

4-Criptografía Simétrica

Fundamentos genéricos de los cifrados simétricos

- **A) Confusión y Difusión:** Introducidos por Claude Shannon como dos bloques básicos de construcción para cualquier sistema criptográfico, cuyos objetivos principales son:
 - Frustrar el criptoanálisis basado en análisis estadístico.
- De nuevo todas las ideas están reflejadas en C. E. Shannon, "Communication theory of secrecy systems," in The Bell System Technical Journal, vol. 28, no. 4, pp. 656-715, Oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.
- **Confusión:** consiste en establecer una dependencia funcional lo más compleja posible entre la clave y el texto cifrado. Para ello se utiliza la técnica de sustitución en el texto claro. Así si el criptoanalista obtuviera alguna estadística del texto cifrado, la forma en que se mezcló la clave en el texto cifrado es tan compleja que no se puede obtener de él.

Fundamentos genéricos de los cifrados simétricos

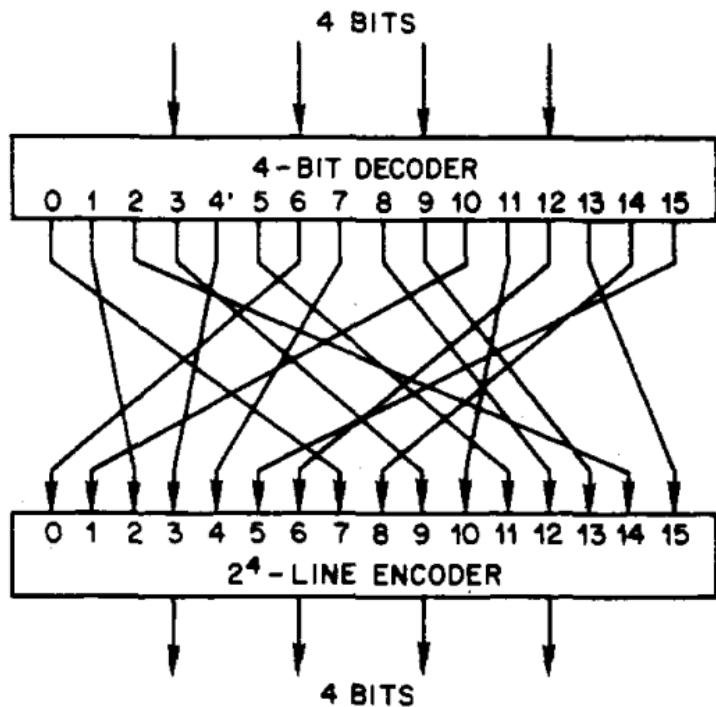
- **Difusión:** la estructura estadística del texto plano se disipa a lo largo de las rondas, es decir cada bit del texto plano afecta al máximo número de bits del texto cifrado, i.e. si se cambia un bit en el texto sin cifrar, deberían cambiarse la mayor cantidad posible de bits en el texto cifrado para que hubiese una buena difusión del criptograma (se obtiene a través Permutaciones).
- Así el objetivo de la difusión es dispersar a lo largo del criptograma las propiedades estadísticas del mensaje claro, separando al máximo los caracteres que aparecen juntos en el mensaje.
- De esta forma se dispersa la información relacionada con la aparición de digramas, trigramas, etc. contenida en el texto claro y cualquier otra información relacionada con la frecuencia de aparición de ciertas combinaciones de texto claro.
- Como hemos comentado ya, la técnica usada para hacer esto son las permutaciones de los caracteres en el criptograma.
- Así un bit a la salida del bloque en el algoritmo depende de manera compleja de todos los bits de entrada en el texto original:
 - $y_i = f_i(x_1, x_2, \dots, x_n)$, donde $\bar{x} = (x_1, x_2, \dots, x_n)$ es el texto plano, e $\bar{y} = (y_1, y_2, \dots, y_n)$ es el texto cifrado, bloque por bloque.

Fundamentos genéricos de los cifrados simétricos

- **B) Producto de Criptosistemas:** se utiliza el producto de dos criptosistemas: transposición y sustitución.
- Transposición o permutación: supongamos n y que $P = C = (Z_m)^n$ y que el conjunto K se compone de todas las permutaciones posibles de n elementos: $|K| = n!$.
 - Ej. Supongamos una permutación para $n = 4$, por ejemplo $k = (2, 4, 1, 3)$: entonces $(x_1, x_2, x_3, x_4) \rightarrow (x_2, x_4, x_1, x_3)$.
- Sustitución: supongamos n y que $P = C = (Z_m)^n$ y que el conjunto K se compone de todas las sustituciones no singulares posibles de n elementos: $|K| = m^n!$.
 - Ej. Así supongamos $m = 2$ (binario) y $n = 3$, una posible sustitución no singular es:

Fundamentos genéricos de los cifrados simétricos

Imágenes extraídas de [Horst Feistel, H. W. Notz, J. Lynn Smith. "Some cryptographic techniques for machine-to-machine data communications." IEEE proceedings, 63\(11\), 1545-1554, 1975.](#)



000 → 111
001 → 001
010 → 010
011 → 011
100 → 100
101 → 101
110 → 110
111 → 000

Reversible Mapping

cleartext	ciphertext
00	11
01	10
10	00
11	01

Irreversible Mapping

cleartext	ciphertext
00	11
01	10
10	01
11	01

} singular block.

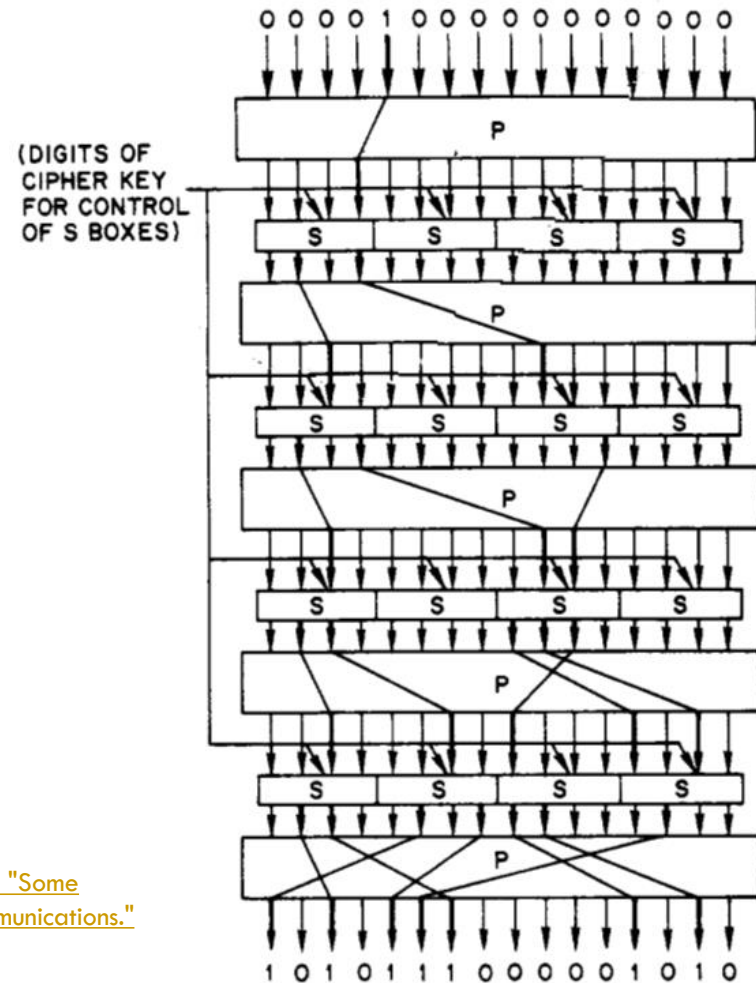
Fundamentos genéricos de los cifrados simétricos

- Si n es pequeño es fácil de romper por la fuerza bruta, pero si es grande es más complicado.
- Pero si es muy grande desde el punto de vista computacional es complicada su implementación.
- Feistel y colegas resuelven este problema en el cifrado de FEISTEL:
 - Horst Feistel, H. W. Notz, J. Lynn Smith. "Some cryptographic techniques for machine-to-machine data communications." IEEE proceedings, 63(11), 1545–1554, 1975.
- En este artículo proponen un enfoque que se aproxima cuando n es grande.

Fundamentos genéricos de los cifrados simétricos

- **C) Cifrado FEISTEL** (es la base del DES, FEAL y otros muchos): Este enfoque está compuesto de componentes fácilmente realizable que implemente el producto de criptosistemas (Substitución y Permutación):

Imagen extraída de [Horst Feistel, H. W. Notz, J. Lynn Smith. "Some cryptographic techniques for machine-to-machine data communications." IEEE proceedings, 63\(11\), 1545–1554, 1975.](#)



Fundamentos genéricos de los cifrados simétricos

- Ya se puede observar por primera vez el fenómeno avalancha de bits.
- Las cajas P son pequeñas permutaciones y las cajas S son pequeñas substituciones.
- Estas substituciones se seleccionan con la clave.
- La unión de estas pequeñas operaciones en cadena genera una confusión y difusión complejas del texto a cifrar.
- Cada repetición en este esquema se suele denominar ronda de cifrado.
- La sucesión de rondas produce la complejidad deseada en el cifrado.

Fundamentos genéricos de los cifrados simétricos

- Normalmente la forma genérica del cifrado Feistel se suele dar como en función de sus rondas.
- Así el bloque a cifrar (típicamente 64 o 128 bits) se divide en dos mitades: L y R.
- Se define una función F y sub claves K_i .
- Se hacen operaciones repetitivas (rondas) con estos elementos: L, R, F y K_i .
- Así una ronda i de manera genérica se puede definir como:
 - $$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus F(R_{i-1}, K_i). \end{cases}$$
- La particularidad del cifrado Feistel es que tiene una función inversa aunque la función F no sea reversible (realmente se utiliza la propiedad de la función XOR: $A \oplus A = 0$, lo veremos con detalle en el DES).
- Por tanto F puede ser tan compleja como uno quiera, permitiendo una gran confusión.
- El descifrado con Feistel se hace invirtiendo el orden las sub claves utilizadas, ya que la propia estructura Feistel es reversible, lo veremos con detalle en el DES.

Fundamentos genéricos de los cifrados simétricos

- Las características y dependencias del de diseño FEISTEL pueden ser las siguientes:
 - Tamaño de Bloque: cuando más grande es el tamaño de bloque más complicado es el análisis estadístico. Pero no hay que pasarse porque se reduce la velocidad de cifrado.
 - Tamaño de la clave: cuando mayor es el tamaño de la clave más resistente al ataque de fuerza bruta, aunque también decrece la velocidad del algoritmo.
 - Número de rondas: Lo esencial de este enfoque es que una única ronda no ofrece seguridad, pero la combinación de varias rondas es la que genera el proceso de confusión y difusión. De nuevo muchas rondas puede ralentizar el algoritmo.
 - Algoritmo de generación de sub claves: tiene una gran complejidad para aumentar la fortaleza al criptoanálisis.
 - La función F de mezcla de la clave en cada ronda: tiene que ser robusta al criptoanálisis, por eso hay que evitar que sea una función afín: $F(x \oplus y) \neq F(x) \oplus F(y)$.

El *Data Encryption Standard* (DES): Historia

- En 1973 la “*National Bureau of Standards*”, NBS, anunció un concurso para la transmisión y almacenamiento de información de manera segura, con una serie de especificaciones:
 - Nivel de seguridad alto.
 - Algoritmo fácil de entender especificando todos sus detalles.
 - Su seguridad no se debería ver decrementada por su publicación.
 - Debería ser disponible para cualquier usuario.
 - Debería ser utilizado en varias aplicaciones.
 - Permitir la implementación en “*hardware*” de bajo coste.
 - Etc.
- El resultado del concurso fue: sin ninguna respuesta!!!!

El *Data Encryption Standard* (DES): Historia

- En agosto de 1974 tuvo lugar el segundo concurso, cuyo resultado es que IBM propuso la modificación de un algoritmo que ya existía para la comunicación segura entre bancos en UK, denominado LUCIFER (diseñado al amparo de IBM por Horst Feistel y Don Coppersmith a principios de la década del 1970).
- LUCIFER tenía Tamaños de clave 48, 64 o 128 bits, y tamaños de bloque 48, 32 o 128 bits.
- En marzo de 1975 este algoritmos fue modificado por la NSA (*National Security Agency*), y se propuso a la opinión general del público. Se impuso una limitación en la clave y tamaño de bloque: Tamaño de clave 64 bits (56 bits efectivos), y tamaño de bloque 64 bits.

El Data Encryption Standard (DES): Historia

- Finalmente en 1977 el originario algoritmos de IBM se constituyó en el legendario *Data Encryption Standard* (DES), en EEUU.
- Ha sido el cifrado más criptoanalizado hasta la fecha por los criptoanalistas modernos.
- Nunca se ha podido romper por técnicas estadísticas de manera práctica, y el único enfoque práctico es fuerza bruta (por supuesto que para ataque tipo A solo hay fuerza bruta):
 - Son posibles varios tres tipos de ataques teóricos:
 - Criptoanálisis diferencial: [Biham, E. & Shamir, A \(1993\). Differential cryptanalysis of the data encryption standard. Shamir, Adi. New York: Springer-Verlag. pp. 487–496. doi:10.1007/978-1-4613-9314-6. ISBN 978-0387979304. OCLC 27173465. S2CID 6361693.](#) (requiere 2^{47} textos planos elegidos).
 - Criptoanálisis lineal: [Matsui, Mitsuru \(1993-05-23\). "Linear Cryptanalysis Method for DES Cipher". Advances in Cryptology — EUROCRYPT '93. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 765: 386–397. doi:10.1007/3-540-48285-7 33. ISBN 978-3540482857.](#) (requiere 2^{43} textos planos elegidos).
 - Ataque de Davies: (el mejorado por Biham requiere 2^{50} textos planos elegidos y con una tasa de éxito del 51%).
 - [Davies, D. W. \(1987\). "Investigation of a potential weakness in the DES algorithm, Private communications". Private Communications.](#)
 - [Biham, E., Biryukov, A. An improvement of Davies' attack on DES. J. Cryptology 10, 195–205 \(1997\). https://doi.org/10.1007/s001459900027](#)
 - Podéis ver un TFG interesante sobre esta temática:
 - [Sergio Galán. Estudio y análisis del criptoanálisis lineal y diferencial: técnicas y herramientas \(2021\).](#)
- Estos criptoanálisis requieren una complejidad teórica un poco menor que un ataque por fuerza bruta (2^{56}).
- Pero debido a que requieren una cantidad tan grande e irreal de textos planos conocidos o escogidos para poder realizar el criptoanálisis, estos ataques NO se tienen en cuenta en criptoanálisis práctico del DES. (Oct 2000 el AES).

El *Data Encryption Standard* (DES): Historia

- El ataque por fuerza bruta al DES se realizó de manera efectiva en el proyecto de DES challenge, propuesto por RSA Security en 1997 (fundada por Ron Rivest, Adi Shamir and Leonard Adleman en 1982, que fue la filial de seguridad de EMC corporation, adquirida a su vez en 2016 por Dell Technologies).
- DES challenge I: Rocke Verser (Loveland, Colorado) ideó un ataque de fuerza de bruta mediante el cálculo con varios usuarios en internet. El proyecto DESDHALL empezó el 13 de marzo de 1997 y terminó el 17 de junio de 1997 (un poco más de 3 meses).
- DES challenge II: Tuvo dos competiciones en enero de 1998 y otra en julio de 1998:
 - DES challenge II-1.
 - DES challenge II-2.

El Data Encryption Standard (DES): Historia

- DES challenge II-1: Este fue encabezado por “distributed.net”, mediante un esfuerzo coordinado de recursos computacionales distribuidos. Necesitaron 39 días.
- DES challenge II-2: Este fue ejecutado por Electronic Frontier Foundation (EFF, fundadores Kapor, Gilmore and Barlow) que diseñó un ordenador especialmente orientado a probar claves del DES. Este computador (DEEP CRACK) tuvo un coste aproximado de unos 250.000\$. En este caso fueron casi 3 días.

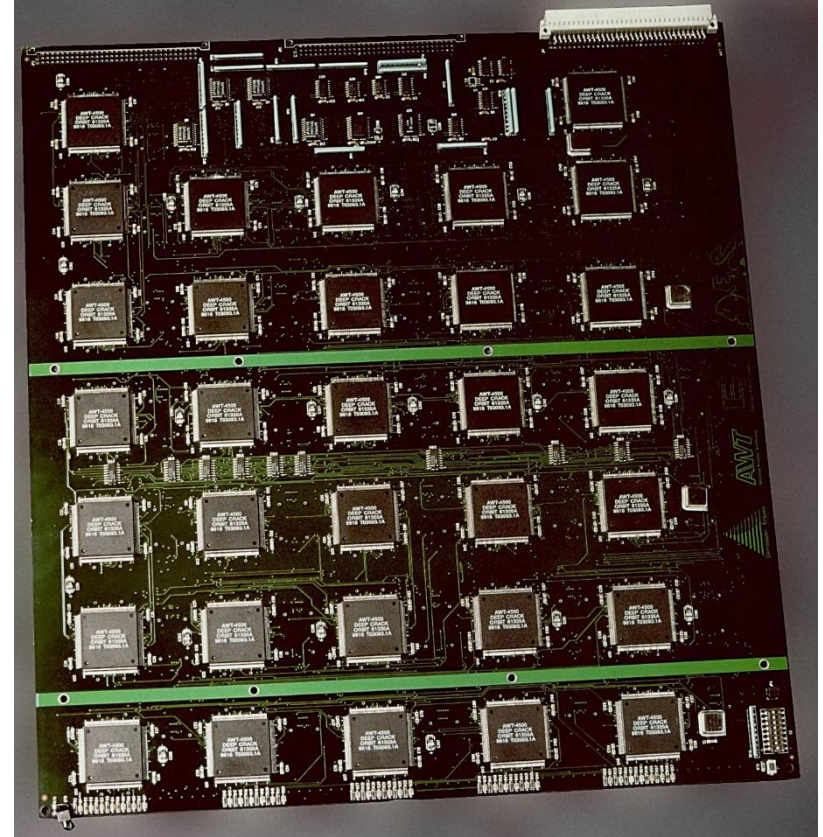


Imagen extraída de https://en.wikipedia.org/wiki/EFF_DES_cracker.

El Data Encryption Standard (DES): Historia y características

Supports ANSI X3.92 Data Encryption Algorithm - DES/DES Chi...

<http://www.allproducts.com.tw/manufacture11/isi/sc72020.htm>

- **DES challenge III:** Aquí en 1999 ya se supera la barrera de las 24 horas mediante una combinación de "[distributed.net](#)" y el **DEEP CRACK** de **EFF**.
- **David C. McNett (19 January 1999).** "US Government's Encryption Standard Broken in Less Than a Day". [distributed.net](#). Retrieved 27 February 2014.
- Ante la rotura de la barrera del día para el DES se tomaron varias acciones que veremos más adelante.
- Algunas **características de DES** son:
 - Tamaño de bloque: 64 bits. Tamaño de clave: 8 bytes (con paridad, 56 efectivos).
 - Normas ANSI: X3.92 (descripción del algoritmo) y X3.108 (descripción de los modos de operación).
 - Fácilmente implementable en circuitos integrados.
 - Es un cifrador de tipo FEISTEL, con 16 Rondas, y una permutación inicial y otra final.
 - Para poder utilizar las sumas (xor) se utilizan permutaciones con expansiones y compresiones para igualar el número de bits.
 - La permutación inicial y final no tiene interés criptográfico.
 - Cada ronda es un cifrado FEISTEL:

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus F(R_{i-1}, K_i). \end{cases}$$

allproducts.com

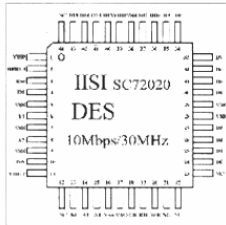
[Home](#) | [Trade Leads](#) | [Adv. Sourcing Service](#) | [Feed Back](#)
[Product Search](#) | [Supplier Search](#)

[Company](#) | [Showroom](#) | [Inquiry](#)

International Integrated Systems, Inc. (IISI)

Product ID: DES Chip (SC72020)

Supports ANSI X3.92 Data Encryption Algorithm - DES



Specifications:

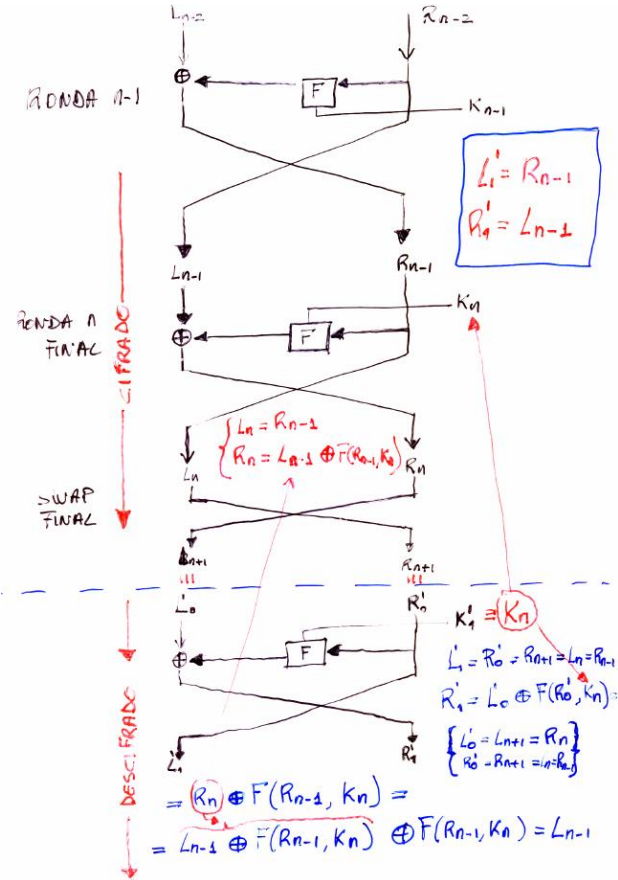
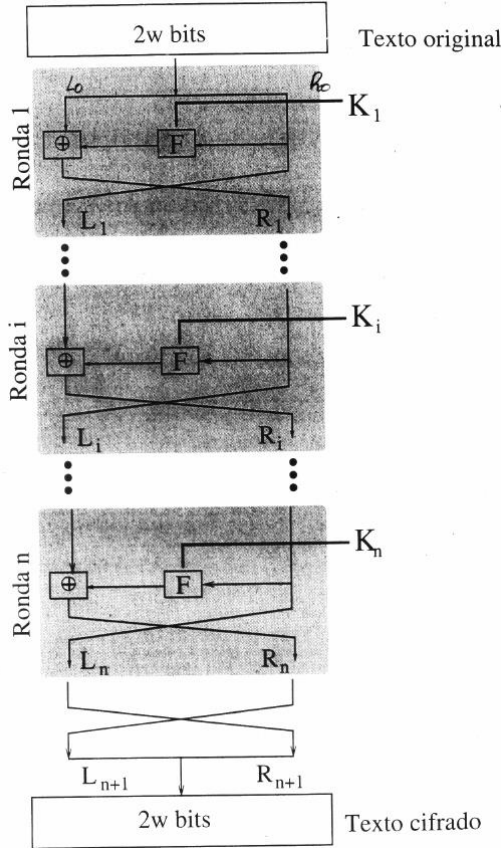
- Size: 18 mm x 18 mm x 3 mm
- Materials: 44-pin plastic QFP
- Transfer Rate: 10 Mbps with 30Mz clock, 667n sec for DES kernel
- 56-bit key: Encrypt/Decrypt 64-bit data words using 56-bit key word
- Byte/Word: Supports BYTE(8-bit)/WORD(16-bit) microprocessor interface
- Clock rate: Up to 30MHz
- Type I/Type II: Type I format(Mortorola chip) and Type II format(Intel Chip)
- Package: Plastic QFP6-44pin

1 of 2

28/03/03 10:...

El Data Encryption Standard (DES): Características

Estructura FEISTEL particular para el DES.



Importancia del "SWAP" final para que el descifrado tenga la misma estructura que el cifrado (únicamente invirtiendo el orden de las claves).

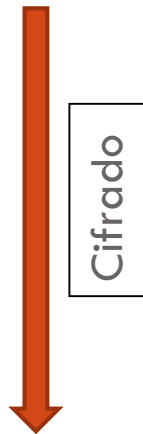
El Data Encryption Standard (DES): Características

➤ El DES tiene dos críticas:

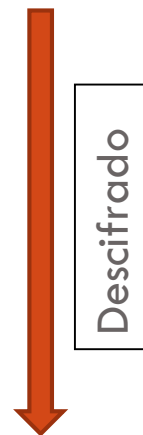
$$\text{Ronda } i: \text{FEISTEL}(F, K_i) = \begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus F(L_i, K_i). \end{cases}$$

- Se disminuyó la seguridad del algoritmo originario LUCIFER.
- El diseño de las cajas S en principio fue secreto (¿¿¿¿puerta trasera????).
- En 1989 se implanta el triple DES o EDE ([FIPS PUB 46-3](#)) debido a la rotura por fuerza bruta del [DES challenge III](#).
- El algoritmo DES lo podemos poner esquemáticamente como ([FIPS PUB 46](#)):

➤ Texto Original (64 bits)
IP
Ronda 1: FEISTEL(F, K₁)
Ronda 2: FEISTEL(F, K₂)
.....
Ronda 16: FEISTEL(F, K₁₆)
SWAP
IP⁻¹
Texto Cifrado (64 bits)



➤ Texto Cifrado (64 bits)
IP
Ronda 1: FEISTEL(F, K₁₆)
Ronda 2: FEISTEL(F, K₁₅)
.....
Ronda 16: FEISTEL(F, K₁)
SWAP
IP⁻¹
Texto Original (64 bits)



El Data Encryption Standard (DES): Permutación inicial

- Las IP y IP^{-1} son las siguientes:

```
/* permutación IP */
static const unsigned short IP[BITS_IN_IP] = {
    58, 50, 42, 34, 26, 18, 10, 2,
    60, 52, 44, 36, 28, 20, 12, 4,
    62, 54, 46, 38, 30, 22, 14, 6,
    64, 56, 48, 40, 32, 24, 16, 8,
    57, 49, 41, 33, 25, 17, 9, 1,
    59, 51, 43, 35, 27, 19, 11, 3,
    61, 53, 45, 37, 29, 21, 13, 5,
    63, 55, 47, 39, 31, 23, 15, 7
};
```

En la posición 25 viene el bit
que estaba en la posición 64

En la posición 64 vuelve el bit
que estaba en la posición 25

```
/* inversa de IP */
static const unsigned short IP_INV[BITS_IN_IP] = {
    40, 8, 48, 16, 56, 24, 64, 32,
    39, 7, 47, 15, 55, 23, 63, 31,
    38, 6, 46, 14, 54, 22, 62, 30,
    37, 5, 45, 13, 53, 21, 61, 29,
    36, 4, 44, 12, 52, 20, 60, 28,
    35, 3, 43, 11, 51, 19, 59, 27,
    34, 2, 42, 10, 50, 18, 58, 26,
    33, 1, 41, 9, 49, 17, 57, 25
};
```

El Data Encryption Standard (DES): El algoritmo de subclaves

- Esquema FEISTEL más la generación de las claves de cada ronda:
- Existe un esquema para generar la clave que le entra cada ronda, mediante la clave inicial del cifrado.
- Utiliza la idea que ya habíamos visto: una semilla genera una secuencia cifrante.

```
/* "permutación" PC1 */
static const unsigned short PC1[BITS_IN_PC1] = {
    57, 49, 41, 33, 25, 17, 9,
    1, 58, 50, 42, 34, 26, 18,
    10, 2, 59, 51, 43, 35, 27,
    19, 11, 3, 60, 52, 44, 36,
    63, 55, 47, 39, 31, 23, 15,
    7, 62, 54, 46, 38, 30, 22,
    14, 6, 61, 53, 45, 37, 29,
    21, 13, 5, 28, 20, 12, 4
};

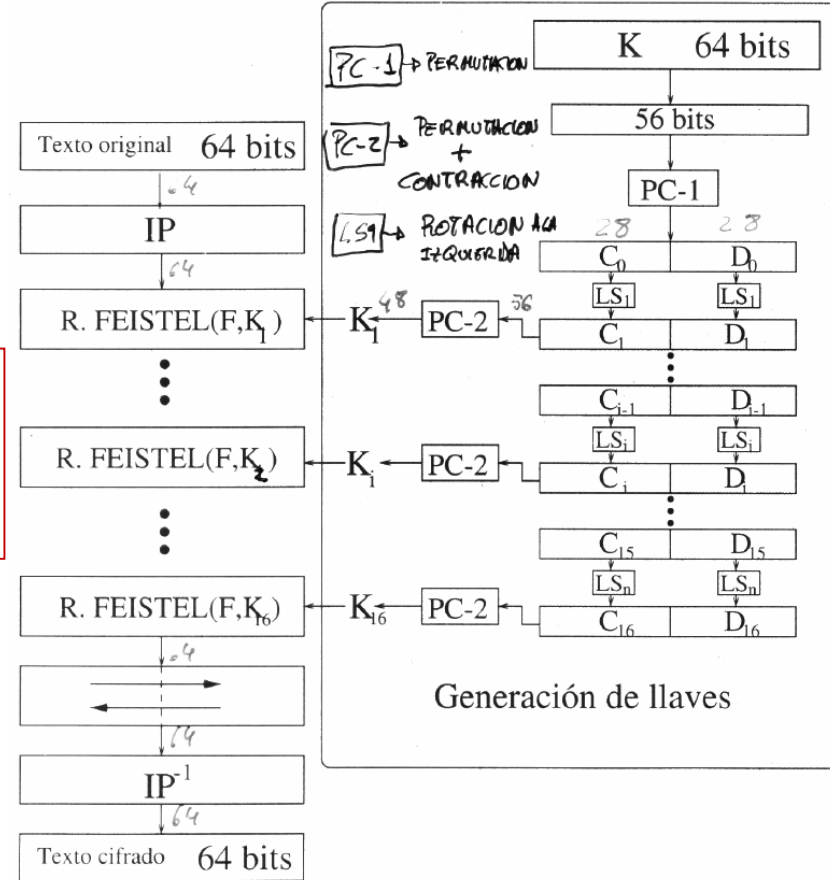
/* "permutación" PC2 */
static const unsigned short PC2[BITS_IN_PC2] = {
    14, 17, 11, 24, 1, 5,
    3, 28, 15, 6, 21, 10,
    23, 19, 12, 4, 26, 8,
    16, 7, 27, 20, 13, 2,
    41, 52, 31, 37, 47, 55,
    30, 40, 51, 45, 33, 48,
    44, 49, 39, 56, 34, 53,
    46, 42, 50, 36, 29, 32
};
```

PC1: Solo una permutación.

PC2: Una permutación+contracción

Pasamos de 56 bits a 48 bits:

PC-2 ignora los bits 9, 18, 22, 25, 35, 38, 43, 54.



El *Data Encryption Standard* (DES): El algoritmo de subclaves

- Inicialmente, se seleccionan 56 bits (los ocho restantes, 8, 16, 24, 32, 40, 48, 56, 64, se especificaron para su uso como bits de paridad) de la clave de los 64 iniciales mediante PC-1:
- Los ocho bits restantes se descartan o se usan como bits de verificación de paridad.
- Luego, los 56 bits se dividen en dos mitades de 28 bits:
 - Cada mitad se trata posteriormente por separado, mediante rotaciones.
 - Así, ambas mitades se rotan a la izquierda uno o dos bits (especificados para cada ronda):

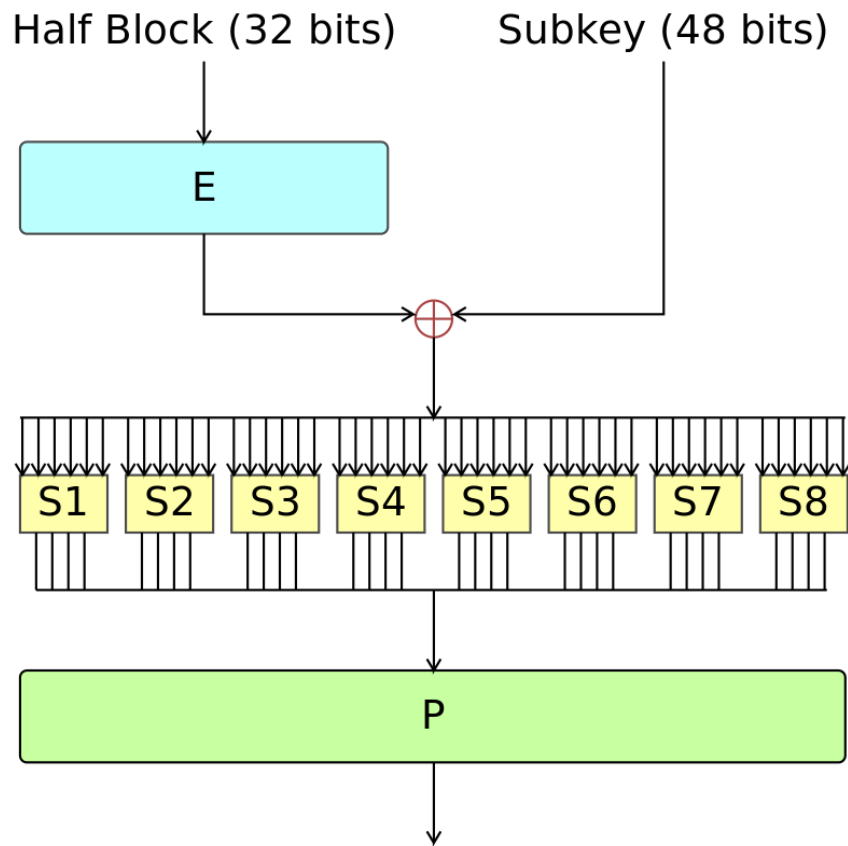
```
/* número de bits que hay que rotar cada semiclave según el número de ronda */  
static const unsigned short ROUND_SHIFTS[ROUNDS] = {  
    1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1  
};
```

- Luego PC-2 permuta y contrae los 56 bits de la subclave rotados a 48 bits: 24 bits de la mitad izquierda y 24 de la derecha .

El Data Encryption Standard (DES): La función F

➤ La función F de cada ronda tiene las siguientes fases:

- **Expansión:** el medio bloque de 32 bits se expande a 48 bits mediante la permutación de expansión.
- **Mezcla de claves:** el resultado se combina con una subclave mediante una operación XOR.
- **Sustitución:** después de mezclar la subclave, el bloque se divide en ocho piezas de 6 bits antes de ser procesado por las cajas S o cajas de sustitución.
- **Permutación:** las 32 salidas de las 8 cajas S se reorganizan según una permutación fija, la caja P.



El *Data Encryption Standard* (DES): La función F

- La expansión y permutación de la función F son las siguientes:

```
/* expansión E */
static const unsigned short E[BITS_IN_E] = {
    32, 1, 2, 3, 4, 5,
    4, 5, 6, 7, 8, 9,
    8, 9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
    16, 17, 18, 19, 20, 21,
    20, 21, 22, 23, 24, 25,
    24, 25, 26, 27, 28, 29,
    28, 29, 30, 31, 32, 1
};

/* permutación P */
static const unsigned short P[BITS_IN_P] = {
    16, 7, 20, 21,
    29, 12, 28, 17,
    1, 15, 23, 26,
    5, 18, 31, 10,
    2, 8, 24, 14,
    32, 27, 3, 9,
    19, 13, 30, 6,
    22, 11, 4, 25
};
```

El Data Encryption Standard (DES): La función F

- Recordar que cada subclave tiene 48 bits y entra en cada ronda a la función F.
- Por eso se puede sumar con la salida de la expansión a través de la función XOR.

**Función de F
de la ronda 2**

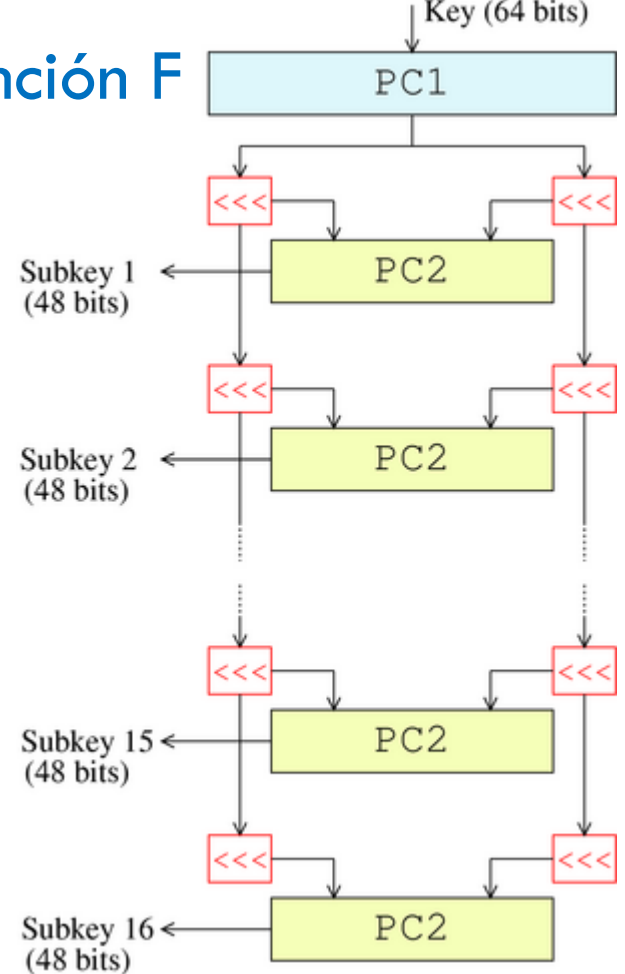
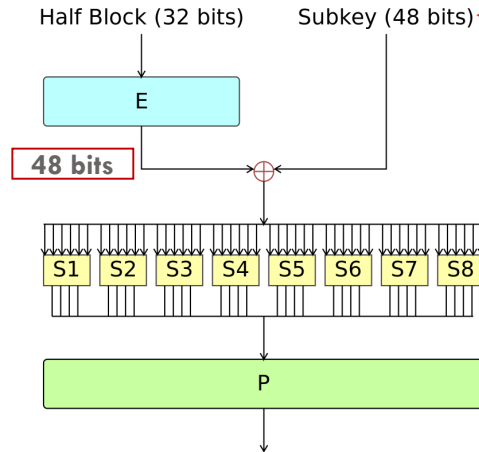


Imagen extraída de https://en.wikipedia.org/wiki/Data_Encryption_Standard.

El Data Encryption Standard (DES): Las cajas S

El corazón de F: las cajas S

- Cada una de las cajas S_i es una sustitución no afín.

- La entrada son 6 bits.

- La salida son 4 bits.

- Organización de los bits de entrada:

$(a_1, a_2, a_3, a_4, a_5, a_6)$

número de fila = (a_1, a_6)

número de columna = (a_2, a_3, a_4, a_5) .

- Ejemplo. La caja S_4 .

Bit | Bits 20, 21, 22, y 23 forman:

19 24 | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

|-----|

0 0 | 7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15

0 1 | 13 8 11 5 6 15 0 3 4 7 2 12 1 10 14 9

1 0 | 10 6 9 0 12 11 7 13 15 1 3 14 5 2 8 4

1 1 | 3 15 0 6 10 1 13 8 9 4 5 11 12 7 2 14

La entrada a la caja es (011001)

fila = 01 y

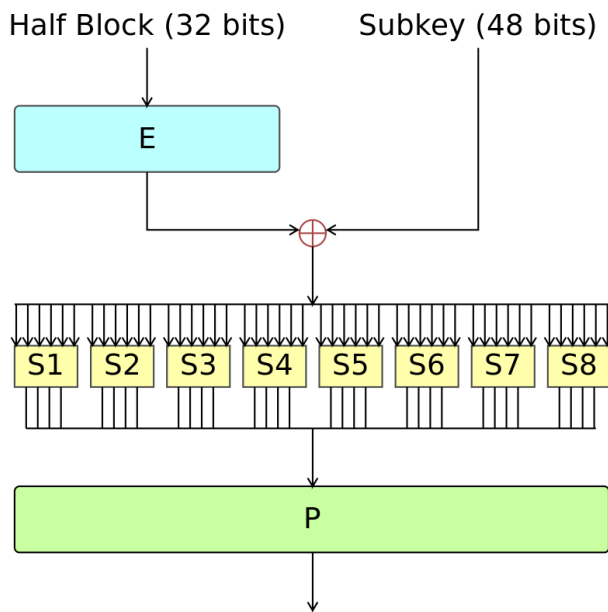
columna = 1100 = 12(decimal).

$S_4(011001) = 1$

- No hay repeticiones en la filas.
- No hay repeticiones en la columnas
- Luego es no singular o invertible.

El Data Encryption Standard (DES): Las cajas S

- La 8 cajas de sustitución dentro de la función F funcionan de manera análoga:



```
static const unsigned short S_BOXES[NUM_S_BOXES][ROWS_PER_SBOX][COLUMNS_PER_SBOX] = {  
    {  
        { 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7 },  
        { 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8 },  
        { 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0 },  
        { 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 }  
    },  
    {  
        { 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10 },  
        { 3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5 },  
        { 0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15 },  
        { 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 }  
    },  
    {  
        { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8 },  
        { 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1 },  
        { 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7 },  
        { 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 }  
    },  
    {  
        { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15 },  
        { 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9 },  
        { 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4 },  
        { 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 }  
    },  
    {  
        { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9 },  
        { 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6 },  
        { 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14 },  
        { 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 }  
    },  
    {  
        { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11 },  
        { 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8 },  
        { 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6 },  
        { 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 }  
    },  
    {  
        { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1 },  
        { 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6 },  
        { 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2 },  
        { 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 }  
    },  
    {  
        { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7 },  
        { 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2 },  
        { 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8 },  
        { 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 }  
    }  
};
```

El *Data Encryption Standard* (DES): El efecto avalancha

- Toda esta estructura para el DES, genera lo que se conoce el efecto avalancha de este algoritmo.
- Es decir un pequeño cambio de en cualquiera de los bits del texto original, produce un cambio muy grande en el texto de cifrado.
- Este efecto avalancha también se produce en la generación de las 16 subclaves del algoritmo DES.
- Es tal este efecto con solo la modificación de un único bit en el texto original, se generan muchos cambios a la salida en el texto cifrado.
- Este algoritmo muestra un gran efecto avalancha, al igual que lo hace el AES cuando lo veamos más adelante.
- En criptografía el efecto avalancha es un efecto muy deseable y lo cumplen muchos algoritmos criptográficos.
- Este efecto está directamente relacionado con los fundamentos de confusión y difusión propuestos por Shannon (que ya comentamos anteriormente).

El Data Encryption Standard (DES): El efecto avalancha

➤	Texto plano	Clave
R	# bits diferentes	# bits diferentes
1	1	0
2	6	2
3	21	14
4	35	28
5	39	32
6	34	30
7	32	32
8	31	35
9	29	34
10	42	40
11	44	38
12	32	31
13	30	28
14	26	26
15	29	34
16	24	35

Input

000

Hash
function

Hash sum

8AEFB06C 426E07A0
A671A1E2 488B4858
D694A730

001

Hash
function

E193A01E CF8D30AD
0AFFEFD3 32CE934E
32FFCE72

010

Hash
function

47AB9979 443FB7ED
1C193D06 773333BA
7876094F

- Texto original: 64 0's: 0.....0.
Se cambia el primer cero.

El *Data Encryption Standard* (DES): Principios de diseño

- En el libro D. Stinson, pag 79 de la primera edición vienen muchas propiedades de principio de diseño, aquí resaltamos las más importantes.
- **A)** La ronda tiene que ser no lineal: en el caso del DES es la ronda es no lineal ya que las cajas S son no lineales: así las rondas del DES cumplen que $f(A \oplus B) \neq f(A) \oplus f(B)$.
- **B)** SAC (Strict Avalanche Criterion): Cualquier bit a la salida , i, de las cajas S debe cambiar con probabilidad $1/2$, cuando cualquier bit de la entrada, j, es complementado para todo i y j:
 - $b_1 \ b_2 \ b_3 \ b_4 \ b_5 \ b_6 \rightarrow S \rightarrow c_1 \ c_2 \ c_3 \ c_4$
 $\forall i, j, \text{ se cumple que } P(c_i=1 \mid \bar{b}_j)=P(c_i=0 \mid \bar{b}_j)=1/2.$
- El SAC se aplica a todas las rondas.

El *Data Encryption Standard* (DES): Principios de diseño

- **C) BIC (Bit Independence Criterion):** Cualquier par de bits, j, k , de la salida de las cajas S debe cambiar independiente cuando cualquier bit de la entrada, i , es complementado para todo i, j y k :
 - $b_1 b_2 b_3 b_4 b_5 b_6 \rightarrow S \rightarrow c_1 c_2 c_3 c_4$
 $\forall i, j, \text{ y } k, \text{ se cumple que } P(c_j, c_k | \bar{b}_i) = P(c_j | \bar{b}_i) P(c_k | \bar{b}_i) .$
- El BIC se aplica a todas las rondas.
- El número de rondas del DES, incrementa la resistencia al ataque del criptoanálisis diferencial.
- Existen varios métodos para la construcción de cajas S :
 - Aleatoria.
 - Aleatoria con comprobación.
 - Manual (así se diseñaron las cajas del DES).

El Data Encryption Standard (DES): Propiedades

- **A) Complementaria:** sea \bar{x} el complementario de x , entonces si $y = E_k(x)$, entonces $\bar{y} = E_{\bar{k}}(\bar{x})$.
- **B) Llaves débiles:** $E_k(E_k(x)) = x$, existen 4 claves débiles.
- **C) Llaves semidébiles:** $E_{k_1}(E_{k_2}(x)) = x$, existen 6 pares claves semidébiles.
- **D) El DES no forma un grupo,** es decir no existe una clave k_3 que para cualquier par de claves k_1 y k_2 se puede tener que $E_{k_3} = E_{k_1}(E_{k_2}(x))$. Esto no permite componer el DES con el mismo aumentando la fortaleza del número de claves.

El Data Encryption Standard (DES): Modos de Operación

- Ahora tenemos ya la estructura básica de funcionamiento de un cifrado de bloque de 64 bit, y la pregunta es cómo se utiliza esta estructura base para cifrar.
- Esto es lo que se conoce como los modos de operación del DES (recomendación de modos de operación para cifrados por bloques).

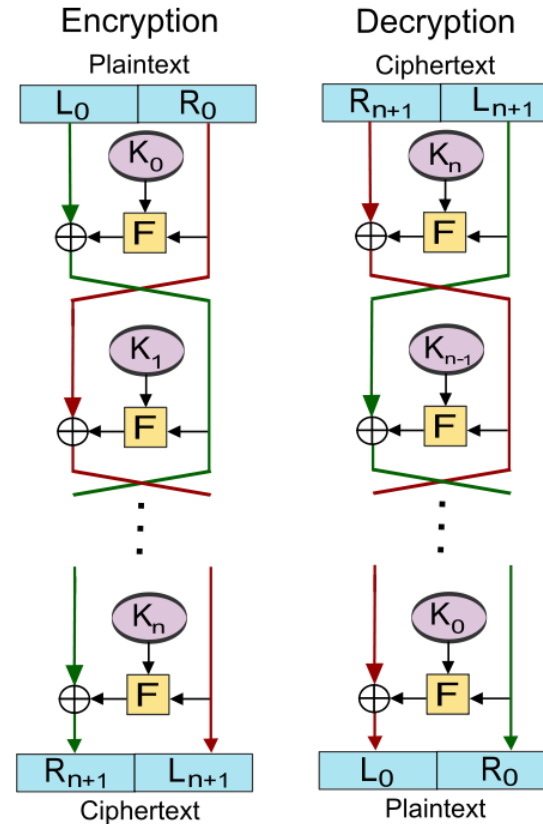
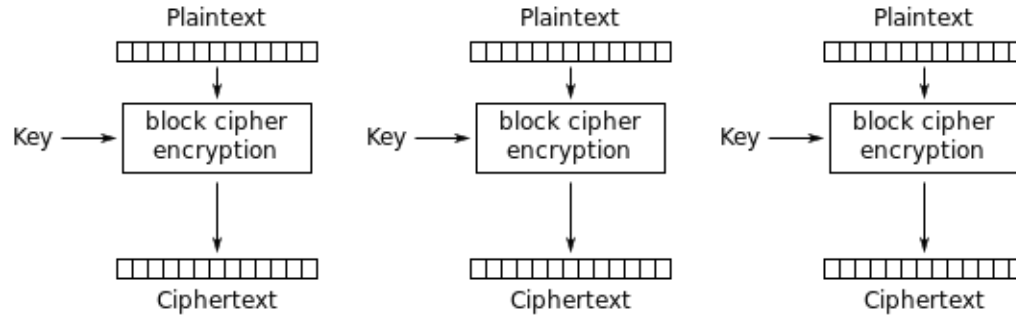


Imagen extraída de
http://commons.wikimedia.org/wiki/File:Feistel_cipher_diagram_en.svg

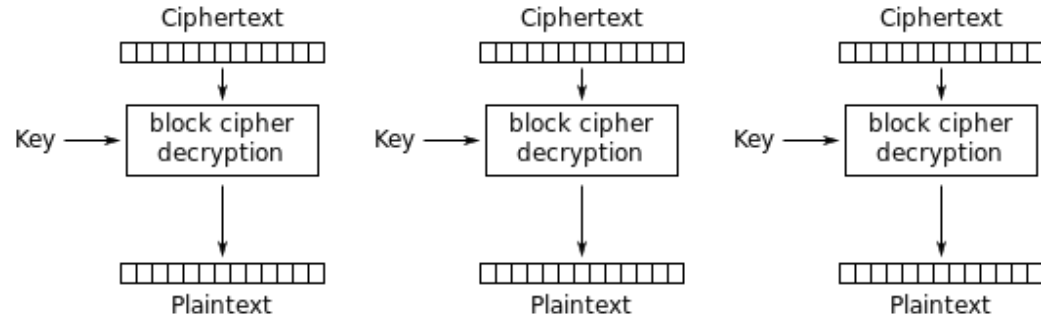
El Data Encryption Standard (DES): Modos de Operación



Electronic Codebook (ECB) mode encryption

FIPS 81 - Des Modes of Operation (HTML)

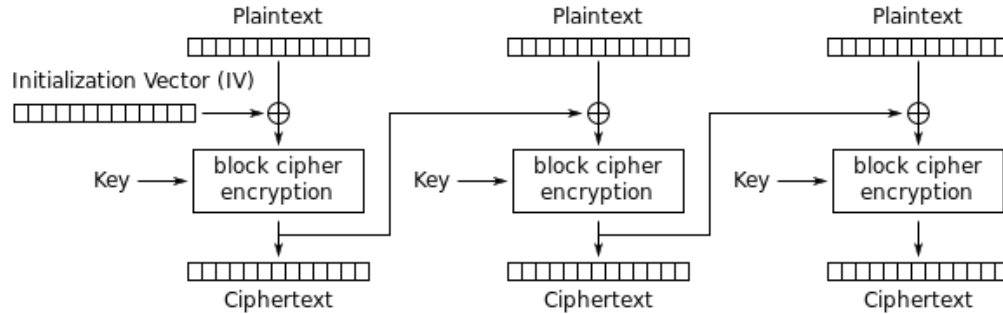
Mirar el detalle de este modo en [recomendación de modos de operación para cifrados por bloques](#) y en el libro.



Electronic Codebook (ECB) mode decryption

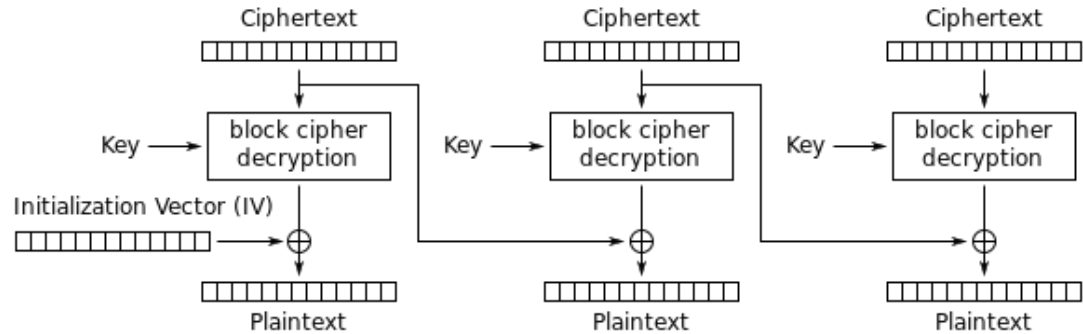
Imagen extraída de https://en.wikipedia.org/wiki/File:ECB_encryption.svg

El Data Encryption Standard (DES): Modos de Operación



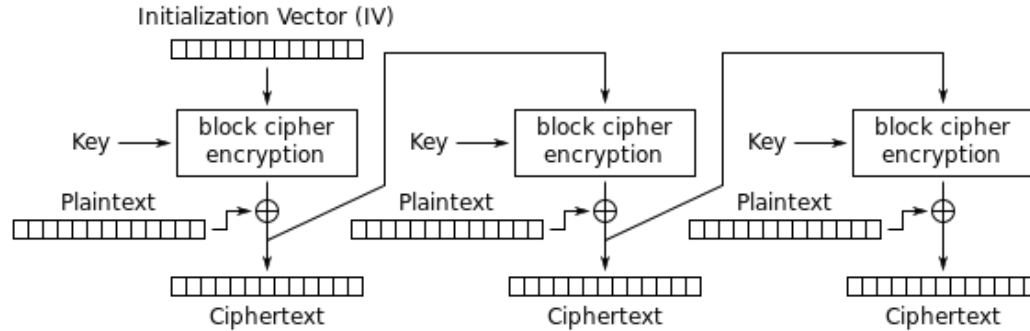
Cipher Block Chaining (CBC) mode encryption

Mirar el detalle de este modo en [recomendación de modos de operación para cifrados por bloques](#) y en el libro.



Cipher Block Chaining (CBC) mode decryption

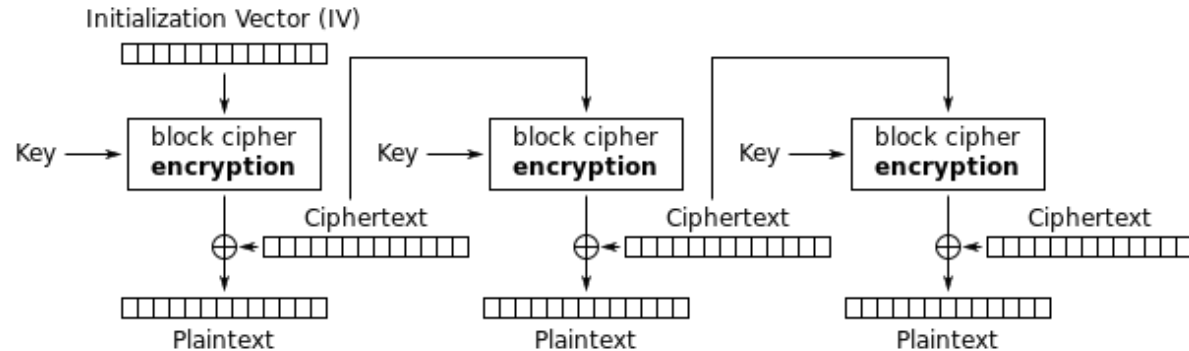
El Data Encryption Standard (DES): Modos de Operación



Cipher Feedback (CFB) mode encryption

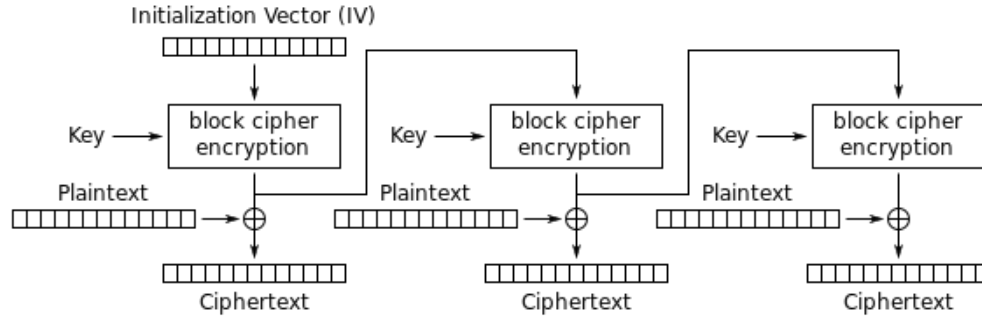
Mirar el detalle de este modo en [recomendación de modos de operación para cifrados por bloques](#) y en el libro.

Mirar el detalle de cómo se pueden cifrar bloques de s bits menor de 64 bits en [recomendación de modos de operación para cifrados por bloques](#) (fig3).



Cipher Feedback (CFB) mode decryption

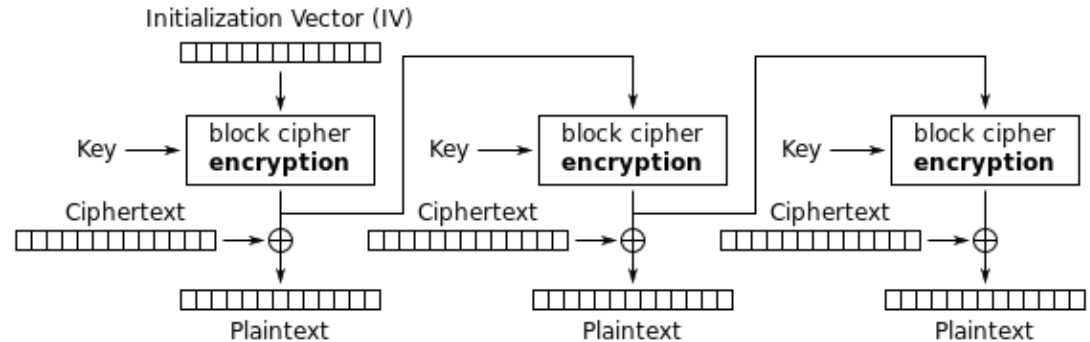
El Data Encryption Standard (DES): Modos de Operación



Output Feedback (OFB) mode encryption

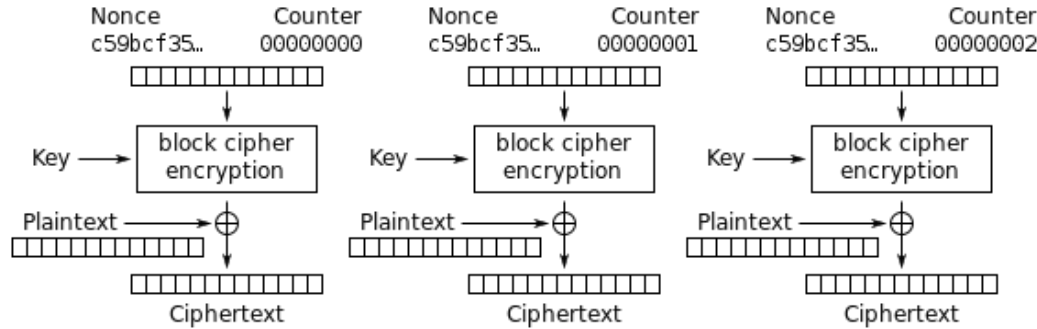
Mirar el detalle de cómo se pueden cifrar bloques de **s bits menor de 64 bits** en recomendación de modos de operación para cifrados por bloques (fig3).

Mirar el detalle de este modo en recomendación de modos de operación para cifrados por bloques y en el libro.

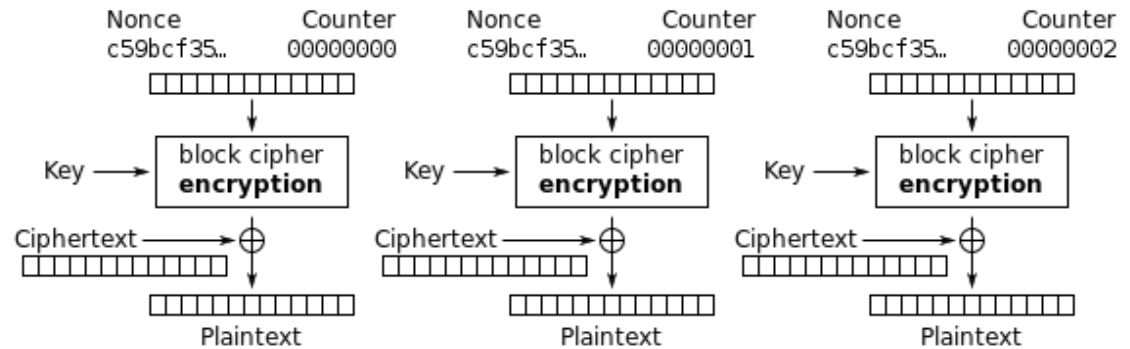


Output Feedback (OFB) mode decryption

El Data Encryption Standard (DES): Modos de Operación



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Mirar el detalle de este modo en [recomendación de modos de operación para cifrados por bloques](#) y en el libro.

El Data Encryption Standard (DES): Versiones Mejoradas

➤ El DES doble:

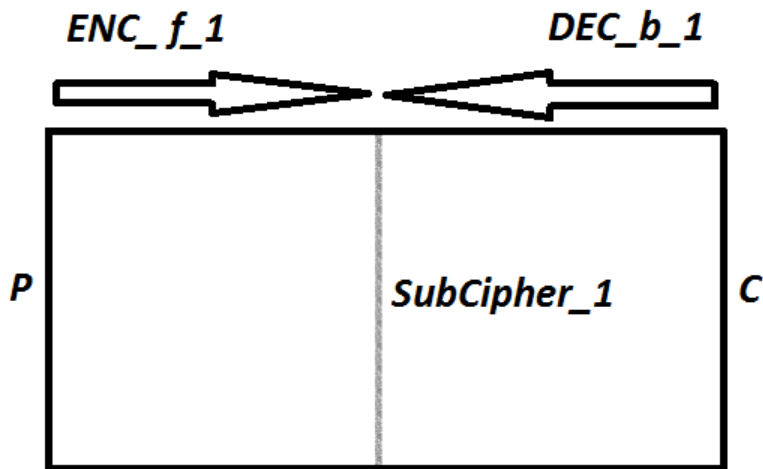
■ $C = EK_2(EK_1(P))$

■ $P = DK_1(DK_2(C))$

➤ Se ataca en el medio:

➤ $E_{K_1}(P) = D_{K_2}(C)$ (2 tablas):

- Comparaciones: $2^{56} \times 2^{56}$ (despreciable vs ejecuciones).
- Ejecuciones: $2^{56} \times 2$.



➤ El TDES o 3DES:

■ $C = EK_3(DK_2(EK_1(P)))$

■ $P = DK_1(EK_2(DK_3(C)))$

➤ Opciones de claves:

- Las tres claves independientes ($3 \times 56 = 168$ bits).
- $K_3 = K_1$ ($2 \times 56 = 112$ bits).
- $K_1 = K_2 = K_3$ (56 bits, por compatibilidad con el antiguo DES).

➤ Información adicional:

- [Triple DES, conocido por TDES o TDEA \(Triple Data Encryption Algorithm\).](#)
- [NIST Special Publication 800-20: Modes of Operation Validation System for the Triple Data Encryption Algorithm \(TMOVS\): Requirements and Procedures.](#)

➤ Su uso:

- [TDES en la actualidad: TDES se utiliza actualmente en estándar de interoperabilidad \(EMV\) de tarjetas y terminales de punto de venta, del tipo Master y Visa. Aparte de este algoritmo se utilizan RSA y SHA.](#)
- [\(SP 800-67 Rev. 2\) - Recommendation for the Triple Data Encryption Algorithm \(TDEA\) Block Cipher.](#)
- [Aunque recientemente el NIST \(27/11/2017\) publicó un boletín que Triple-DES quedará en desuso en el futuro y se rechazará en protocolos como TLS e IPsec.](#)
- [\(800-131A Rev. 2\) - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths.](#)

El *Advanced Encryption Standard* (AES): Historia

- Debido que el DES empezaba a flaquear, sacaron el 3DES o TDES: aunque es lento cuando se ejecuta en *software*:
 - Recordar que el DES fue implementado a mediados de los 70 y fue diseñado para poder meterse en *hardware* (funciones XOR, por ejemplo).
 - El TDES tiene tres veces más ejecuciones, siendo menos óptimo en *software*.
 - Si solo se tuviese en cuenta la seguridad TDES es suficientemente seguro.
 - Aunque el tamaño de bloque del TDES sigue siendo 64 bits: esto merma un poco su seguridad, aunque sigue siendo robusto (recordar el modo de operación ECB con las repeticiones, con tamaños de bloque más grande es mejor).
 - Por todas estas razones el NIST en 1997 hizo una llamada por nuevas propuestas de encriptación para ser utilizado por el gobierno y por el sector privado.

El *Advanced Encryption Standard* (AES): Historia

- La condiciones generales de la propuesta para este nuevo algoritmo eran:
 - Cifrado por bloques.
 - Algoritmo público.
 - Tamaño mínimo de bloque 128 bits.
 - Tamaño de la clave 128, 192 y 256 bits.
- El concurso fue abierto en septiembre de 1997 por NIST buscando algo tan fuerte como el TDES, pero mucho más eficiente.
- Se aceptaron en una primera fase 15 algoritmos de encriptación en agosto de 1998.
- De estos 15 algoritmos solo 5 algoritmos pasaron la criba (remarcados).

El Advanced Encryption Standard (AES): Historia

- Esta información está extraída del libro Joan Daemen, Vicent Rijmen, "The design of Rijndael AES-The Advanced Encryption Standard":

CAST-256	Entrust (CA)	Company
Crypton	Future Systems (KR)	Company
DEAL	Outerbridge, Knudsen (USA-DK)	Researchers
DFC	ENS-CNRS (FR)	Researchers
E2	NTT (JP)	Company
Frog	TecApro (CR)	Company
HPC	Schroepel (USA)	Researcher
LOKI97	Brown et al. (AU)	Researchers
Magenta	Deutsche Telekom (DE)	Company
Mars	IBM (USA)	Company
RC6	RSA (USA)	Company
Rijndael	Daemen and Rijmen (BE)	Researchers
SAFER+	Cylink (USA)	Company
Serpent	Anderson, Biham, Knudsen (UK-IL-DK)	Researchers
Twofish	Counterpane (USA)	Company

El *Advanced Encryption Standard* (AES): Historia

- Los criterios de selección fueron:
 - Seguridad (fue el criterio más importante):
 - Que fuese resistente al criptoanálisis diferencial y lineal.
 - Que tuviese una fuerte base matemática.
 - Aleatoriedad a la salida del algoritmo.
 - Seguridad relativa respecto a los otros candidatos.
- Costes:
 - Licencias y aspectos legales del algoritmo.
 - Eficacia computacional en varias plataformas.
 - Recursos necesarios de memoria.
- Algoritmo e implementaciones:
 - Que fuese flexible para diferentes tamaños de clave.
 - Que fuese flexible para diferentes plataformas.
 - Que tuviese capacidad de desarrollar a partir de este cifrados de flujo y algoritmos de hash.

El *Advanced Encryption Standard* (AES): Historia

➤ **Mars:**

- Innovador en su estructura.
- Muy eficiente en P. Pro/II/III por las rotaciones dependientes de los datos y multiplicaciones de 32 bits.
- No es un buen candidato para *Smart Cards*, ya que utiliza muchos recursos de memoria.

➤ **RC6:**

- El más rápido en PII y PIII, pero no en procesadores de 64 bits.
- No es un buen candidato para *Smart Cards*, ya que utiliza muchos recursos de memoria.
- Cuasi-estructura Feistel con dos claves (es como un doble Feistel).
- La función F utiliza dos subclaves.
- Tiene operaciones muy sencillas.

El *Advanced Encryption Standard* (AES): Historia

➤ **SERPENT:**

- Ultraconservador en seguridad:
 - Utiliza el doble de rondas que el mínimo necesario para los ataques conocidos.
- El más lento de todos.

➤ **Twofish:**

- Muy eficiente.
- Fácil de implementar en *hardware*.
- Estructura Feistel.
- Las cajas S son dinámicas (no estáticas como el DES).
- Por el contrario la generación de las subclaves es lenta.

➤ **Rijndael:**

- Parece ser eficiente en muchas plataformas y apto para *Smart Cards*.
- Funciona bien en CPUs de 32 y 64 bits.
- Es eficiente en *hardware*, y rápido en la construcción de secuencia de claves.
- Es paralelizable y funciona como función Hash.

AES: Rijndael, conceptos previos

- Rijndael → se dice Reindael.
- Se basa en campos de Galois.
- Los campos de Galois son campos limitados finitos, $GF(p^n)$, donde p es primo
- Así los campos de Galois cumplen:
 - Campo finito:
 - Por ejemplo:
 - $(\mathbb{Z}_m, +) \rightarrow$ Grupo conmutativo abeliano
 - $(\mathbb{Z}_m, +, *) \rightarrow$ Anillo
 - Si $m = p$ primo $\rightarrow \mathbb{Z}_p^* \cup 0 = \mathbb{Z}_p$
 - $(\mathbb{Z}_p, +, *) \rightarrow$ Campo finito
 - $|GF(p^n)| = p^n$.
- En esencia un campo de Galois es un anillo conmutativo, en el cual todos los elementos distintos de 0 tienen inverso multiplicativo:
 - Anillo con las operaciones $(+, *)$, y además inverso en la operación $*$.
- Las operaciones se implementan en polinomios. Para una descripción más profunda de las operaciones podéis mirar los libros de la asignatura. Aquí vamos a hacer una breve descripción de estas operaciones.

AES: Rijndael, Definiciones sobre polinomios

- **Definición 1:** Definimos el espacio $Z_p[x]$ con $f(x) \in Z_p[x]$ el polinomio $f(x) = a_n x^n + \dots + a_1 x^1 + a_0$, con $a_i \in Z_p$, con $i=1, \dots, n$.
- Ejemplo: $f(x) = x^3 + x + 1$, y $g(x) = x^2 + x$, son polinomios $\in Z_2[x]$.
 - $f(x) + g(x) = x^3 + x^2 + 1$.
 - $f(x) * g(x) = x^5 + x^4 + x^3 + x$.
 - La aritmética modular entre los coeficientes de los polinomios se realiza en Z_2 .
- **Definición 2:** Se define como el grado del polinomio, como dado $f(x) \in Z_p[x]$, el $\text{grado}(f(x)) \equiv$ exponente máximo cuyo coeficiente $a_i \neq 0$.
- **Definición 3:** Dados $f(x), g(x) \in Z_p[x]$, decimos que $f(x) \mid g(x)$, si podemos encontrar un $q(x)$, tal que verifica que $g(x) = q(x) * f(x)$.

AES: Rijndael, Definiciones sobre polinomios

- **Definición 4:** Es la equivalente a aritmética modular en enteros. Dados $f(x), g(x), h(x) \in Z_p[x]$, decimos que $f(x) \equiv g(x) \bmod h(x)$, si se cumple que $h(x) \mid (f(x) - g(x)) \rightarrow f(x) = g(x) + k(x) * h(x)$.
- **Hecho 1:** Dados $f(x), g(x) \in Z_p[x]$, podemos encontrar un $f(x) = q(x) * g(x) + r(x)$, tal que $q(x), r(x) \in Z_p[x]$ sean únicos. El polinomio $q(x)$ es un cociente, el polinomio $r(x)$ es un resto.
- Ejemplo 1: $f(x) \bmod g(x) = r(x)$.
- Ejemplo 2: $f(x) = x^6 + x^5 + x^3 + x^2 + x + 1$, y $g(x) = x^4 + x^3 + 1$, son polinomios $\in Z_2[x] \rightarrow f(x) = x^2 * g(x) + (x^3 + x^1 + 1)$.
- **Definición 5:** Definimos un nuevo conjunto $Z_p[x]/m(x)$, donde $m(x) \in Z_p[x]$ siendo este lo que se denomina un polinomio irreducible.

AES: Rijndael, Definiciones sobre polinomios

- **Definición 6:** El polinomio $m(x) \in Z_p[x]$, es al menos de grado 1 y es irreducible si no se puede expresar como producto de dos polinomios de grado positivo. Es equivalente a un número primo en el espacio de polinomios $Z_p[x]$. Recordemos que un número primo es aquel que solo es divisible por 1 y por el mismo. Es decir que $m(x)$ y es irreducible si no existe un $f(x) \in Z_p[x]$ tal que $f(x) \mid m(x)$.
- Vamos ahora precisar más la definición del nuevo conjunto $Z_p[x]/m(x)$:
 - Este conjunto son todos los polinomios $\in Z_p[x]$ de grado menor que $n = \text{grado}(m(x))$.
 - Las operaciones $+$ y $*$, se realizan en mod $m(x)$.
 - Este nuevo conjunto, $Z_p[x]/m(x)$, se puede demostrar que es un anillo conmutativo.
 - Así se puede demostrar que esto es lo que se denomina campo de Galois.

AES: Rijndael, Definición de campo de Galois

- Por tanto tenemos que $Z_p[x]/m(x) \equiv \text{GF}(p^n)$:
 - p es primo.
 - n es el grado de polinomio irreducible $m(x)$.
 - p^n es la cardinalidad del campo.
 - Hay dos tipos de operaciones:
 - Operaciones entre polinomios mod $m(x)$.
 - Operaciones entre los coeficientes de los polinomios se hacen en Z_p .
 - Así los elementos del campo son del siguiente tipo (siempre con grado $n-1$ o menor):
 - $a(x) = a_{n-1}x^{n-1} + \dots + a_1x^1 + a_0$, con $a_i \in Z_p$, con $i=1, \dots, n-1$.
 - $m(x)$ es un polinomio irreducible de grado n que no puede ser factorizado en polinomios grado positivos menor que n .

AES: Rijndael, Definición de campo de Galois

- Ejemplo, $p=2$: $GF(p^3) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$, hay al menos dos polinomios irreducibles:
 - $m_1(x) = x^3 + x + 1$.
 - $m_2(x) = x^3 + x^2 + x + 1$.
- Vamos a ver por ejemplo que se cumple la propiedad de cierre:
 - $p_1(x) = x^2 + 1$.
 - $p_2(x) = x^2 + x$.
 - $p_1(x) * p_2(x) \bmod (x^3 + x + 1) = (x^4 + x^3 + x^2 + x) \bmod (x^3 + x + 1) = x+1$. Por lo tanto el resultado $\in GF(p^3)$.
 - | | |
|-------|------|
| 11110 | 1011 |
| 1011 | 11 |
| <hr/> | |
| 01000 | |
| 1011 | |
| <hr/> | |
| 0011 | |

AES: Euclides y Euclides extendido en campo de Galois

- Vamos a extender el algoritmo de Euclides y Euclides Extendido a campos de Galois.
- Recordar que el algoritmo de Euclides se basa en la propiedad de que $d(x) = \text{mcd}(a(x), b(x)) = \text{mcd}(b(x), a(x) \bmod b(x))$

$$d(x) = \text{mcd}(a(x), b(x))$$

$$r_0(x) = a(x), r_1(x) = b(x)$$

$$r_0(x) = q_1(x) r_1(x) + r_2(x)$$

$$r_1(x) = q_2(x) r_2(x) + r_3(x)$$

$$r_2(x) = q_3(x) r_3(x) + r_4(x)$$

$$\dots\dots\dots r_{n-2}(x) = q_{n-1}(x) r_{n-1}(x) + r_n(x)$$

$$r_{n-1}(x) = q_n(x) r_n(x) + 0$$

$$d(x) = r_n(x)$$

$$\begin{aligned} d(x) &= \text{mcd}(a(x), b(x)) = \text{mcd}(r_0(x), r_1(x)) = \text{mcd}(r_1(x), r_2(x)) = \dots = \\ &= \text{mcd}(r_{n-2}(x), r_{n-1}(x)) = \text{mcd}(r_{n-1}(x), r_n(x)) = \text{mcd}(r_n(x), 0) = r_n(x) \end{aligned}$$

AES: Euclides y Euclides extendido en campo de Galois

- El algoritmo extendido de Euclides calcula el inverso multiplicativo de un polinomio en este caso en aritmética modular $m(x)$.
- Supongamos que $\text{mcd}(m(x), a(x)) = 1$.
- En el algoritmo de Euclides anterior podemos poner todos los restos $r_i(x)$ en función solo de $r_0(x)$ y $r_1(x)$ (siendo $r_0(x)=m(x)$, $r_1(x)=a(x)$), de manera recursiva obteniendo así la expresión del tipo para el último resto $r_n(x)$, que es $\text{mcd}(m(x), a(x))=1$:
 - $1 = r_n(x) = m(x) u_n(x) + a(x) v_n(x)$
- Por tanto con esta expresión ya tenemos el inverso multiplicativo de a en aritmética modular $m(x)$, por la propia definición de congruencia:
 - $m(x) u_n(x) + a(x) v_n(x) = 1 \rightarrow a(x) v_n(x) \equiv 1 \pmod{m(x)} \rightarrow v_n(x) = a(x)^{-1} \pmod{m}$

AES: Euclides y Euclides extendido en campo de Galois

- Ejemplo: $GF(2^3) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$, calcular $[x^2]^{-1}$ con el siguiente polinomio irreducible:

➤ $m(x) = x^3 + x + 1$.

➤ $\text{mcd}(m(x), x^2)$

1. $x^3 + x + 1 = x * x^2 + (x + 1)$

2. $x^2 = (x + 1) * (x + 1) + 1$

3. $x + 1 = (x + 1) * 1 + 0$

4. $\text{mcd}(m(x), x^2) = 1$

$$\begin{array}{r} 1011 \overline{)100} \\ \underline{100} \\ 0011 \end{array}$$

$\frac{x^3+x+1}{x^2}$

$$\begin{array}{r} 100 \overline{)11} \\ \underline{11} \\ 010 \end{array}$$

$\frac{x^2}{x+1}$

$$\begin{array}{r} 11 \overline{)11} \\ \underline{11} \\ 010 \end{array}$$

$$\begin{array}{r} 11 \overline{)11} \\ \underline{11} \\ 010 \end{array}$$

$$\begin{array}{r} 11 \overline{)11} \\ \underline{11} \\ 010 \end{array}$$

$$\begin{array}{r} 11 \overline{)11} \\ \underline{11} \\ 010 \end{array}$$

$$\begin{array}{r} 11 \overline{)11} \\ \underline{11} \\ 010 \end{array}$$

$$\begin{array}{r} 11 \overline{)1} \\ \underline{1} \\ 01 \end{array}$$

$$\begin{array}{r} 11 \overline{)1} \\ \underline{1} \\ 01 \end{array}$$

$$\begin{array}{r} 11 \overline{)1} \\ \underline{1} \\ 01 \end{array}$$

$$\begin{array}{r} 11 \overline{)1} \\ \underline{1} \\ 01 \end{array}$$

$$\begin{array}{r} 11 \overline{)1} \\ \underline{1} \\ 01 \end{array}$$

$$\begin{array}{r} 11 \overline{)1} \\ \underline{1} \\ 01 \end{array}$$

$$\frac{x+1}{1}$$

AES: Euclides y Euclides extendido en campo de Galois

➤ Ejemplo: $GF(2^3) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$, calcular $[x^2]^{-1}$ con el siguiente polinomio irreducible:

➤ $m(x) = x^3 + x + 1.$

➤ $mcd(m(x), x^2)$

$$x^3 + x + 1 = x * x^2 + (x + 1) \quad (**) \rightarrow x + 1 = (x^3 + x + 1) + x * x^2$$

$$x^2 = (x + 1) * (x + 1) + 1 \quad (*) \rightarrow 1 = x^2 - (x + 1) * (x + 1)$$

$$x + 1 = (x + 1) * 1 + 0$$

Despejamos de (*) $1 = x^2 - (x + 1) * (x + 1) =$
 $x^2 + (x + 1) * (x + 1) =$

Despejamos $(x + 1)$ de (**) y sustituimos =

$$x^2 + (x + 1) * (x^3 + x + 1 + x * x^2) =$$

$$x^2 + (x + 1) * (x^3 + x + 1) + x * (x + 1) * x^2 =$$

$$(x^3 + x + 1) * (x + 1) + x^2(x^2 + x + 1) = 1 \rightarrow [x^2]^{-1} = (x^2 + x + 1)$$

AES: Euclides y Euclides extendido en campo de Galois

- Comprobar que es correcto $[x^2]^{-1} = (x^2 + x + 1)$.
- Más ejemplos de división de polinomios en aritmética modular 2:

$$\begin{array}{r}
 1x^{13}+1x^{12}+0x^{11}+1x^{10}+0x^9+1x^8+1x^7+0x^6+1x^5+1x^4+0x^3+0x^2+0x^1+0 \\
 \underline{1x^{13}+0x^{12}+0x^{11}+1x^{10}+1x^9} \\
 0x^{13}+1x^{12}+0x^{11}+0x^{10}+1x^9+1x^8 \\
 \quad \underline{1x^{12}+0x^{11}+0x^{10}+1x^9+1x^8} \\
 \quad 0x^{12}+0x^{11}+0x^{10}+0x^9+0x^8+1x^7+0x^6+1x^5+1x^4+0x^3 \\
 \quad \quad \underline{1x^7+0x^6+0x^5+1x^4+1x^3} \\
 \quad \quad 0x^7+0x^6+1x^5+0x^4+1x^3+0x^2+0x^1 \\
 \quad \quad \quad \underline{1x^5+0x^4+0x^3+1x^2+1x^1} \\
 \quad \quad \quad 0x^5+0x^4+1x^3+1x^2+1x^1+0
 \end{array}
 \quad
 \begin{array}{r}
 1x^4+0x^3+0x^2+1x^1+1 \\
 \hline
 1x^9+1x^8+0x^7+0x^6+0x^5+0x^4+1x^3+0x^2+1x^1+0
 \end{array}$$

AES: Euclides y Euclides extendido en campo de Galois

- Mismo ejemplo de división de polinomios en aritmética modular 2:

$$\begin{array}{r} 11010110110000 \mid 10011 \\ 10011 \\ \hline 010011 \\ 10011 \\ \hline 0000010110 \\ 10011 \\ \hline 0010100 \\ 10011 \\ \hline 001110 \end{array}$$

Base matemática del AES

- La base matemática del AES son los campos de Galois de orden 2^8 , con el polinomio irreducible: $m(x) = x^8 + x^4 + x^3 + x + 1$.
- Es decir es el conjunto que viene dado por
$$\text{GF}(2^8) \equiv \mathbb{Z}_2[x]/m(x) \equiv \mathbb{Z}_2[x]/x^8 + x^4 + x^3 + x + 1$$
- La representación que utilizamos es la binaria:
 - $p(x) = x^6 + x^4 + x^2 + x + 1 \equiv 01010111_2 \equiv '57'_{16}$.
- Se suele denotar la función binaria y hexadecimal como:
 - $b(p(x)) = 01010111_2, H(p(x)) = '57'_{16}$.
- Supongamos que tenemos 2 polinomios:
 - $p_a(x) = a_7x^7 + \dots + a_1x^1 + a_0 \rightarrow b(p_a(x)) = (a_7 \dots a_1, a_0)$.
 - $p_b(x) = b_7x^7 + \dots + b_1x^1 + b_0 \rightarrow b(p_b(x)) = (b_7 \dots b_1, b_0)$.
 - $H(p_a(x)) = ([a_7 \dots a_4][a_3 \dots a_0])$

Base matemática del AES: Operaciones básicas

- Ahora vamos a ver como son detalladamente las operaciones del AES.

- **Suma:**

$$p_a(x) + p_b(x) = \sum_{i=0}^7 ((a_i + b_i) \bmod 2) x^i$$
$$b(p_a(x) + p_b(x)) = b(p_a(x)) \oplus b(p_b(x))$$

- **Producto:** El producto se realiza a través del producto de polinomios con coeficientes en módulo 2. El resultado de la operación se reduce mediante el polinomio irreducible $m(x) = x^8 + x^4 + x^3 + x + 1 = 100011011_2 \equiv '11B'_{16}$.
- Ejemplo: $'57' \cdot '88' = 'C1'$
 $(x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) = (x^7 + x^6 + 1)$ (ejercicio para casa).

Base matemática del AES: Optimización del producto

- Se realiza la optimización del producto en el AES para implementar esta operación a nivel de Byte.
- La idea es poner la multiplicaciones en función de $x.a(x)$.
- Supongamos $a(x) = a_7x^7 + \dots + a_1x^1 + a_0$.
- Así se define la función $xtime(a(x)) = x.a(x)$.
- Si multiplicamos $x.a(x) = a_7x^8 + \dots + a_1x^2 + a_0x \bmod m(x)$.
- Podemos distinguir dos casos:
 - **Caso 1:** $a_7 = 0 \rightarrow x.a(x) = a_6x^7 + \dots + a_1x^2 + a_0x = L(b(a(x)))$.
Es decir es un desplazamiento de los bits del polinomio hacia la izquierda.
 - **Caso 2:** $a_7 = 1 \rightarrow x.a(x) = a_7x^8 + \dots + a_1x^2 + a_0x \bmod m(x)$.

Base matemática del AES: Optimización del producto

- En este caso 2 , se puede hacer la siguiente observación:
 - $x^8 \bmod m(x) = m(x) - x^8 = x^4 + x^3 + x + 1$.
 - Así $x \cdot a(x) = x^8 + \dots + a_1 x^2 + a_0 x \bmod m(x) =$
 $[x^8 \bmod m(x)] + [a_6 x^7 + \dots + a_1 x^2 + a_0 x \bmod m(x)] =$
 $[x^4 + x^3 + x + 1] + [a_6 x^7 + \dots + a_1 x^2 + a_0 x] =$
 $L(b(a(x))) \oplus' 1B'$.
- Así podemos poner:
 - $$\text{xtime}(a(x)) = x \cdot a(x) = \begin{cases} \text{Si } a_7 = 0 \rightarrow L(b(a(x))) \\ \text{Si } a_7 = 1 \rightarrow L(b(a(x))) \oplus' 1B'. \end{cases}$$

Base matemática del AES: Optimización del producto

- ¿Y para qué sirve todo esto? Vamos a ver un ejemplo para su aplicación.
- Ejemplo: $'57'. '13'$, tenemos que darnos cuenta que $'13' = 1 + x + x^4$
$$'57'. '13' = '57'. ('01' \oplus '02' \oplus '10') =$$
$$'57'. '01' \oplus '57'. '02' \oplus '57'. '10' =$$
- Ahora podemos calcular todos los *xtime* de manera recursiva:
$$'57'. '02' = \text{xtime}('57') = 'AE'$$
$$'57'. '04' = \text{xtime}('AE') = '47'$$
$$'57'. '08' = \text{xtime}('47') = '8E'$$
$$'57'. '10' = \text{xtime}('8E') = '07'$$
- Así tenemos $'57'. '01' \oplus '57'. '02' \oplus '57'. '10' = '57' \oplus 'AE' \oplus '07' = 'FE'$
 $= '57' \oplus \text{xtime}('57') \oplus \text{xtime}(\text{xtime}(\text{xtime}(\text{xtime}('57')))))$.
- Esta implementación **recursiva** es muy eficiente.

Operaciones AES: Polinomios con coeficientes en $\text{GF}(2^8)$

- Vamos a una vuelta de tuerca más, va a suponer los siguientes polinomios cuyos coeficientes son polinomios en $\text{GF}(2^8)$ (son palabras de bytes):
- $A(x) = \mathbf{a_3}x^3 + \mathbf{a_2}x^2 + \mathbf{a_1}x^1 + \mathbf{a_0}$, con $\mathbf{a_i} \in \text{GF}(2^8)$.
- $B(x) = \mathbf{b_3}x^3 + \mathbf{b_2}x^2 + \mathbf{b_1}x^1 + \mathbf{b_0}$, con $\mathbf{b_i} \in \text{GF}(2^8)$.
- El producto de ambos polinomios es:
- $$C(x) = A(x) * B(x) = a_3b_3x^6 + a_2b_3x^5 + a_1b_3x^4 + a_0b_3x^3 + a_3b_2x^5 + a_2b_2x^4 + a_1b_2x^3 + a_0b_2x^2 + a_3b_1x^4 + a_2b_1x^3 + a_1b_1x^2 + a_0b_1x^1 + a_3b_0x^3 + a_2b_0x^2 + a_1b_0x^1 + a_0b_0 = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x^1 + c_0.$$
- Siendo los coeficientes c_i :
- $$c_0 = a_0b_0$$
- $$c_1 = a_1b_0 \oplus a_0b_1$$
- $$c_2 = a_2b_0 \oplus a_1b_1 \oplus a_0b_2$$
- $$c_3 = a_3b_0 \oplus a_2b_1 \oplus a_1b_2 \oplus a_0b_3$$
- $$c_4 = a_3b_1 \oplus a_2b_2 \oplus a_1b_3$$
- $$c_5 = a_3b_2 \oplus a_2b_3$$
- $$c_6 = a_3b_3$$

Operaciones AES: Polinomios con coeficientes en $\text{GF}(2^8)$

- Por favor notar que
 - $\oplus \rightarrow \text{xor}$.
 - $a_i b_j \rightarrow \text{xtime}$.
- Lo que interesa es que al multiplicar estos coeficientes, el resultado no sobrepase el vector formado por cuatro bytes.
- Por tanto en este caso se reduce por un polinomio de orden 4 (en este caso no es irreducible): $M(x) = x^4 + 1$.
- Para llevar a cabo esta reducción se utiliza la propiedad que este caso dividir x^i por $M(x) = x^4 + 1$, equivale a:
 - $x^i \bmod x^4 + 1 = x^{i \bmod 4}$.
- Así el producto modular $D(x) = A(x) * B(x) \bmod M(x) = d_3 x^3 + d_2 x^2 + d_1 x^1 + d_0$, con $d_i \in \text{GF}(2^8)$.

Operaciones AES: Polinomios con coeficientes en $GF(2^8)$

- Los coeficientes d_i :
 - $d_0 = a_0b_0 \oplus a_3b_1 \oplus a_2b_2 \oplus a_1b_3$
 - $d_1 = a_1b_0 \oplus a_0b_1 \oplus a_3b_2 \oplus a_2b_3$
 - $d_2 = a_2b_0 \oplus a_1b_1 \oplus a_0b_2 \oplus a_3b_3$
 - $d_3 = a_3b_0 \oplus a_2b_1 \oplus a_1b_2 \oplus a_0b_3$
- Es decir:
 - $d_0 \rightarrow (c_0 \oplus c_4)x^0$
 - $d_1 \rightarrow (c_1 \oplus c_5)x^1$
 - $d_2 \rightarrow (c_2 \oplus c_6)x^2$
 - $d_3 \rightarrow c_3x^3$

- En notación matricial tenemos:

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

- El producto se realiza mediante la función xtime.

Operaciones AES: XTIME con polinomios con coeficientes en $\text{GF}(2^8)$

- La función XTIME multiplica un polinomio $B(x)$ con coeficientes en $\text{GF}(2^8)$ por el polinomio x :

- $\text{XTIME}(B(x)) =$
 $xB(x) = C(x) = b_3x^4 + x^3 + b_1x^2 + b_0x \bmod M(x), \text{ con } b_i \in \text{GF}(2^8).$

- En notación matricial:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

- $\text{XTIME}(B(x)) = \text{XTIME}(b_3, b_2, b_1, b_0) = (b_2, b_1, b_0, b_3) = (c_3, c_2, c_1, c_0)$

- **HASTA AQUÍ PRUEBA INTERMEDIA**

AES: Rijndael: objetivos del diseño

- Todas las operaciones que hemos visto anteriormente conforman la base matemática del AES.
- El Rijndael puede tener un tamaño de bloque de 128, 192 y 256 bits, pero en el AES se limita solo a 128 bits.
- Los principales objetivos de diseño del AES son:
 - Resistente a criptoanálisis diferencial y lineal.
 - Eficacia en diferentes plataformas:
 - Sí que lo es, ya que hemos visto que las principales operaciones que se realizan son desplazamientos y rotaciones de bytes.
 - La idea es una fácil y óptima implementación en “*Smart Cards*”.
 - Código compacto y sencillo.
 - Fuerte base matemática.

AES, Rijndael: fundamentos del diseño

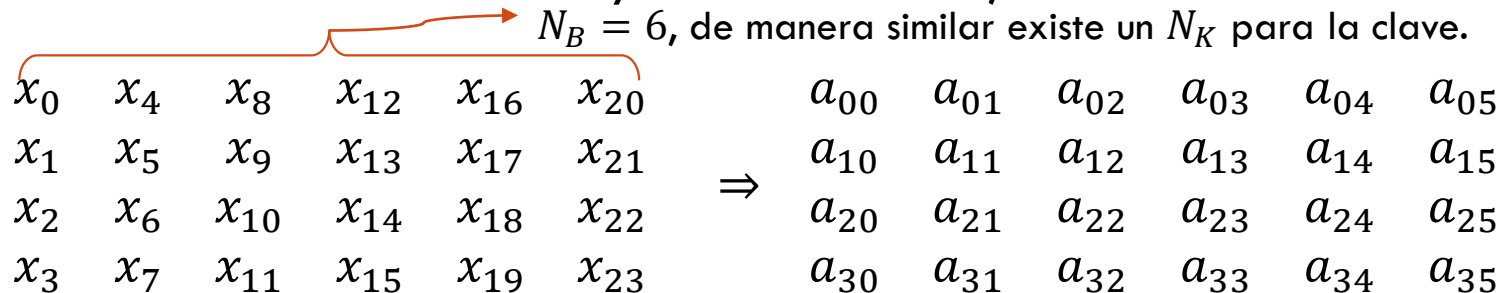
- Este algoritmo es el nuevo estándar de cifrado simétrico desde 2001.
- No está basado en el cifrado Feistel, sino en redes de sustitución y permutación (la función directa e inversa son diferentes).

Tamaño de clave (W/B/b)	4/16/128	6/24/192	8/32/256
Tamaño de bloque (W/B/b)	4/16/128	4/16/128	4/16/128
Numero de Rondas	10	12	14
Clave expandida (W/B)	44/176	52/208	60/240

- El bloque sufre cuatro transformaciones en cada ronda:
 - SubBytes
 - ShiftRows
 - MixColumns
 - AddRoundKey

AES, Rijndael: STATE

- Las diferentes transformaciones del algoritmo operan en un estadio intermedio denominado STATE.
- Supongamos que la cadena de bytes a cifrar viene determinado por la secuencia:
 - x_0, x_1, \dots, x_n , con $x_i \in \text{GF}(2^8)$.
- Supongamos un tamaño de bloque a cifrar de 192 bits (24 bytes), notar por favor que esto es solo posible en el Rijndael (en el AES el tamaño de bloque es siempre 128 bits).
- Así el STATE para un bloque de 192 bits tendrá el siguiente aspecto (observar el orden de los bytes en el STATE):



AES, Rijndael: N_B y N_k

- Se define el parámetro N_B , para el tamaño del bloque en Rijndael:
 - $N_B = \frac{\text{longitud de bloque}}{32}$.
 - $N_B = 4 \rightarrow$ tamaño de bloque 128 bits.
 - $N_B = 6 \rightarrow$ tamaño de bloque 192 bits.
 - $N_B = 8 \rightarrow$ tamaño de bloque 256 bits.
- Se define el parámetro N_k , para el tamaño de clave en AES:
 - $N_k = \frac{\text{longitud de clave}}{32}$.
 - $N_k = 4 \rightarrow$ tamaño de clave 128 bits.
 - $N_k = 6 \rightarrow$ tamaño de clave 192 bits.
 - $N_k = 8 \rightarrow$ tamaño de clave 256 bits.

AES, Rijndael: Número de Rondas N_R

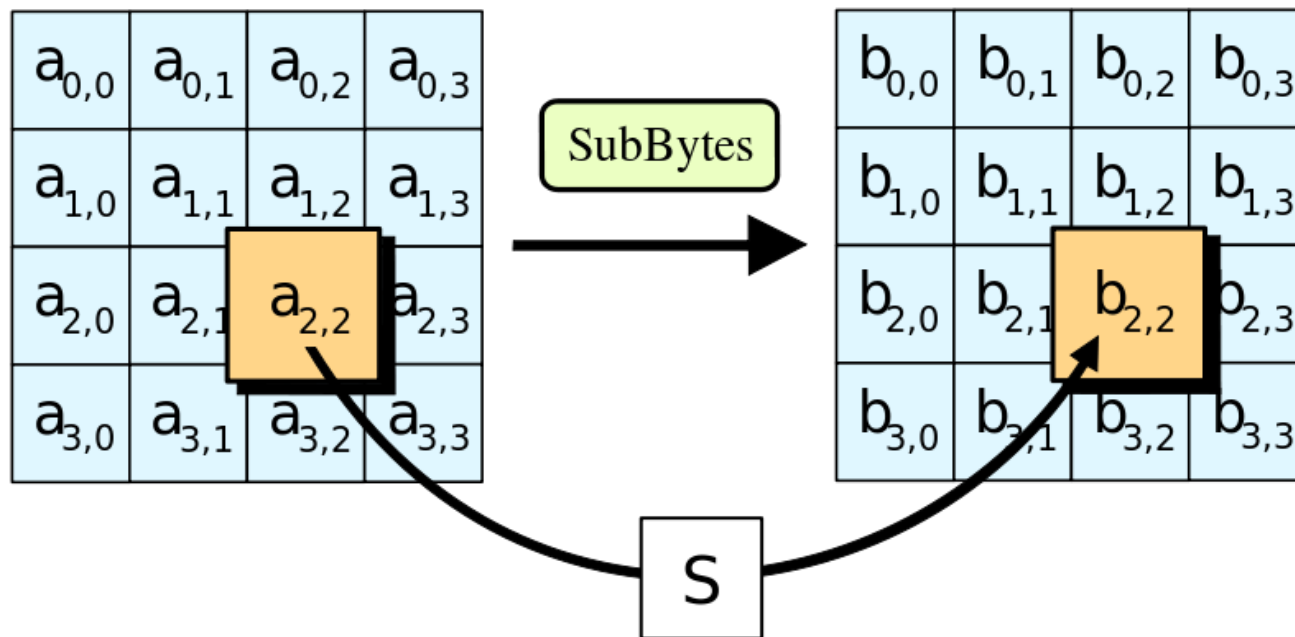
- El número de rondas N_R para el Rijndael varía en función de N_B y N_K , como se muestra en la siguiente tabla:

N_R	$N_B = 4$	$N_B = 6$	$N_B = 8$
$N_k = 4$	10	12	14
$N_k = 6$	12	12	14
$N_k = 8$	14	14	14

- Recordar que el AES es solo $N_B = 4$.
- Como hemos comentado antes en cada ronda se ejecutan:
 - SubBytes
 - ShiftRows
 - MixColumns (la roda final no ejecuta esta función)
 - AddRoundKey

AES, Rijndael: SubBytes

Imagen extraída de
<http://commons.wikimedia.org/wiki/File:AES-SubBytes.svg>



- State: transformaciones que sufre el bloque a lo largo del cifrado.
- Caja de sustitución por bytes del bloque.
- Operaciones matemáticas basadas en campos finitos (polinomios de Galois)

AES, Rijndael: SubBytes

- Los pasos para calcular la función “SubBytes” son:
1. Se calcula el multiplicativo inverso para cada $a_{ij} \rightarrow a'_{ij}$ en el STATE en $GF(2^8)$ con $m(x) = x^8 + x^4 + x^3 + x + 1$, el caso especial del byte 00 se transforma en byte 00.
 2. Se aplica la siguiente transformación afín al resultado al byte resultante a'_{ij} :

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a'_7 \\ a'_6 \\ a'_5 \\ a'_4 \\ a'_3 \\ a'_2 \\ a'_1 \\ a'_0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

AES, Rijndael: SubBytes

- Realmente esto es como una caja de sustitución para cada byte, ya que cada byte del STATE se transforma en otro byte.
- De esta forma se obtiene la caja de sustitución S para el AES:
 - $b_{ij} = S\text{-}box[a_{ij}]$.
- Ejemplo:
 - $9A \rightarrow B8$
- Calcular para casa $\text{SubBytes}(x^7 + 1)$ utilizando: Euclides, Euclides extendido y la transformación afín anterior.

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

The column is determined by the least significant nibble, and the row by the most significant nibble. For example, the value $9a_{16}$ is converted into $b8_{16}$.

AES, Rijndael: InvSubBytes

➤ Los pasos para calcular la función “InvSubBytes” son:

1. Se aplica la siguiente transformación afín inversa a la anterior:

$$\begin{pmatrix} a'_7 \\ a'_6 \\ a'_5 \\ a'_4 \\ a'_3 \\ a'_2 \\ a'_1 \\ a'_0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

2. Se calcula el multiplicativo inverso para cada $a'_{ij} \rightarrow a_{ij}$ en el STATE en $\text{GF}(2^8)$ con $m(x) = x^8 + x^4 + x^3 + x + 1$, el caso especial del byte 00 se transforma en byte 00.

AES, Rijndael: InvSubBytes

- Realmente esto es como una caja de sustitución para cada byte inversa.
- De esta forma se obtiene la caja de sustitución IS para el AES:
 - $b_{ij} = IS-box[a'_{ij}]$.
- Ejemplo:
 - B8 → 9A

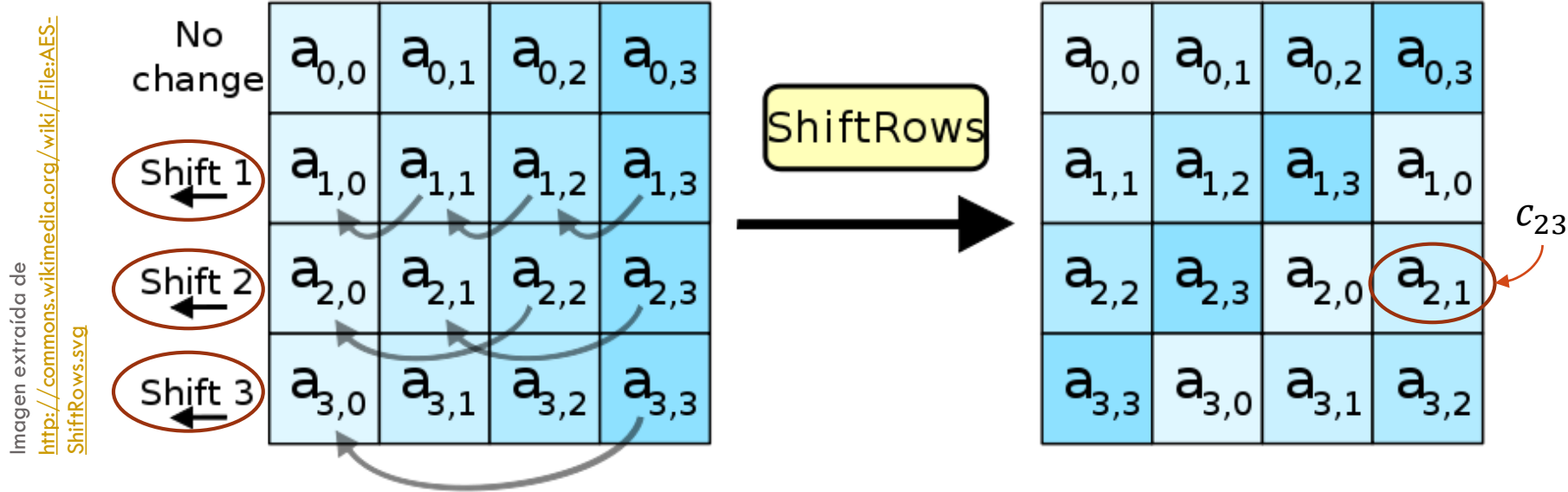
Inverse S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

AES, Rijndael: Base del diseño de SubBytes

- Las cajas S del AES están diseñadas para ser resistentes ante los criptoanálisis conocidos.
- El AES fue diseñado para para obtener una baja correlación entre la entrada y la salida cifrada.
- Es decir no hay una función matemática que describa fácilmente la salida en función de la entrada.
- El vector que se suma en las transformaciones afines es para evitar los puntos fijos de las cajas AES, $S\text{-Box}(a)=a$, ni puntos fijos opuestos, $S\text{-Box}(a)=\bar{a}$.
- S-box es invertible: $IS\text{-Box}[S\text{-Box}(a)]=a$.
- Pero no es auto-invertible: $S\text{-Box}(a) \neq IS\text{-Box}(a)$.

AES, Rijndael: ShiftRows



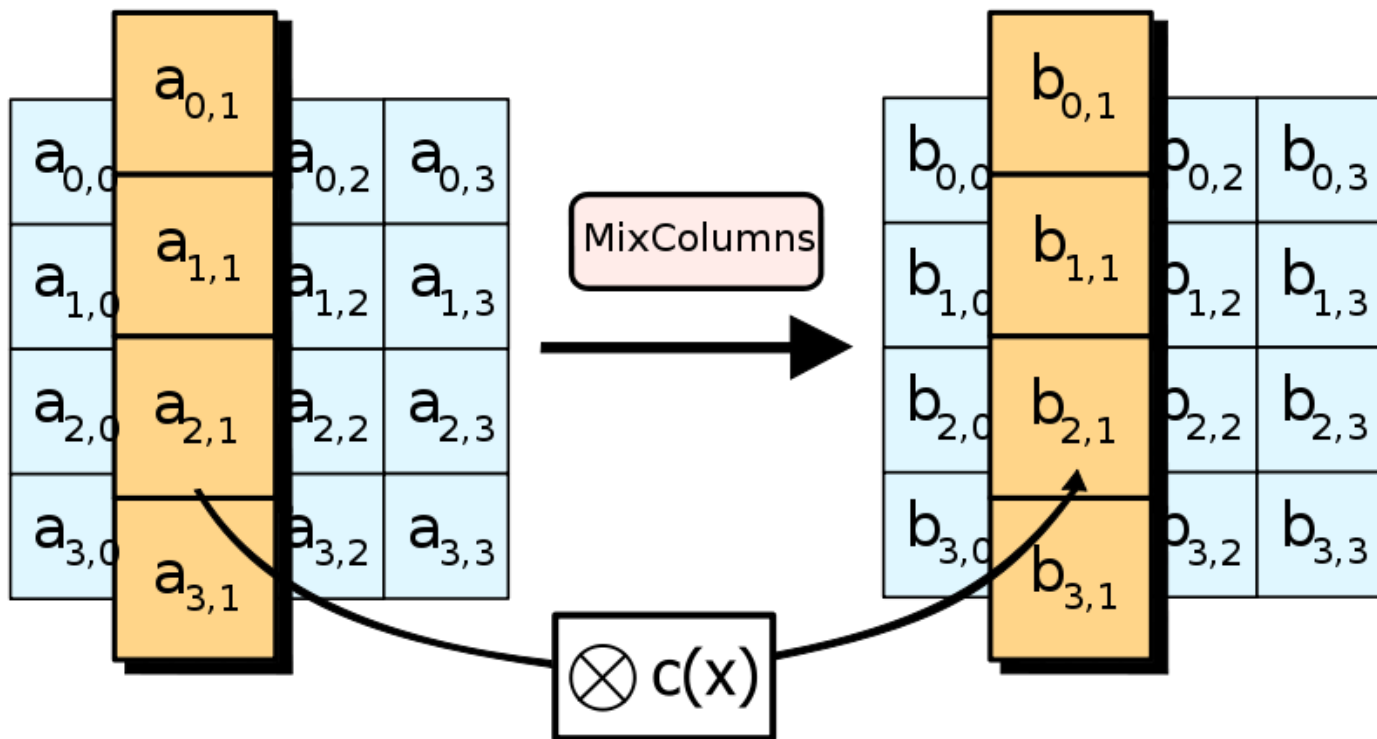
- Los bytes del STATE son rotados por filas, es decir se descolocan los bytes en el state, pero no se cambia su valor.
- Se utiliza la función XTIME que multiplica un polinomio $B(x)$ con coeficientes en $GF(2^8)$ (una fila del STATE) por el polinomio x que vimos anteriormente.

AES, Rijndael: **InvShiftRows** y Base del diseño de ShiftRows

- El resultado de esta transformación ShiftRows da una matriz resultado C , cuyos elementos $c_{ij} = b_{i(j-c_i) \bmod 4}$, con $c_0 = 0, c_1 = 1, c_2 = 2, c_3 = 3$.
- El elemento c_{ij} representa la transformación de $\text{byte}_{c_{ij}}$, es decir a que posición movemos el b_{ij} :
 - Así el byte b_{23} se transforma en $c_{23} = b_{2(3-c_2) \bmod 4} = b_{2(3+(4-c_2)) \bmod 4} = b_{2(3+(4-2)) \bmod 4} = b_{2(5) \bmod 4} = b_{21}$.
 - Es decir en la posición c_{23} está el b_{21} después de aplicar la función “ShiftRows”.
- La “**InvShiftRows**” ejecuta el desplazamiento circular en dirección contraria: $b_{ij} = c_{i(j+c_i) \bmod 4}$, con $c_0 = 0, c_1 = 1, c_2 = 2, c_3 = 3$.
- Esta función hace un buen mezclado de columnas: hay que darse cuenta que en las columnas del STATE es donde se coloca la secuencia de bytes a cifrar.
- Así esta función mezcla bytes de diferentes columnas.
- Es decir los bytes de una columna se esparcen en el resto de las columnas.

AES, Rijndael: MixColumns

Imagen extraída de
<http://commons.wikimedia.org/wiki/File:AES-MixColumns.svg>



- Cada columna se multiplica por valor fijo o polinomio constante (polinomio de Galois), mezclándose así todos los bytes por columnas.

AES, Rijndael: MixColumns

- Las columnas del STATE son consideradas como polinomios de grado menor o igual que 3 con coeficientes en $GF(2^8)$, siendo reducido el resultado de las operaciones por el polinomio: $M(x) = x^4 + 1$.
- Las operaciones esta función se hacen únicamente entre columnas:
 - Cada columna se multiplica por el polinomio:
 - $C(x) = '03'x^3 + '01'x^2 + '01'x^1 + '02'$.
 - Este polinomio es coprimo con $M(x) = x^4 + 1$, por lo que se puede realizar la transformación inversa.
- Así cada columna se trasforma como:

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Se denomina matriz M .

AES, Rijndael: InvMixColumns

- Para la operación inversa tenemos que buscar $C(x)C(x)^{-1} = '01'$.
- Existe ya $C(x)$ que es coprimo con $M(x) = x^4 + 1$.
- $C(x)^{-1} = '0B'x^3 + '0D'x^2 + '09'x^1 + '0E'$
- Así cada columna se transforma de manera inversa como:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0B & 09 & 0E \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} \quad \text{Se denomina } M^{-1}.$$

- Se puede probar que:

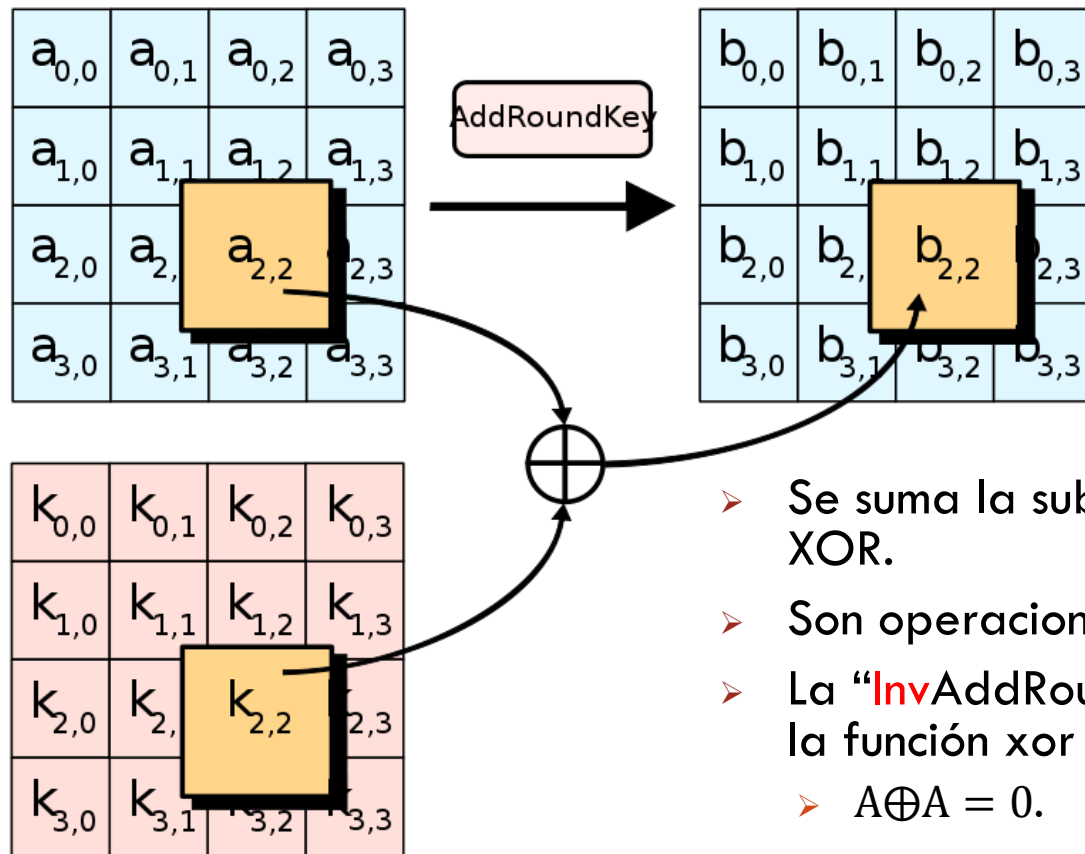
$$\text{➤ } MM^{-1} = \begin{pmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 01 \end{pmatrix}.$$

AES, Rijndael: Base del diseño de MixColumns

- La elección de los coeficientes '01', '02' y '03' asegura un buen mezclado entre los bytes de cada columna.
- Esta transformación junto con "ShiftRows", asegura que después de pocas rondas todos los bits de salida dependan de todos los bits de entrada.
- La elección de '01', '02' y '03' se debe también a que hay una gran optimización en la operaciones, ya que son polinomios muy pequeños (1 , x y $x + 1$):
 - es decir multiplicar por la unidad, o un solo xtime.
- Por otro lado la transformación inversa, "**Inv**MixColumns", es mucho más costosa ya que los polinomios involucran más funciones xtime: '0E', '0B', '0D', y '09'.
- Esto quiere decir que el cifrado es más rápido que el descifrado, pero no hay problema porque para esto existen los modos de operación:
 - CFB, OFB y CTR, en donde el algoritmo AES solo se utiliza en su modo de cifrado y nunca se descifra (no se utiliza la función, "**Inv**MixColumns").
 - Otra opción si se quiere utilizar el AES en modo descifrado y que sea rápido, es tener 4 tablas de las 256 posibilidades de la multiplicación de los cuatro polinomios: '0E', '0B', '0D', y '09'.

AES, Rijndael: AddRoundKey

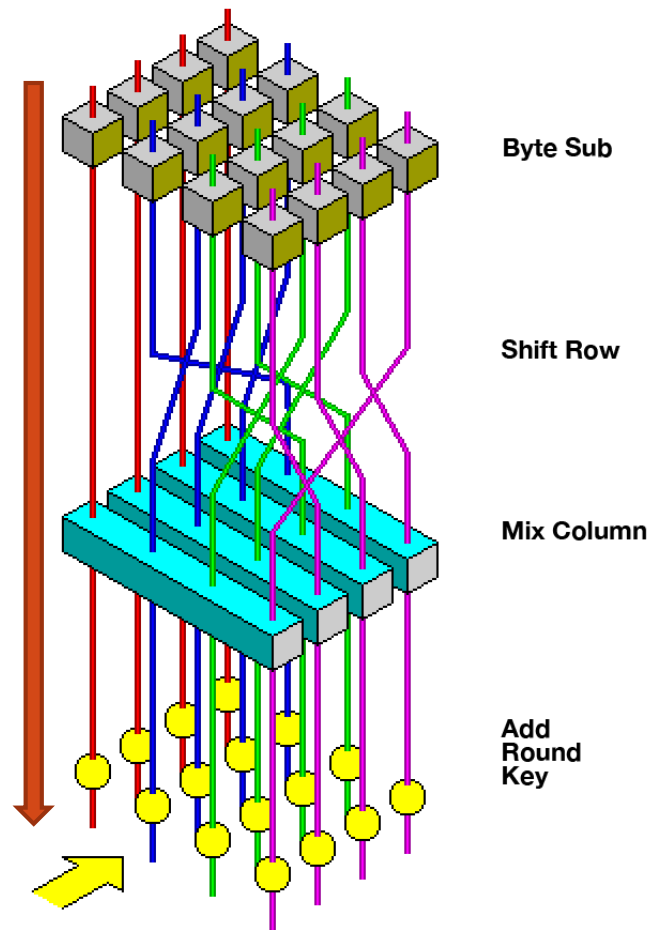
Imagen extraída de
<http://commons.wikimedia.org/wiki/File:AES-AddRoundKey.svg>



- Se suma la subclave mediante la función XOR.
- Son operaciones a nivel de byte.
- La “**Inv**AddRoundKey” es idéntica ya que la función xor es la propia inversa:
 - $A \oplus A = 0$.

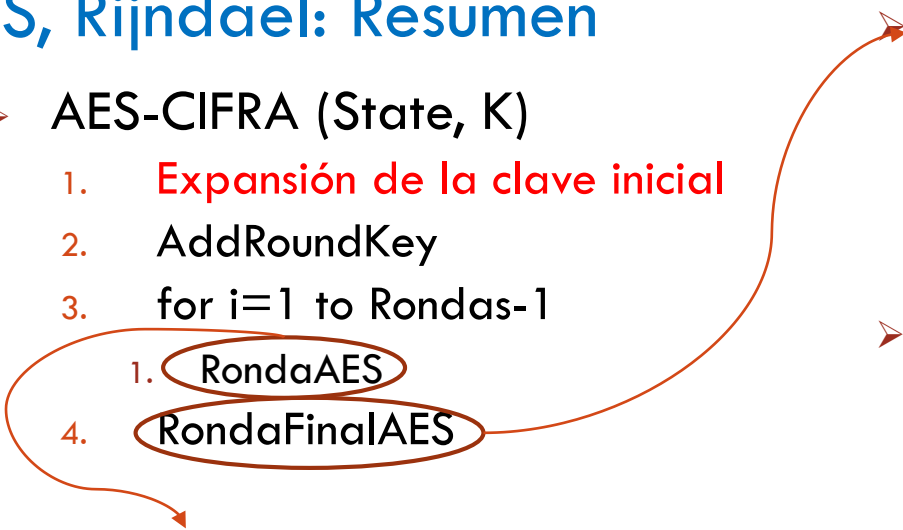
AES, Rijndael: Resumen

- Funciones de una ronda:
 - SubBytes: $b_{ij} = S\text{-}box[a_{ij}]$.
 - ShiftRows: $c_{ij} = b_{i(j-c_i) \bmod 4}$.
 - MixColumns: $d = M\bar{c}$.
 - AddRoundKey: $e_{ij} = d_{ij} \oplus k_{ij}$.
- Para cada función del AES existe su inversa bien definida:
 - **Inv**SubBytes
 - **Inv**ShiftRows
 - **Inv**MixColumns
 - AddRoundKey
- Por lo tanto en el descifrado se realiza el proceso inverso con las funciones inversas, que ya hemos visto.



AES, Rijndael: Resumen

➤ AES-CIFRA (State, K)

1. Expansión de la clave inicial
 2. AddRoundKey
 3. for i=1 to Rondas-1
 1. RondaAES
 4. RondaFinalAES
- 
- A red curved arrow originates from the 'RondaFinalAES' step (item 4) and points to the 'RondaAES' step (item 1.1). Another red curved arrow originates from the 'RondaFinalAES' step and points to the 'RondaFinalAES (State, K)' section header.

➤ RondaAES (State, K)

1. SubBytes
2. ShiftRows
3. MixColumns
4. AddRoundKey

RondaFinalAES (State, K)

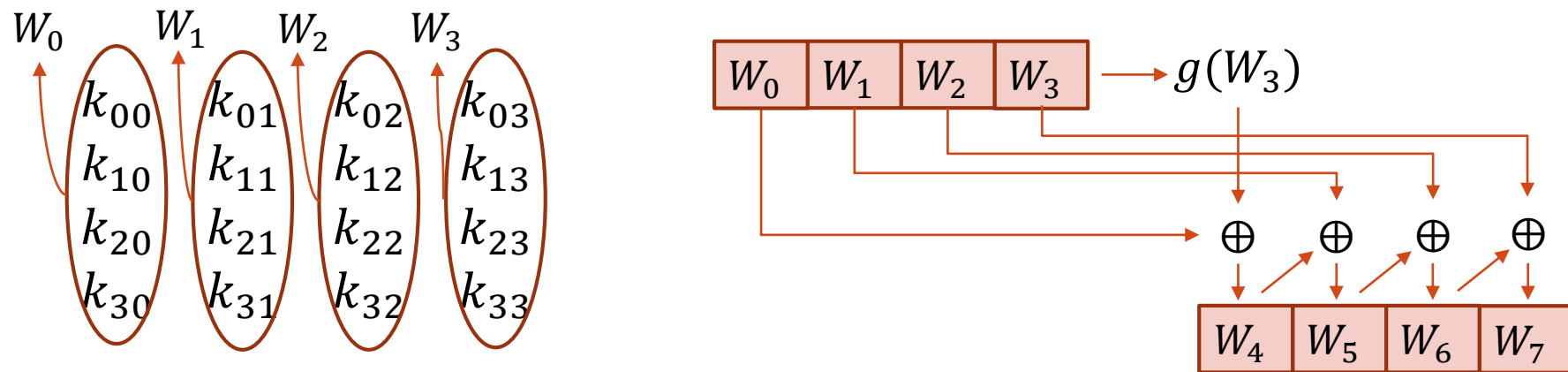
1. SubBytes
2. ShiftRows
3. AddRoundKey

➤ Referencias interesantes:

- FIPS 197
<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- Gráfico animado AES
<http://www.formaestudio.com/rijndaelinspector/>
- Códigos de diferentes cifrados:
<http://embeddedsw.net/CipherReferenceHome.html>

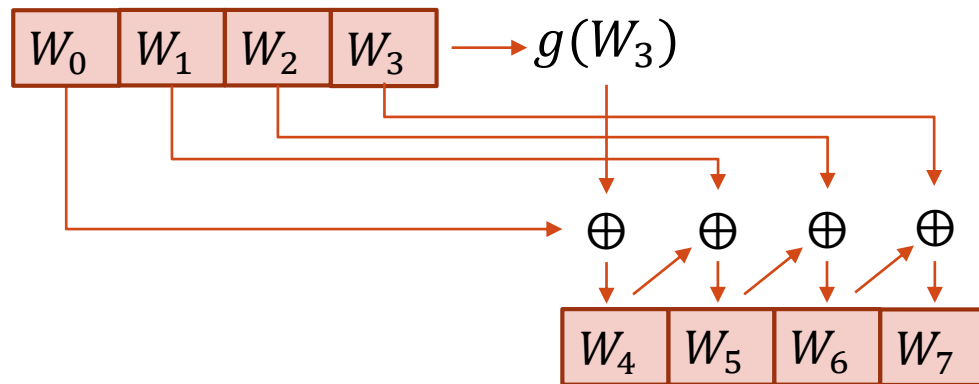
AES, Rijndael: Expansión de la clave

- En general el número de bits de la clave expandida tiene:
 - Número de bits del bloque $\times (N_R + 1)$.
- Vamos a ver el algoritmo para 128 bits de llave que se utilizan 10 rondas, como se vio en la transparencias del número rondas.
- Para otros tamaños de clave la expansión es diferente.
- Supongamos que tenemos la clave de 128 bits (se separa en palabras):



AES, Rijndael: Expansión de la clave

- Así la subclave para el primer “AddRoundKey” viene determinada por la clave $[W_0, W_1, W_2, W_3]$, y partir de esta se genera la siguiente subclave mediante este procedimiento.



- Observar que:
 - W_4 depende de la función $g(W_3)$ y la palabra W_0 .
 - W_5 depende de las palabras W_1 y W_4 .
 - W_6 depende de las palabras W_2 y W_5 .
 - W_7 depende de las palabras W_3 y W_6 .

AES, Rijndael: Expansión de la clave

- Así se genera la siguiente subclave para el segundo “AddRoundKey”:
 - $[W_4, W_5, W_6, W_7]$.
- Este proceso se repite para obtener el número necesario de subclaves para todas la funciones “AddRoundKey” del algoritmo.
- La **función g** consiste en varios pasos:
 1. RotWord.
 2. SubWord.
 3. $\oplus Rcon$ (suma de una constante al resultado).
- Supongamos una palabra cualquiera que está compuesta por 4 bytes:
 - $W_i = [b_0, b_1, b_2, b_3]$.

AES, Rijndael: Expansión de la clave

POSIBLE Tarea de Evaluación
Continúa: Implementación del AES en micros de 32 y 8 bits ("smart cards").

- Así $g(W_i)$ se calcula de la siguiente forma:
1. $\text{RotWord}(W_i) = [b_1, b_2, b_3, b_0]$.
 2. $\text{SubWord}[b_1, b_2, b_3, b_0] = [S\text{-}box[b_1], S\text{-}box[b_2], S\text{-}box[b_3],$

Values of rc_i in hexadecimal

i	1	2	3	4	5	6	7	8	9	10
rc_i	01	02	04	08	10	20	40	80	1B	36

Imagen extraída de
https://en.wikipedia.org/wiki/AES_key_schedule

AES, Rijndael: Modos de operación

- Se utilizan los mismo modos de operación ya estudiados en el DES.
- Además de esos modos se utiliza un modo de operación que está basado en Campos de Galois y el modo contador:
 - Galois/Counter Mode (GCM):
 - NIST Special Publication SP800-38D defining GCM and GMAC.
- Este modo de operación es utilizado mucho hoy en día en los navegadores, ya con este modo el AES se convierte en un cifrador de flujo:
 - Por ejemplo veréis en la seguridad de conexión en el navegador: AES-256-GCM.

POSIBLE Tarea de Evaluación Continúa: Estudio del modo de operación “Galois/Counter Mode (GCM)”.