Pràctica 2 - Criptografia

Descripció de la Pràctica a realitzar

Exercici 1 (5 punts)

En aquest primer exercici implementarem una variant del criptosistema *DES*, que anomenarem *UOC-DES*.

La variant *UOC-DES* implementarà el *DES* tal com està descrit en el mòdul didàctic de "xifres de clau compartida: xifres de bloc", però modificant el funcionament de les taules *S* de la següent manera:

> Cada caixa, S_j, rebrà el seu bloc corresponent de 6 bits $B_j = b_1b_2b_3b_4b_5b_6$, i en retornarà un de 4 bits de llargada, d'acord amb la taula inclosa a les pàgines 16 i 17 dels materials i el criteri per mitjà del qual s'assignarà una fila i una columna d'una caixa a B_j és el següent: l'índex *j* fixarà la caixa S_j, l'enter corresponent a b_1b_2 seleccionarà la fila i l'enter que correspon a $b_3b_4b_5b_6$ determinarà la columna.

Per a desenvolupar el *UOC-DES* cal que implementeu les següents funcions que us permetran validar la correcció de la vostra implementació amb el joc de proves que us proporcionem:

1. Implementeu una funció que permeti xifrar un bloc de 64 bits fent servir el *UOC-DES*. **(1 punt)**
   La funció prendrà com a arguments el bloc de bits a xifrar i una clau.
   a) `clar`: el primer argument contindrà el bloc de 64 bits a xifrar.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   La funció retornarà el bloc de bits xifrat fent servir *UOC-DES*.

   Implementada al fitxer adjunt *PRAC2.sws*.

2. Implementeu una funció que permeti desxifrar un bloc de 64 bits fent servir *UOC-DES*. **(1 punt)**
   La funció prendrà com a arguments el bloc xifrat i la clau.
   a) `xifrat`: el primer argument contindrà el bloc de 64 bits amb el contingut xifrat.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   La funció retornarà el bloc desxifrat.

   Implementada al fitxer adjunt *PRAC2.sws*.

3. Implementeu una funció que permeti xifrar una cadena de bits amb *UOC-DES* fent servir el mode d'operació *ECB*. **(0,5 punts)**
   La funció prendrà com a arguments una cadena de bits a xifrar i una clau.
   a) `clar`: el primer argument contindrà una cadena de bits amb el contingut a xifrar.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   La funció retornarà la cadena de bits xifrada fent servir *UOC-DES* en mode *ECB*.
   Per tal de simplificar la funció, assumirem que les cadenes rebudes sempre tenen una mida múltiple de 64 bits.

   Implementada al fitxer adjunt *PRAC2.sws*.

4. Implementeu una funció que permeti desxifrar cadenes de bits fent servir *UOC-DES* en mode *ECB*. **(0,5 punts)**
   La funció prendrà com a arguments el contingut xifrat i la clau. De la mateixa manera que amb la funció de xifratge, assumirem que les cadenes de bits a desxifrar sempre tenen una mida múltiple de 64 bits.
   a) `xifrat`: el primer argument contindrà una cadena de bits amb el contingut xifrat.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   La funció retornarà el contingut desxifrat.

5. Implementeu una funció que permeti xifrar una cadena de bits amb *UOC-DES* fent servir el mode d'operació *CBC*. **(0,5 punts)**
   La funció prendrà com a arguments una cadena de bits a xifrar, una clau i el vector inicial.
   a) `clar`: el primer argument contindrà una cadena de bits amb el contingut a xifrar.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   c) `vector_inicial`: el tercer argument contindrà una cadena de bits amb el vector inicial.
   La funció retornarà la cadena de bits xifrada fent servir *UOC-DES* en mode *CBC*.
   Per tal de simplificar la funció, assumirem que les cadenes rebudes sempre tenen una mida múltiple de 64 bits.

   Implementada al fitxer adjunt *PRAC2.sws*.

6. Implementeu una funció que permeti desxifrar cadenes de bits fent servir *UOC-DES* en mode *CBC*. **(0,5 punts)**
   La funció prendrà com a arguments el contingut xifrat, la clau i el vector d'inicialització fet servir. De la mateixa manera que amb la funció de xifratge, assumirem que les cadenes de bits a desxifrar sempre tenen una mida múltiple de 64 bits.
   a) `xifrat`: el primer argument contindrà una cadena de bits amb el contingut xifrat.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   c) `vector_inicial`: el tercer argument contindrà una cadena de bits amb el vector inicial.
   La funció retornarà el contingut desxifrat.

   Implementada al fitxer adjunt *PRAC2.sws*.

7. Implementeu una funció que permeti xifrar una cadena de bits amb *UOC-DES* fent servir el mode d'operació *CFB*. **(0,5 punts)**
   La funció prendrà com a arguments una cadena de bits a xifrar, una clau i el vector inicial.
   a) `clar`: el primer argument contindrà una cadena de bits amb el contingut a xifrar.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   c) `vector_inicial`: el tercer argument contindrà una cadena de bits amb el vector inicial.
   La funció retornarà la cadena de bits xifrada fent servir *UOC-DES* en mode *CFB*.
   Per tal de simplificar la funció, assumirem que les cadenes rebudes sempre tenen una mida múltiple de 64 bits. El mode *CFB* permet que la mida dels blocs a xifrar sigui diferent de la mida de bloc de l'algorisme de xifra que es fa servir. Tot i això, **assumirem que la mida dels blocs a xifrar és sempre la mida del bloc de l'algorisme de xifra**, en el nostre cas, els 64 bits del *UOC-DES*.

   Implementada al fitxer adjunt *PRAC2.sws*.

8. Implementeu una funció que permeti desxifrar cadenes de bits fent servir *UOC-DES* en mode *CFB*. **(0,5 punts)**
   La funció prendrà com a arguments el contingut xifrat, la clau i el vector d'inicialització fet servir. De la mateixa manera que amb la funció de xifratge, assumirem que les cadenes de bits a desxifrar sempre tenen una mida múltiple de 64 bits.
   a) `xifrat`: el primer argument contindrà una cadena de bits amb el contingut xifrat.
   b) `clau`: el segon argument contindrà una cadena de bits que representa la clau de xifratge.
   c) `vector_inicial`: el tercer argument contindrà una cadena de bits amb el vector inicial.
   La funció retornarà el contingut desxifrat.

   Implementada al fitxer adjunt *PRAC2.sws*.

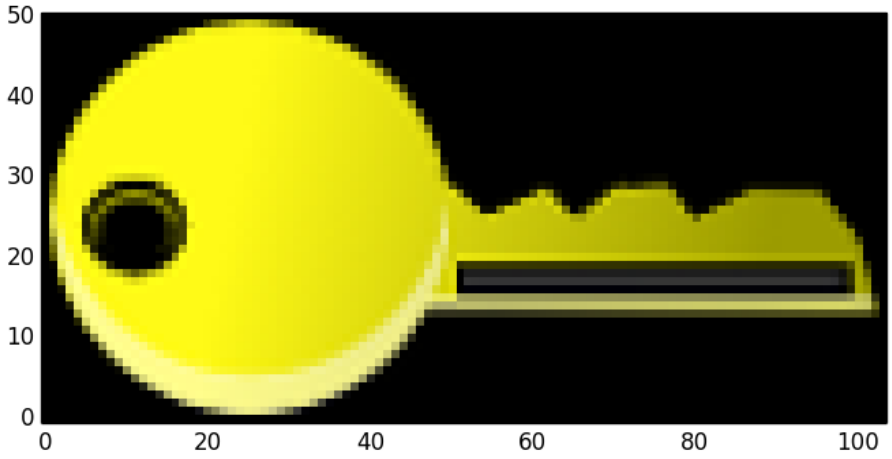
Exercici 2 (5 punts)

En aquesta segona part farem servir les funcions implementades a la primera part de la pràctica per analitzar el comportament dels diferents modes de xifrat, tot observant els efectes de xifrar imatges.
Atès que l'objectiu d'aquesta pràctica és estudiar els modes de xifra de bloc i no pas el tractament d'imatges amb *SAGE*, us proporcionem un petit tutorial que cobreix els aspectes no-criptogràfics de la implementació. El trobareu al fitxer *Pràctica 2 – Intro.sws*.

En la resposta a cada pregunta, copieu la imatge resultant de cada procés.

1. Fent servir les funcions que us detallem al tutorial, carregueu i dibuixeu la imatge continguda en el fitxer *image_64.bmp*.

```
img = image.imread(DATA+'image_64.bmp')
imshow(img,origin='lower',interpolation='nearest')
savefig('')
```



Després, fent servir tant les funcions desenvolupades a la primera part de la pràctica com les funcions de tractament d'imatges que us donem al tutorial, xifreu la imatge fent servir el mode *ECB* i mostreu la imatge resultant. És a dir,

1.- Carregueu la imatge i convertiu-la a una cadena de bits.

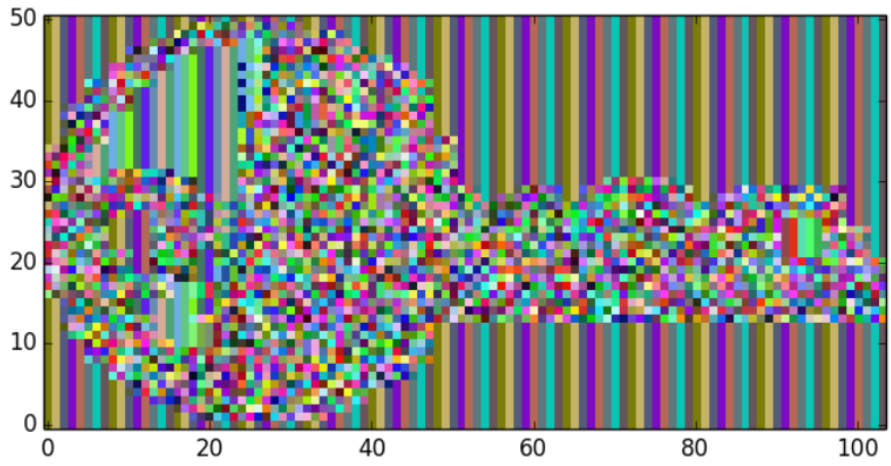
```
img = image.imread(DATA+'image_64.bmp')
(cadenaDeBits, dimensions) = Image_to_BinList(img)
```

2.- Xifreu la cadena de bits amb *ECB* fent servir una clau aleatòria.

```
key = "11221122"
bin_key = String_to_BitList(key)
Kn = create_sub_keys(bin_key)
cadenaDeBitsXifrada = UOC_DES_ECB_Cipher(cadenaDeBits, Kn)
```

3.- Convertiu la cadena de bits resultant (el contingut xifrat) a un array tridimensional i mostreu la imatge que representa.

```
matriuImatgeXifrada = BinList_to_Image(cadenaDeBitsXifrada,dimensions)
imshow(matriuImatgeXifrada,origin='lower',interpolation='nearest')
savefig('')
```



En la imatge xifrada que obteniu, es protegeix totalment la informació que es mostra en la imatge original? Es pot dir alguna cosa de la imatge original si només es té la corresponent imatge xifrada?

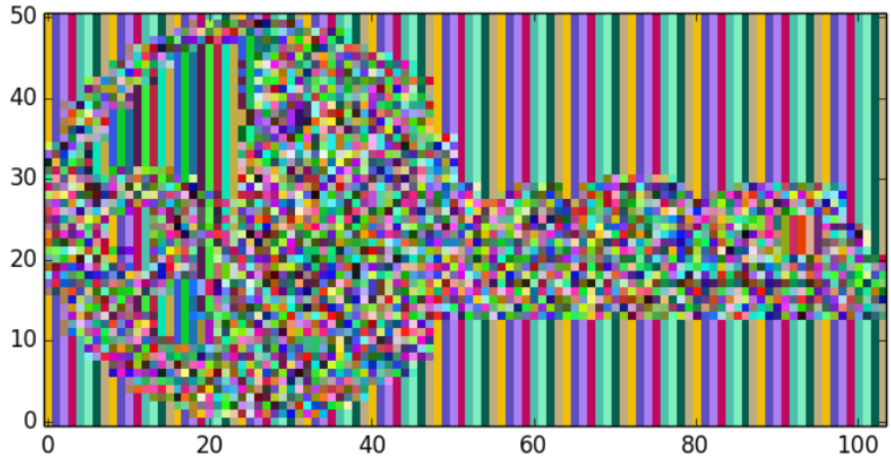
No, no es protegeix totalment la informació de la imatge original. Com podem observar a la imatge xifrada, encara es pot intuir el que hi havia a la imatge sense xifrar.

Per què es produeix aquest fenomen? **(1 punt)**

Per la manera en que treballa l'algorisme. En subdividir el blocs del missatge i xifrar independentment cada un d'ells fent ús de la mateixa clau, els patrons de repetició que puguen tenir els bloc originals també es presenten en els blocs xifrats.

2. Repetiu el procediment fent servir una clau de xifratge diferent.

key = "22112211"



Què observeu en aquest cas?

Únicament ha canviat els tons de color amb què es representa la imatge xifrada.

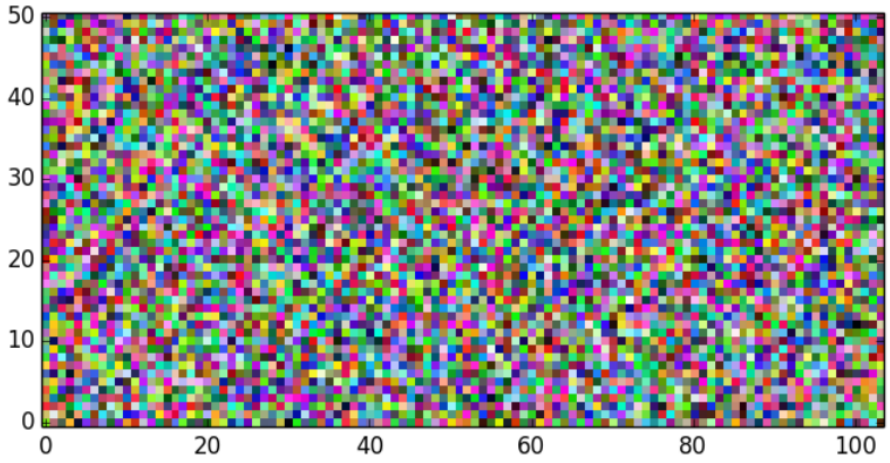
Per què es produeix aquest fenomen? **(1 punt)**

Un canvi de clau implica un canvi en el resultat de les operacions *XOR* que es fan a nivell d'algorisme de xifrat (respecte al cas anterior, clar), la qual cosa únicament modificarà el valor final dels bits de cada bloc –que representen cada un dels píxels (R,G,B)–, per això el canvi de color.

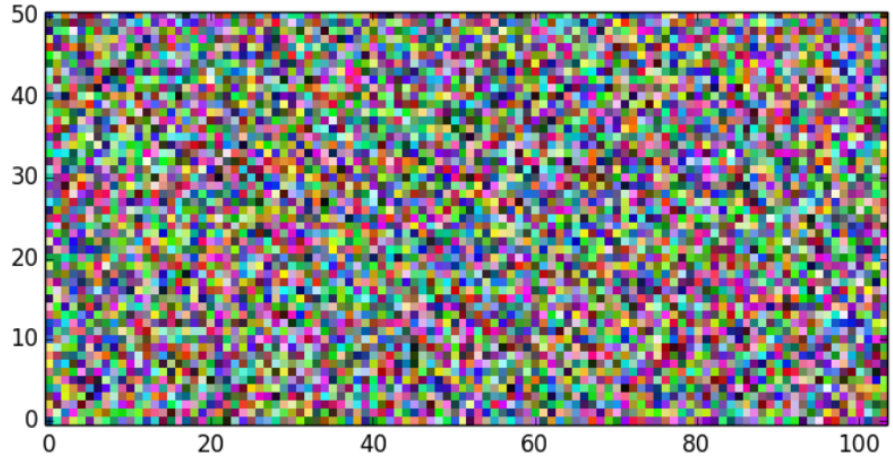
3. Realitzeu el mateix procediment fent servir el mode *CBC* en comptes d'*ECB*.

```
img = image.imread(DATA+'image_64.bmp')
(cadenaDeBits, dimensions) = Image_to_BinList(img)

key = "11221122"
bin_key = String_to_BitList(key)
Kn = create_sub_keys(bin_key)
iv = "01010101"
bin_iv = String_to_BitList(iv)

cadenaDeBitsXifrada = UOC_DES_CBC_Cipher(cadenaDeBits, Kn, bin_iv)

matriuImatgeXifrada = BinList_to_Image(cadenaDeBitsXifrada,dimensions)
imshow(matriuImatgeXifrada,origin='lower',interpolation='nearest')
savefig('')
```

Quin n'és el resultat?



Key = "22112211"



Què podeu observar en aquest cas?

En el mode *CBC* **Sí** que queda protegit el contingut de la imatge original, atès que, observant la imatge xifrada no és possible intuir el contingut de la imatge sense xifrar. No obstant això, encara que canvie la clau, com que el patró a xifrar és el mateix en ambdós casos, obtenim el mateix patró xifrat –canviant únicament els tons de color amb que es representa la imatge xifrada.

A què es deu la diferència entre aquest resultat i l'obtingut a la pregunta 1? **(1 punt)**

A la manera en que treballa l'algorisme. A diferència del *ECB*, amb el *CBC* un bloc xifrat passa a formar part del següent cicle de xifrat, eliminant així la possibilitat de replicació dels patrons que puguin haver del missatge original en el missatge xifrat.

4. Una vegada observats els efectes d'alguns modes sobre el contingut xifrat, passem a analitzar les conseqüències d'alguns processos sobre el text en clar obtingut després de desxifrar un missatge. En primer lloc, veiem la influència del vector inicial. Per fer-ho,

   1.- Carregueu la imatge (*imatge_64.bmp*) i convertiu-la a una cadena de bits.

```
img = image.imread(DATA+'image_64.bmp')
(cadenaDeBits, dimensions) = Image_to_BinList(img)
```

   2.- Xifreu la cadena de bits amb *CBC* fent servir la clau "WWWWWWWW" i el vector d'inicialització "EBEBEBEB".

```
key = "WWWWWWWW"
bin_key = String_to_BitList(key)
Kn = create_sub_keys(bin_key)
iv = "EBEBEBEB"
bin_iv = String_to_BitList(iv)
cadenaDeBitsXifrada = UOC_DES_CBC_Cipher(cadenaDeBits, Kn, bin_iv)
```

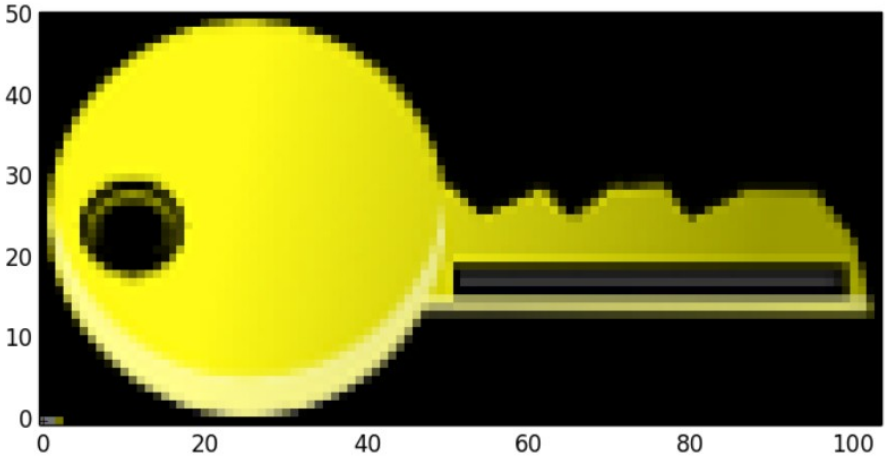
   3.- Desxifreu el contingut xifrat amb la clau correcta però fent servir un vector d'inicialització incorrecte ("00000000").

```
iv = "00000000"
bin_iv = String_to_BitList(iv)
cadenaDeBitsDesxifrada = UOC_DES_CBC_Decipher(cadenaDeBitsXifrada, Kn, bin_iv)
```

4.- Convertiu la cadena de bits resultant (el contingut desxifrat) en un array tridimensional i mostreu la imatge que representa.

```
matriuImatgeDesxifrada = BinList_to_Image(cadenaDeBitsDesxifrada,dimensions)
imshow(matriuImatgeDesxifrada,origin='lower',interpolation='nearest')
savefig('')
```

Quina imatge s'obté?



Per què s'obté aquesta imatge tot i fer servir un vector d'inicialització incorrecte? **(1 punt)**

A l'algorisme *CBC,* en el procés de desxifratge, el vector d'inicialització únicament afecta al primer bloc desxifrat, atès que és el bloc de text xifrat el que s'usa en el següent cicle de desxifratge, per tant, obtenim la imatge original, però amb algun "desperfecte" en el primer bloc desxifrat, com s'observa a la fila 0.

5. Per veure la influència de la clau,

1.- Carregueu la imatge i convertiu-la a una cadena de bits.

```
img = image.imread(DATA+'image_64.bmp')
(cadenaDeBits, dimensions) = Image_to_BinList(img)
```

2.- Xifreu la cadena de bits amb *CBC* fent servir la clau "XXXXXXX " i un vector d'inicialització aleatori.

```
key = "XXXXXXXX"
bin_key = String_to_BitList(key)
Kn = create_sub_keys(bin_key)
iv = "01010101"
bin_iv = String_to_BitList(iv)

cadenaDeBitsXifrada = UOC_DES_CBC_Cipher(cadenaDeBits, Kn, bin_iv)
```

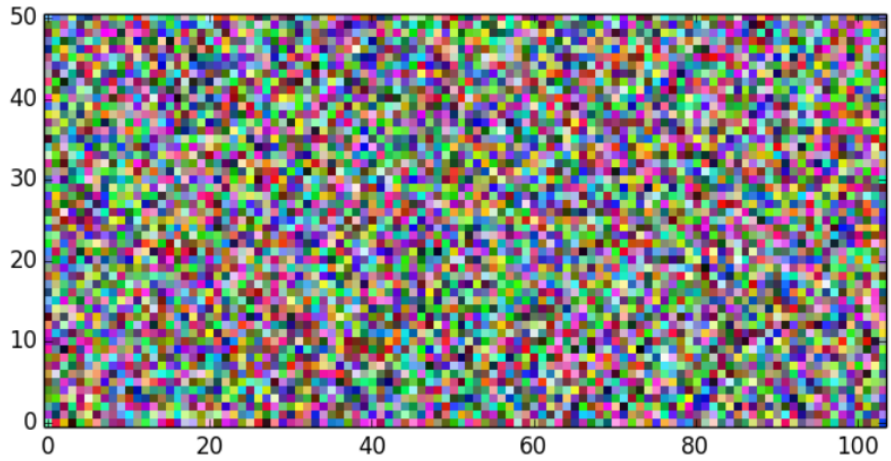
3.- Desxifreu el contingut xifrat amb la clau "XXXXXXX@" (fixeu-vos que només difereix en un sol bit de la clau de xifrat) i fent servir el mateix vector d'inicialització utilitzat en el pas 2.

```
key = "XXXXXXX@"
bin_key = String_to_BitList(key)
Kn = create_sub_keys(bin_key)
cadenaDeBitsDesxifrada = UOC_DES_CBC_Decipher(cadenaDeBitsXifrada, Kn, bin_iv)
```

4.- Convertiu la cadena de bits resultant (el contingut desxifrat) en un array tridimensional i mostreu la imatge que representa.

```
matriuImatgeDesxifrada = BinList_to_Image(cadenaDeBitsDesxifrada,dimensions)
imshow(matriuImatgeDesxifrada,origin='lower',interpolation='nearest')
savefig('')
```

Quina imatge s'obté?



Què en podem dir de la importància de la clau respecte a la importància del vector d'inicialització? **(1 punt)**

A diferència del vector d'inicialització, la clau emprada s'aplica en cada cicle de desxifrat a cada bloc xifrat, per tant, afectarà globalment al resultat, garantint així que, sense la clau, no es pugui obtenir el missatge original.