Practical Session 2 – RC4
Criptografía – Grupo 2
Facultad de Ingeniería
Departamento de Computación

As we discussed on previous sessions, RC4 [1] (Rivest Cipher 4 also known as ARC4 or ARCFOUR) is a stream cipher famous for being very simple and fast at software level.
Various vulnerabilities have been found in RC4, making it vulnerable especially when the beginning of the output keystream is not discarded, or when nonrandom or related keys are used. For this reason, in 2015 IETF has published RFC 7465 to prohibit the use of RC4 in TLS (a very important protocol used by Internet browsers to secure communications) [2].
RC4 generates a pseudorandom stream of bits (called keystream). As with any stream cipher, these can be used for encryption by combining it with the plaintext using bit-wise exclusive-or. Decryption is performed the same way (since exclusive-or with given data is an involution).
To generate the *keystream*, the cipher makes use of a secret internal state which consists of two parts:

1. A permutation of all 256 possible bytes (denoted "S" below).
2. Two 8-bit index-pointers (denoted "i" and "j").

The permutation is initialized with a variable length key, typically between 40 and 256 bits, using the *key-scheduling* algorithm (KSA). Once this has been completed, the stream of bits is generated using the *pseudo-random generation algorithm* (PRGA). Then, this stream of bits is x-ored with the given plain text.

Key-scheduling algorithm (KSA)

The key-scheduling algorithm is used to initialize the permutation in the array "S". "keylength" is defined as the number of bytes in the key and can be in the range $1 \leq \text{keylength} \leq 256$, typically between 5 and 16, corresponding to a key length of 40 – 128 bits. First, the array "S" is initialized to the identity permutation. S is then processed for 256 iterations in a similar way to the main PRGA, but also mixes in bytes of the key at the same time.

```
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
```

Pseudo-random generation algorithm (PRGA)

The output byte is selected by looking up the values of S(i) and S(j), adding them together modulo 256, and then using the sum as an index into S; S(S(i) + S(j)) is used as a byte of the key stream, K.

For as many iterations as are needed, the PRGA modifies the state and outputs a byte of the keystream. In each iteration, the PRGA increments *i*, looks up the *ith* element of S, S[*i*], and adds that to *j*, exchanges the values of S[*i*] and S[*j*], and then uses the sum S[*i*] + S[*j*] (modulo 256) as an index to fetch a third element of S, (the keystream value K below) which is XORed with the next byte of the message to produce the next byte of either ciphertext or plaintext. Each element of S is swapped with another element at least once every 256 iterations.

```
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
```

Test Vectors

These test vectors [3] are not official, but very useful to test your own RC4 implementation. The keys and plaintext are ASCII, the keystream and ciphertext are in hexadecimal.

Key	Keystream	Plaintext	Ciphertext
Key	EB9F7781B734CA72A719...	Plaintext	BBF316E8D940AF0AD3
Wiki	6044DB6D41B7...	pedia	1021BF0420
Secret	04D46B053CA87B59...	Attack at dawn	45A01F645FC35B383552544B9BF5

Implementation

1. Make your individual submission on Alphagrader your RC4 implementation in the programming language of your choice. The testing cases are the ones presented on the table above.

References

[1] Schneier, B., “Applied Cryptography: Protocols, Algorithms, and Source Code in C”, 2nd Edition, 1996.
[2] A. Popov, “RFC 7465 - Prohibiting RC4 Cipher Suites”, February 2015.
[3] https://en.wikipedia.org/wiki/RC4#Test_vectors