

RC4 Stream Cipher

Dominik Schürmann

Grundlagen der Sicherheit in Netzen und verteilten Systemen

15. November 2009



TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG

Stromchiffre RC4

Stromchiffre RC4

- Wird genutzt in...
 - SSL
 - WEP
- RC4 steht...
 - ursprünglich für „Ron's Code“
 - später offiziell „Rivest Cipher 4“

Security by Obscurity?

- Ursprünglich entworfen 1987 von Ron Rivest (RSA Security)
- Bis 1994 geheimer Code
- September 1994 taucht ein Code-Leak auf der "Cyberpunks" Mailingliste auf
- RSA Security hat den Algorithmus bis heute aufgrund von Patentproblemen nicht offiziell veröffentlicht

Was tut es?

- Der RC4 Algorithmus ist ein Pseudozufallsgenerator
- RC4 generiert einen pseudozufälligen Bytestream der gleichen Länge wie der zu verschlüsselnde Text
- Jedes Klartextbyte wird dann mit dem generierten Byte über XOR verknüpft und somit verschlüsselt

Erzeugung einer Substitutionstabelle

- 1 Schlüssel K definieren (von 40 bis 256 bits)
- 2 Lineares auffüllen der Tabelle:
 $S_0 := 0, S_1 := 1, \dots, S_{255} := 255$
- 3 Tabelle anpassen, k sei $|K|$
für i von 0 bis 255
 $j := (j + S_i + K_i \bmod k) \bmod 256$
vertausche S_i und S_j
nächstes i

Pseudozufälligen Stream erzeugen

1 Algorithmus gibt bei jeder Iteration ein Byte zurück
(1 Byte = 8 Bits = 2^8 Zustände = 256)

2 Stream erzeugen:

$i := 0$

$j := 0$

Während Stream erzeugt wird

$i := (i + 1) \bmod 256$

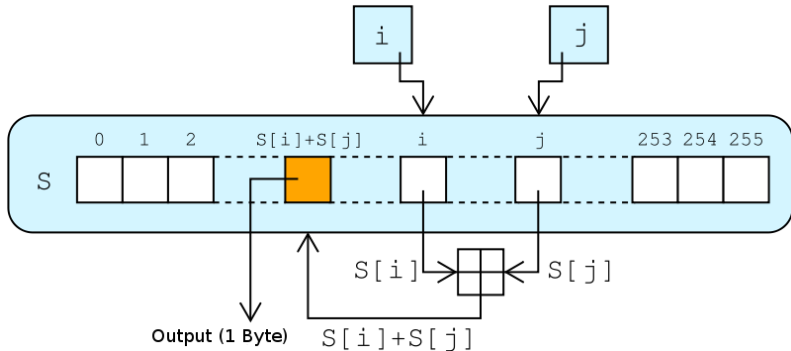
$j := (j + S_i) \bmod 256$

vertausche S_i und S_j

output $S_{(S_i+S_j \bmod 256)}$

weiter erzeugen

Grafik



Key schedule

Key schedule

- Bei einer schlechten Implementierung wird der Key K nach einer abgeschlossenen Verschlüsselung wieder für eine neue verwendet
- Somit entsteht der gleiche Pseudozufällige Stream
- Hat jemand nun ein Paar aus Klartext/Chiffretext kommt er zwar nicht auf den Schlüssel K, aber auf den Stream, denn:

$$k \text{ XOR } z = c$$

$$(k \text{ XOR } z) \text{ XOR } k = z$$

wobei k = Klartextbyte, z = Zufallsbyte, c = Chiffrebyte

Fluhrer, Mantin and Shamir attack

- 2001 Fluhrer, Mantin und Shamir finden eine neue Schwachstelle
- Die ersten Bytes sind nicht zufällig und lassen Rückschlüsse auf den Schlüssel K zu
- Bei schlechter Implementierung bei der der RC4 Schlüssel nicht komplett zufällig erzeugt wird und z.B. bei jeder Verschlüsselung der RC4 Schlüssel nur aus einer Aneinanderreihung von Langzeitschlüssel+Nonce besteht kann der Langzeitschlüssel durch die Beobachtung von Chiffretexten herausgefunden werden.
- Dies und die schlechte Implementierung des Initialisierungsvektors helfen WEP zu knacken

Weitere Schwachstellen

- 2005: Klein's Attack: Bei „Fluhrer, Mantin, and Shamir attack“ brauchte man 10.000.000 frames um WEP zu knacken, Klein's Implementierung braucht nur 40,000 frames um mit 50% Wahrscheinlichkeit, oder 85,000 frames um mit 95% Wahrscheinlichkeit den richtigen Schlüssel zu finden.

Fragen?