Criptografía
Práctica SL4
Informe de prácticas

1. **Obtenga y/o genere fichero de texto, una imagen BMP, un fichero formado por datos aleatorios y un fichero formado por un único byte repetido y encripte y desencripte estos ficheros usando diferentes algoritmos (AES, DES, CAMELLIA, BLOWFISH) y diferentes modos de cifrado (ECB, CBC, CFB). Recopile cuantos resultados pueda de este proceso (memoria usada, tiempo necesario para encriptar y desencriptar,…). Muestre en una o varias tablas los resultados obtenidos (tiempos, tamaños,…) y coméntelos.**

   Los ficheros utilizados durante los test son los siguientes (Se ha tratado de conseguir que todos tuvieran un tamaño similar: 350-400KB):

   - **Imagen BMP**: Se ha utilizado la imagen que se ofrece junto con el guion de la práctica (test.bmp).
   - **Fichero de Texto**: Obtenido de la red, consiste en parte de un libro en inglés.
   - **Fichero de Texto Aleatorio**: Se ha generado mediante un script script en Python y la redirección por línea de comandos. El Script se muestra a continuación.
   - **Fichero de Texto con Byte Repetido**: Se ha generado repitiendo el carácter “a” hasta alcanzar el tamaño requerido.

   El script para generar el fichero aleatorio es el siguiente:

```python
import random, string

def randomword(length):
   return ''.join(random.choice(string.lowercase) for i in range(length))

print randomword(400000)
```

   En cuanto a los algoritmos de cifrado utilizados en la comparativa, se han utilizado los siguientes (Todos ellos son de cifrado simétrico, es decir, utilizan la misma clave tanto para el cifrado como para el descifrado):

   - **AES**: Se basa en cifrado por bloques, tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 o 256 bits. Los cálculos se hacen en un espacio finito o de Galois. Se basa en 4 operaciones básicas denominadas *subBytes*, *shiftRows*, *mixColumns* y *addRoundKey*, que se repiten un número determinado de veces tanto

en el cifrado como en el descifrado. Actualmente es el estándar de seguridad utilizado por el Gobierno de Estados Unidos.

- **DES**: El tamaño del bloque es de 64 bits. La clave tiene una longitud de64 bits, aunque en realidad, sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad, y después son descartados. Para el proceso de cifrado utiliza una función matemática denominada F de Feistel. Fue el estándar hasta que en 2001 fue remplazado por AES.

- **CAMELLIA**: Al igual que DES, también utiliza la función de Feistel con 18 o 24 rondas dependiendo del tamaño de la clave. Cada seis rondas, se aplica una capa de transformación lógica: la llamada "función FL" o su inversa. Camellia utiliza cuatro matrices de 8 × 8 bits con transformaciones afines de entrada y salida y operaciones lógicas. El cifrado también utiliza el blanqueamiento de claves de entrada y salida.

- **BLOWFISH**: Tiene un tamaño de bloque de 64 bits y una longitud de clave variable de 32 bits a 448 bits. Utiliza la función de Feistel en 16 rondas y grandes matrices dependientes de la clave. En estructura se asemeja a CAST.

Otro de los puntos a tener en cuenta durante la comparativa es el modo de cifrado de bloque:

- **ECB**: Es el modo de cifrado más simple, en el cual el mensaje es dividido en bloques, cada uno de los cuales se cifra de manera separada. La desventaja de este método es que bloques idénticos de mensaje sin cifrar producirán idénticos textos cifrados. Por esto, no proporciona una auténtica confidencialidad y no es recomendado para protocolos criptográficos, como apunte no usa el vector de inicialización, sino que únicamente se basa en la clave del algoritmo de encriptación.

- **CBC**: En este modo, antes de ser cifrado, a cada bloque de texto se le aplica una operación XOR con el previo bloque ya cifrado. De esta manera, cada bloque cifrado depende de todos los bloques de texto claros usados hasta ese punto. Además, para hacer cada mensaje único se debe usar un vector de inicialización en el primer bloque.

- **CFB**: Este modo es similar al anterior, solo que en lugar de cifrar el texto plano y hacer la operación XOR con el mensaje anterior (o vector de inicialización en el primer caso), lo que hace es cifrar el mensaje anterior (o IV) y después hacer el XOR con el texto plano.

Para tratar de automatizar el proceso de obtención de resultados se ha desarrollado un script para Shell que cifra y descifra los ficheros escribiendo los resultados de tiempo, memoria, etc en otro fichero denominado readme. El código fuente de este script es el siguiente:

```bash
#!/bin/bash

#
# Descrition: Obtains performance results of ciphers.
#
# Author:   Sergio García Prado
#           garciparedes.me
#

password="password"

CypherAlg=( "aes-256" "des" "camellia-256" "bf")
CypherMode=( "ecb" "cbc" "cfb")


mkdir -p encrypted/$1


exec 3>&1 4>&2 >encrypted/$1/readme 2>&1

for i in "${CypherAlg[@]}"
do
    for j in "${CypherMode[@]}"
    do
        echo
        echo $1 $i-$j
        echo "Cipher:"
        /usr/bin/time -v openssl $i-$j -in $1 -out encrypted/$1/$1.$i-$j -k ${password}
        echo "Decipher:"
        /usr/bin/time -v openssl $i-$j -d -in encrypted/$1/$1.$i-$j -out $1 -k  ${password}
    done
done

# restore stdout and stderr
exec 1>&3 2>&4
```

Los resultados obtenidos han sido los siguientes:

Tiempo (segundos)			**Imagen BMP**	**Fichero Texto**	**Fichero Aleatorio**	**Fichero Byte Repetido**
AES-256	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				
DES	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				
CAMELLIA-256	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				

BLOWFISH	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				

Memoria (kb)			**Imagen BMP**	**Fichero Texto**	**Fichero Aleatorio**	**Fichero Byte Repetido**
AES-256	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				
DES	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				
CAMELLIA-256	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				
BLOWFISH	ECB	Cifrado				
		Descifrado				
	CBC	Cifrado				
		Descifrado				
	CFB	Cifrado				
		Descifrado				

2. **Escriba un programa, en el lenguaje de programación que desee, que calcule la media aritmética de los bytes de un fichero.**

   Para el cálculo de la media aritmética de los bytes del fichero se ha creado un script en el lenguaje Python, que recibe el nombre del fichero como argumento al programa e imprime por salida estándar el resultado. El código fuente del script es el siguiente:

```python
#!/usr/bin/env python

#
# Descrition: Calculates Aritmetical Median of argv[1] file.
#
# Author:   Sergio García Prado
#           garciparedes.me
#

import sys

with open(sys.argv[1], "rb") as f:
    byteArr = map(ord, f.read())
    f.close()
    size = len(byteArr)

    # Median
    sum = 0
    for byte in byteArr:
        sum += byte
    median = float(sum)/size
    print median
```

3. **Comprima con *gzip* los cuatro ficheros usados en la cuestión 1. Usando el programa anterior, obtenga la media aritmética de los bytes que componen los cuatro ficheros usados en la cuestión 1, sus correspondientes archivos comprimidos y sus correspondientes archivos encriptados con AES-ECB, AES-CBC, DES-ECB y DES-CBC. Muestre los resultados en una tabla y coméntelos. ¿De qué puede ser un indicador la media aritmética de los bytes de un fichero?**
   [SUS RESPUESTAS]

Media Aritmética		**Imagen BMP**		**Fichero Texto**		**Fichero Aleatorio**		**Fichero Byte Repetido**	
		P	**C**	**P**	**C**	**P**	**C**	**P**	**C**
AES-256	**ECB**								
	CBC								
	CFB								
DES	**ECB**								
	CBC								
	CFB								
CAMELLIA-256	**ECB**								
	CBC								
	CFB								
BLOWFISH	**ECB**								
	CBC								
	CFB								

4. **Escriba un programa, en el lenguaje de programación que desee, que calcule la entropía de la información de un fichero.**

```python
#!/usr/bin/env python

#
# Descrition: Calculates Shannon Entropy of argv[1] file.
#
# Author:   Sergio García Prado
#           garciparedes.me
#

import sys
import math
import pandas

with open(sys.argv[1], "rb") as f:
    byteArr = map(ord, f.read())
    f.close()
    fileSize = len(byteArr)

    #Obtain frequency table
    freqList = pandas.Series(byteArr).value_counts(normalize = True)

    # Shannon entropy
    ent = 0.0
    for freq in freqList:
        ent = ent + freq * math.log(freq, 2)
    ent = -ent
    print ent
```

5. **Usando el programa anterior, obtenga la entropía de la información de los cuatro ficheros usados en la cuestión 1, sus correspondientes archivos comprimidos con *gzip* y sus correspondientes archivos encriptados con AES-ECB, AES-CBC, DES-ECB y DES-CBC. Muestre los resultados en una tabla y coméntelos. ¿Por qué $\mathcal{H}(X) \in [0,8]$? ¿Existe alguna relación entre la entropía, $\mathcal{H}(X)$, de los ficheros sin comprimir y el tamaño de los correspondientes ficheros comprimidos?**

Entropía		Imagen BMP		Fichero Texto		Fichero Aleatorio		Fichero Byte Repetido	
		P	C	P	C	P	C	P	C
AES-256	ECB								
	CBC								
	CFB								
DES	ECB								
	CBC								
	CFB								
CAMELLIA-256	ECB								
	CBC								
	CFB								
BLOWFISH	ECB								
	CBC								
	CFB								

6. **Encripte la imagen *test.bmp* usando el algoritmo AES con los modos de cifrado ECB y CBC. Salve una copia de la imagen cifrada, copie la cabecera de 54 bytes de la imagen original en el fichero cifrado y muestre (corte y pegue en la respuesta) las tres imágenes. Comente y explique, a la luz de los resultados, por qué sucede esto.**
   [SUS RESPUESTAS]