Universitat Politècnica de Catalunya

Criptografía

Práctica 2

Joan Marc Pastor Mackenzie

1. El cuerpo finito GF(2^8)

Las funciones GF_product_p, GF_es_generador, GF_tables, GF_product_t i GF_invers se encuentran implementadas en el documento **1/GF.py**.

Los tests realizados para hacer las comparativas de tiempo se pueden encontrar en **1/ExecutionTimes.py**. El tiempo de ejecución GF_product_t también incluye la generación de las tablas.

	$\sum_{a=1}^{255} \text{exec\_time}(\text{GF\_product\_p}(a, b))$	$\sum_{a=1}^{255} \text{exec\_time}(\text{GF\_product\_t}(a, b))$
b = 0x02	0.997558 milliseconds	0.951904 milliseconds
b = 0x03	0.997802 milliseconds	0.997558 milliseconds
b = 0x09	1.025146 milliseconds	0.996582 milliseconds
b = 0x0B	0.997314 milliseconds	0.997802 milliseconds
b = 0x0D	0.997802 milliseconds	0.994628 milliseconds
b = 0x0E	0.994140 milliseconds	0.922607 milliseconds

La diferencia entre los tiempos de ejecución entre cada función es mínima. Este resultado es lo esperado ya que para generar las tablas hace falta realizar 255 ejecuciones de GF_product_p. Si suponemos que las tablas están previamente generadas, el tiempo de ejecución de GF_product_t sería reducido drásticamente. Este último caso no se pudo medir con la librería *time* de Python ya que al ser un tiempo tan pequeño devuelve 0.

2. Advanced Encryption Standard (AES)

2. 1 Efectos de las funciones elementales

Los tests para analizar los siguientes comportamientos se pueden encontrar en **2/2.1.elemental_function_tests.py**.

Implementación de AES: https://github.com/boppreh/aes

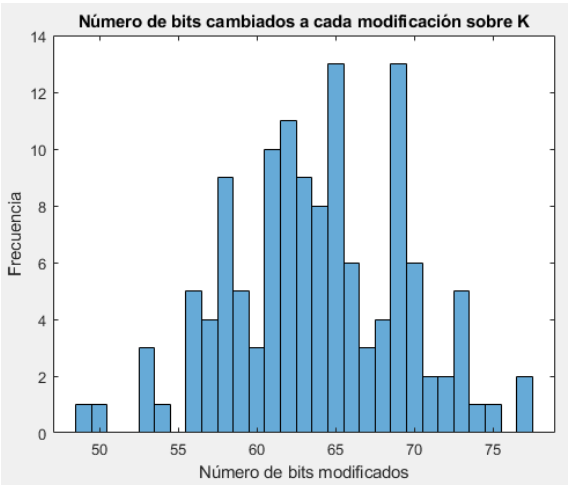
1. ByteSub por identidad
Intercambiar los bytes de posición mediante la S-Box provoca que no haya linealidad. Es por eso que, si se anula el ByteSub, el bloque resultante es igual a la acumulación de los bloques (i,j) mediante operadores XOR.
Test: 2/elemental_function_tests.py -> test_no_ByteSub()

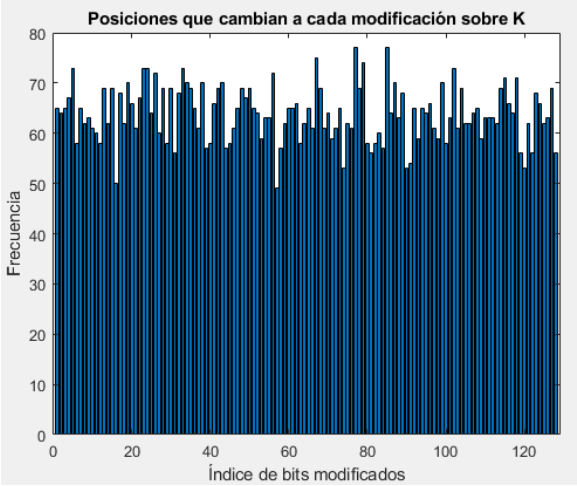
2. ShiftRows por identidad
Sin la operación ShiftRows, la modificación de un bit en un bloque de mensaje provoca solo la modificación de la columna del bloque cifrado donde se encontraba el bit modificado. ShiftRows sirve para añadir difusión al cifrado.
Test: 2/elemental_function_tests.py -> test_no_ShiftRows()

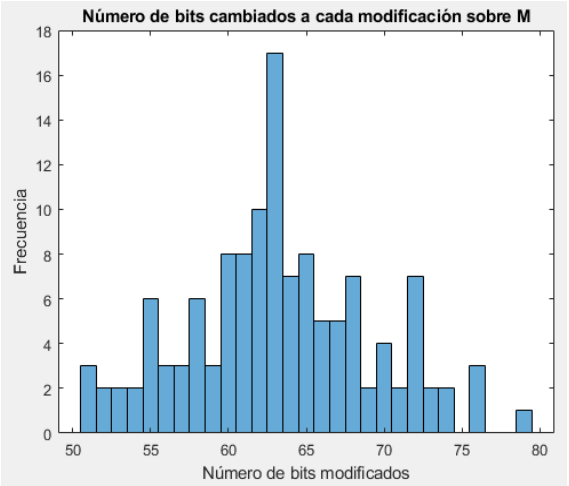
3. MixColumns por identidad
Al modificar un bit en el mensaje sin MixColumns provoca que se modifique el Byte perteneciente a ese bit en el bloque cifrado. Más difusión.
Test: 2/elemental_function_tests.py -> test_no_MixColumns()

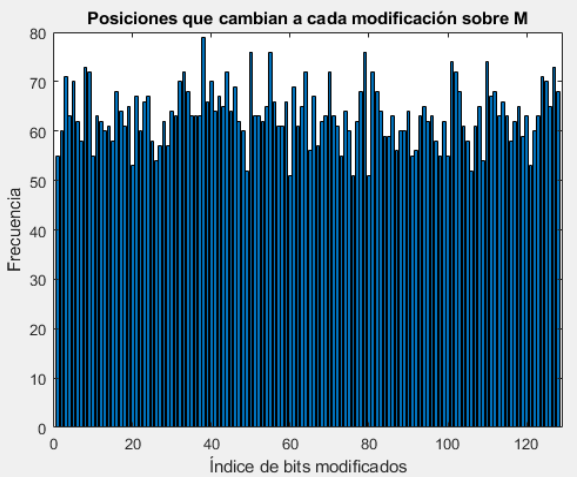
2.2 Propagación de pequeños cambios

Valores obtenidos con el script **2/2.2.propagation_tests.py**.
Key: 0x0102030405060708090a0b0c0d0e0f
Message: 0x00112233445566778899aabbccddeeff









2.3 Uso como a función unidireccional

Por tal de conseguir un ciphertext con el máximo números de 0 al principio, definimos un ciphertext de 128 bits con valor 0x00 y lo desciframos con una clave K cualquiera. Con la clave K usada podemos encriptar el mensaje obtenido para obtener un ciphertext con valor 0.

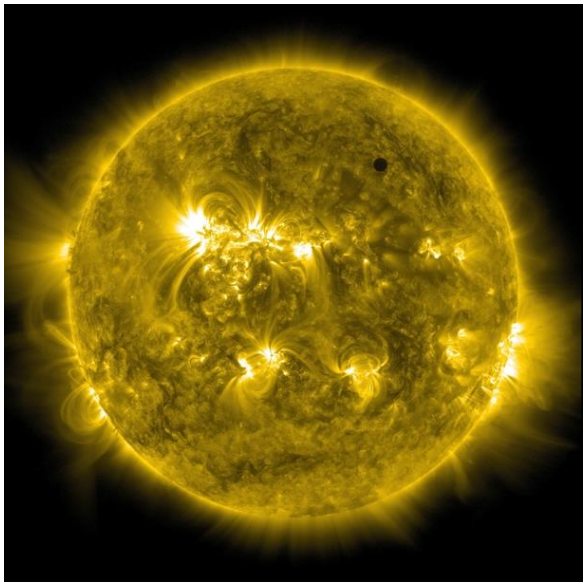
Una combinación podría ser:
Key: 0x00000000000000000000000000000000
Message: 0x140f0f1011b5223d79587717ffd9ec3a
Cipher obtenido: 0x00000000000000000000000000000000

Código: **2/2.3.cipher_with_0_value.py**

3. Criptografía de Clave Secreta

1.

Script: 3/decryptFirstFile.py
Encrypted file: 3/2019_09_25_17_02_22_joan.marc.pastor.enc
Decrypted file: 3/2019_09_25_17_02_22_joan.marc.pastor.dec
Content: JPEG Image



2.

Script: 3/decryptSecondFile.py
Candidate decrypted files: 3/out
PreMasterKey: 0x71717171717171712b2b2b2b2b2b2b2b
Encrypted file: 3/ 2019_09_25_17_02_22_joan.marc.pastor.puerta_trasera.enc
Decrypted file: 3/ 2019_09_25_17_02_22_joan.marc.pastor.puerta_trasera.dec
Content: MP4 File