# Criptografía simétrica

- La criptografía de clave secreta o simétrica se refiere al conjunto de métodos que permiten tener comunicación segura entre las partes siempre y cuando anteriormente se hayan intercambiado la clave correspondiente.

- Ha sido la más usada en toda la historia y ha sido implementada en diferentes dispositivos, manuales, mecánicos, eléctricos, hasta los algoritmos actuales que son programables en cualquier ordenador.

- Todos los sistemas criptográficos clásicos se basan en criptografía simétrica.

# Criptografía simétrica

- Generalmente el **algoritmo** de cifrado es **conocido** por lo que la **fortaleza** del mismo dependerá de su **complejidad** interna y sobre todo de la **longitud de la clave** empleada.

- Para que un algoritmo de este tipo sea considerado **fiable** debe cumplir varios requisitos básicos:
  - **conocido el criptograma** (texto cifrado) **no** se pueden obtener de él ni el **texto** en claro ni la **clave**.
  - **conocidos el texto en claro y el texto cifrado** debe resultar **más caro en tiempo o dinero descifrar la clave** que el **valor** posible de la **información** obtenida por terceros.

# Criptografía simétrica

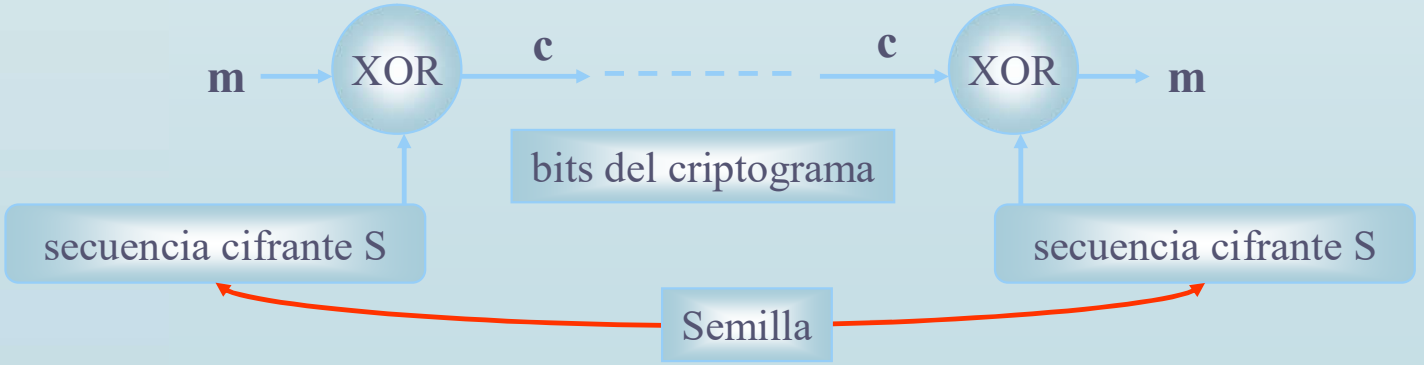
- El principal problema para este sistema de cifrado consiste en que para cada par de usuarios que quieran establecer comunicación se requiere una clave diferente, es decir, que un usuario de una red debe almacenar tantas claves como personas con las que quiera mantener una comunicación segura.

- Al principio (cuando las redes contaban con pocos usuarios) este hecho no constituía ningún problema, pero actualmente, con la cantidad de usuarios que existen en las redes se convierte en impracticable.

- Otro problema que presentan es el hecho de la distribución de claves y el peligro de que muchas personas deban conocer una misma clave.

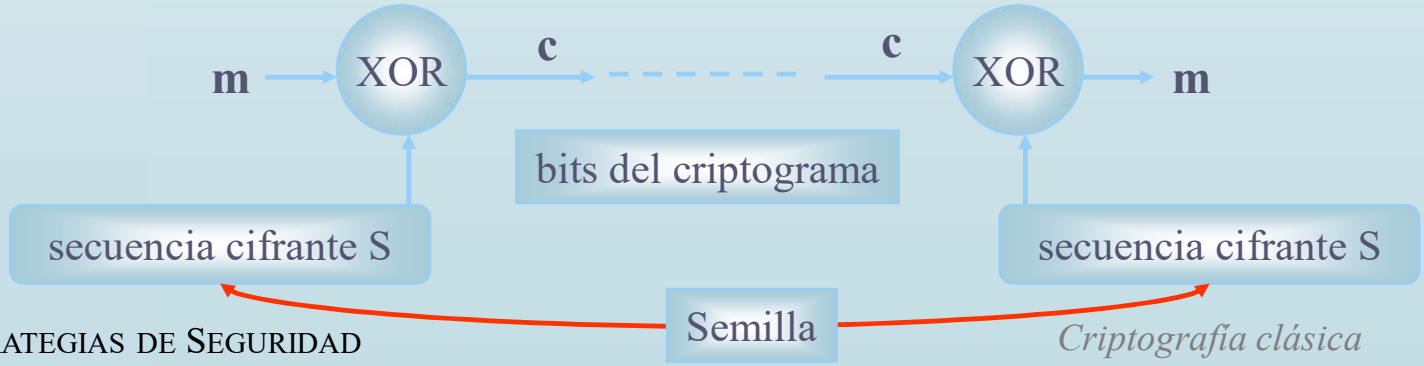
# 3.1 Características generales

- Como ya se ha visto, el cifrado Vernam verifica las condiciones de secreto perfecto definidas por Shannon.
- Es, en estos momentos, el **único procedimiento de cifrado incondicionalmente seguro**.
- Sin embargo presenta el inconveniente evidente de que requiere un bit de clave por cada bit de texto claro.
  - Puesto que el hacer llegar tal cantidad de clave a emisor y receptor por un canal seguro desbordaría la propia capacidad del canal, **en la práctica** se utiliza el método de cifrado en flujo, cuyo esquema fundamental es:



# 3.1 Características generales

- E1 emisor A, con **una clave corta (secreta)** y un **algoritmo determinista (público)**, genera una secuencia binaria S cuyos elementos se suman módulo 2 con los correspondientes bits de texto claro m, dando lugar a los bits de texto cifrado c.
  - Esta secuencia c es la que se envía a través de un canal público.
  - En recepción, B, con **la misma clave y el mismo algoritmo determinístico**, genera la misma secuencia cifrante S, que se suma módulo 2 con la secuencia cifrada c, dando lugar a los bits de texto claro m.
- Obsérvese que el cifrado en flujo es asimismo una **involución**, pues el procedimiento de cifrado y descifrado es idéntico.



# 3.1 Características generales

- Como la **secuencia cifrante** se ha obtenido a partir de un **algoritmo determinístico**, el cifrado en flujo ya no considera secuencias perfectamente aleatorias, **sino** solamente **pseudoaleatorias**.

- Sin embargo, lo que se **pierde** en cuanto a **seguridad**, por no verificarse en rigor las condiciones de Shannon, se **gana** en **viabilidad práctica** a la hora de utilizar este procedimiento de cifrado ya que **la única información que han de compartir emisor y receptor es la clave secreta**, cuya longitud oscila entre **128-256 bits**.

- En la actualidad, los bits de clave se suelen hacer llegar a ambos destinatarios mediante un procedimiento de clave pública; una vez que ambos disponen ya de la clave, se procede a aplicar el esquema tradicional del cifrado en flujo.

# 3.1 Características generales

## REQUERIMIENTOS DE UNA SECUENCIA CIFRANTE

- Es difícil evaluar cuándo una secuencia binaria es suficientemente segura para su utilización en criptografía, ya que no existe un criterio general y unificado que lo certifique.

- Sin embargo sí se pueden señalar una serie de **requerimientos generales que toda secuencia cifrante ha de satisfacer** para su correcta aplicación al cifrado en flujo, entre ellos podemos señalar

# 3.1 Características generales

## REQUERIMIENTOS DE UNA SECUENCIA CIFRANTE

### Período

- El período de la secuencia cifrante ha de ser al menos tan largo como la longitud del texto a cifrar.
- En la práctica, se generan **secuencias con período del orden de $10^{38}$ ($2^{128}$)** o superiores que cumplen sobradamente este requisito criptográfico.

### Distribución de ceros y unos

- En una secuencia aleatoria, diferentes muestras de una determinada longitud han de estar **uniformemente distribuidas a lo 1argo de ella**.

# 3.1 Características generales

## REQUERIMIENTOS DE UNA SECUENCIA CIFRANTE

### Imprevisibilidad

- La secuencia cifrante ha de ser **imprevisible**; es decir,
  - **dada una porción de secuencia** de cualquier longitud, un criptoanalista no podría **predecir el siguiente dígito** con una **probabilidad de acierto superior a 1/2.**

### Facilidad de generación

- La secuencia tiene que ser **fácil de generar con medios electrónicos** para su aplicabilidad en el proceso real de cifrado/descifrado.
- En este epígrafe se incluyen una serie de aspectos técnicos:
  - velocidad de generación, coste, tamaño, número de circuitos utilizados, consumo, etcétera, que han de tenerse en cuenta a la hora de implementar el generador de secuencia cifrante.

# 3.2 Generadores pseudoaleatorios

- La seguridad de muchos sistemas criptográficos está basada en el uso de números aleatorios como
  - claves de sesión
  - números primos
  - valores de desafío
  - secuencias cifrantes

*Estos valores han de ser lo suficientemente impredecibles para que un atacante no sea capaz de averiguarlos mediante el uso de técnicas probabilísticas*

*Se obtienen de secuencias aleatorias que pueden ser*
- *realmente aleatorias o,*
- *simplemente, comportarse como tal*.

# 3.2 Generadores pseudoaleatorios

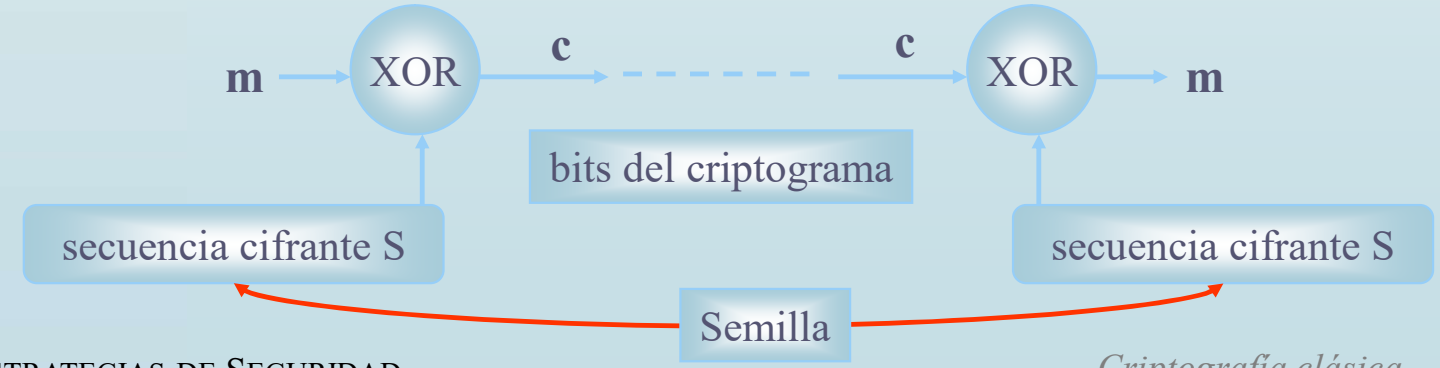
- Los generadores realmente aleatorios presentan ciertos inconvenientes para su uso criptográfico.
  - Un **atacante podría observar o manipular la fuente de aleatoriedad** de forma que le permitiera predecir la secuencia con cierta probabilidad.
  - Se ha de **monitorizar** continuamente el **funcionamiento** de la fuente para evitar que **deje de ser suficientemente aleatoria**.

# 3.2 Generadores pseudoaleatorios

- Para determinadas aplicaciones criptográficas, resulta mucho más conveniente utilizar **generadores pseudoaleatorios**
  - basados en **algoritmos deterministas**
  - las **secuencias** generadas
    - son perfectamente **reproducibles** en función de la **entrada o semilla**
    - no son genuinamente aleatorias pero **se comportan como tal**

# 3.2 Generadores pseudoaleatorios

- En las aplicaciones de los generadores pseudoaleatorios a la seguridad, necesitamos producir secuencias
  - de grandes períodos
  - complejidades lineales altas y
  - una buena distribución estadística que las haga impredecibles.
- La secuencias generadas se pueden utilizar para cifrar la información haciendo uso del método Vernam
  - XOR bit a bit entre el texto en claro y la secuencia cifrante.



# 3.2 Generadores pseudoaleatorios

Veamos algunos de los métodos más simples y conocidos para la generación de secuencias pseudoaleatorias.

- Uno de los métodos más simples para la **generación de secuencias pseudoaleatorias es el basado en congruencias lineales** que no es criptográficamente seguro.

- Otro método muy simple es un **Registro de Desplazamiento Realimentado Linealmente** (*Linear Feedback Shift Register*, **LFSR**).
  - Por sí solo es inseguro pero combinado apropiadamente puede ser muy seguro
    - A5 – inseguro
    - SNOW - seguro

**¿ A5, SNOW, LFSR?**

# 3.2 Generadores pseudoaleatorios

## ALGORITMOS DE CIFRADO EN FLUJO

- Entre los sistemas de cifrado en flujo más conocidos y utilizados destacamos:
  - **RC4**: Algoritmo de RSA, Rivest Cipher #4, desarrollado en el año 1987.
  - **SPRITZ**: variante mejorada de RC4 desarrollada en 2014.

    http://people.csail.mit.edu/rivest/pubs/RS14.pdf
    http://crypto.2014.rump.cr.yp.to/3de41b60e32a494c8f0fc9c21c67063a.pdf
  - **Salsa20**: uno de los algoritmos ganadores de eSTREAM PROJECT

    https://en.wikipedia.org/wiki/Salsa20
  - **A5**: Algoritmo no publicado propuesto en 1994. Versiones A5/1 fuerte (Europa) y A5/2 débil (exportación). Utilizado para cifrar el enlace en la telefonía móvil GSM (Global Systems for Mobile communications) o telefonía 2G.
  - **SNOW 3G**, núcleo de la integridad y la confidencialidad de las comunicaciones 4G.

    https://rua.ua.es/dspace/bitstream/10045/40395/1/RECSI-2014_12.pdf

# 3.2 Generadores pseudoaleatorios

## GENERADORES BASADOS EN CONGRUENCIAS LINEALES

- Este procedimiento de generación de números pseudoaleatorios está basado en relaciones de recurrencia del tipo

$$x_{i+1} = ax_i + b \mod n$$

donde ***(a,b,n)*** son los parámetros que caracterizan al generador y pueden utilizarse como **clave secreta**.

- Se toma un valor $\mathbf{x_0}$ como **semilla** para inicializar del proceso..

# 3.2 Generadores pseudoaleatorios

## GENERADORES BASADOS EN CONGRUENCIAS LINEALES

- Si los **parámetros** han sido **elegidos** de forma **conveniente**, **los números resultantes** $x_i$ **no** se **repetirán** hasta haber **cubierto íntegramente el intervalo** $[0, n-1]$.

- **Boyar** demostró que las secuencias obtenidas a partir de congruencias lineales **no son criptográficamente seguras**, pues dada una **porción suficientemente larga** de las mismas, se podían **deducir los parámetros n, a, b**.

# 3.2 Generadores pseudoaleatorios

## EJEMPLO DE GENERADOR BASADO EN CONGRUENCIAS LINEALES

Sean:
$a = 5$ $b = 1$
$n = 16$ $x_0 = 10$

$x_{i+1} = ax_i + b \bmod n = 5x_i + 1 \bmod 16$

FlujoLab

http://www.criptored.upm.es/software/sw_m001m.htm

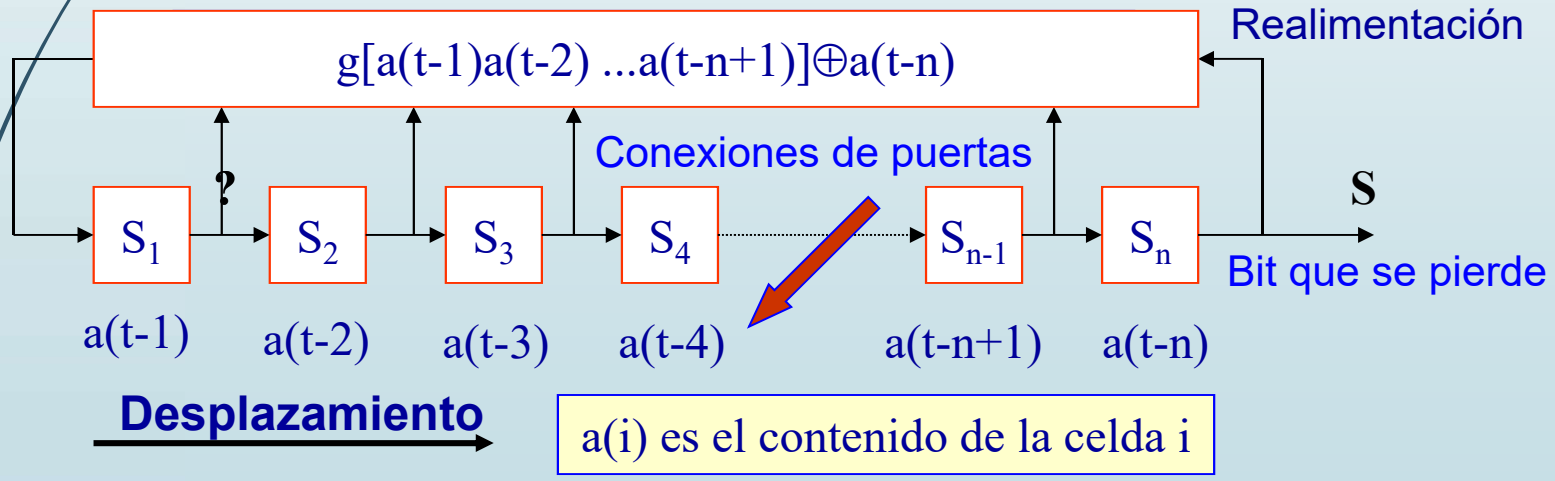
S = 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5, 10….

$x_1 = 5 \cdot 10 + 1 \bmod 16 = 3$ $x_2 = 5 \cdot 3 + 1 \bmod 16 = 0$
$x_3 = 5 \cdot 0 + 1 \bmod 16 = 1$ $x_4 = 5 \cdot 1 + 1 \bmod 16 = 6$
$x_5 = 5 \cdot 6 + 1 \bmod 16 = 15$ $x_6 = 5 \cdot 15 + 1 \bmod 16 = 12$
$x_7 = 5 \cdot 12 + 1 \bmod 16 = 13$ $x_8 = 5 \cdot 13 + 1 \bmod 16 = 2$
$x_9 = 5 \cdot 2 + 1 \bmod 16 = 11$ $x_{10} = 5 \cdot 11 + 1 \bmod 16 = 8$
$x_{11} = 5 \cdot 8 + 1 \bmod 16 = 9$ $x_{12} = 5 \cdot 9 + 1 \bmod 16 = 14$
$x_{13} = 5 \cdot 14 + 1 \bmod 16 = 7$ $x_{14} = 5 \cdot 7 + 1 \bmod 16 = 4$
$x_{15} = 5 \cdot 4 + 1 \bmod 16 = 5$ $x_{16} = 5 \cdot 5 + 1 \bmod 16 = \mathbf{10}$

## 3.3 Registros de desplazamiento realimentados

- Un registro de desplazamiento realimentado es un circuito digital secuencial constituido por n celdas (etapas o *flip-flops*) que almacenan un 1 o un 0 y una función de realimentación g que permite expresar cada nuevo elemento de la secuencia a(t), con t>n, en función de los *n* elementos anteriores
  *a(t - n), a(t - n - 1), .... a(t -1).*

- El contenido de las celdas se desplaza un lugar en el mismo sentido a cada impulso de reloj, de manera que el nuevo elemento de la secuencia *a(t)* se realimenta directamente a la primera celda.




Genera una secuencia con un período máximo $2^n$

## 3.3 Registros de desplazamiento realimentados

- Se denomina estado del registro al contenido de las celdas entre dos impulsos; el estado inicial del registro corresponde al contenido de las celdas en el momento de comenzar el proceso.

- E1 diagrama de estados de un registro de desplazamiento (y consecuentemente la secuencia generada) es cíclico siempre que la función de realimentación sea no singular; es decir, de la forma

$$a(t)=g[a(t-1),a(t-2),\ldots,a(t-n-1)]\oplus a(t-n)$$

(⊕ representa la operación lógica XOR)

pues en caso contrario el nuevo elemento *a(t)* no tendría constancia de *a(t - n)*, que es el dígito que se pierde en el siguiente desplazamiento.

## 3.3 Registros de desplazamiento realimentados

- E1 período de la secuencia producida dependerá del número de celdas del registro y de las características de la función *g*.
  - Lógicamente, el máximo período que puede alcanzar una secuencia de este tipo corresponde al máximo número de estados distintos, siendo éste $2^n$ para el caso de un registro de *n* celdas.

- La clave en este tipo de generadores está constituida por el contenido inicial del registro y/o el conocimiento de la función de realimentación.

## 3.3 Registros de desplazamiento realimentados

- Dependiendo de si la función g es o no lineal, así será, respectivamente, el registro de desplazamiento realimentado.

- **R**egistros de **D**esplazamiento **R**ealimentados **N**o **L**inealmente **NLFSR**
  **N**on **L**inear **F**eedback **S**hift **R**egister
- **R**egistros de **D**esplazamiento **R**ealimentados **L**inealmente **LFSR**
  **L**inear **F**eedback **S**hift **R**egister

# 3.3.1 Generadores no lineales NLFSR

**REGISTROS DE DESPLAZAMIENTO REALIMENTADOS NO LINEALMENTE (NLFSR)**

*Non Linear Feedback Shift Register*

- E1 problema que presenta este tipo de generador es que no hay un método sistemático para su análisis y manipulación, las secuencias generadas por estos registros pueden tener ciclos pequeños que se repiten indefinidamente a lo largo de todas ellas, lo que criptográficamente hablando es peligroso.

# 3.3.2 Generadores lineales LFSR

**REGISTROS DE DESPLAZAMIENTO REALIMENTADOS LINEALMENTE (LFSR)**

*Linear Feedback Shift Register*

- Se cuentan entre los dispositivos más importantes para la generación de secuencias pseudoaleatorias.
- Su función de realimentación g es lineal de la forma

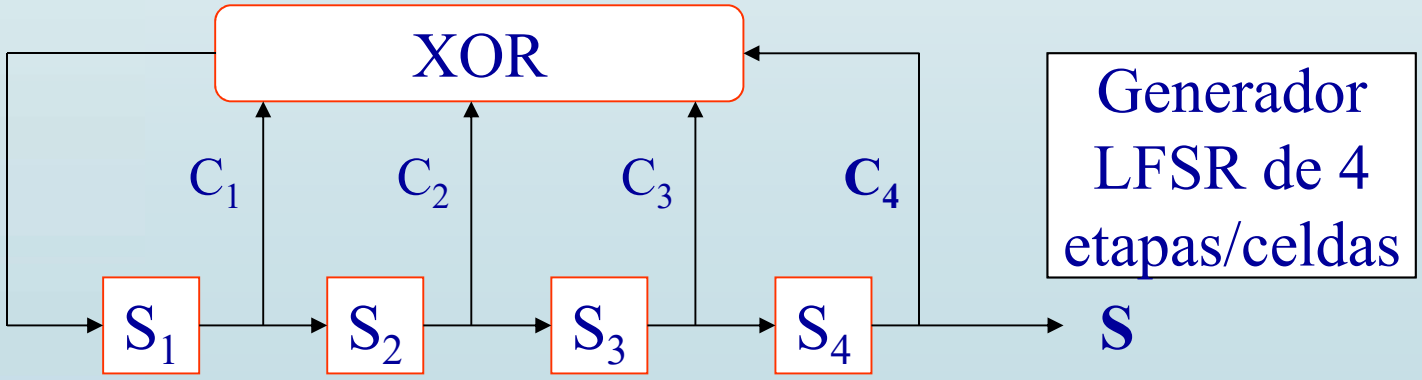
$$a(t) = c_1\, a(t\text{-}l) \oplus c_2\, a(t\text{-}2) \oplus c_n\, a(t\text{–}n)$$

con $c_i \in \{1,0\}$ y $c_n=1$.

# 3.3.2 Generadores lineales LFSR

- Su modelización es sencilla, al igual que su implementación electrónica.

- Obviamente, el estado inicial tiene que ser distinto del estado todo ceros, para evitar la secuencia idénticamente nula; el mayor número de estados diferentes será, pues, $2^n$-1.



# 3.3.2 Generadores lineales LFSR

- Todo registro de desplazamiento realimentado linealmente tiene asociado un polinomio de realimentación de grado *n*

$$f(x) = 1+c_1 x+c_2 x^2+\ldots+c_{n-1} x^{n-1}+x^n$$

con coeficientes binarios 1 ó 0, respectivamente, según que la correspondiente etapa esté o no conectada a la puerta XOR

- Estudiando las características de este polinomio, se pueden determinar las características de la secuencia generada por el registro.

LFSR con polinomios factorizables
LFSR con polinomios irreducibles
LFSR con polinomios primitivos

- De este modo se distinguen varios tipos de generadores

# 3.3.2 Generadores lineales LFSR

## LFSR CON POLINOMIO FACTORIZABLE

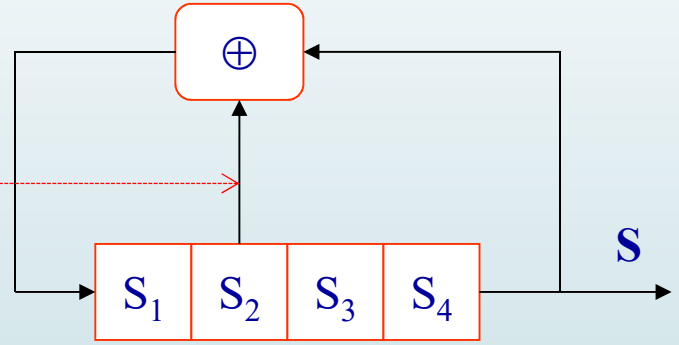
- Dan lugar a secuencias caracterizadas por:
  - La longitud de la secuencia depende del estado inicial.
  - El máximo período T verifica $n<T<2^n-1$, pudiendo aparecer períodos secundarios que son divisores de T.

# 3.3.2 Generadores lineales LFSR

## LFSR CON POLINOMIO FACTORIZABLE

- Dan lugar a secuencias caracterizadas por:
  - La longitud de la secuencia depende del estado inicial.
  - El máximo período T verifica $n<T<2^n-1$, pudiendo aparecer períodos secundarios que son divisores de T.

Sea $f(x) = x^4 + x^2 + 1 = 1 + x^2 + x^4$




Es factorizable porque:

$f(x) = f(x_1) \bullet f(x_2)$

$f(x) = (x^2+x+1)\,(x^2+x+1)$

$f(x) = x^4+2x^3+3x^2+2x+1$

Reduciendo mod 2:

$f(x_1) \bullet f(x_2) = x^4+x^2+1$

Problema

T depende de la semilla

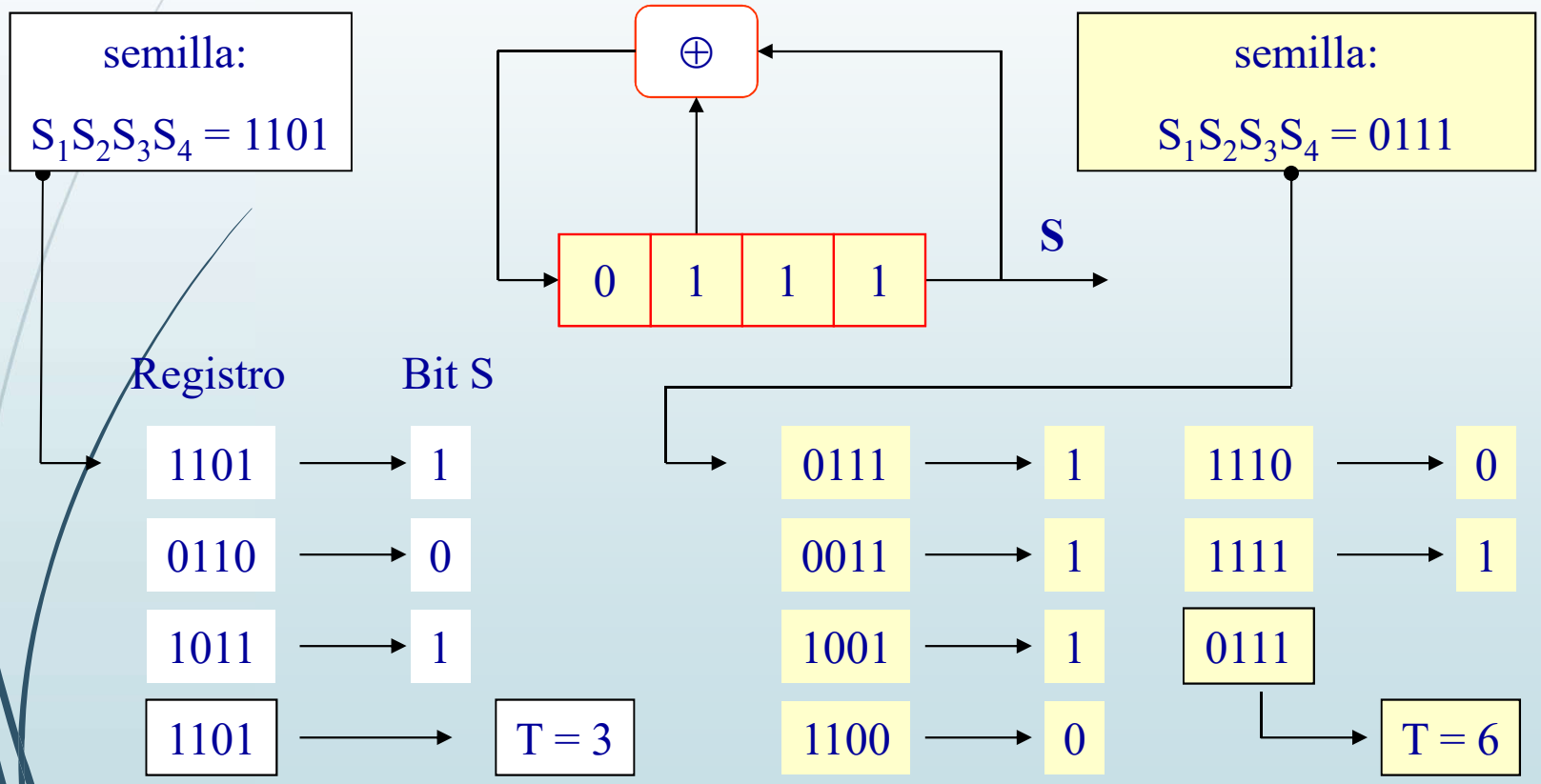
$n \le T \le 2^n - 1$

Períodos secundarios divisores

# 3.3.2 Generadores lineales LFSR

## LFSR CON POLINOMIO FACTORIZABLE

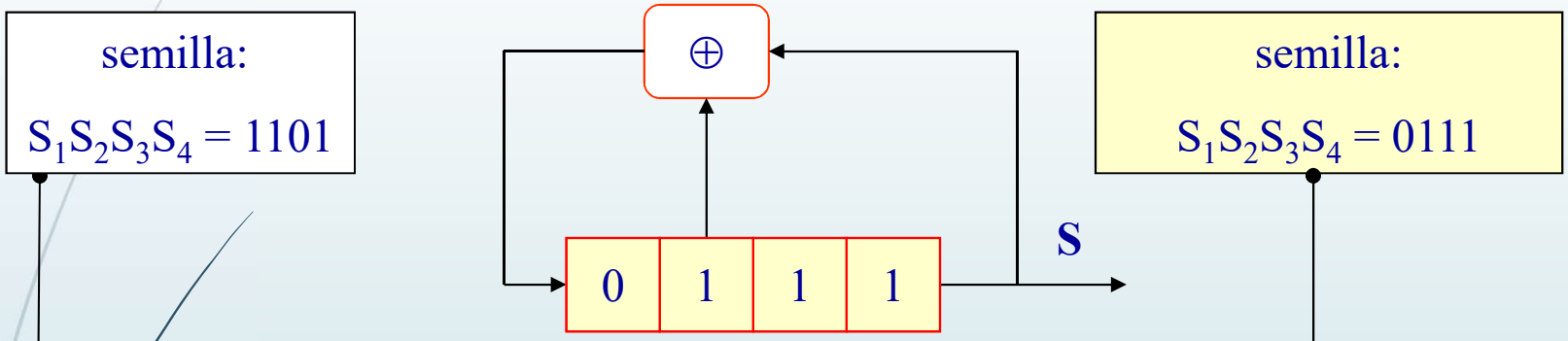
- Generador factorizable de cuatro celdas $f(x) = 1+x^2+x^4$



# 3.3.2 Generadores lineales LFSR

## LFSR CON POLINOMIO FACTORIZABLE

- Generador factorizable de cuatro celdas $f(x) = 1+x^2+x^4$




Dependiendo de la semilla, el período T de la secuencia toma valores distintos.
En este caso, el período principal es $T_1 = 6$ y el período secundario $T_2 = 3$

# 3.3.2 Generadores lineales LFSR

**LFSR CON POLINOMIO IRREDUCIBLE**

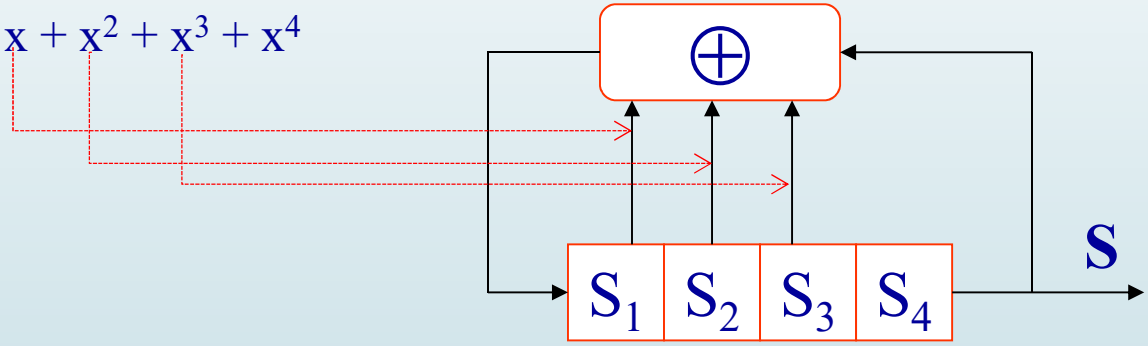
- Dan lugar a secuencias caracterizadas por:
  - La longitud de la secuencia no depende del estado inicial.
  - El período T es un divisor de $2^n$-1.

# 3.3.2 Generadores lineales LFSR

## LFSR CON POLINOMIO IRREDUCIBLE

- Dan lugar a secuencias caracterizadas por:
  - La longitud de la secuencia no depende del estado inicial.
  - El período T es un divisor de $2^n$-1.

Sea $f(x) = 1 + x + x^2 + x^3 + x^4$




Es imposible factorizar en módulo 2 el polinomio f(x) como producto de dos polinomios $f(x_1)$ y $f(x_2)$ de grado menor

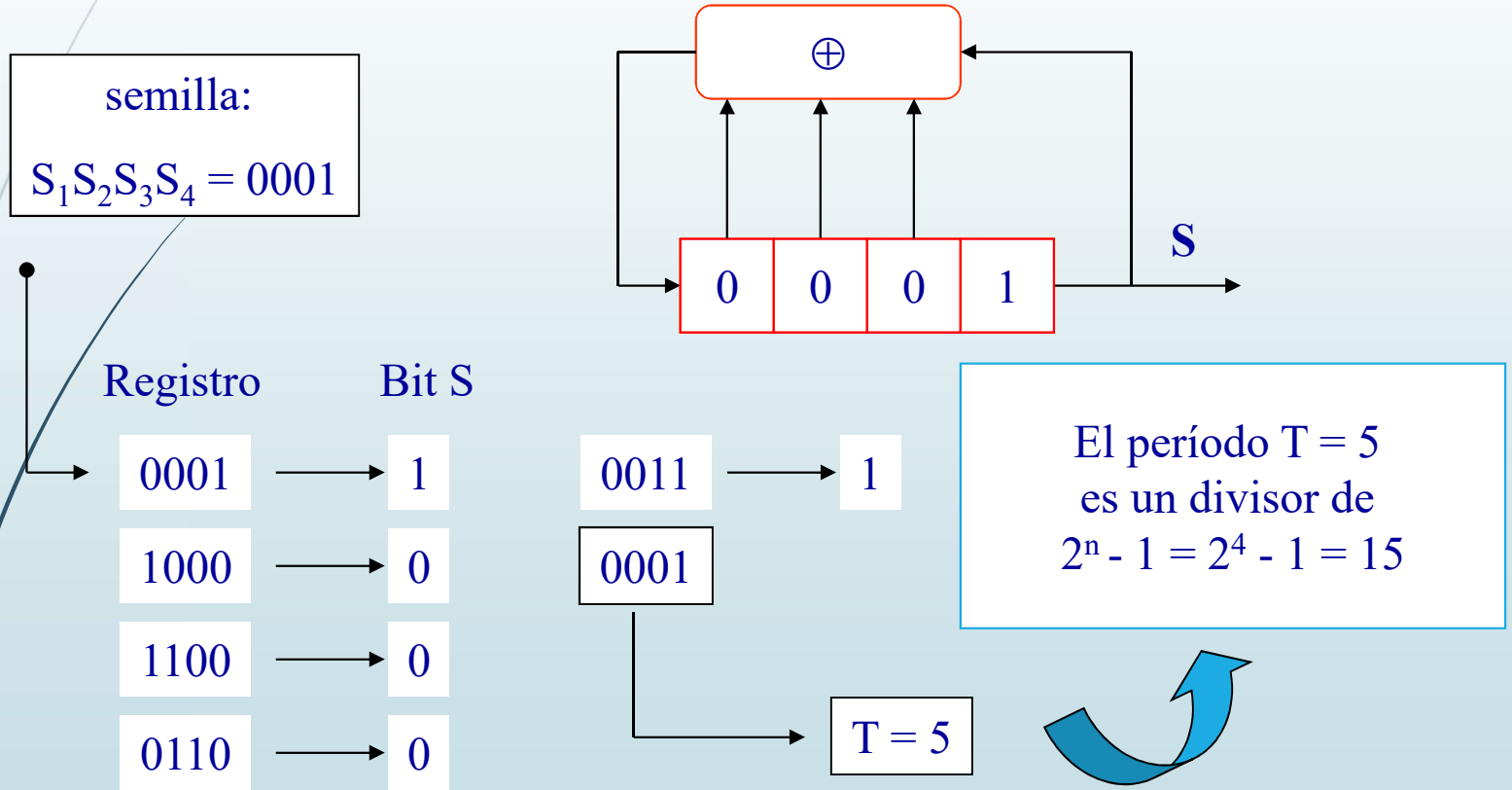
Problema

Ahora T ya no depende de la semilla pero será un divisor de $T_{máx} = 2^n - 1$

# 3.3.2 Generadores lineales LFSR

## LFSR CON POLINOMIO IRREDUCIBLE

- Generador irreducible de cuatro celdas $f(x) = 1 + x + x^2 + x^3 + x^4$

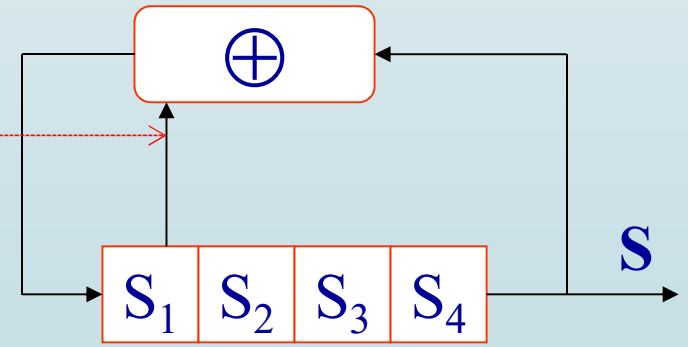


# 3.3.2 Generadores lineales LFSR

**LFSR CON POLINOMIO PRIMITIVO**

- Dan lugar a secuencias caracterizadas por:
  - La longitud de la secuencia no depende del estado inicial.
  - El período T es $2^n$-1
- Estos generadores son los que ofrecen una secuencia de período máximo $2^n$-1; luego son los recomendados para su aplicación criptográfica.
- Existen algoritmos para la determinación de polinomios primitivos con coeficientes binarios:
  - GOLOMB, S. W., *Shift Register Sequences*

Sea $f(x) = 1 + x + x^4$




f(x) no es factorizable como $f(x_1) \cdot f(x_2)$ en módulo dos.Es además un generador del grupo.

# 3.3.2 Generadores lineales LFSR

**LFSR CON POLINOMIO PRIMITIVO**

- El número de polinomios primitivos de grado n obedece a la expresión

$$\frac{\Phi(2^n - 1)}{n}$$

donde $\Phi(x)$ es la función de Euler que denota el número de enteros positivos menores que x y primos con él.

- Este número crece exponencialmente con n, por ejemplo:
  - para n=11 existen 176 polinomios primitivos,
  - mientras que para n = 24 existen 276480.

- Una secuencia generada por un registro de desplazamiento de polinomio primitivo se denomina secuencia de máxima longitud o, abreviadamente, ***m-secuencia***.

# 3.3.2 Generadores lineales LFSR

## LFSR CON POLINOMIO PRIMITIVO

- Generador primitivo de cuatro celdas $f(x) = 1 + x + x^4$