEP è lo standard di sicurezza originale per il Wi-Fi (802.11). È considerato uno dei migliori esempi didattici di **come NON progettare un protocollo crittografico**.

### 3.1 Problema 1: Gestione dell'IV (Confidenzialità)

WEP utilizza RC4. Per evitare il riutilizzo del keystream, concatena la chiave segreta condivisa ( $K$ ) con un IV trasmesso in chiaro.

$$\text{Keystream} = \text{RC4}(\text{IV} \parallel K)$$

I difetti implementativi sono fatali:

- **IV troppo corto:** L'IV è di soli **24 bit**. Qui la generazione è lasciata all'implementazione (*MAI fare una cosa del genere in protocolli di sicurezza*).
- **Collisioni Garantite:** Con  $2^{24}$  possibili valori (circa 16 milioni), su una rete Wi-Fi trafficata gli IV si ripetono in poche ore (o molto meno a causa del Paradosso del Compleanno - circa 4000 frame per avere il 50% di probabilità di collisione).
- **Implementation Fail:** Molte schede di rete ripartivano da IV=0 al riavvio, o usavano contatori semplici, rendendo le collisioni prevedibili.

**Conseguenza:** Appena un IV si ripete, l'attaccante recupera lo XOR di due plaintext. Se riesce a indovinare parte di un messaggio (es. intestazioni HTTP), recupera il keystream e può decifrare tutti i pacchetti futuri con quell'IV.

### 3.2 Problema 2: Autenticazione (Challenge-Response Fallita)

WEP non usa solo la cifratura per la confidenzialità, ma anche per l'autenticazione ("Shared Key Authentication"). L'obiettivo è provare di conoscere la chiave  $K$  senza rivelarla.

Il protocollo è il seguente:

1. Access Point (AP) invia una stringa casuale (**Challenge**) in chiaro al client.
2. Il client cifra il Challenge usando WEP ( $RC4(IV, K)$ ) e invia la **Response**.
3. L'AP decifra e verifica se corrisponde al Challenge.

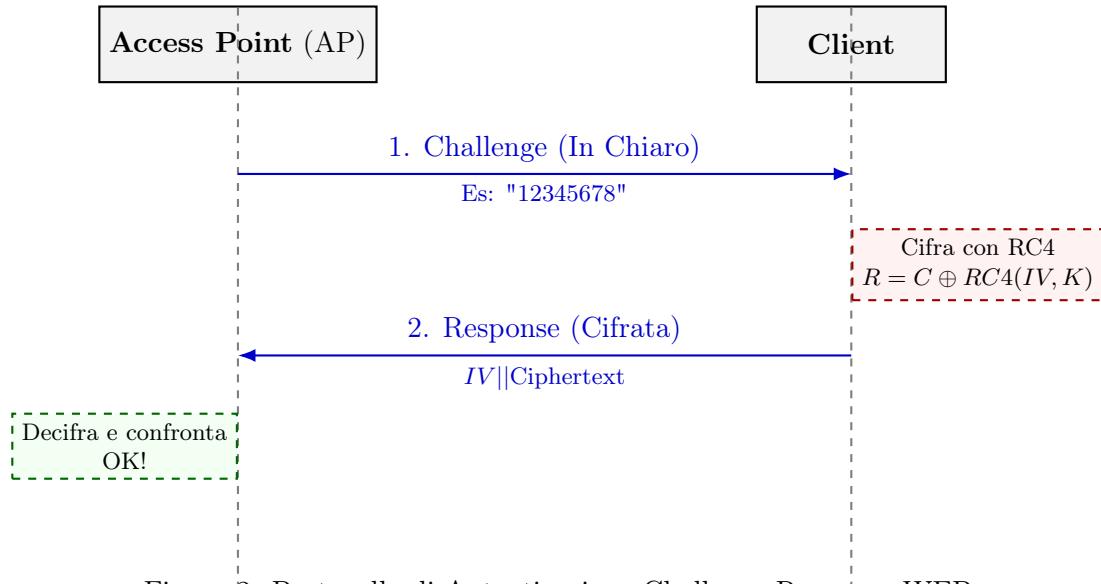


Figura 2: Protocollo di Autenticazione Challenge-Response WEP

Fatto così, non assomiglia ad un **KPA**?

#### L'Attacco (Keystream Recovery)

Questo meccanismo è un suicidio crittografico. Un attaccante passivo vede:

- Il **Plaintext** (il Challenge in chiaro).
- Il **Ciphertext** (la Response).

Poiché  $C = P \oplus$  Keystream, l'attaccante calcola semplicemente:

$$\text{Keystream} = P \oplus C$$

Ora l'attaccante possiede il keystream valido per quell'IV.

**Risultato:** L'attaccante non ha bisogno di conoscere la chiave  $K$ . Può autenticarsi alla rete aspettando una nuova Challenge dall'AP, cifrandola con il keystream rubato e inviando la risposta. *Authentication Bypass* totale.

### No need to know key to enter network!

⇒ JUST one [IV, RC4(IV,K)] pair → game over!

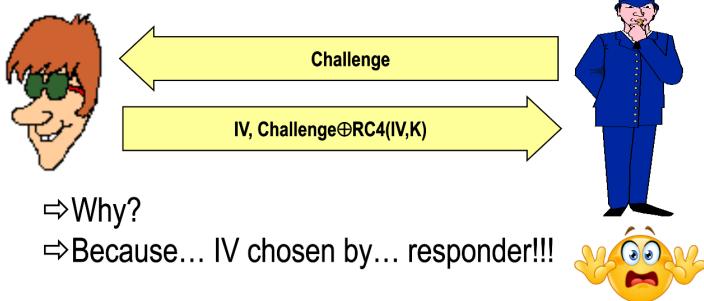


Figura 3: Attacco all'autenticazione WEP

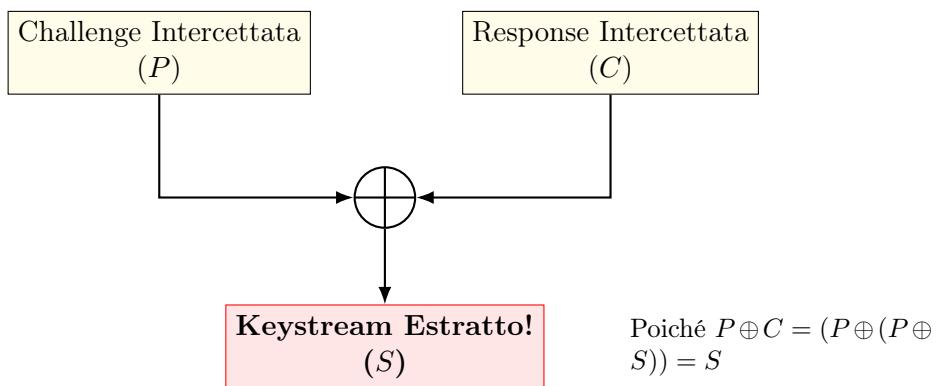


Figura 4: Attacco Known Plaintext su WEP: recupero immediato del Keystream

### 3.3 Problema 3: Integrità e Malleabilità (Linearità)

WEP deve garantire che i messaggi non vengano modificati in transito. Per farlo, utilizza il **CRC-32** (Cyclic Redundancy Check), calcolando un checksum del messaggio (ICV) e cifrandolo insieme al messaggio.

**Il Difetto:** Il CRC-32 è una funzione lineare rispetto all'operazione XOR:

$$\text{CRC}(A \oplus B) = \text{CRC}(A) \oplus \text{CRC}(B)$$

Anche la cifratura a flusso è lineare (XOR). Questa "doppia linearità" permette due attacchi devastanti.

#### A. Message Modification (Bit-Flipping Attack)

Un attaccante vuole modificare un messaggio cifrato  $C$  (che contiene  $P$ ) in un messaggio  $P' = P \oplus \Delta$  (es. cambiare l'importo di un bonifico o un indirizzo IP). L'attaccante non conosce  $P$  né la chiave, ma può modificare il ciphertext intercettato  $C$  in  $C'$ :

1. Sceglie la modifica  $\Delta$  (pattern di bit da invertire).
2. Calcola  $\text{CRC}(\Delta)$ .
3. Modifica il ciphertext:  $C' = C \oplus (\Delta \parallel \text{CRC}(\Delta))$ .

Quando la vittima decifra  $C'$ , ottiene:

$$P' = P \oplus \Delta$$

E il controllo di integrità (ICV) calcolato sul nuovo messaggio corrisponderà perfettamente, perché la modifica fatta al checksum compensa esattamente la modifica fatta ai dati. Il sistema accetta il messaggio manipolato come valido.

## B. Message Injection

Sfruttando la falla dell'autenticazione (recupero del keystream), un attaccante può iniettare traffico arbitrario nella rete.

1. L'attaccante costruisce un pacchetto malevolo  $M$  (es. un comando ARP o DNS spoofing).
2. Calcola il  $\text{CRC}(M)$ .
3. Se possiede una coppia conosciuta (IV, Keystream) (ottenuta dall'attacco all'autenticazione), cifra il pacchetto:

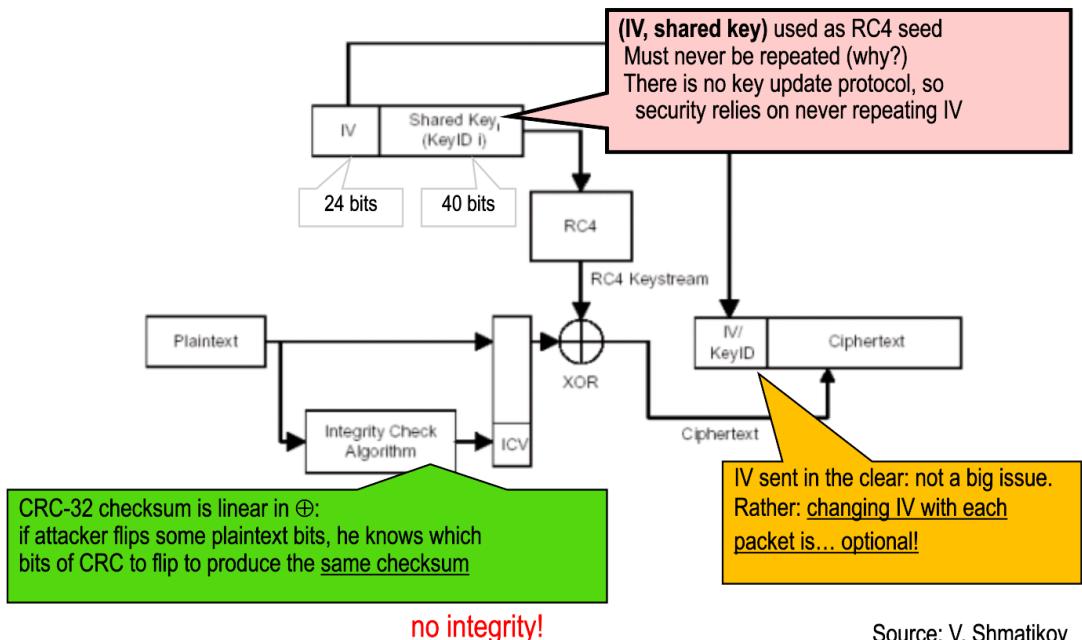
$$C = (M \parallel \text{CRC}(M)) \oplus \text{Keystream}$$

4. Invia il pacchetto con quell'IV in chiaro. L'AP lo riceverà, lo decifrerà correttamente e lo riterrà autentico.

## 4 Sintesi delle Lezioni Apprese da WEP

- **Non usare IV corti:** 24 bit sono insufficienti. WPA usa 48 bit.
- **Non usare integrità lineare:** CRC-32 è per rilevare errori casuali (rumore), non attacchi malevoli. Servono MAC (Message Authentication Codes) crittografici come HMAC-SHA1 o AES-CMAC.
- **L'autenticazione non è cifratura:** Usare la primitiva di cifratura per fare autenticazione challenge-response in modo ingenuo espone il keystream.
- **Stream Ciphers richiedono cura:** La sincronizzazione e l'unicità del keystream sono fragili.

Questi errori hanno portato allo sviluppo di WPA (una "pezza" temporanea che usava ancora RC4 ma con gestione chiavi migliore, TKIP) e finalmente **WPA2/WPA3** basati su AES (Block Cipher).



Source: V. Shmatikov

Figura 5: Riepilogo dell'evoluzione della sicurezza Wi-Fi