BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

BÁO CÁO MÔN HỌC

TÌM HIỂU VỀ HỆ MẬT RC4

Ngành: **Công nghệ thông tin**

Chuyên ngành: **Công nghệ phần mềm**

Giảng viên hướng dẫn : Ths. Trương Đông Nam

Sinh viên thực hiện :

Hồ Lê Minh Tâm.....1611250609

Võ Trường Thạnh.....1711060857

Lý Thanh Hùng 1611060590

Phạm Nguyễn Hoàng Vĩnh Phúc 1611060524

TP. Hồ Chí Minh, 2021

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC KÍ HIỆU, VIẾT TẮT

- RC4: Rivest Cipher 4 – ARCFOUR
- $\oplus$: XOR

CHƯƠNG 1: GIỚI THIỆU RC4

- RC4 (Rivest Cipher 4 – ARCFOUR) là một mật mã dòng (stream cipher).
- Được thiết kế bởi Ron Rivest (thuộc viện bảo mật RSA) vào năm 1987.
- RC4 được giới thiệu lần đầu vào năm 1994.
- RC4 thực hiện mã hoá từng byte một (hoặc các đơn vị lớn hơn tại một thời điểm). Đầu vào khoá là bộ tạo bit giả ngẫu nhiên tạo ra số luồng 8-bit không thể đoán trước được. Đầu ra gọi là dòng khoá, được kết hợp từng byte một với mất mã dòng bằng cách sử dụng phép toán XOR.
- Ưu điểm của RC4 nhằm vào tính chất đơn giản, tốc độ và dễ triển khai cùng phần mềm.
- RC4 từng trở nên phổ biến và là tiêu chuẩn bảo mật trong các giao thức SSL (1995), TLS (1999), bảo mật mạng không dây WEP (1997), bảo mật mạng không dây WPA (2003/2004).
- Sau đó, các lỗi bảo mật dần được khám phá trong quy trình mã hoá khiến cho RC4 không còn an toàn nữa. Các tổ chức bảo mật cũng như những công ty phần mềm lớn đã đưa ra khuyến nghị thay thế RC4 bằng thuật mã hoá khác.

Ưu điểm của RC4

- Dễ sử dụng, dễ thực thi, yêu cầu ít bộ nhớ hơn các mã dòng khác.
- RC4 hoạt động nhanh, do có tính chất đơn giản nên RC4 hoạt động hiệu quả hơn.
- RC4 hoạt động với các luồng dữ liệu lớn một cách nhanh chóng và dễ dàng.
- RC4 có thể sử dụng nhiều độ dài khoá khác nhau
- RC4 được sử dụng rộng rải chủ yếu vì độ dài khoá tuỳ chọn ngắn hơn là 40 bit.

Hạn chế của RC4

- Có nhiều lỗ hổng bảo mật nghiêm trọng đã được tìm thấy trong RC4, làm giảm độ tin cậy của RC4.
- Một trong 256 khoá có thể có một khoá yếu, mật mã dễ dàng bị phá nếu dựa vào khoá yếu này.
- Một khoá được sinh bởi RC4 chỉ nên được dùng một lần.

– RC4 không có bước xác thực, vì vậy những cuộc tấn công như Man in the Middle có thể xảy ra và người dùng mật mã RC4 có thể bị đánh cắp dữ liệu.

CHƯƠNG 2: CÁCH RC4 HOẠT ĐỘNG

2.1 TẠO KHOÁ K (Key-scheduling algorithm (KSA))

- RC4 tạo ra dòng khoá chứa các bít giả ngẫu nhiên (keystream).
- Giống như nhiều mã dòng khác, key nay được dùng để kết hợp giữa các kí tự mã hoá với dòng văn bản gốc để tạo ra đoạn văn bản được mã hoá. Việc giải mã cũng được thực hiện tương tự.
- Các khoá K được tạo ra ngẫu nhiên và do đó các kí tự mã hoá cũng được tạo ra một cách ngẫu nhiên.

- Để tạo ra các khoá K, mật mã RC4 thực hiện hai phần:
    - Tạo hoán vị của tất cả 256 byte (được kí hiệu S).
    - Dùng hai con trỏ 8-bit (kí hiệu i, j)

- Để khởi tạo khoá K, thuật toán lập khoá được dùng để khởi tạo hoán vị kí tự trong mảng "S".
- Keylength được định nghĩa là số byte trong khoá và có thể nằm trong phạm vi 1 ≤ keylength ≤ 256, thường là từ 5 đến 16, tương ứng với độ dài khoá là 40-128 bit.
- Đầu tiên, mảng "S" được khởi tạo để hoán vị các kí tự.
- Sau đó mảng "S" được xử lí 256 lần lặp để trộn các kí tự để tạo ra khoá K.

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap values of S[i] and S[j]
endfor
```

2.2 TẠO VĂN BẢN MÃ HOÁ (Pseudo-random generation algorithm (PRGA))

- Đối với mỗi lần lặp, PRGA sẽ sửa đổi trạng thái và xuất ra một byte của dòng khoá.
- Trong mỗi lần lặp lại, PRGA sẽ:
- Tăng i lên một đơn vị.
- Tìm giá trị thứ I trong mãng “S[i]”, và thêm giá trị của i vào j.
- Hoán vị giá trị của S[i] và S[j], sau đó tính tổng của S[i] và S[j] làm chỉ số để tìm nạp phần tử thứ ba của S.
- Sau đó dùng quy tắc XOR để loại trừ từng bit với byte tiếp theo của đoạn mã để tạo ra đoạn mã hoá hoặc đoạn giải mã.
- Mỗi phần tử của S được hoán đổi với phần tử khác ít nhất một lần sau mỗi 256 lần lặp.

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
endwhile
```

- Do đó, điều này tạo ra một dòng K [0], K [1], ... được XOR với bản rõ để thu được bản mã.
- Vậy ciphertext [l] = plaintext [l] $\oplus$ K [l].

CHƯƠNG 3: TÍNH BẢO MẬT CỦA RC4

- Không giống như một mật mã dòng hiện đại (chẳng hạn như trong eSTREAM), RC4 không có một nonce riêng biệt cùng với khóa. Điều này có nghĩa là nếu một khóa dài hạn duy nhất được sử dụng để mã hóa nhiều luồng, giao thức phải chỉ định cách kết hợp khóa nonce và khóa dài hạn để tạo khóa luồng cho RC4. Một cách tiếp cận để giải quyết vấn đề này là tạo một khóa RC4 "mới" bằng cách băm một khóa dài hạn với một nonce. Tuy nhiên, nhiều ứng dụng sử dụng RC4 chỉ đơn giản là ghép key và nonce; KSA yếu của RC4 có thể làm phát sinh các cuộc tấn công chính liên quan, như cuộc tấn công Fluhrer, Mantin và Shamir (nổi tiếng vì đã phá vỡ tiêu chuẩn WEP).

- Bởi vì RC4 là một mật mã dòng, nó dễ bị phá mã hơn so với mật mã khối thông thường. Nếu không được sử dụng cùng với mã xác thực tin nhắn mạnh (MAC), thì mã hóa dễ bị tấn công lật bit. Mật mã cũng dễ bị tấn công bằng mật mã luồng nếu không được triển khai đúng cách.

- Tuy nhiên, đáng chú ý là RC4, là một mật mã dòng, trong một khoảng thời gian, là mật mã phổ biến duy nhất miễn nhiễm với cuộc tấn công BEAST năm 2011 trên TLS 1.0. Cuộc tấn công khai thác một điểm yếu đã biết trong cách tạo chuỗi khối mật mã được sử dụng với tất cả các mật mã khác được hỗ trợ bởi TLS 1.0, tất cả đều là mật mã khối.

- Vào tháng 3 năm 2013, có những kịch bản tấn công mới được đề xuất bởi Isobe, Ohigashi, Watanabe và Morii, cũng như AlFardan, Bernstein, Paterson, Poettering và Schuldt sử dụng các thống kê các chuỗi K có khả năng lặp lại trong bảng khóa RC4 để khôi phục văn bản gốc với số lượng lớn mã hóa TLS.

- Việc sử dụng RC4 trong TLS bị cấm bởi RFC 7465 được xuất bản vào tháng 2 năm 2015.

CHƯƠNG 4: ỨNG DỤNG CỦA RC4

- RC4 được sử dụng phổ biến trong:
- Các giao thức Lớp cổng bảo mật (SSL).
- Bảo mật lớp truyền tải (TLS).
- Tiêu chuẩn mạng LAN không dây IEEE 802.11.
- Giao thức bảo mật Wi-Fi WEP (Giao thức tương đương không dây).
- Bảo mật WPA.
- BitTorrent protocol encryption.
- Microsoft Office XP.
- Microsoft Point-to-Point Encryption.
- Secure Shell.
- Remote Desktop Protocol.
- Kerberos.
- SASL Mechanism Digest-MD5.
- Gpcode.AK (virus máy tính vào đầu 6/2008 trên hệ điều hành window. Virus tấn công mã hoá dữ liệu trên máy tính của nạn nhân bằng cách sử dụng thuật toán mã hoá RC4 và RSA-1024).
- PDF.
- Skype.

CHƯƠNG 5: CÁC BIẾN THỂ CỦA RC4

5.1 RC4A

- RC4A là một biến thể từ RC4 gốc.
- RC4A được đề xuất bởi Souradyuti Paul và Bart Preneel.

- RC4A sử dụng hai mảng trạng thái S1 và S2, và hai chỉ mục j1 và j2. Mỗi khi i được tăng lên, hai byte được tạo ra:
- Đầu tiên, thuật toán RC4 cơ bản được thực hiện bằng S1 và j1, nhưng ở bước cuối cùng, S1 [i] + S1 [j1] được tra cứu trong S2.
- Sau đó, hoạt động được lặp lại (không tăng i lần nữa) trên S2 và j2, và S1 [S2 [i] + S2 [j2]] là đầu ra.

Thuật toán biểu diễn:

```
i := 0
j1 := 0
j2 := 0
while GeneratingOutput:
   i := i + 1
   j1 := j1 + S1[i]
   swap values of S1[i] and S1[j1]
   output S2[S1[i] + S1[j1]]
   j2 := j2 + S2[i]
   swap values of S2[i] and S2[j2]
   output S1[S2[i] + S2[j2]]
endwhile
```

- Mặc dù thuật toán yêu cầu cùng một số lượng tính toán trên mỗi byte đầu ra, nhưng có độ tối ưu lớn hơn RC4, cung cấp khả năng cải thiện tốc độ.
- Mặc dù mạnh hơn RC4, thuật toán này cũng đã bị tấn công, với Alexander Maximov và một nhóm từ NEC đang phát triển các cách để phân biệt đầu ra của nó với một chuỗi kí tự ngẫu nhiên.

5.2 VMPC

- Thành phần hoán vị được sửa đổi khác nhau (VMPC) là một biến thể RC4 khác. Nó sử dụng quy tắc sinh khoá K tương tự như RC4, với j: = S [(j + S [i] + key [i mod keylength]) mod 256] lặp lại 3 × 256 = 768 lần thay vì 256 và có thêm 768 lần lặp tùy chọn để kết hợp một vectơ ban đầu.
- Thuật toán này cũng đã bị tấn công tương tự như RC4A và có thể được phân biệt trong phạm vi 2^{38} -byte đầu ra
- Chức năng tạo đầu ra hoạt động như sau:

```
i := 0
while GeneratingOutput:
   a := S[i]
   j := S[j + a]

   output S[S[S[j] + 1]]
   Swap S[i] and S[j]
        (b := S[j]; S[i] := b; S[j] := a))
   i := i + 1
endwhile
```

5.3 RC4+

- RC4+ là phiên bản sửa đổi của RC4 với quy tắc sinh khoá K phức tạp hơn (phức tạp hơn khoảng ba lần so với RC4 hoặc giống như RC4-drop512) và một chức năng đầu ra phức tạp hơn thực hiện bốn lần tra cứu bổ sung trong mảng “S”.
- Cho mỗi byte đầu ra, chậm hơn khoảng 1,7 lần so với RC4 cơ bản.

```
while GeneratingOutput:
   i := i + 1
   a := S[i]
   j := j + a

   Swap S[i] and S[j]
        (b := S[j]; S[j] := S[i]; S[i] := b;)
   c := S[i<<5 ⊕ j>>3] + S[j<<5 ⊕ i>>3]
   output (S[a+b] + S[c⊕0xAA]) ⊕ S[j+b]
endwhile
```

- Thuật toán này hiện vẫn chưa bị phá thành công.

5.4 Spritz

- Vào năm 2014, Ronald Rivest đã có thông báo về một bản thiết kế lại cập nhật có tên là Spritz.
- Một trình tăng tốc phần cứng của Spritz đã được xuất bản trong Secrypt, 2016 và cho thấy rằng do nhiều lệnh gọi lồng nhau được yêu cầu để tạo ra các byte đầu ra.
- Spritz thực hiện khá chậm so với các hàm băm khác như SHA-3 và những thuật toán RC4 khác.

Thuật toán biểu diễn:

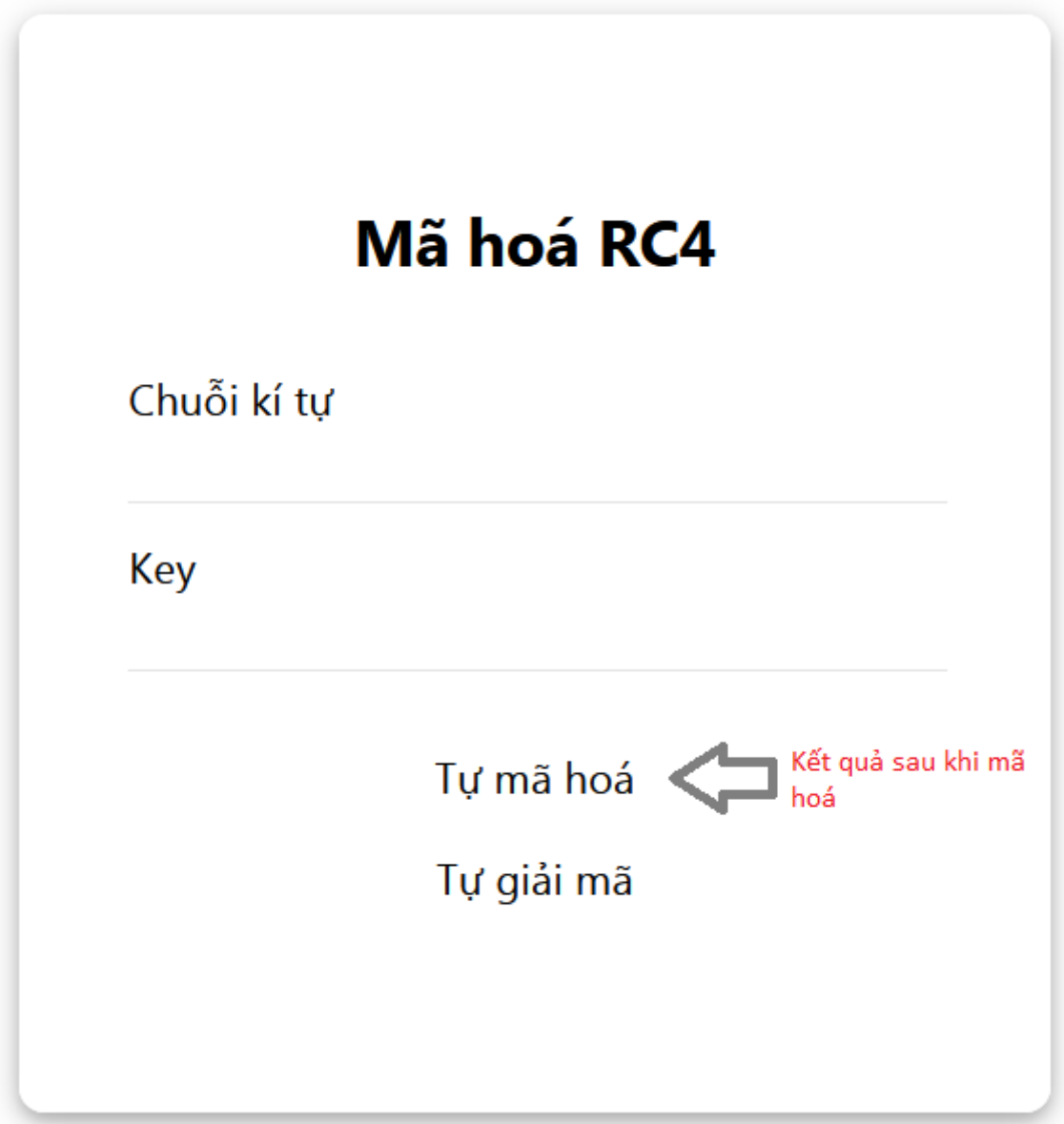
```
while GeneratingOutput:
    i := i + w
    j := k + S[j + S[i]]
    k := k + i + S[j]
    swap values of S[i] and S[j]
    output z := S[j + S[i + S[z + k]]]
endwhile
```

- Giá trị w, tương đối nguyên tố với kích thước của mảng S. Vì vậy, sau 256 lần lặp của vòng lặp bên trong này, giá trị i (tăng lên bởi w mỗi lần lặp) đã nhận tất cả các giá trị từ 0 ... 255 và mọi byte trong mảng S đã được hoán đổi ít nhất một lần.
- Giống như các hàm khác, Spritz có thể được sử dụng để xây dựng hàm băm mật mã, trình tạo bit ngẫu nhiên xác định (DRBG), thuật toán mã hóa hỗ trợ mã hóa được xác thực với dữ liệu liên quan (AEAD), …
- Năm 2016, Banik và Isobe đề xuất một cuộc tấn công có thể phân biệt Spritz với tập nhiễu thông tin ngẫu nhiên.

CHƯƠNG 6: THỰC HÀNH CÙNG RC4

- Bài thực hành ứng dụng thuật toán RC4 của nhóm chúng em.
- Link bài thực hành: https://orangefoxie.github.io/rc4demo.github.io/

6.1 GIAO DIỆN DEMO

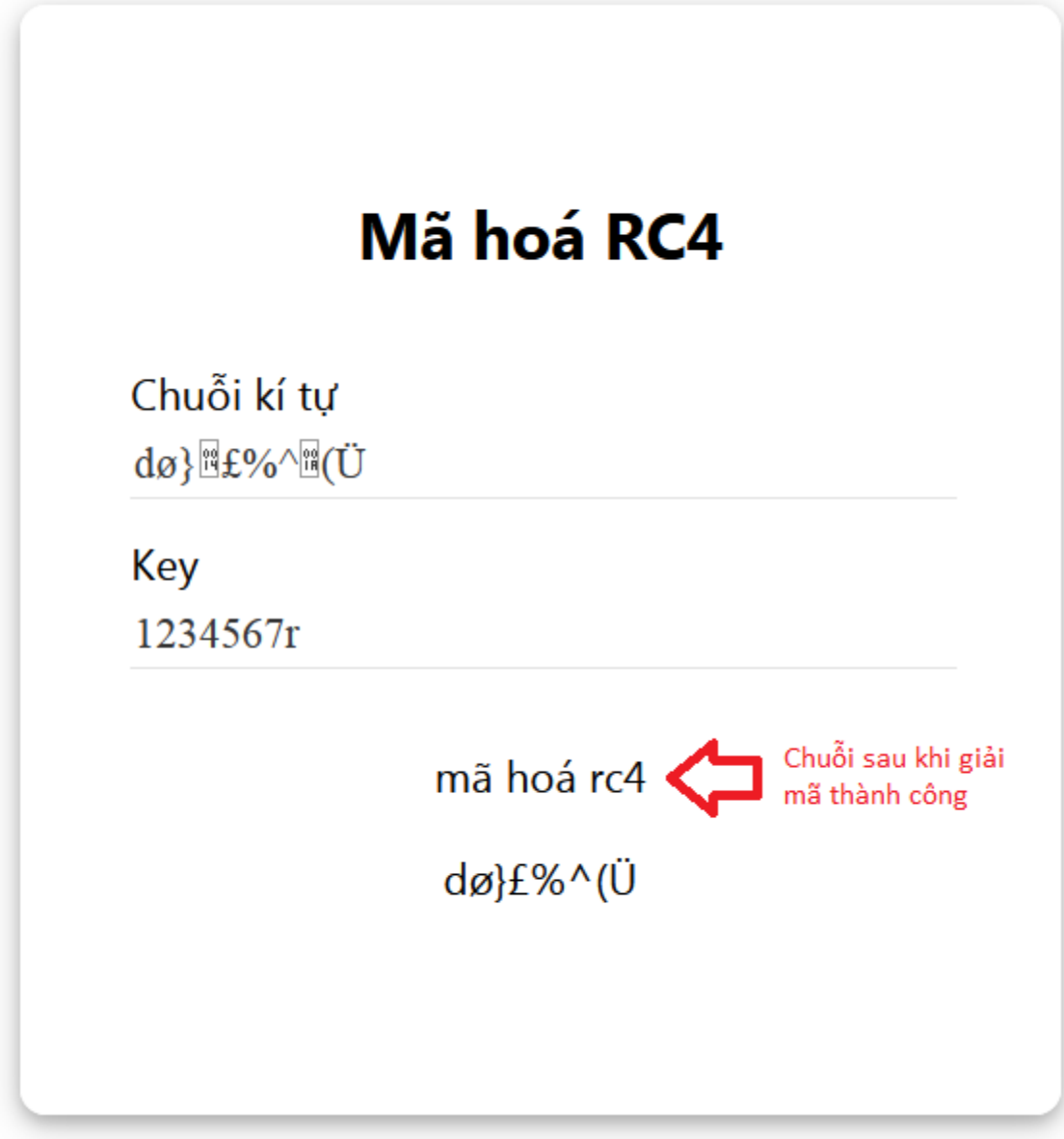


Hình 6.1 Giao diện demo

6.2 KHI NHẬP CHUỖI KÍ TỰ CẦN MÃ HOÁ

Hình 6.2 Chuỗi sau khi được mã hoá thành công

6.3 KHI NHẬP CHUỖI KÍ TỰ CẦN GIẢI MÃ



Hình 6.3 Chuỗi kí tự sau khi được giải mã thành công

CHƯƠNG 7: KẾT QUẢ THU ĐƯỢC

7.1 NHÓM ĐÃ ĐẠT ĐƯỢC

- Trong quá trình thực hiện đề tài, nhóm chúng em đã được tìm hiểu về thuật toán mã hoá RC4.
- Bên cạnh đó nhóm còn được tìm hiểu về cách hoạt động cũng như tìm được cách để làm demo.
- Ngoài ra nhóm chúng em còn biết được thêm những ứng dụng hữu ích của RC4.

7.2 NHỮNG ĐIỀU CẦN CẢI THIỆN

- Bản demo vẫn còn những thiếu sót trong tính toán dẫn đến một số đoạn văn bản sau khi mã hoá sẽ có một số kí tự bị lỗi không thể hiển thị đúng.
- Đôi lúc vì mã khoá K phức tạp nên việc giải mã của bản demo cũng gặp sai sót trong tính toán, dẫn đến việc giải mã không thành công do văn bản giải mã khác với văn bản gốc.

CHƯƠNG 8: TÀI LIỆU THAM KHẢO

- https://en.wikipedia.org/wiki/RC4
- https://www.geeksforgeeks.org/what-is-rc4-encryption/
- https://www.geeksforgeeks.org/rc4-encryption-algorithm/
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjYipjl953xAhUqE6YKHeUrDRkQFjADegQIAhAE&url=https%3A%2F%2Fwww.encryptionconsulting.com%2Feducation-center%2Fwhat-is-rc4%2F&usg=AOvVaw2IBafRvMLU0uylFUsp83YW
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjYipjl953xAhUqE6YKHeUrDRkQFjAHegQIBBAD&url=https%3A%2F%2Fpaginas.fe.up.pt%2F~ei10109%2Fca%2Frc4.html&usg=AOvVaw3nRt5vx7MCiV8WYh02o0ni