

2.3 Shift Register-Based Stream Ciphers (Cifrado de flujo basado en registro de desplazamiento)

2.3.1 Linear Feedback Shift Registers (LFSR) (Registros de desplazamiento con feedback lineal)

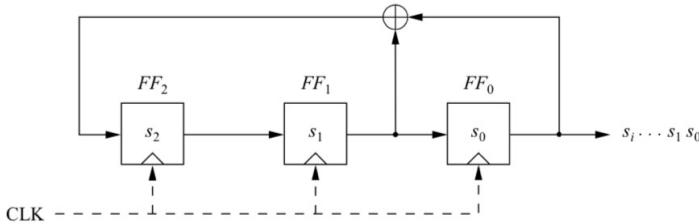
Un LFSR consiste de:

- flip-flops: elementos de almacenamiento cronometrados
- feedback path (ruta de retroalimentación)

- El grado de un LFSR es el número de flip-flops.
- la red de retroalimentación calcula el input para el último flip-flop como una XOR-suma de ciertos flip-flops en el registro de desplazamiento. (^{shift} register)

Ejemplo: LFSR simple

Consideramos un LFSR de grado 3 con flip-flops FF_2, FF_1, FF_0 con ruta de retroalimentación



Denotamos por s_i a los bits de estado interno los cuales se mueven uno hacia la derecha con cada tacto del reloj

El bit que se encuentra más hacia la derecha es el bit de salida actual.

El bit que se encuentra más a la izquierda se calcula en la ruta de retroalimentación, la cual es la suma XOR de algunos de los valores flip-flop en el periodo de tiempo previo

Como XOR es una operación lineal, a los circuitos se les llama registros de desplazamiento de retroalimentación lineal (linear feedback shift registers).

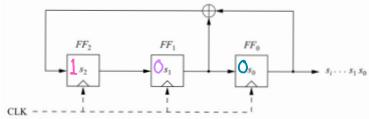
Ejemplo: Supongamos el estado inicial $s_2=1, s_1=0, s_0=0$.

clk	FF ₂	FF ₁	FF ₀ = s_i
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

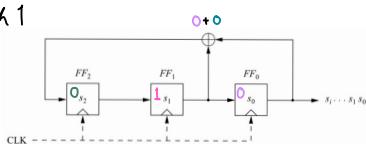
esta columna son las salidas del LFSR.

la tabla nos da una sucesión de los estados del LFSR.

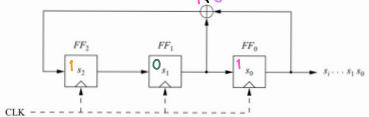
click 0



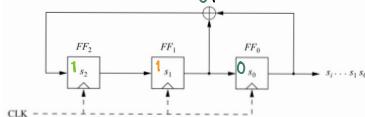
click 1



click 2



click 3



En este ejemplo podemos notar que el LFSR se empieza a repetir después de un ciclo de 6 clicks.

\Rightarrow la salida del LFSR tiene un periodo de longitud 7 de la forma
 0010111 0010111 0010111 ...

Hay una fórmula que determina el funcionamiento de este LFSR. Veamos cómo los bits de salida s_i son calculados asumiendo los bits de estado inicial s_0, s_1, s_2 :

$$s_3 \equiv s_1 + s_0 \pmod{2}$$

$$s_4 \equiv s_2 + s_1 \pmod{2}$$

$$s_5 \equiv s_3 + s_2 \pmod{2}$$

 \vdots

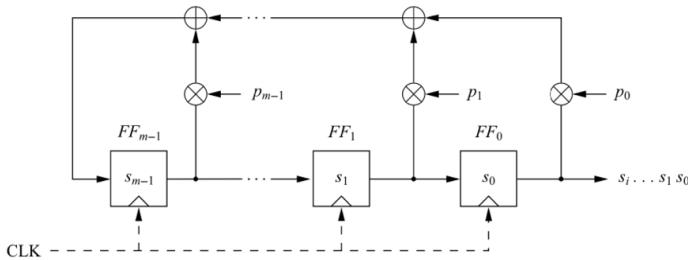
En general, el bit de salida se calcula como sigue

$$s_{i+3} \equiv s_{i+1} + s_i \pmod{2}$$

dond $i = 0, 1, 2, \dots$

Descripción matemática de un LFSR

Un LFSR general de grado m tiene la siguiente forma



tiene m flip-flops y m posibles lugares de retroalimentación (feedback) todas combinadas con la operación XOR.

Sin importar si un camino de feedback está activo o no, está definido

por los coeficientes de feedback p_0, p_1, \dots, p_{m-1} :

- Si $p_i = 1$ (interruptor cerrado) el feedback está activo
- Si $p_i = 0$ (interruptor abierto), el flip-flop de salida no es utilizado en el feedback.

Con esto obtenemos una descripción matemática

Si multiplicamos la salida del flip-flop i por su coeficiente p_i el resultado es

- el valor de salida, si $p_i = 1$ (corresponde a un switch cerrado)
- 0, si $p_i = 0$ (corresponde a un switch abierto).

Los valores de los coeficientes del feedback son cruciales para la sucesión de outputs producida por el LFSR.

Supongamos un LFSR con valores iniciales s_0, \dots, s_{m-1} .

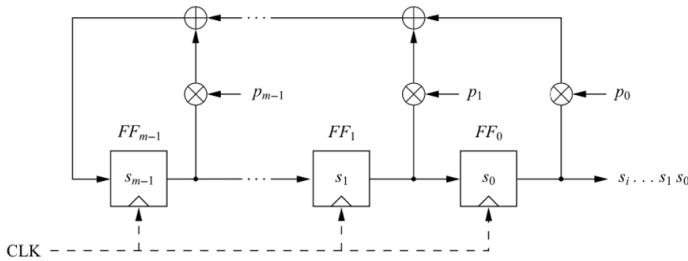
El siguiente bit de salida del LFSR s_m (que es también el input del flip-flop a la izquierda de todos) puede ser calculado como la XOR-suma de los productos de flip-flops de salida con sus coeficientes de feedback correspondientes:

$$S_m \equiv S_{m-1}p_{m-1} + \dots + S_1p_1 + S_0p_0 \pmod{2}$$

El siguiente LFSR de salida es entonces

$$S_{m+1} \equiv S_m p_{m-1} + \dots + S_2 p_1 + S_1 p_0 \pmod{2}$$

(ver el diagrama)



En general, la sucesión de bits de salida se puede describir como

$$S_{i+m} \equiv \sum_{j=0}^{m-1} p_j S_{i+j} \pmod{2}; \quad S_i, p_j \in \{0, 1\}, \quad i = 0, 1, 2, \dots \quad (2.1)$$

Los valores de salida de los LFSR están dados por una combinación de los valores de salida previos. A los LFSR a veces se les llama **recurrencias lineales**.

Observación

- Dado que se tiene un número finito de estados de recurrencia, la sucesión de salida se repite periódicamente.
- Un LFSR puede producir sucesiones de salida de diferentes longitudes (periodo) dependiendo de los coeficientes de feedback.

Teorema

La secuencia de longitud máxima generada por un LFSR de grado m tiene longitud $2^m - 1$.

(bosquejo)

Demostración todos los valores en la posición que se encuentran.

El estado de un LFSR está únicamente determinado por los m bits internos registrados.

Dado un estado, el LFSR determinísticamente asume su siguiente estado.

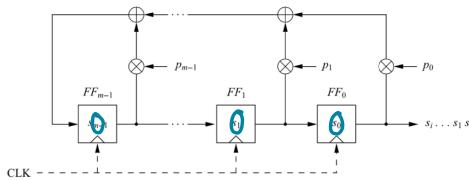
→ En el momento en el que un LFSR asume un estado previo, se comienza a repetir.

Como un estado de m bits puede asumir $2^m - 1$ estados no cero

\Rightarrow la longitud máxima de una sucesión de salida antes de comenzar a repetirse (el periodo máximo) es 2^{m-1} □

Nota: ¿por qué excluimos el estado cero?

Si un LFSR asume el estado cero \Rightarrow se queda atorado en dicho estado



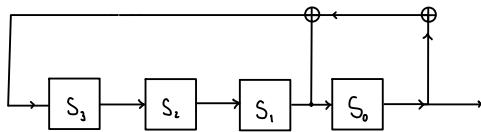
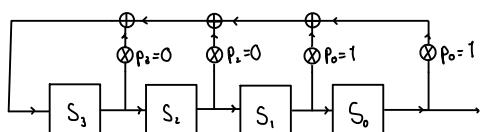
Observación

Solo ciertas configuraciones (p_0, \dots, p_{m-1}) producen longitudes máximas en un LFSR.

Ejemplos:

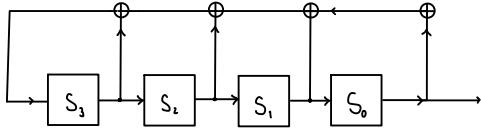
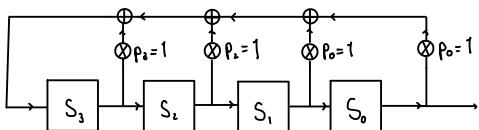
① LFSR con sucesión de salida de longitud máxima:

Dado un LFSR de grado $m=4$ y feedback path $(p_3=0, p_2=0, p_1=1, p_0=1)$ la sucesión de salida tiene periodo $2^m - 1 = 2^4 - 1 = 15$ (long máxima)



② LFSR de longitud no máxima

Dado un LFSR de grado $m=4$ y feedback path $(p_3=1, p_2=1, p_1=1, p_0=1)$ la sucesión de salida tiene periodo 5 (< 15) \Rightarrow no tiene longitud máxima.



Nota

- Regulamente los LFSR se especifican con polinomios.

Un LFSR con coeficientes feedback $(p_{m-1}, \dots, p_1, p_0)$ se representa por el polinomio

$$P(x) = x^m + p_{m-1}x^{m-1} + \dots + p_1x + p_0$$

- Los LFSR de mayor longitud tienen asociado (se representan por) un **polinomio primitivo** (un tipo de polinomio irreducible)

(0,1,2)	(0,1,3,4,24)	(0,1,46)	(0,1,5,7,68)	(0,2,3,5,90)	(0,3,4,5,112)
(0,1,3)	(0,3,25)	(0,5,47)	(0,2,5,6,69)	(0,1,5,8,91)	(0,2,3,5,113)
(0,1,4)	(0,1,3,4,26)	(0,2,3,5,48)	(0,1,3,5,70)	(0,2,5,6,92)	(0,2,3,5,114)
(0,2,5)	(0,1,2,5,27)	(0,4,5,6,49)	(0,1,3,5,71)	(0,2,93)	(0,5,7,8,115)
(0,1,6)	(0,1,28)	(0,2,3,4,50)	(0,3,9,10,72)	(0,1,5,6,94)	(0,1,2,4,116)
(0,1,7)	(0,2,29)	(0,1,3,6,51)	(0,2,3,4,73)	(0,11,95)	(0,1,2,5,117)
(0,1,3,4,8)	(0,1,30)	(0,3,52)	(0,1,2,6,74)	(0,6,9,10,96)	(0,2,5,6,118)
(0,1,9)	(0,3,31)	(0,1,2,6,53)	(0,1,3,6,75)	(0,6,97)	(0,8,119)
(0,3,10)	(0,2,3,7,32)	(0,3,6,8,54)	(0,2,4,5,76)	(0,3,4,7,98)	(0,1,3,4,120)
(0,2,11)	(0,1,3,6,33)	(0,1,2,6,55)	(0,2,5,6,77)	(0,1,3,6,99)	(0,1,5,8,121)
(0,3,12)	(0,1,3,4,34)	(0,2,4,7,56)	(0,1,2,7,78)	(0,2,5,6,100)	(0,1,2,6,122)
(0,1,3,4,13)	(0,2,35)	(0,4,57)	(0,2,3,4,79)	(0,1,6,7,101)	(0,2,123)
(0,5,14)	(0,2,4,5,36)	(0,1,5,6,58)	(0,2,4,9,80)	(0,3,5,6,102)	(0,37,124)
(0,1,15)	(0,1,4,6,37)	(0,2,4,7,59)	(0,4,81)	(0,9,103)	(0,5,6,7,125)
(0,1,3,5,16)	(0,1,5,6,38)	(0,1,60)	(0,4,6,9,82)	(0,1,3,4,104)	(0,2,4,7,126)
(0,3,17)	(0,4,39)	(0,1,2,5,61)	(0,2,4,7,83)	(0,4,105)	(0,1,127)
(0,3,18)	(0,3,4,5,40)	(0,3,5,6,62)	(0,5,84)	(0,1,5,6,106)	(0,1,2,7,128)
(0,1,2,5,19)	(0,3,41)	(0,1,63)	(0,1,2,8,85)	(0,4,7,9,107)	
(0,3,20)	(0,1,2,5,42)	(0,1,3,4,64)	(0,2,5,6,86)	(0,1,4,6,108)	
(0,2,21)	(0,3,4,6,43)	(0,1,3,4,65)	(0,1,5,7,87)	(0,2,4,5,109)	
(0,1,22)	(0,5,44)	(0,3,66)	(0,8,9,11,88)	(0,1,4,6,110)	
(0,5,23)	(0,1,3,4,45)	(0,1,2,5,67)	(0,3,5,6,89)	(0,2,4,7,111)	

$$1+3x+4x^2+5x^3+112x^4$$

Algunos ejemplos de polinomios primitivos

Hay 69, 273, 666 polinomios primitivos de grado $m=31$.

2.3.2 Known-Plaintext Attack Against Single LFSRs

LFSRs son lineales (relación lineal entre inputs y outputs)

Si usamos un LFSR como stream cipher la llave secreta K es el vector de coeficiente de feedback $(p_{m-1}, \dots, p_1, p_0)$.

Un ataque es posible si el atacante conoce un fragmento del texto plano y su correspondiente texto cifrado.



Poder: Puede intentar un gran número de posibles de m valores sin problema alguno.

Con estos $2m$ pares de bits de texto plano y texto cifrado Oscar reconstruye los primeros $2m$ Key stream bits:

$$s_i \equiv x_i + y_i \pmod{2}, \quad i = 0, 1, \dots, 2m-1$$

Objetivo: encontrar la llave dada por los coeficientes de feedback p_i .

Recordemos que tenemos una forma de describir la relación entre los bits desconocidos p_i y la llave:

$$s_{i+m} \equiv \sum_{j=0}^{m-1} p_j \cdot s_{i+j} \pmod{2}; \quad s_i, p_j \in \{0, 1\}, \quad i = 0, 1, 2, \dots$$

Observación:

- Obtenemos una ecuación diferente para cada valor de i .
- Las ecuaciones son linealmente independientes.

Dado esto, Óscar puede generar m ecuaciones para los primeros m valores de i :

$$\begin{aligned} i=0, \quad S_m &\equiv p_{m-1}S_{m-1} + \dots + p_1S_1 + p_0S_0 \pmod{2} \\ i=1, \quad S_{m+1} &\equiv p_{m-1}S_m + \dots + p_1S_2 + p_0S_1 \pmod{2} \\ \vdots & \vdots \\ i=m-1, \quad S_{2m-1} &\equiv p_{m-1}S_{2m-2} + \dots + p_1S_m + p_0S_{m-1} \pmod{2} \end{aligned}$$

Óscar tiene un sistema de m ecuaciones lineales con m incógnitas
⇒ puede encontrar la solución

Observación

- En cuanto Óscar conozca $2m$ bits de salida de un LFSR de grado m , los coeficientes p_i pueden encontrarse resolviendo un sistema de ecuaciones lineales.

⇒ Puede construir el LFSR e ingresar cualesquiera m bits de salida consecutivas que conozca

⇒ Puede correr el LFSR y producir la secuencia de salida completa

⇒ LFSR pos si solos son **extremadamente inseguros !!!**

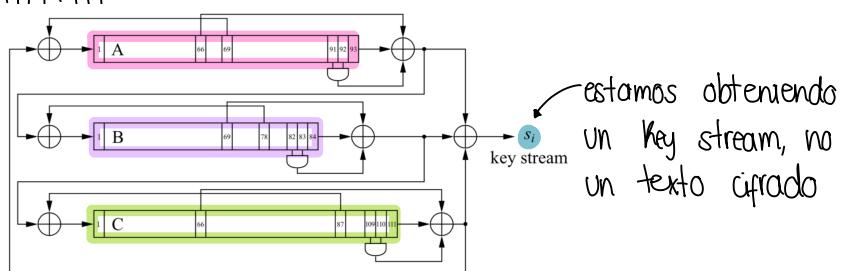
Existen stream ciphers que usan combinaciones de LFSRs para construir criptosistemas más seguros. El siguiente es un ejemplo de estos.

2.3.3 Trivium

- Relativamente nuevo
- Llave de 80-bits
- Basado en la combinación de 3 shift registers
- Hay componentes no lineares usados para derivar el output de cada registro.

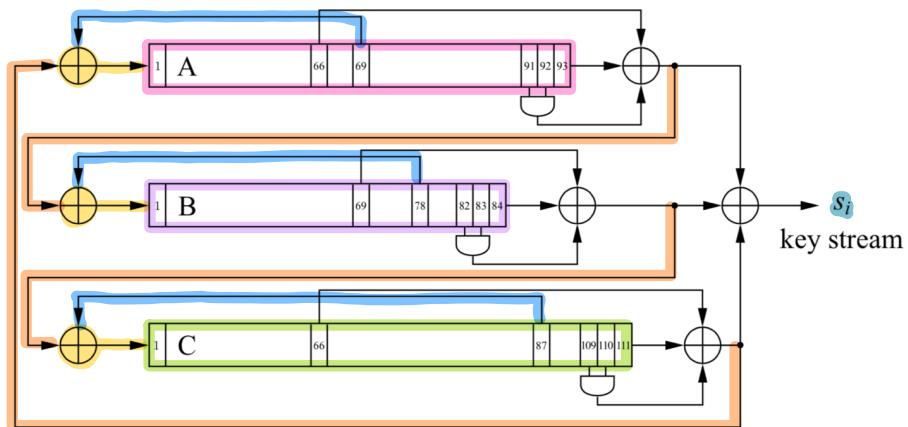
Descripción:

- Se compone de 3 shift registers: A B y C. De longitudes 93, 84 y 111, respectivamente
- La XOR-suma de los outputs de los 3 registros forman la llave s_i
- El output de cada registro está conectado al input de otro registro.
⇒ los registros pueden verse como un registro circular de longitud $288 = 93 + 84 + 111$



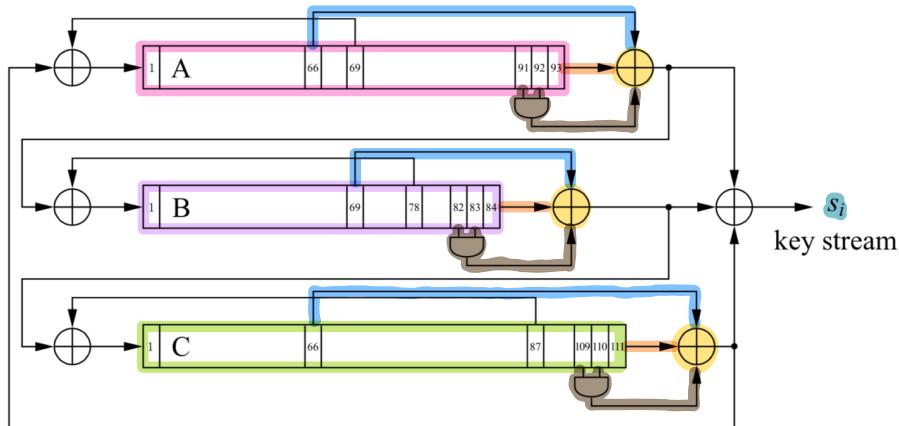
El **input** de cada registro se calcula como la **XOR-suma** de dos bits:

- El output bit de otro registro (**ejemplo**: el output del registro A es parte del input del registro B)
- Un bit de registro de una ubicación específica es llevado al input (**ejemplo**: el bit 69 del registro A es llevado a su input)



El **output** de cada registro es calculado como la xor-suma de 3 bits:

- El **bit** que se encuentra más a la derecha
- Un **bit** registrado en una ubicación específica es llevado al output (ejemplo: el bit 66 del registro A es llevado a su input)
- El **output** de una función lógica **AND** cuyo input son dos bits de registro específicos.



	register length	feedback bit	feedforward bit	AND inputs
A	93	69	66	91, 92
B	84	78	69	82, 83
C	111	87	66	109, 110

Observación:

- La operación AND es igual a la multiplicación módulo 2
- Si multiplicamos dos incógnitas, y los contenidos del registro son las incógnitas que Óscar quiere conocer
⇒ las ecuaciones resultantes ya no son lineales dado que contiene productos de incógnitas

AND		
0	0	0
0	1	0
1	0	0
1	1	1

⇒ Los caminos feedforward que involucran la operación AND son cruciales para la seguridad de Trivium puesto que preveen ataques que explotan la linealidad del cifrado.

Cifrados con Trivium

Casi todos los cifrados modernos tienen dos parámetros de entrada:

- una llave K
- un vector de inicialización IV

Llave regular usada en todo cripto sistema simétrico

El vector IV como un generador de elementos aleatorios (randomizer) y toma un nuevo valor en cada sesión de encriptación

El IV no tiene que ser un secreto, pero si cambiar en cada sesión

A los valores aleatorios que genera se les llaman **nonces** haciendo referencia a que son "number used once"

Propósito: dos Key streams producidas por el cifrado sean diferentes aún cuando la llave no haya cambiado.

Si este no es el caso un ataque que conozca el texto plano de un primer cifrado ⇒ puede calcular la Key stream correspondiente

⇒ El segundo cifrado puede ser descifrado inmediatamente usando la misma Key stream

Si el IV no cambia, el cifrado stream cipher es altamente determinístico.

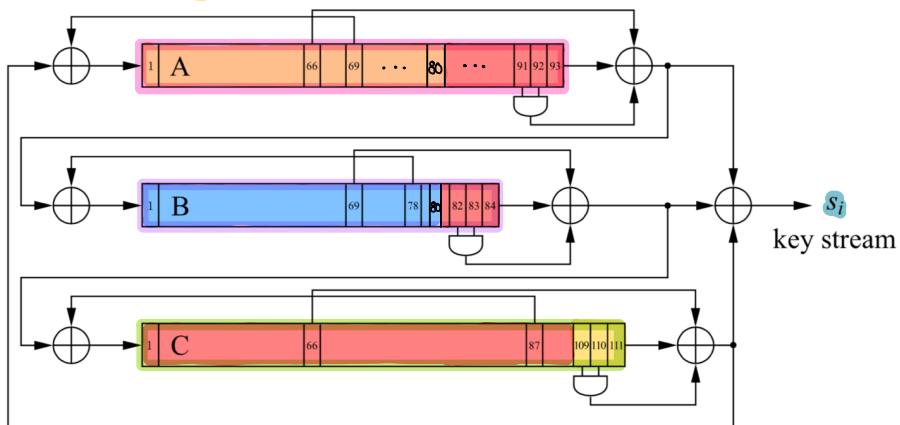
Veamos cómo es que esto funciona:

- Inicialización:

Un IV de 80 bits se carga en las 80 ubicaciones más a la izquierda del registro A.

Una llave de 80 bits se carga en las 80 ubicaciones más a la izquierda del registro B.

Todos los otros bits registrados se quedan como 0 a excepción de los 3 bits más a la derecha en el registro C (los bits C_{109}, C_{110} y C_{111}) los cuales quedan como 1.



- Fase de calentamiento:

El cifrado inicia con $4 \times 288 = 1152$ clicks del reloj. No se generan valores de salida del cifrado.

- Fase de cifrado:

Los bits producidos después de la fase de calentamiento (a partir del click 1153 del reloj) forman un Key stream

→ la fase de calentamiento es necesaria para randomizar el cifrado lo

suficiente, se asegura de que el Key stream dependa tanto de la llave K como del IV.

- Algo atractivo del Trivium es que es compacto, especialmente si se implementa en hardware.

Consta básicamente de un shift register de 288 bits y un par de operaciones Booleanas.

- Se estima que un hardware de implementación de este cifrado ocupa un área de entre 3500 y 5500 gate equivalences.

(Un gate equivalence es el área del chip ocupada por un 2-input NAND gate)

- Hasta el momento (en el que se escribió el libro) no hay ataques conocidos a este cifrado. (era relativamente nuevo en ese momento)

Muchos otros stream ciphers fueron encontrados no ser seguros.

2.4 Discusión y lectura a futuro

Stream ciphers establecidos

- La seguridad de muchos stream ciphers es desconocida y varios otros han sido rotos.
- Varios fueron usados pero debido a varios defectos actualmente no se recomienda (basados en el entendimiento actual del criptoanálisis).
- Este pesimismo cambió con el eSTREAM project

eSTREAM Project

objetivo: hacer avances en el conocimiento de diseños de stream cipher en estado del arte.

- eSTREAM fue organizado la Red Europea de Excelencia en Criptografía (CRYPT)
- Se submitieron 34 candidatos y al final del proyecto se encontraron cuatro cifrados software orientados (HC-128, Rabbit, Salsa 20/12 y SOSEMANUK) y 3 cifrados hardware orientados (Grain v1, MICKEY v2 y Trivium). Estos son relativamente nuevos (2008) y no se sabe si son realmente criptográficamente fuertes)

True random Number Generation

- Han habido varios TRNG que demuestran no comportarse de forma suficientemente aleatoria
 - ↳ Esto resulta en un problema de seguridad (dependiendo de cómo sean utilizados)

Existen estandares para evaluar formalmente los TRNG y herramientas que ponen a prueba sus propiedades estadísticas

2.5 Conclusiones

- Stream ciphers son menos populares que los cifrados de bloque en general (ejemplo: seguridad en internet). Sin embargo, tienen sus excepciones, RC4 es un stream cipher popular.

- Stream ciphers en ocasiones requieren menos recursos (ejemplo: tamaño del código o área del chip) comparado con cifrados de bloque haciendo su uso atractivo en ambientes de espacio restringido (ejemplo: celulares)
- Requieren el uso de un generador aleatorio de números criptográficamente seguro de forma más demandante.