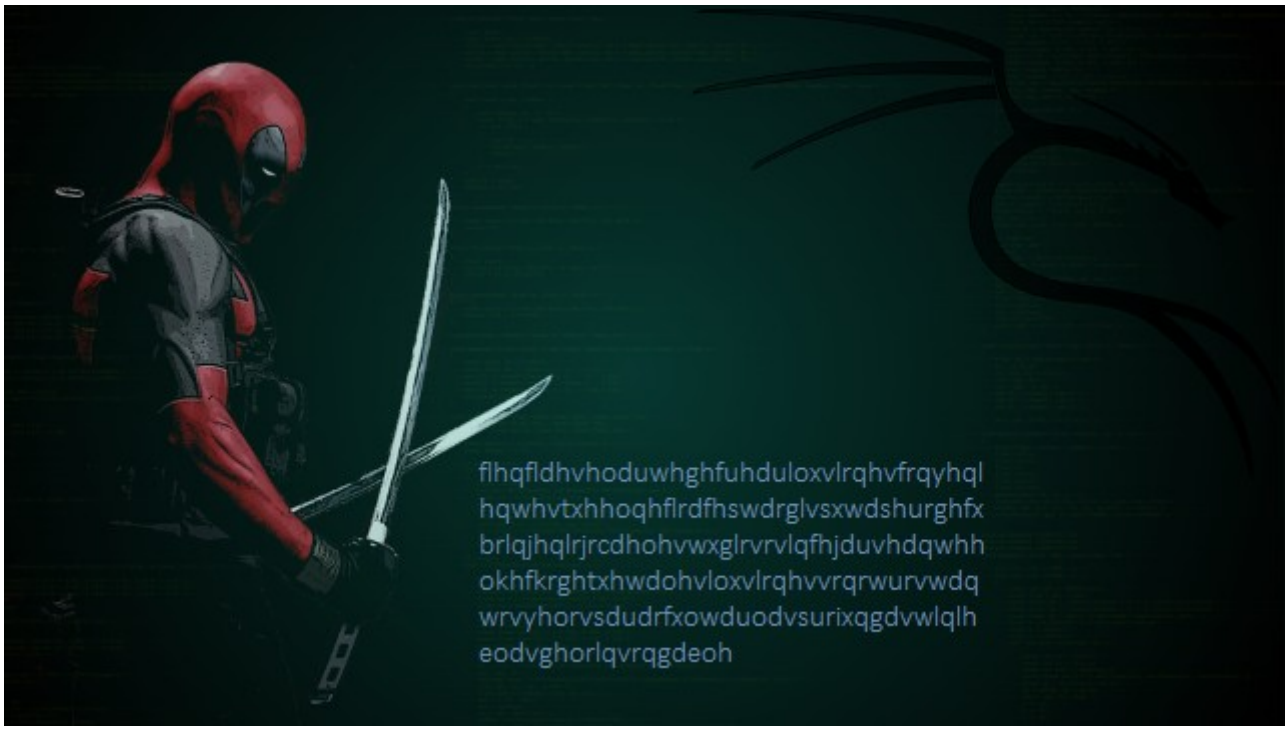
Solución del tercer criptoreto de http://www.criptored.upm.es/paginas/retos.htm

El enunciado hace referencia a un archivo de imagen dragon.png.



Estudiándola, se observan varios detalles:

- sobre la imagen aparece un texto, que parece cifrado.
- después de la marca de fin de fichero del formato PNG, hay añadidos una cantidad de bytes aparentemente aleatorios.

Fase 1: Descifrado del texto.

Haciendo unas pocas pruebas se ve que el texto:

flhqfldhvhoduwhghfuhduloxvlrqhvfrqyhql
hqwhvtxhhoqhflrdfhswdrglvsxwdshurghfx
brlqjhqlrjrcdhohvwxglrvrvlqfhjduvhdqwhh
okhfkrghtxhwdohvloxvlrqhvvrqrwurvwdq
wrvyhorvsdudrfxowduodvsurixqgdvwlqlh
eodvghorlqvrqgdeoh

, está cifrado con el cifrado Cesar (https://es.wikipedia.org/wiki/Cifrado_C%C3%A9sar). Descifrándolo (sustituyendo cada letra por la letra tres posiciones a la izquierda según el alfabeto), queda:

cienciaeselartedecrearilusionesconveni
entesqueelnecioaceptaodisputaperodecu
yoingeniogozaelestudiososincegarseantee
lhechodequetalesilusionessonotrostan
tosvelosparaocultarlasprofundastinie
blasdeloinsondable

, frase de Carl Gustav Jung.

Parece que la frase sólo era para indicar que había información oculta con esteganografía...

Fase segunda: Openstego

Según las **pistas**:

Pista1 criptoreto: es necesario pasar 3 fases de Stego para ver el criptograma...
Pista 2: el fichero PNG contiene mucha información ;)
Pista 3: el texto que se ve en la imagen es solo una pista que te indica que se mire Stego..
Pista 4: Quizás openstego sea un buen comienzo...

Para extraer información de la imagen utilizando Openstego, es necesaria una contraseña.
Probando con varias palabras sencillas, un poco de fuerza bruta con diccionario, etc... finalmente doy con la contraseña: “criptored”.
Utilizando Openstego y “criptored” como contraseña, extrae un fichero llamado “msdragon.txt”, con contenido: “alanturingcrypto”.

$ java -jar openstego-0.6.1/lib/openstego.jar extract -a RandomLSB -sf dragon.png -xd data -p criptored
Extracted file: data/msgdragon.txt

$ hexdump -C data/msgdragon.txt
```
00000000  61 6c 61 6e 74 75 72 69  6e 67 63 72 79 70 74 6f  |alanturingcrypto|
00000010
```

Ummm... es un texto de 16 bytes....
Y los bytes del final del fichero (tras la marca de final de imagen PNG), comienzan por 16 bytes “\x10”, y luego bytes aparentemente aleatorios.... tal vez sea cifrado AES, key: “alanturingcrypto”, “\x10”*16 puede ser un padding, o un iv, o....

Tras varios intentos con diferentes modos de cifrado de bloques, no sale nada... toca esperar a las siguientes pistas.

Fase tercera: Corkami

Pista 5: A corkami le gustan MUCHO los pngs..

Pista 6: Corkami sin openstego no es nadie...

Las pistas nos guían hacia https://code.google.com/p/corkami/, de ange albertini, autor de los maravillosos posters de formatos de ficheros: https://imgur.com/a/MtQZv/

Leyendo documentos del repositorio, parece que este tiene mucha relación con el criptoreto:

https://code.google.com/p/corkami/source/browse/trunk/src/angecryption/slides/AngeCryption.pdf

En ese documento @corkami genera un fichero PNG, que tras descifrarlo (con cierta clave y cierta IV) el resultado es un fichero ZIP. El truco está en calcular un IV adecuado.

Parece que el autor del criptoreto ha hecho algo similar. Por esa razón en el archivo dragon.png, tras la marca de final de fichero del formato PNG, aparecen "\x10"*16 (padding), y a continuación muchos bytes en apariencia aleatorios.

Haciendo varias pruebas para conseguir diferentes formatos (ZIP, PDF, JPEG, PNG, ...), el que resulta un fichero completo con sentido es otro fichero PNG.

Usando python para calcular el IV adecuado, y cifrando con AES:

```
import struct
from Crypto.Cipher import AES
data = open('dragon.png').read()
def xor(a,b):
      return ''.join([chr(ord(i)^ord(j)) for i,j in zip(a,b)] )
def corkami(header):
    cipher = AES.new("alanturingcrypto", AES.MODE_CBC, "\x00"*16)
    h1 = cipher.decrypt(header)
    iv = xor(h1, data[:16])
    cipher = AES.new("alanturingcrypto", AES.MODE_CBC, iv)
    a = cipher.encrypt(data)
    return a
data2 = corkami("\x89PNG\r\n\x1a\n" +  struct.pack(">I", 191600) + "aaaa")
open('dragon2.png','wb').write(data2)
```

#191600 es la cantidad de bytes del PNG anterior, bytes que al cifrarlos quedan en la sección “AAAA” del archivo PNG nuevo (que es una sección que (la mayoría de) los visores de PNG no tienen en cuenta y lo pasarán por alto).

El archivo dragon2.png:

- es una imagen que representa un ojo.
- de nuevo tiene datos después de la marca de fin de fichero PNG.

```
00041f90  93 27 e8 d9 7f 2b a5 f2  a6 b0 97 5a 6a a9 c5 b5  |.'...+.....Zj...|
00041fa0  ff 0f a9 b6 0c 6d d2 0b  0e 32 00 00 00 00 49 45  |.....m...2....IE|
00041fb0  4e 44 ae 42 60 82 49 50  52 55 54 49 56 4f 54 4c  |ND.B`.IPRUTIVOTL|
00041fc0  49 54 4e 45 20 0d 0a 05  05 05 05 05 04 04 04 04  |ITNE ...........|
00041fd0
```

La parte final: “\x04”*4, es el padding del cifrado AES del paso anterior.

Los datos “extras” son el texto: “IPRUTIVOTLITNE \r\n” + “\x05”*5

El “\x05”*5 son otro padding, y el “\r\n” es un salto de línea en windows.

Fase cuarta: Criptograma

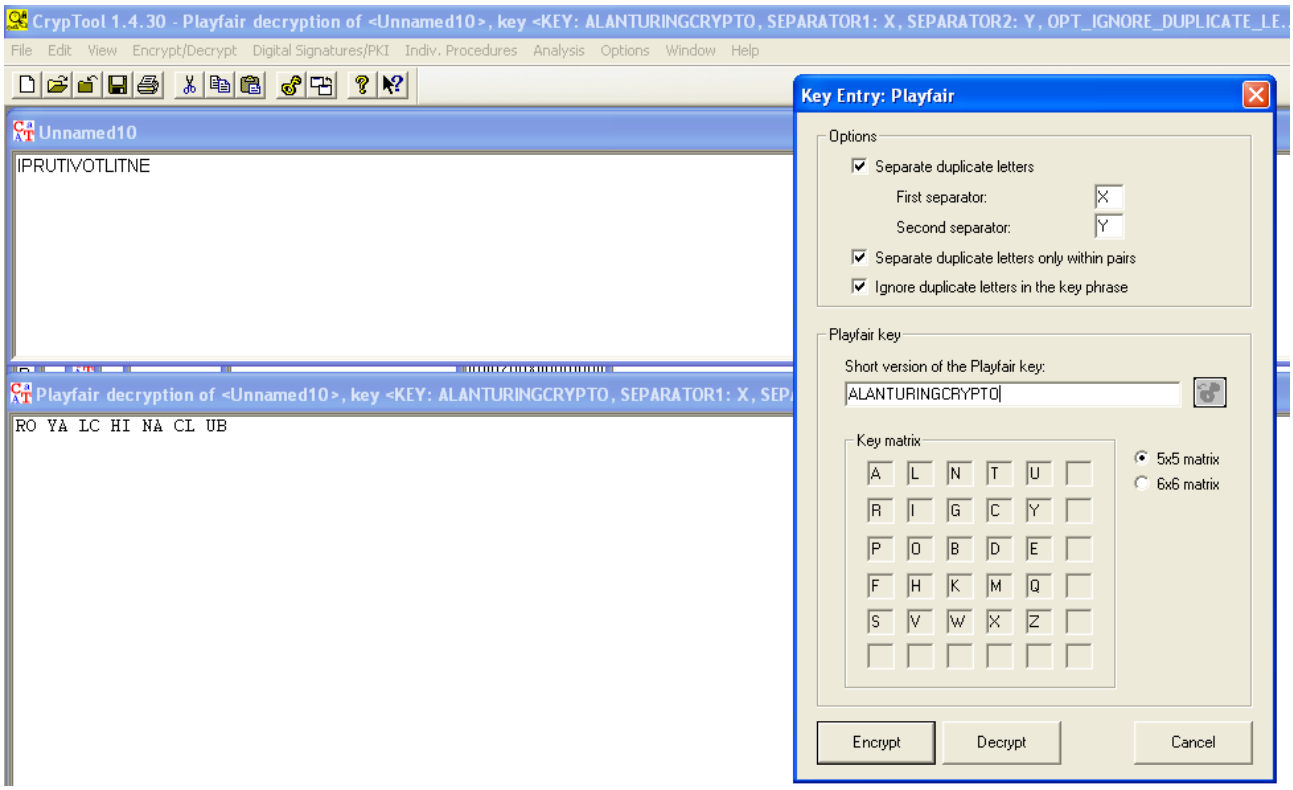
Analizando la imagen no parece que tenga nada más oculto, por lo que parece que la siguiente fase es analizar el criptograma: “IPRUTIVOTLITNE ”

Dado que el texto cifrado está formado por letras y está en mayúsculas (convención habitualmente usada en cifrados clásicos) puedo pensar que está cifrado con una cifra clásica.

Es un texto cifrado demasiado corto como para poder realizar algún tipo de análisis estadístico.

Mientras espero a nuevas pistas, hago una prueba básica con varios de los cifrados clásicos https://en.wikipedia.org/wiki/Classical_cipher y utilizando Cryptool, y usando como passwords palabras que hayan salido ya a lo largo del criptoreto, y suena la flauta (cifrado playfair, clave: alanturingcrypto):

https://en.wikipedia.org/wiki/Playfair_cipher



Solución: royal china club

Buscando el royal china club, en United Kingdom, se encuentra en el 40-42 de Baker Street, Londres.

2015-04-06
Dani Torregrosa, danitorwS
w0pr