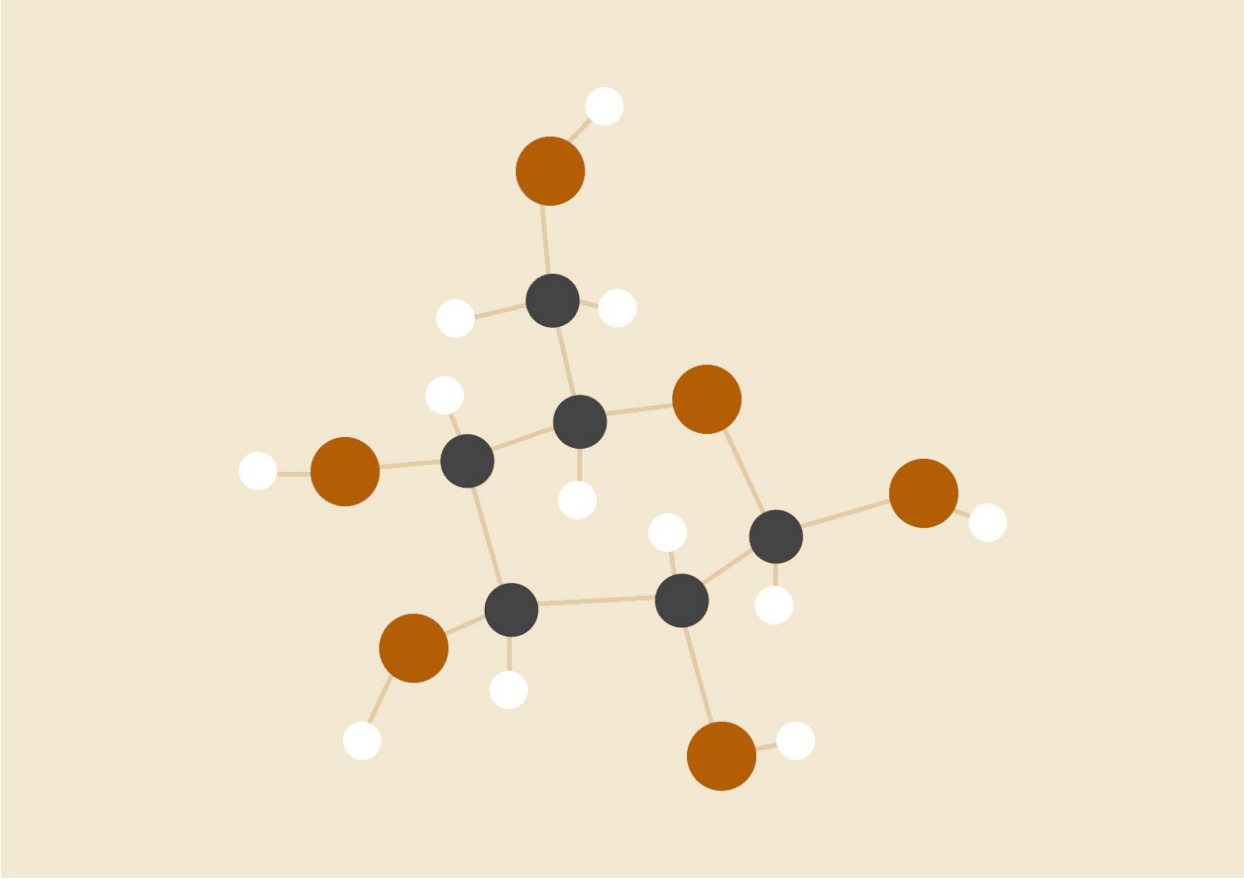
Práctica 2: Seguridad Perfecta y Criptografía Simétrica

Fundamentos de Criptografía y Seguridad Informática



Alfonso Camblor y Alberto Espinosa

G08

2020-2021

Universidad Autónoma de Madrid - Escuela Politécnica Superior - Ingeniería Informática

Índice

1. Seguridad Perfecta

Para probar este primer apartado, hemos creado dos funciones diferentes de cifrado, en las cuales a y b son generadas aleatoriamente, dependiendo del método de ejecución.

Para el método equiprobable, hemos utilizado la propia función de *rand()* que nos proporciona el lenguaje de C, ya que esta genera números aleatorios de una forma uniforme.

Para el método no equiprobable, hemos implementado la función *metodo_no_equiprobable(),* que siguiendo la gráfica inversa a una función exponencial, se genera un valor aleatorio.

Ejecutando el programa con los diferentes métodos, hemos generado dos ficheros *equiprobable.txt* y *no_equiprobable.txt* donde se muestran los resultados obtenidos mediante los métodos de equiprobable y no equiprobable respectivamente.

2. Implementación del DES

- ## Programación del DES

Para implementar el algoritmo DES se ha seguido el esquema que describe las diferentes partes, de forma que se implementa en contenedores cada una de los métodos necesarios para en conjunto realizar la correcta encriptación de los datos, siguiendo por ello un enfoque ascendente.

Inicialmente se encuentra la función de permutación, que permite de forma genérica aplicar todas las permutaciones necesarias durante la ejecución del algoritmo referentes a cada una de las tablas implicadas, indicando en el código cada una de las operaciones en forma de comentarios para facilitar el entendimiento.

La siguiente función a implementar es la encargada del método S-Box en el esquema, que consiste en la división de la clave y posterior sustitución por medio de las S-Boxes especificadas en el algoritmo DES.

A continuación se realiza una función auxiliar capacitada para rotar los bits de

forma genérica, dado que hay multitud de rotaciones durante la ejecución de DES, por lo que al hacerla genérica logramos una modularización del código.

La penúltima función es la de generación de subclaves, de forma que previamente a la iteración de las rondas del DES estamos generando a partir de la clave todas las subclaves necesarias para el correcto funcionamiento, de forma que el cálculo ocurra previamente y no interfiera con la ejecución directa, obteniendo así los bits previamente y aplicándolos a los métodos correspondientes a la ronda actual.

La última función es la del algoritmo DES general, uniendo las iteraciones entre rondas de forma que seamos capaces de obtener el texto cifrado a partir de una única llamada a esta función, que se encarga, siguiendo el esquema, de las sucesivas llamadas a funciones que son capaces de cifrar la información introducida.

• Programación del Triple-DES

Para la implementación del triple DES se realizan tres llamadas sucesivas al método DES implementado anteriormente, aplicando en cada una de las iteraciones Clave1, Clave2 y Clave3 , correspondientemente.

Para obtener las diferentes claves, empezamos leyendo como texto la clave completa para así obtener las diferentes subclaves de partir en 3 trozos dicha clave completa. Una vez ocurre esto, simplemente se llama al algoritmo DES 3 veces sucesivas con cada clave, necesitando la clave invertida para descom

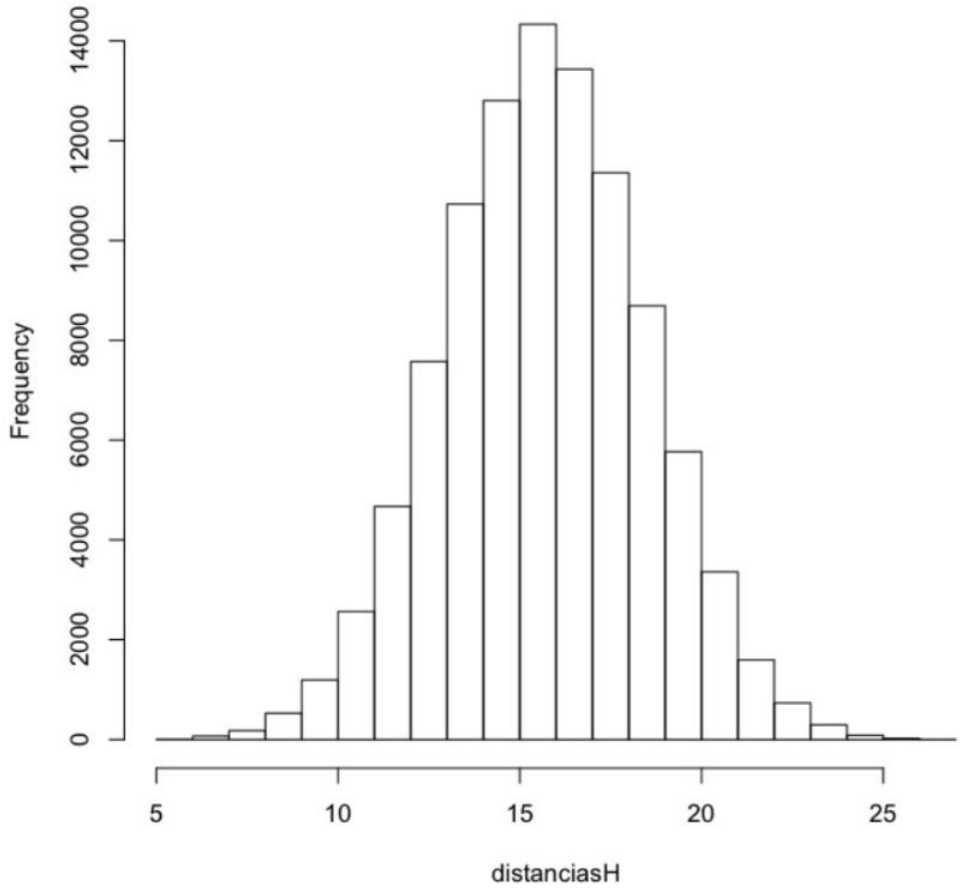
3. Principios de diseño del DES

• No-Linealidad de las S-Boxes

Para demostrar la no linealidad de las S-Boxes del DES, debemos demostrar que se cumple en todos los casos la siguiente sentencia:

$$f(a \text{ xor } b) \neq f(a) \text{ xor } f(b)$$

Para ello, tomamos dos números aleatorios de 48 bits *a*, *b* y *c* = *a xor b* y los hacemos pasar por todas las S-Boxes para obtener así *f(a)*, *f(b)* y *f(a xor b)*. Para comparar los resultados, hemos utilizado la distancia de *Hamming* entre *f(a xor b)* y *f(a) xor f(b)*. Para todos los test realizados, obtenemos un histograma de este estilo:



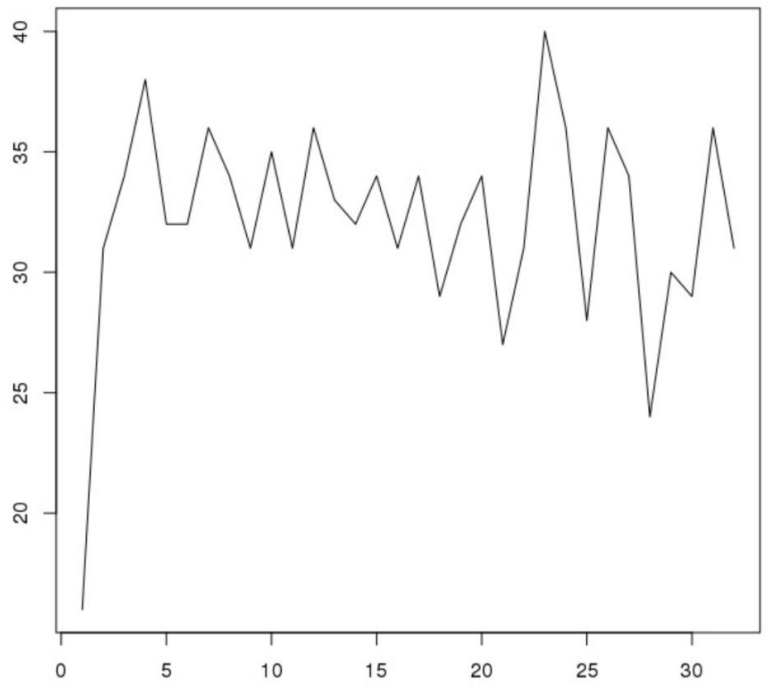
Analizando el gráfico, podemos observar que los resultados difieren de media en 16 bits, lo que nos transmite la seguridad de afirmar la no linealidad de las S-Box del DES.

Efecto de avalancha

El criterio de avalancha del DES postula que un bit de la salida debe cambiar si complementamos un bit de la entrada con una probabilidad de 0,5. Para probar este concepto, hemos creado un programa que, una vez generada las entradas aleatorias y se pasan por la S-Box, obtenemos la salida con la que comprobamos el número de 1's que hemos obtenido en la salida por cada bit hasta los 32 de la salida:

```
bit 0 300223 cambios (50.037 %)

bit 1 300686 cambios (50.114 %)

bit 2 299965 cambios (49.994 %)

...

bit 29 298956 cambios (49.826 %)

bit 30 300423 cambios (50.070 %)

bit 31 299732 cambios (49.955 %)
```

También hemos utilizado el concepto de la distancia de Hamming para probar este criterio. Para ello, hemos comparado la distancia entre la salida con la entrada original en función del número de rondas del DES. Para ello, generamos varias salidas diferentes (unas 32):



Observando el gráfico, podemos ver que de media se cambian unos 30-32 bits. Sin embargo, vemos que en los primeros valores del eje x, no se realizan tantos

cambios. Según nuestro estudio, esto se trata de una de las desventajas de la S-Box del DES frente a la del AES.

4. Principios de diseño del AES.

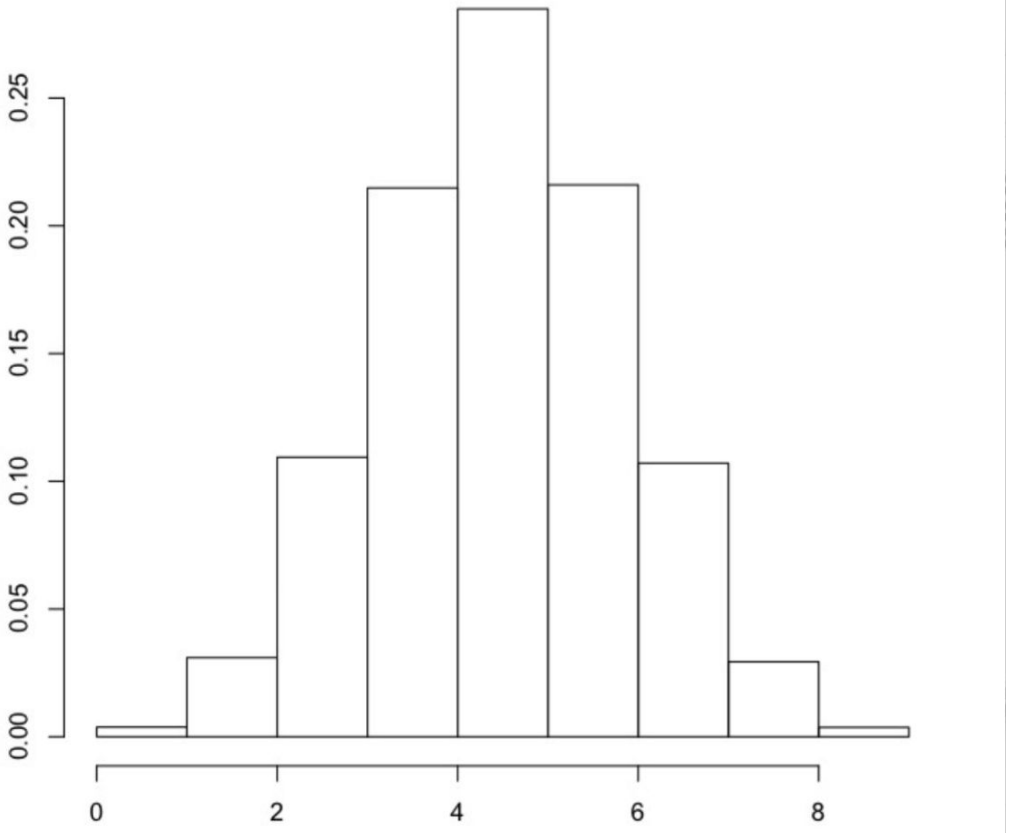
• No-Linealidad de las S-Boxes

En este apartado vamos a comprobar la no linealidad de las S-Boxes del AES, por lo tanto tenemos que demostrar:

f(a xor b) != f(a) xor f(b)

Hemos utilizado un método similar al realizado para demostrar la no linealidad de las S-Boxes en el DES. Dados dos número aleatorios de 8 bits, a y b, y tomando otro, c = a xor b, los hacemos pasar por todas las S-Boxes para obtener así *f(c) = f(a xor b)* junto con *f(a)* y *f(b)*.

Para comparar los resultados hemos utilizado la distancia de Hamming entre *f(c)=f(a xor b)* y *f(a) xor f(b)*. Para ello realizamos varios tests dando diferentes valores de *a* y *b* y generamos un histograma para analizar los resultados:



En el eje x tenemos las distancias de Hamming. Podemos observar que en todos los test realizados, siempre obtenemos algunos valores en los que la distancia es 0, por lo que *f(a xor b) = f(a) xor f(b)* para algunos casos. Debido a esta contradicción, demostramos la no linealidad de las S-Boxes del AES.

Generación de las S-Boxes

Para la generación de las S-Boxes nos hemos basado en el cálculo de la *Rijndael S-Box*, que es la caja de sustitución utilizada en el cifrado de *Rijndael*, cifrado del cual se basa el AES.

La S-Box directa mapea una entrada de 8 bits *c,* y la convierte en otra secuencia de 8 bits *s* (*S(c)*), en las que ambas son interpretados como polinomios en GF(2^8). Primero, la entrada es convertida en su inverso multiplicativo mediante esta transformación afín:

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

donde [s_7, ..., s_0] es la salida de la *S-Box* y [b_7, ..., b_0] es el inverso multiplicativo convertido en vector. La transformación afín es la suma de múltiples rotaciones del byte como vector añadiendo la operación *XOR*:

$$s = b \oplus (b \lll 1) \oplus (b \lll 2) \oplus (b \lll 3) \oplus (b \lll 4) \oplus 63_{16}$$

La S-Box inversa es simplemente la S-Box directa revertida utilizando la transformación afín tal y como se muestra a continuación:

Si ejecutamos el programa "*sbox_AES*" con los diferentes parámetros, obtenemos

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$b = (s \lll 1) \oplus (s \lll 3) \oplus (s \lll 6) \oplus 5_{16}$$

las S-Box establecidas por convenio:

0x63	0x7c	0x77	0x7b	0xf2	0x6b	0x6f	0xc5	0x30	0x01	0x67	0x2b	0xfe	0xd7	0xab	0x76
0xca	0x82	0xc9	0x7d	0xfa	0x59	0x47	0xf0	0xad	0xd4	0xa2	0xaf	0x9c	0xa4	0x72	0xc0
0xb7	0xfd	0x93	0x26	0x36	0x3f	0xf7	0xcc	0x34	0xa5	0xe5	0xf1	0x71	0xd8	0x31	0x15
0x04	0xc7	0x23	0xc3	0x18	0x96	0x05	0x9a	0x07	0x12	0x80	0xe2	0xeb	0x27	0xb2	0x75
0x09	0x83	0x2c	0x1a	0x1b	0x6e	0x5a	0xa0	0x52	0x3b	0xd6	0xb3	0x29	0xe3	0x2f	0x84
0x53	0xd1	0x00	0xed	0x20	0xfc	0xb1	0x5b	0x6a	0xcb	0xbe	0x39	0x4a	0x4c	0x58	0xcf
0xd0	0xef	0xaa	0xfb	0x43	0x4d	0x33	0x85	0x45	0xf9	0x02	0x7f	0x50	0x3c	0x9f	0xa8
0x51	0xa3	0x40	0x8f	0x92	0x9d	0x38	0xf5	0xbc	0xb6	0xda	0x21	0x10	0xff	0xf3	0xd2
0xcd	0x0c	0x13	0xec	0x5f	0x97	0x44	0x17	0xc4	0xa7	0x7e	0x3d	0x64	0x5d	0x19	0x73
0x60	0x81	0x4f	0xdc	0x22	0x2a	0x90	0x88	0x46	0xee	0xb8	0x14	0xde	0x5e	0x0b	0xdb
0xe0	0x32	0x3a	0x0a	0x49	0x06	0x24	0x5c	0xc2	0xd3	0xac	0x62	0x91	0x95	0xe4	0x79
0xe7	0xc8	0x37	0x6d	0x8d	0xd5	0x4e	0xa9	0x6c	0x56	0xf4	0xea	0x65	0x7a	0xae	0x08
0xba	0x78	0x25	0x2e	0x1c	0xa6	0xb4	0xc6	0xe8	0xdd	0x74	0x1f	0x4b	0xbd	0x8b	0x8a
0x70	0x3e	0xb5	0x66	0x48	0x03	0xf6	0x0e	0x61	0x35	0x57	0xb9	0x86	0xc1	0x1d	0x9e
0xe1	0xf8	0x98	0x11	0x69	0xd9	0x8e	0x94	0x9b	0x1e	0x87	0xe9	0xce	0x55	0x28	0xdf
0x8c	0xa1	0x89	0x0d	0xbf	0xe6	0x42	0x68	0x41	0x99	0x2d	0x0f	0xb0	0x54	0xbb	0x16

0x52	0x09	0x6a	0xd5	0x30	0x36	0xa5	0x38	0xbf	0x40	0xa3	0x9e	0x81	0xf3	0xd7	0xfb
0x7c	0xe3	0x39	0x82	0x9b	0x2f	0xff	0x87	0x34	0x8e	0x43	0x44	0xc4	0xde	0xe9	0xcb
0x54	0x7b	0x94	0x32	0xa6	0xc2	0x23	0x3d	0xee	0x4c	0x95	0x0b	0x42	0xfa	0xc3	0x4e
0x08	0x2e	0xa1	0x66	0x28	0xd9	0x24	0xb2	0x76	0x5b	0xa2	0x49	0x6d	0x8b	0xd1	0x25
0x72	0xf8	0xf6	0x64	0x86	0x68	0x98	0x16	0xd4	0xa4	0x5c	0xcc	0x5d	0x65	0xb6	0x92
0x6c	0x70	0x48	0x50	0xfd	0xed	0xb9	0xda	0x5e	0x15	0x46	0x57	0xa7	0x8d	0x9d	0x84
0x90	0xd8	0xab	0x00	0x8c	0xbc	0xd3	0x0a	0xf7	0xe4	0x58	0x05	0xb8	0xb3	0x45	0x06
0xd0	0x2c	0x1e	0x8f	0xca	0x3f	0x0f	0x02	0xc1	0xaf	0xbd	0x03	0x01	0x13	0x8a	0x6b
0x3a	0x91	0x11	0x41	0x4f	0x67	0xdc	0xea	0x97	0xf2	0xcf	0xce	0xf0	0xb4	0xe6	0x73
0x96	0xac	0x74	0x22	0xe7	0xad	0x35	0x85	0xe2	0xf9	0x37	0xe8	0x1c	0x75	0xdf	0x6e
0x47	0xf1	0x1a	0x71	0x1d	0x29	0xc5	0x89	0x6f	0xb7	0x62	0x0e	0xaa	0x18	0xbe	0x1b
0xfc	0x56	0x3e	0x4b	0xc6	0xd2	0x79	0x20	0x9a	0xdb	0xc0	0xfe	0x78	0xcd	0x5a	0xf4
0x1f	0xdd	0xa8	0x33	0x88	0x07	0xc7	0x31	0xb1	0x12	0x10	0x59	0x27	0x80	0xec	0x5f
0x60	0x51	0x7f	0xa9	0x19	0xb5	0x4a	0x0d	0x2d	0xe5	0x7a	0x9f	0x93	0xc9	0x9c	0xef
0xa0	0xe0	0x3b	0x4d	0xae	0x2a	0xf5	0xb0	0xc8	0xeb	0xbb	0x3c	0x83	0x53	0x99	0x61
0x17	0x2b	0x04	0x7e	0xba	0x77	0xd6	0x26	0xe1	0x69	0x14	0x63	0x55	0x21	0x0c	0x7d