**IDEA**

**Introducción**

IDEA es un cifrador de clave simétrica por bloques de dominio público. El mismo fue diseñado por Xuejia Lai yJames L. Massey para Escuela Politécnica Federal de Zúrich y su primera publicación data del año 1991. El algoritmo es una versión mejorada del crifrado PES (Proposed encryption standard).

El cifrador opera sobre bloques de 64 bits (estos bloques son la información que se va a cifrar), cifrando el contenido usando una clave de 128 bits. La principal característica de este algoritmo es que hace uso de operaciones de tres grupos algebraicos distintos. Otra característica especial es que las operaciones tanto para cifrado como para descifrado son las mismas, solo cambiando la forma de uso de la clave (más adelante se revisará este punto).

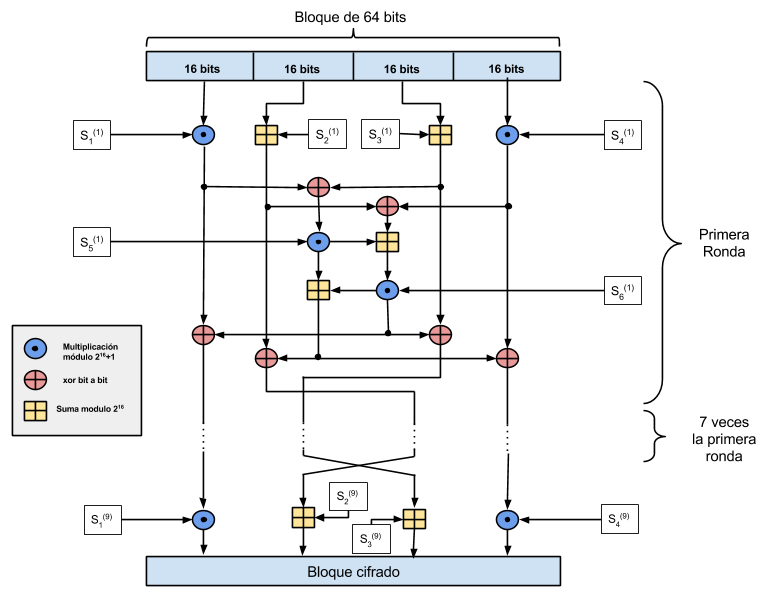
**Pasos del algoritmo**

**Algoritmo de cifrado**

Como se mencionó, el algoritmo hace uso de tres operaciones de grupos algebraicos distintos. Las mismas son las siguientes:

* XOR bit a bit de bloques de 16-bits
* Suma en módulo 216 de dos enteros de 16-bits
* Multiplicación en módulo 216+1 de dos enteros de 16-bits, ajustandose a al valor 216 a los valores de entrada igual a cero y llevando a cero los resultados del producto que den 216

Mediante estas tres operaciones, el algoritmo se ejecuta en 8 rondas iguales más una ronda final especial (también denominada “media ronda”) como se muestra en la siguiente figura:



Las entradas con notación SN(M) hacen referencia a la sub-clave N de la ronda M, cuya longitud es de 16 bits.

**Generación de sub-claves**

En el gráfico anterior se observa que el algoritmo utiliza 6 sub-claves por ronda más 4 claves en la ronda final. Operando sobre la cantidad de claves podes mos ver que la cantidad de claves que va a hacer uso el algoritmo es de 52 (8 x 6 + 4 x 1).

El proceso para obtener las 52 claves se puede resumir en los siguientes pasos:

* Se subdivide la clave original (de 128 bits) en grupos de 16 bits. De la primera subdivisión se obtienen las 6 sub-claves utilizadas en la primera ronda más las claves S1(2) y S2(2).
* Para obtener las claves faltantes para la segunda ronda, las mismas se obtienen al realizar un shift a izquierda de 25 bits de la clave original. De la nueva clave de 128 bits que queda formada, se procede a extraer las cuatro claves faltantes para la segunda ronda (las otras 4 claves sobrantes se utilizan de igual forma que se utilizaron las sobrantes del primer paso).
* El punto anterior se repite hasta bien haber obtenido las 52 claves que requiere el algoritmo (en el último shift van a sobrar 4 claves, estás no se utilizan y son descartadas)

**Descifrado**

El algoritmo necesario para realizar el descifrado es exactamente el mismo que se requirió para el cifrado. Los cambios que se producen solo están reflejados en las claves utilizadas.

Si las claves requeridas para cifrar hubiesen sido las siguientes:

| **N° Ronda** | **Sub claves** |
| --- | --- |
| Ronda 1 | S1(1) S2(1) S3(1) S4(1) S5(1) S6(1) |
| Ronda 2 | S1(2) S2(2) S3(2) S4(2) S5(2) S6(2) |
| Ronda 3 | S1(3) S2(3) S3(3) S4(3) S5(3) S6(3) |
| Ronda 4 | S1(4) S2(4) S3(4) S4(4) S5(4) S6(4) |
| Ronda 5 | S1(5) S2(5) S3(5) S4(5) S5(5) S6(5) |
| Ronda 6 | S1(6) S2(6) S3(6) S4(6) S5(6) S6(6) |
| Ronda 7 | S1(7) S2(7) S3(7) S4(7) S5(7) S6(7) |
| Ronda 8 | S1(8) S2(8) S3(8) S4(8) S5(8) S6(8) |
| Ronda 9 | S1(9) S2(9) S3(9) S4(9) |

Entonces, las requeridas para el descifrado serían las siguientes:

| **N° Ronda** | **Sub claves** |
| --- | --- |
| Ronda 1 | (S1(9))-1 -S2(9) -S3(9) (S4(9))-1 S5(8) S6(8) |
| Ronda 2 | (S1(8))-1 -S2(8) -S3(8) (S4(8))-1 S5(7) S6(7) |
| Ronda 3 | (S1(7))-1 -S2(7) -S3(7) (S4(7))-1 S5(6) S6(6) |
| Ronda 4 | (S1(6))-1 -S2(6) -S3(6) (S4(6))-1 S5(5) S6(5) |
| Ronda 5 | (S1(5))-1 -S2(5) -S3(5) (S4(5))-1 S5(4) S6(4) |
| Ronda 6 | (S1(4))-1 -S2(4) -S3(4) (S4(4))-1 S5(2) S6(3) |
| Ronda 7 | (S1(3))-1 -S2(3) -S3(3) (S4(3))-1 S5(2) S6(2) |
| Ronda 8 | (S1(2))-1 -S2(2) -S3(2) (S4(2))-1 S5(1) S6(1) |
| Ronda 9 | (S1(1))-1 S2(1) S3(1) (S4(9))-1 |

En donde:

* “-SN(M)” es el complemente a 2 del valor de la sub-clave al momento de cifrar
* “(SN(M))-1“ es la inversa en complementa a 216+1 de dicho número. Para obtener este valor se puede hacer uso del algoritmo euclidiano extendido.

**Cálculo del inverso**

Como se mencionó en el punto anterior, para lograr revertir las operaciones de multiplicación se requiere del uso del inverso multiplicativo del número utilizado inicialmente. Para tal fin, se hace uso del algoritmo euclidiano extendido. Sin embargo, antes de dar inicio a la explicación de dicho algoritmo, vamos a aclarar algunos puntos a tener en cuenta en el cálculo de inversos multiplicativos de números en módulo N.

Cuando definimos el inverso multiplicativo de un número "a" en módulo "p", en realidad estamos buscando un número "x" que cumpla:

a.x = 1 mod p

O sea, el inverso genera un número que es múltiplo de "p" sumado una unidad. Por ejemplo:

tomamos p = 17 y a = 2

a.x = 1 mod 17 = (17.n) + 1 // "n" es un número natural

Para este caso, podemos ver que x = 9 es solución. A su vez, tomando este valor de x e incrementando o decrementando el valor del módulo podemos obtener nuevos valores que funcionan como el inverso de "a". Sin embargo, x = 9 es el único inverso multiplicativo dentro del módulo p.

Según la teoría de los campos de galois (teoría que excede al presente resumen), en un conjunto de números enteros en los que esté definida las operaciones de suma y multiplicación modular, el mismo podrá lograr que exista la inversa multiplicativa de sus elementos (excepto el 0, claro está) si el módulo es en base a una potencia de un número primo.

Volviendo a las operaciones modulares de IDEA, podemos ver que el algoritmo realiza multiplicacione en módulo 216+1, el cual es un número primo (nota de color: también es un número de Fermat) y, dado lo mencionado para los campos de galois, sabemos que todos los números menores a 216+1 poseen un número que es su inverso multiplicativo dentro del módulo (lo cual es muy importante!! ya que nos garantiza que exista la inversa de todos los números que representamos con 16 bits, dándo una operación reversible ideal para un algoritmo de cifrado).

Entonces, pudimos ver que todos los números en módulo 216+1 poseen un inverso multiplicativo, a excepción del número cero, el cual no puede tenerlo. Si el algoritmo no hiciera ninguna aclaración con el número cero, entonces cualquier multiplicación que involucrara a este valor desembocaría en la imposibilidad de revertir la operación. Como era de esperar, IDEA tomó en cuenta esta limitación y le asignó un inverso multiplicativo "virtual" al cero. Este inverso multiplicativo es el número 216, lo cual no genera problema alguno con la representación numerica utilizada, ya que los datos de entrada sus valores de 16 bits (i.e. se puede representar hasta el valor 216-1).

**Algoritmo de Euclides extendido**

Pseudocódigo:

**EXTENDED EUCLIDES(m, b)**

# (A1, A2, A3)=(1, 0, m);

# (B1, B2, B3)=(0, 1, b);

# if (B3 == 0)

# return A3 = gcd(m, b); *//no inverse*

# if (B3 == 1)

# return B3 = gcd(m, b); B2 = b^–1 mod m

# Q = A3 div B3

# (T1, T2, T3)=(A1 – Q B1, A2 – Q B2, A3 – Q B3)

# (A1, A2, A3)=(B1, B2, B3)

# (B1, B2, B3)=(T1, T2, T3)

# goto 2

