# Exercice 1

1. Complexité

Temps:

* 2 boucles de 2^56 itérations => 2\*2^56 = 2^57 opérations

Mémoire:

* 2 listes avec 2^56 éléments => 2\*2^56 = 2^57 éléments

Niveau de sécurité:

* Uniquement 2^57 et non 2^112 !!!!

1. Je considère.
2. 112 bits car uniquement deux clefs.

L’attaque meet-in-the-middle n’est plus possible, il faut au moins une boucle à 112 itérations.

1. Si K1=K2, cela revient à faire que DES. Car on chiffre puis déchiffre puis rechiffre.
2. 3 fois plus long car il faut le faire 3 fois.

# Exercice 2

1. Le même cryptage est utilisé pour toutes les entrées de la BDD.

La methode change rien on a deux lettres = 4 chiffres (je cherchais à retrouver la clef en fait) :

2)

Ja |ck |2 |10 |50 |00

Q9 |2D |FP |VX |C9 |IO

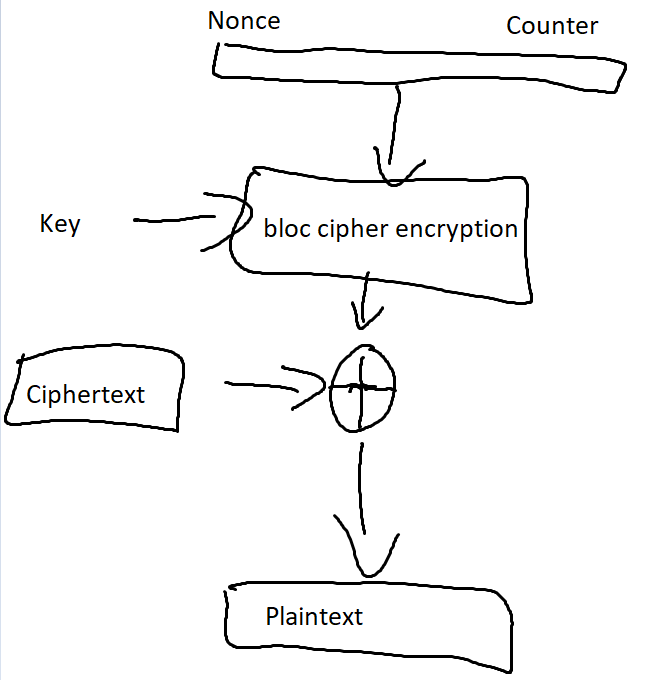
Ja |ne |2 |50 |00 |00

Q9 |AX |FP |C9 |IO |IO

3)Si un bloc correspond à un pixel, alors on va juste remplacer les pixel par d’autre couleur, mais la structure restera bien visible.

# Exercice 3

1. On connaît toutes les clés pour chaque bloc car on peut facilement déduire le counter du block et le nonce est connu.



1. Le nonce doit être envoyé au destinataire avec le chiffré. Il n'a pas besoin de rester secret. Mais il doit être aléatoire. Cela garantit que si l’on chiffre deux fois le même message, le chiffré sera différent.

Il est impossible de déduire l’encryption à partir d’un message connu car la clé change à chaque block. L’autre avantage est que l’on a pas besoin de décoder les blocs précédents pour décoder le bloc actuel, il suffit de savoir sa position.

1. Facilement parallélisable et “random access”.

# Exercice 4

1. Non
2. - ajouter un MAC ou une signature

- utiliser un mode qui garantit l’authenticité comme SCM