

## Alleged RC-4

En una parte de la criptografía simétrica existen los “stream ciphers”, es decir algoritmos de cifrado que cifran de byte en byte, esto para reducir al máximo el ancho de banda, es decir, el número de bits que pueden ser transmitidos a la vez. Lo que permite tener un método de cifrado de gran rapidez que es muy necesario en aplicaciones donde el ancho de banda es reducido.

Uno de estos sistemas de cifrado, es el conocido como rc\_4 inventado por R. Rivest criptógrafo reconocido mundialmente y fundador de la compañía RSA security. Se había mantenido en secreto el código, sin embargo en 1994 un anónimo da a conocer un código de alleged\_rc4 afirmando ser el original, posteriormente quienes tenían licencia del código afirmaron la compatibilidad.

Dejando a un lado el alegato sobre la autenticidad del código, nos proponemos a describirlo con la ayuda del programa anexo (alleged\_rc4.exe), el funcionamiento de este sistema cubre las características más precisas de un buen algoritmo criptográfico, sencillo, seguro y rápido.

La sencillez se refleja en la siguiente descripción en dos renglones del sistema rc\_4

- 1) A partir de la clave secreta se elige una permutación del grupo simétrico  $S_{256}$ , es decir, una forma particular de ordenar los números del 0 al 255.
- 2) Se toma el primer byte del mensaje y se realiza el cifrado efectuando un xor con una parte de la permutación, este proceso se sigue para cada byte del mensaje.

Tomaremos como punto de partida la anterior descripción para seguir detallando al sistema rc\_4, el primer renglón se le conoce como la extensión de la clave, el segundo renglón describe el proceso de cifrado. Cabe hacer notar que el proceso de descifrado es igual al cifrado por lo tanto solo nos dedicaremos al primero.

### 1) Extensión de la clave

#### 1.1) Se inicializan las siguientes variables

```
for(i = 0; i < 256; i++) state[i] = i;  
x = 0;  
y = 0;  
index1 = 0;
```

index2 = 0;

Es decir el arreglo state de 256 entradas se inicializa con la formula  $state[i] = i$ .

1.2) A partir de la clave privada, se reordena el arreglo state[], de la siguiente manera

```
for(i= 0; i < 256; i++)
{
    index2 = (key[index1] + state[i] + index2) mod 256;
    swap_byte(state[i], state[index2]);
    index1 = (index1 + 1) mod key_len;
}
```

Recorriendo cada elemento del arreglo state[], primero se recalcula la variable index2.

Posteriormente se realiza una transposición (swap), es decir, se intercambia el elemento en state[i] por state[index2].

Finalmente se recalcula la variable index1.

Lo anterior da un nuevo orden a los elementos del arreglo state[], que no es nada más que una permutación calculada a partir de la clave privada, es decir se elige un elemento del grupo simétrico  $S_{256}$ .

## 2) Proceso de cifrado

2.1) El proceso de cifrado es ahora muy simple

```
for(i = 0; i < men_len; i++)
{
    x = (x + 1) mod 256;
    y = (state[x] + y) mod 256;
    swap_byte(state[x], state[y]);
    xorIndex = (state[x] + state[y]) mod 256;
    men[i] ^= state[xorIndex];
};
```

Para cada byte del mensaje, es decir desde  $i=0$  hasta  $i=\text{longitud del mensaje (men\_len)}$ , se calculan las variables

x, y con las formulas expuestas.

Se realiza una transposición (swap) del elemento state[x] y state[y].

Se calcula la variable xorIndex.

Y finalmente se cifra el byte `men[i]` con la formula `men[i] xor state[xorIndex]`.

Se recomienda ver el proceso del cifrado con el programa, donde se ven todos los cálculos que se realizan para cifrar un mensaje.