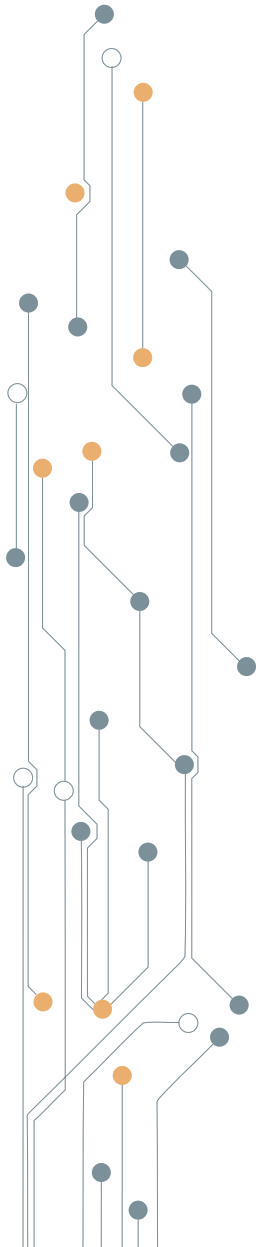
Algoritmos de cifra simétrica en flujo

Telefónica EDUCACIÓN DIGITAL

Índice



1. Esquema y fundamentos de la cifra en flujo

El texto en claro se cifrará bit a bit, o byte a byte, con una secuencia de bits o bytes que forman la clave, conocida como secuencia de clave S_i. No se trata de una clave cuyo valor es único y que se usa durante el proceso de cifra como se usará en la cifra en bloques, sino que ésta va cambiando. Por este mismo motivo, es recomendable que, entre otras cosas, dicha secuencia de clave tenga al menos tantos bits como el mensaje que se va a cifrar, para que no sea periódica y, por tanto, no dé pistas al criptoanalista.

La figura 5.1 muestra el esquema de un sistema de cifra en flujo.

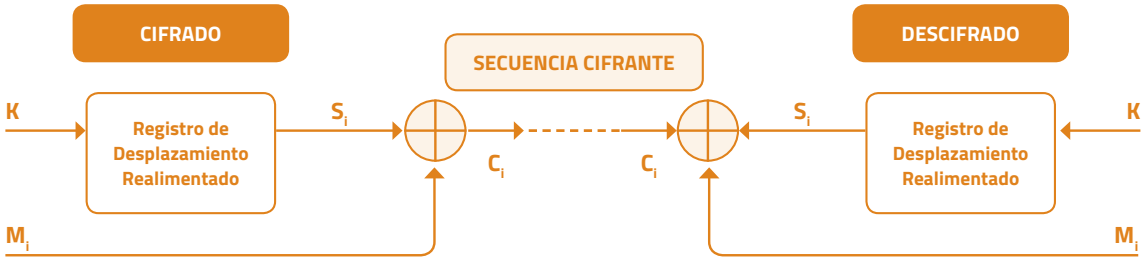


Figura 4.1. Clasificación de los sistemas de cifra moderna.

Por lo tanto, estos sistemas de cifra se resumen en las siguientes tres características:

a) Un algoritmo de cifrado y descifrado con la función XOR.

b) Una secuencia cifrante binaria y pseudoaleatoria S_i que se obtiene a partir una clave secreta K, compartida por emisor y receptor, y un algoritmo generador de esa secuencia con registros de desplazamiento.

c) Una misma operación de cifrado que de descifrado debido el carácter involutivo de la función XOR.

2. Secuencias cifrantes y postulados de Golomb

Toda la fortaleza de un sistema de cifra en flujo residirá en las características de esa secuencia de clave Si, en tanto el algoritmo de cifra es muy básico, una simple función XOR. No se utilizan en este caso técnicas de difusión ni de confusión, que sin embargo sí son habituales y necesarias en sistemas de cifra simétrica en bloques. Una de las condiciones más importantes que deben cumplir estas secuencias para que puedan ser consideradas como pseudoaleatorias, son las que tienen que ver con los postulados de Golomb que se indican a continuación.

Primer postulado de Golomb

Los bits 0 y 1 de la secuencia cifrante deberán ser equiprobables. Por lo tanto la secuencia deberá tener mitad de unos y mitad de ceros, aunque lo normal es que tengan un uno más que ceros puesto que las secuencias que se utilizan en la práctica serán impares, al provenir de un sistema lineal. Esto quiere decir que la probabilidad de encontrarse con un 0 o con un 1 en cualquier posición de esa secuencia, deberá ser del 50%.

Por ejemplo, cumple con este primer postulado de Golomb la secuencia K_1 = 00111110001101110101000010010 11 (16 uno y 15 ceros), pero no lo cumple la secuencia K_2 = 110101011001000 (7 unos y 8 ceros).

Segundo postulado de Golomb

La equiprobabilidad de unos y ceros debería seguir manifestándose, independientemente de los bits que hayamos leído o conocido con anterioridad en dicha secuencia. Esto es, si hemos recibido la cadena 1000, la probabilidad de que el quinto bit de esa cadena sea un 0 es decir 10000 o bien un 1, es decir 10001, seguirá siendo del 50%. La secuencia anterior K_1 cumple con este segundo postulado de Golomb y en este ejemplo se marca con los bits en cuestión en negrita: K_1 = 00111110001101110101000010010 11.

Lo importante de este segundo postulado de Golomb es que, si se cumple, significa que la secuencia Si pasará por todos sus estados (restos) posibles y por tanto la secuencia será máxima, que es una de las cosas que se buscan. En el caso de la secuencia anterior K1, se trata de un registro con 5 celdas, que entrega un periodo igual a $2^5-1 = 31$ y que, excepto el resto 00000 que veremos está prohibido porque el sistema es lineal, pasa por todos los demás restos, desde 00001, 00010,... hasta 11110, 11111.

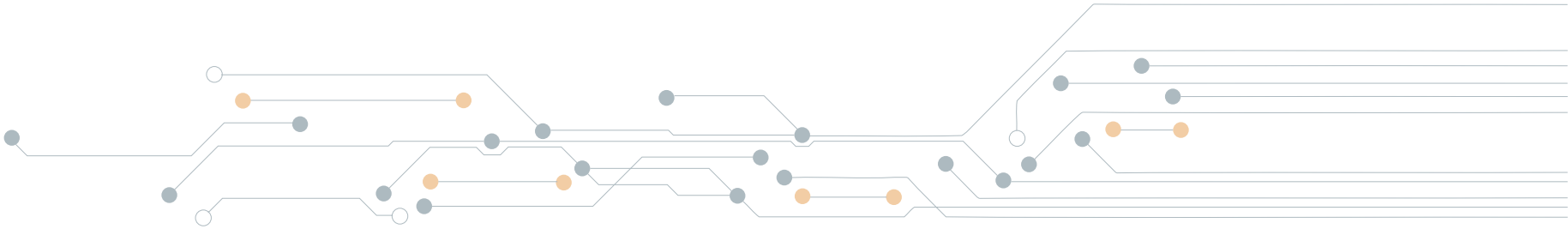
Para que esto suceda, deberá cumplirse que las rachas de ceros y de unos, esto es un conjunto de ceros entre dos unos, y un conjunto de unos entre dos ceros, siga una distribución geométrica. Por ejemplo, una secuencia máxima de 63 bits ($2^6 - 1$) conj un registro de 6 celdas, deberá tener:

- 16 rachas de longitud uno: ocho del tipo 010 y ocho del tipo 101.
- 8 rachas de longitud dos: cuatro del tipo 0110 y cuatro del tipo 1001.
- 4 rachas de longitud tres: dos del tipo 01110 y dos del tipo 10001
- 2 rachas de longitud cuatro: una del tipo 011110 y una del tipo 100001.
- 1 racha de longitud cinco (n-1) sólo de ceros: del tipo 1000001.
- 1 racha de longitud seis (n) sólo de unos: del tipo 01111110.

Tercer postulado de Golomb

El tercer postulado dice que la autocorrelación fuera de fase AC(k) entre la secuencia original y dicha secuencia desplazada k bits deberá ser contante. Se define AC(k) = (Aciertos – Fallos)/T, siendo Aciertos el número de bits que coinciden en igual posición entre las dos cadenas y Fallos el número de bits que no coinciden, con T igual al periodo de la secuencia.

Si esto se cumple, significa que la probabilidad de éxito para atacar a la secuencia cifrante analizando un conjunto de bits en una parte de ella o bien en otra, será la misma. Es decir, no habrá ventajas al estudiar la secuencia en una zona u otra.



3. Registros de desplazamiento

Un registro de desplazamiento es un circuito digital secuencial que está compuesto por n celdas que almacenan un 1 o un 0, y que al ritmo que marca un reloj, los bits se van desplazando a la celda contigua y por tanto entregando un bit de salida. Ese bit de salida, que se pierde y que en nuestro caso se usa como parte de la secuencia S_i de clave para cifrar, se realimenta hacia el sistema mediante una función que puede ser lineal o no lineal y en la que intervienen, además, los valores que contienen en ese momento una o más celdas.

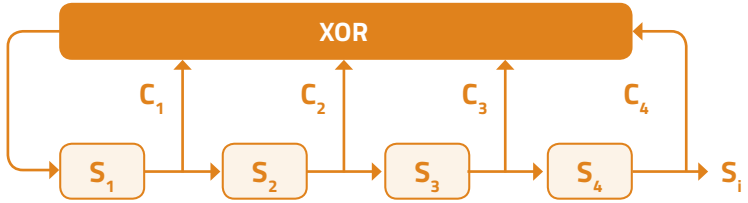


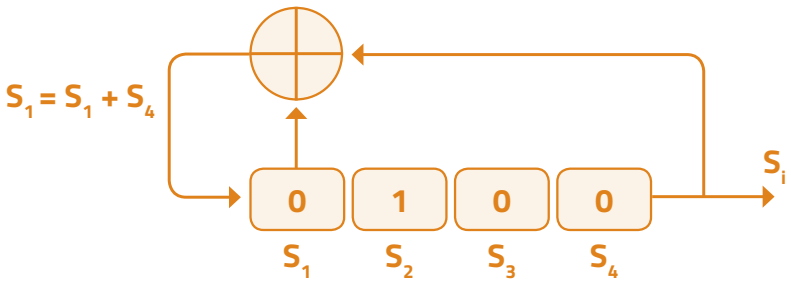
Figura 5.2. Esquema de la cifra en flujo.

La figura 5.2 muestra un registro lineal (sólo una función xor como realimentación) conocido como LFSR (*Linear Feedback Shift Register*) de 4 celdas, donde están conectadas a ese or exclusivo la celda 4 (al ser la última, lo estará siempre), la celda 3 y la celda 1, no así la celda 2.

La disposición de conexión de celdas al xor se asocia a un polinomio, en el caso de la figura 2 al polinomio $x^4 + x^3 + x + 1$. Dicho polinomio marcará si el registro en cuestión es de los que entregan un periodo máximo, que es lo que nos interesa y si cumplen, además, con los tres postulados de Golomb. Dichos polinomios se conocen como primitivos.

Como los bits de la secuencia Si del registro de la figura 2 salen hacia la derecha, los bits de dicha secuencia serán $S_4S_3S_2S_1S_5S_6$... etc., es decir, primero se transmite la semilla al revés y, a continuación, los bits siguientes según evoluciona el registro.

La figura 5.3 muestra un registro LFSR primitivo de grado 4 y cómo éste evoluciona con una semilla $S_1S_2S_3S_4 = 0100$.



t	Registro	S_i
0	0100	0
1	0010	0
2	0001	1
3	1000	0
4	1100	0
5	1110	0
6	1111	1
7	0111	1

t	Registro	S_i
8	1011	1
9	0101	1
10	1010	0
11	1101	1
12	0110	0
13	0011	1
14	1001	1
15	0100	

Figura 5.3. LFSR primitivo de 4 celdas con semilla 0100 y evolución del registro para generar Si.

4. Ataque a secuencias cifrantes máximas

Con LFSRs primitivos se consiguen m-secuencias, esto es, secuencias con un periodo máximo de valor $T = 2^n - 1$. Si n es un valor del orden de las centenas de bits, el periodo en cuestión tendrá más de mil billones de terabytes, un valor enorme. No existe en el mundo mensaje ni documento de ese tamaño.

No obstante, el hecho de que la secuencia cifrante S_i sea máxima, significa que ésta pasa por todos sus estados o restos. Esto hace que dicha secuencia sea predecible y, por lo tanto, no sea válida para para la cifra. Ello se debe a que esto permite que mediante un ataque, conocido como de Berlekamp-Massey, sea posible romper una secuencia de 2n bits conociendo tan sólo 2^n bits consecutivos de dicha cadena. Es decir, una secuencia de 2^{25} = 33.554.432 bits, más de 33 millones de bits, podría romperse conociendo tan sólo 2*25 = 50 bits consecutivos en cualquier lugar de esa cadena. Por tal motivo, nunca se usará un único LFSR para obtener la secuencia cifrante, sino dos o más de ellos, conectados mediante una función lógica, preferentemente una puerta xor como se muestra en la figura 5.4.

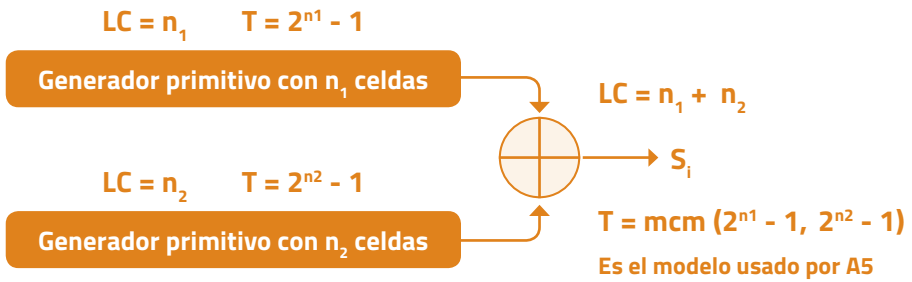


Figura 5.4. Registros LFRS conectados a través de una puerta xor (*LC Linear Complexity*).

El generador de la figura 4 no será predecible, tendrá un periodo igual al mínimo común múltiplo de los periodos de cada LFSR y cumplirá en algunos casos sólo con el primer postulado de Golomb, pero sí será un buen generador de secuencia cifrante.

5. Los algoritmos A5 y RC4

El algoritmo A5

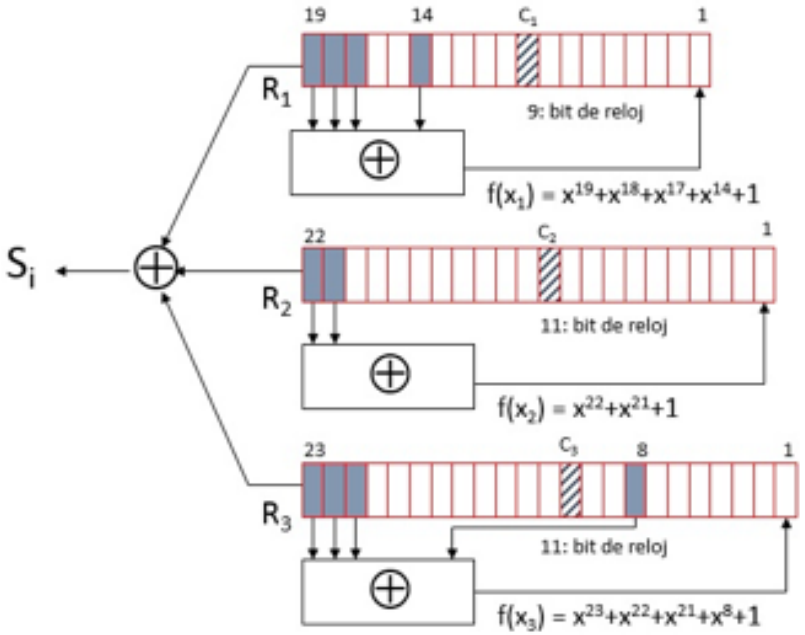


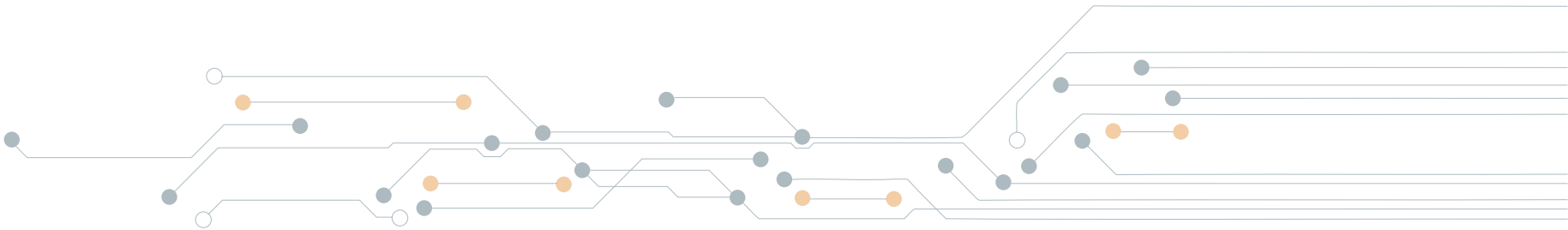
Figura 5.5. Algoritmo de cifra A5.

Creado en 1987 para GSM Global System for Mobile communications, el uso habitual de este algoritmo lo encontramos en el cifrado del enlace entre el abonado y la central de un teléfono móvil o celular tipo GSM. Existen dos versiones A5/1 y A5/2, este último mucho más débil y diseñado a propósito con esas características. Aunque el código fuente se oculta (un grave error), en 1999 se demuestra que el diseño del algoritmo es muy débil, especialmente por su corta longitud de clave o semilla. El sistema A5/1 contaba en 1999 con cerca de 130 millones de usuarios en Europa y otros 100 millones de usuarios en el resto del mundo, y sucumbe por primera vez en diciembre de ese año ante un ataque con texto en claro conocido, realizado por Alex Biryukov, Adi Shamir y David Wagner. Ante estos ataques, se sustituye A5/1 por A5/3, algoritmo llamado Kasumi y que en realidad es un cifrador de bloque y no de flujo. Actualmente en telefonía móvil es más común Snow 3G.

El algoritmo A5 consta de 3 registros LFSR primitivos de 19, 22 y 23 celdas. Por lo tanto, la semilla del sistema es igual a la suma de estas cantidades, 64 bits, un valor muy bajo y desaconsejable para la fecha de su creación. Para que la salida de bits en el XOR de los tres registros sea menos predecible, se usa una función denominada mayoría y que consiste en lo siguiente: el bit de la celda 9 del registro R1 y los bits de las celdas 11 de los registros R2 y R3 pueden pasar durante la generación de la secuencia por estos ocho estados posibles: 000, 001, 010, 011, 100, 101, 110, 111. Así, en función del valor que tenga el bit en cada una de esas tres celdas, los correspondientes registros desplazarán o no de acuerdo a la función mayoría que se muestra en la figura 6. Por tanto, los registros que tienen su bit dentro de la mayoría de ceros o de unos (F) desplazarán y, en cambio, el registro que tenga su bit fuera de esa mayoría, no desplazará. Siempre desplazarán al menos dos.

C_1	C_2	C_3	
0	0	0	F = 0: Desplazan todos
0	0	1	F = 0: No desplaza R_3
0	1	0	F = 0: No desplaza R_2
0	1	1	F = 1: No desplaza R_1
1	0	0	F = 0: No desplaza R_1
1	0	1	F = 1: No desplaza R_2
1	1	0	F = 1: No desplaza R_3
1	1	1	F = 1: Desplazan todos

Figura 5.5. Algoritmo de cifra A5.



El algoritmo RC4

RC4 es el acrónimo de Ron's Code 4, un algoritmo de cifra en flujo byte a byte diseñado por Ron Rivest en 1987. Su uso más habitual se encontraba hasta el año 2014 en el protocolo SSL/TLS, especialmente al utilizar Internet Explorer, aunque se usa también en WEP Wired Equivalent Privacy y en WPA Wi-Fi Protected Access, para el cifrado en redes inalámbricas. Su popularidad estaba basada fundamentalmente por su alta velocidad y la simplicidad de diseño, tanto hardware como software, siendo diez veces más rápido que el algoritmo DES, el estándar mundial de cifra simétrica hasta finales de la década de los noventa, y el doble de rápido que AES, el actual estándar.

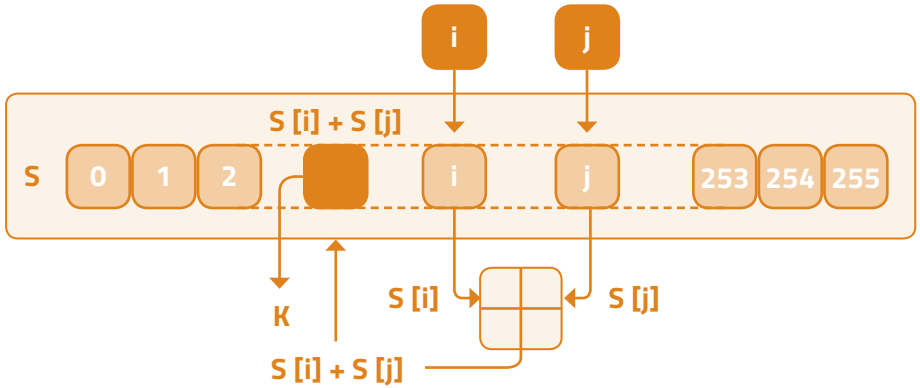


Figura 5.7. Esquema del algoritmo RC4.

```
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
```

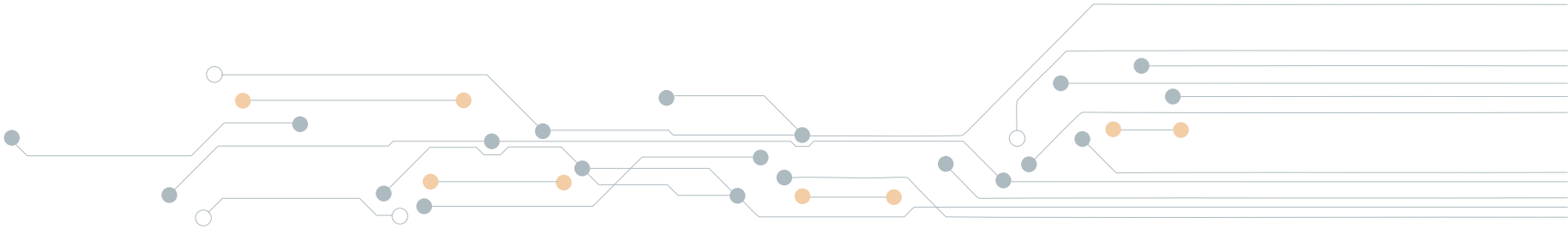
KSA
Key Scheduling Algorithm
Inicialización del array S
1 < keylength < 256 bytes
5 ≤ Típico ≤ 16 bytes (40 - 128 bits)

```
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
```

PRGA
Pseudo-Random Generation Algorithm
Mientras sea necesario, modifica el estado y entrega el byte de clave K.
El byte de salida K se obtiene sumando S[i] + S[j] mod 256

Figura 5.8. Rutinas KSA y PRGA en RC4.

El código del algoritmo RC4 era secreto hasta el año 1994 en que aparece una descripción de su funcionamiento en un post anónimo enviado a la lista de correo de Cipherpunks. Como es una marca registrada, las implementaciones no oficiales de RC4 se conocen con otros nombres como ARC4, ARCFOUR o Alleged-RC4. En febrero de 2015, la organización internacional IETF Internet Engineering Task Force recomienda la eliminación del cifrado RC4 en las conexiones TLS entre clientes y servidores, debido a una serie de vulnerabilidades críticas descubiertas, por lo que actualmente ha quedado prácticamente en desuso, en beneficio del AES.



Telefónica EDUCACIÓN DIGITAL