REPORTE 3. Implementación de un programa de Cifrado Parcial

Contenido

Primera versión de Inversión de Bits (Criterio del último bit invertido).....3
Permutación de segmentos con inversión de bits por medio del criterio del último bit invertido. 8
Segunda versión de Inversión de Bits (Se toma el bit de referencia).....10
Observaciones.....17

Primera versión de Inversión de Bits (Criterio del último bit invertido)

Ahora, explicamos un proceso de inversión basándonos en el criterio 2 que se menciona en las observaciones. Se muestra un ejemplo simple de dos segmentos, cada segmento consta de 4 bytes (32 bits).

El arreglo original es el siguiente (La primer fila representa los valores almacenados, la segunda fila representa el índice del arreglo, en este caso, las dos filas coinciden):

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

Esto significa que tenemos dos segmentos. Si lo anterior lo pasamos a bits, obtenemos lo siguiente:

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

A la hora de ejecutar un ejemplo, obtenemos los siguientes resultados:

Primer Segmento

A continuación, mostramos las iteraciones obtenidas para el primer segmento:

Iteracion	Cosas Obtenidas
1	bitRef: 0 p: 2 bitRefConP: 2 posInversion: 1 bitRefConINversion: 3 posicionArreglo: 0 valor par invertir: 16

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
2	bitRef: 3 p: 1 bitRefConP: 4 posInversion: 0 bitRefConINversion: 4 posicionArreglo: 0 valor par invertir: 8

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
3	bitRef: 4 p: 2 bitRefConP: 6 posInversion: 1 bitRefConINversion: 7 posicionArreglo: 0 valor par invertir: 1

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
4	bitRef: 7 p: 5 bitRefConP: 12 posInversion: 0 bitRefConINversion: 12 posicionArreglo: 1 valor par invertir: 8

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
5	bitRef: 12 p: 6 bitRefConP: 18 posInversion: 1 bitRefConINversion: 19 posicionArreglo: 2 valor par invertir: 16

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas

6	bitRef: 19 p: 3 bitRefConP: 22 posInversion: 2 bitRefConINversion: 20 posicionArreglo: 2 valor par invertir: 8

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
7	bitRef: 20 p: 2 bitRefConP: 22 posInversion: 1 bitRefConINversion: 23 posicionArreglo: 2 valor par invertir: 1

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
8	bitRef: 23 p: 5 bitRefConP: 28 posInversion: 4 bitRefConINversion: 24 posicionArreglo: 3 valor par invertir: 128

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Segundo Segmento

Iteracion	Cosas Obtenidas
9	bitRef: 0

	p: 6 bitRefConP: 6 posInversion: 1 bitRefConINversion: 7 posicionArreglo: 4 valor par invertir: 1

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
10	bitRef: 7 p: 3 bitRefConP: 10 posInversion: 2 bitRefConINversion: 8 posicionArreglo: 5 valor par invertir: 128

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
11	bitRef: 8 p: 5 bitRefConP: 13 posInversion: 1 bitRefConINversion: 12 posicionArreglo: 5 valor par invertir: 8

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
12	bitRef: 12

	p: 4 bitRefConP: 16 posInversion: 1 bitRefConINversion: 17 posicionArreglo: 6 valor par invertir: 64

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
13	bitRef: 17 p: 6 bitRefConP: 23 posInversion: 1 bitRefConINversion: 24 posicionArreglo: 7 valor par invertir: 128

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Ya se han recorrido todos los segmentos existentes, ahora, lo único que falta es intercambiar los bits que fueron elegidos y que están resaltados por un fondo negro, los falores obtenidos son los siguientes:

25	9	27	131	5	141	70	135
00011001	00001001	00011011	10000011	00000101	10001101	01000110	10000111
0	1	2	3	4	5	6	7

Permutación de segmentos con inversión de bits por medio del criterio del último bit invertido.

El proceso de permutación de segmentos utiliza un arreglo auxiliar para evitar el movimiento de datos durante el proceso de cifrado. A continuación, se observa la un ejemplo de permutación de un conjunto de datos formado por dos paquetes RTP.

Permutación del primer paquete RTP.
Una vez que se ha ejecutado el proceso de inversión y permutación de bits, obtenemos el siguiente arreglo auxiliar que nos indica cómo se permutarán los segmentos que forman el paquete.

Los datos que se obtuvieron **después el proceso de inversión de bits**, sin mover la información del primer paquete RTP, fueron los siguientes:

179 40 216 226 64	**80 214 34 185 102**	**8 19 97 80 129**	**89 5 170 48 128**	**255 219 0 67 0**	**70 63 68 224 12**	**2 13 157 54 68**	**23 150 6 230 71**	**76 137 199 199 135**	**3 139 43 71 41**

Ahora, una vez que obtuvimos el arreglo final de índices, se deben de permutar los sigmentos. El resultado obtenido es el siguiente:

4	0	7	2	9	3	8	1	5	6

255 219 0 67 0	**179 40 216 226 64**	**23 150 6 230 71**	**8 19 97 80 129**	**3 139 43 71 41**	**89 5 170 48 128**	**76 137 199 199 135**	**80 214 34 185 102**	**70 63 68 224 12**	**2 13 157 54 68**

Permutación del segundo paquete RTP.

Para el segundo segmento, los datos que se obtuvieron **después del proceso de inversión de bits** fueron los siguientes:

28 149 3 208 82	83 71 13 30 17	86 157 54 16 4	2 12 245 130 16	33 24 26 29 29	30 28 19 27 7	35 162 38 18 4	108 63 60 80 191	166 124 43 64 16	44 38 164 78 129

Para el proceso de inversión y permutación de bits del sgundo paquete, se obtuvo el arreglo auxiliar de índices mostrado en la parte de abajo, éste nos indica cómo se permutarán los segmentos del paquete. También se muestra el paquete RTP obtenido a la hora de aplicar el intercambio de segmentos indicado.

4	2	5	6	8	9	7	1	3	0

33 24 26 29 29	86 157 54 16 4	30 28 19 27 7	35 162 38 18 4	166 124 43 64 16	44 38 164 78 129	108 63 60 80 191	83 71 13 30 17	2 12 245 130 16	28 149 3 208 82

Segunda versión de Inversión de Bits (Se toma el bit de referencia)

Ahora explica de manera detallada el proceso de inversión cuando se toma como nueva referencia, el bit de referencia anterior. En este caso, se toma como criterio de inversión de bits el bit de referencia para el cálculo de bits anteriores db y bits siguientes de. Se muestra un ejemplo simple de dos segmentos, cada segmento consta de 4 bytes (32 bits).

El arreglo original es el siguiente (La primer fila representa los valores almacenados, la segunda fila

representa el índice del arreglo, en este caso, las dos filas coinciden):

0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7

Esto significa que tenemos dos segmentos. Si lo anterior lo pasamos a bits, obtenemos lo siguiente:

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

A continuación se muestran los resultados obtenidos, los bits con fondo negro son los que son invertidos.

Primer Segmento

Iteracion	Cosas Obtenidas
1	bitRef: 0 p: 4 bitRefConP: 4 posInversion: 1 bitRefConINversion: 5 posicionArreglo: 0 valor par invertir: 4

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
2	bitRef: 4 p: 1 bitRefConP: 5 posInversion: 0 bitRefConINversion: 5 posicionArreglo: 0 valor par invertir: 4

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
3	bitRef: 5 p: 3

	bitRefConP: 8 posInversion: 2 bitRefConINversion: 6 posicionArreglo: 0 valor par invertir: 2

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
4	bitRef: 8 p: 3 bitRefConP: 11 posInversion: 2 bitRefConINversion: 9 posicionArreglo: 1 valor par invertir: 64

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
5	bitRef: 11 p: 5 bitRefConP: 16 posInversion: 4 bitRefConINversion: 12 posicionArreglo: 1 valor par invertir: 8

00000000	00000001	00000010	00000011	00000100	00000101	00000110	00000111

0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
6	bitRef: 16 p: 2 bitRefConP: 18 posInversion: 1 bitRefConINversion: 19 posicionArreglo: 2 valor par invertir: 16

0000000	0000001	0000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
7	bitRef: 18 p: 3 bitRefConP: 21 posInversion: 2 bitRefConINversion: 19 posicionArreglo: 2 valor par invertir: 16

0000000	0000001	00000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
8	bitRef: 21 p: 3 bitRefConP: 24 posInversion: 2 bitRefConINversion: 22 posicionArreglo: 2 valor par invertir: 2

0000000	0000001	0000010	00000011	00000100	00000101	00000110	00000111

0	1	2	3	4	5	6	7

Segundo Segmento

Iteracion	Cosas Obtenidas
9	bitRef: 0 p: 1 bitRefConP: 1 posInversion: 0 bitRefConINversion: 1 posicionArreglo: 4 valor par invertir: 64

0000000	0000001	0000010	00000011	0000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
10	BitRef: 1 p: 1 bitRefConP: 2 posInversion: 0 bitRefConINversion: 2 posicionArreglo: 4 valor par invertir: 32

0000000	0000001	0000010	00000011	0000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
11	bitRef: 2 p: 1 bitRefConP: 3 posInversion: 0 bitRefConINversion: 3 posicionArreglo: 4 valor par invertir: 16

0000000	0000001	0000010	00000011	0000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
12	bitRef: 3 p: 5 bitRefConP: 8 posInversion: 4 bitRefConINversion: 4 posicionArreglo: 4 valor par invertir: 8

0000000	0000001	0000010	00000011	0000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
13	bitRef: 8 p: 1 bitRefConP: 9 posInversion: 0 bitRefConINversion: 9 posicionArreglo: 5 valor par invertir: 64

0000000	0000001	0000010	00000011	0000100	0000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
14	bitRef: 9 p: 4 bitRefConP: 13 posInversion: 1 bitRefConINversion: 14 posicionArreglo: 5 valor par invertir: 2

0000000	0000001	0000010	00000011	0000100	0000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
15	bitRef: 13 p: 3 bitRefConP: 16 posInversion: 2 bitRefConINversion: 14 posicionArreglo: 5 valor par invertir: 2

0000000	0000001	0000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
16	bitRef: 16 p: 3 bitRefConP: 19 posInversion: 2 bitRefConINversion: 17 posicionArreglo: 6 valor par invertir: 64

0000000	0000001	0000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
17	bitRef: 19 p: 3 bitRefConP: 22 posInversion: 2 bitRefConINversion: 20 posicionArreglo: 6 valor par invertir: 8

0000000	0000001	0000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Iteracion	Cosas Obtenidas
18	bitRef: 22 p: 4 bitRefConP: 26 posInversion: 1 bitRefConINversion: 27 posicionArreglo: 7 valor par invertir: 16

0000000	0000001	0000010	00000011	00000100	00000101	00000110	00000111
0	1	2	3	4	5	6	7

Ya se han recorrido todos los segmentos existentes, ahora, lo único que falta es intercambiar los bits

que fueron elegidos y que están resaltados por un fondo negro, los falores obtenidos son los siguientes:

2	73	0	3	124	69	78	23
0000000**0**	0**0**00**0**001	000000**1**0	00000011	0**0000**100	0**0**000101	0**0**00**0**110	0000**0**111
0	1	2	3	4	5	6	7

Observaciones

Hay un mayor número de iteraciones cuando se toma el bit de referencia anterior para el criterio de tomar como referencia el último bit invertido, se utilizaron 13 iteraciones para cubrir los dos segmentos. Para el criterio de tomar como referencia el último bit de referencia, se necesitaron 18 iteraciones.

Con base en la prueba realizada en este documento, también puede ocurrir que no haya un cambio de bits en el último segmento del paquete como se puede ver aquí: #resultado final Lo anterior quizá se deba a que a condición de paro todavía no está bien diseñada.

Todavía falta ver la manera de optimizarel código, eso está en proceso.