République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mouloud MAMMERI de TIZI-OUZOU

Faculté de Génie Electrique et d'Informatique

Département d'Informatique

MEMOIRE

De fin d'études

En vue de l'obtention du diplôme de master en Informatique option système informatique

Thème

Conception et réalisation d'une application de cryptage Basée sur les réseaux de Feistel

Dirigé par :
M[r] KHEMLICHE S

Réalisé par :
M[r] OUAREZKI Ghiles
M[r] DJENDER Ali

Promotion: 2011/2012

Remerciements

Nous tenons à présenter nos sincères remerciements à notre promoteur Mr Khemliche pour nous avoir encadrés et guidés tout au long de notre projet et pour tous les conseils judicieux qu'il nous a prodigué.

Aussi nous tenons à lui reconnaitre le temps précieux qu'il nous a consacré.

Que les membres du jury trouvent ici nos très vifs remerciements pour avoir accepté d'honorer par leur jugement notre travail, ainsi que pour le temps qu'ils ont consacré pour nous donner leurs avis, corrections.

Nous leurs exprimons ici nos sincères sentiments.

Enfin, merci à tous ceux qui de près ou de loin ont contribué à la réalisation de ce projet : nos enseignants, nos chères familles et nos amis(es).

Ghiles et Ali

Sommaire

Chapitre 2 : Analyse et conception

Chapitre 3 : Réalisation

Liste des figures

Introduction générale

De tout temps, les codes ont existé. Ils ont d'abord servi à retranscrire des idées, à écrire un langage. L'homme a perçu le besoin de cacher, de dissimuler des informations personnelles ou confidentielles, et cela bien avant l'ère informatique.

Mais avec ces nouveaux moyens de communication est arrivée la nécessité de protéger le contenu de certains messages des inévitables curieux.

Ainsi est apparue la cryptographie qui est la science ou l'art qui consiste à transformer un message clair en un message indéchiffrable pour tous, sauf pour les destinataires du message original (qui peut être l'émetteur lui-même). Il sert donc le plus souvent à prévenir une éventuelle interception par un tiers.

On utilise pour cela un algorithme et une ou plusieurs clés, secrètes ou publiques. L'émetteur crypte son message à l'aide d'un algorithme, le transmet crypté, et l'émetteur peut alors le décrypter à l'aide d'une même clé ou d'une différente :

Si le message est intercepté pendant sa transmission, il ne peut normalement qu'être décodé qu'avec la bonne clé, donc l'information transmise reste confidentielle, si toutefois l'algorithme et la/les clé(s) sont suffisamment complexes!

La cryptographie est utilisée depuis l'Antiquité, mais certaines de ses méthodes les plus importantes, comme la cryptographie asymétrique, datent de la fin du XX^{e} siècle.

Aujourd'hui, elle est utilisé partout où des données "sensibles" sont à protéger : dans les administrations, dans les sites militaires, dans les hôpitaux, dans les banques, sur les cartes bancaires et pour une grande partie des transactions sur Internet.

La cryptographie a évolué en trois périodes historiques :

- La cryptographie mécanique. Il s'agit de la cryptographie qui utilise des moyens mécaniques pour chiffrer un message. Cette cryptographie s'étend de l'antiquité jusqu'à la fin de la seconde guerre mondiale environ. De nos jours, elle n'a plus cours.

- La cryptographie mathématique. Il s'agit de la cryptographie qui utilise les mathématiques pour chiffrer un message. Cette cryptographie a commencé aux environs de la fin de la deuxième guerre mondiale et c'est celle que l'on utilise de nos jours.

- La cryptographie quantique. Il s'agit de la cryptographie dont les bases reposent sur la physique quantique. Nous sommes en train de la voir émerger de nos jours et nul doute qu'elle ne remplace dans les années qui viennent la cryptographie basée sur les mathématiques.

Objectif :

Notre travail ici consiste à concevoir un crypto-système pour le chiffrement/déchiffrement de données sous forme de fichiers (texte, image, vidéo, …etc.).

Pour mieux comprendre le problème, nous allons d'abord parler dans le 1er chapitre des différentes notions de cryptographie qui sont utilisées à l'époque et celles utilisées maintenant. Ensuite dans le 2eme chapitre on fera une brève définition des réseaux de Feistel puis on parlera de l'analyse et conception de notre algorithme de cryptage à travers plusieurs exemples. Enfin, dans le dernier et 3eme chapitre, on illustrera les différentes fonctionnalités de notre application avec des captures d'écran.

1. Introduction :

Le but de ce chapitre, est de présenter les fondements et le fonctionnement de la cryptographie. Il intéressera le lecteur qui connaît peu ou pas le domaine et qui souhaiterait comprendre le fonctionnement et les mécanismes mis en œuvre en cryptographie.

2. Terminologie : [Web01]

Dans cette présentation on s'attachera à utiliser toujours les mêmes termes. Ce paragraphe présente les termes utilisés ainsi que leur signification.

- Texte en clair : c'est le message à protéger.
- Texte chiffré : c'est le résultat du chiffrement du texte en clair.
- Chiffrement (Cryptage) : c'est la méthode ou l'algorithme utilisé pour transformer un texte en clair en texte chiffré.
- Déchiffrement (Décryptage) : c'est la méthode ou l'algorithme utilisé pour transformer un texte chiffré en texte en clair.
- Clé : c'est le secret partagé utilisé pour chiffrer le texte en clair en texte chiffré et pour déchiffrer le texte chiffré en texte en clair. On peut parfaitement concevoir un algorithme qui n'utilise pas de clé, dans ce cas c'est l'algorithme lui-même qui constitue la clé, et son principe ne doit donc en aucun cas être dévoilé.
- Cryptographie : cette branche regroupe l'ensemble des méthodes qui permettent de chiffrer et de déchiffrer un texte en clair afin de le rendre incompréhensible pour quiconque n'est pas en possession de la clé à utiliser pour le déchiffrer.
- Cryptanalyse : c'est l'art de révéler les textes en clair qui ont fait l'objet d'un chiffrement sans connaître la clé utilisée pour chiffrer le texte en clair.
- Cryptologie : il s'agit de la science qui étudie les communications secrètes. Elle est composée de deux domaines d'étude complémentaires : la cryptographie et la cryptanalyse.
- Coder, décoder : c'est une méthode ou un algorithme permettant de modifier la mise en forme d'un message sans introduire d'élément secret. Le Morse est donc un code puisqu'il transforme des lettres en trait et points sans notion de secret. L'ASCII est lui aussi un code puisqu'il permet de transformer une lettre en valeur binaire.

3. Cryptographie historique : [Web01]

Ce paragraphe présente plusieurs algorithmes ou méthodes de cryptographie à une époque où les mathématiques ne régnaient pas encore en maîtres sur ce domaine.

3.1. La technique grecque :

Une méthode de chiffrement datée entre le Xème et VIIème siècle avant Jésus Christ repose sur l'utilisation d'un bâton appelé scytale d'un diamètre fixé. Une lanière en cuir était enroulée en hélice autour de ce bâton et le texte en clair était alors écrit sur la lanière. Ensuite, la lanière était déroulée et pouvait être envoyée (sans le bâton) au destinataire du message.

Scytale grecque

Pour déchiffrer le texte chiffré, il suffisait d'utiliser un bâton possédant exactement le même diamètre que le précédent, d'y enrouler la lanière de cuir et le texte en clair pouvait alors être relu.

Le procédé utilisé par cette méthode est un chiffrement par transposition, c'est-à-dire que les lettres ne sont pas modifiées mais que seul l'ordre des lettres est changé.

3.2. Le code de César :

Le code de César est une méthode connue par tous les écoliers. Il suffit de décaler les lettres d'un certain nombre connu aussi bien par celui qui écrit le message que par celui qui le reçoit.

Par exemple si n=4, cela donne :

- a=E, b=F, c=G, ... w=A, x=B, y=C, z=D.

Si le texte en clair à chiffrer est "rendons a cesar ce qui est a cesar", avec un décalage de quatre lettres, le texte chiffré est "VIRHSRW E GIWEV GI UYM IWX E GIWEV".

Ce procédé est un procédé de chiffrement par substitution mono-alphabétique. Il est très simple mais il ne résiste pas à une analyse basée sur la fréquence des caractères puisque chaque lettre est toujours remplacée par la même lettre. D'ailleurs dans l'exemple fourni, le mot cesar est chiffré deux fois de la même manière en GIWEV de même que le a est chiffré deux fois en E.

Un autre algorithme, nommé ROT13, basé sur le même fonctionnement a existé au tout début de l'informatique. Le décalage était de treize lettres, et c'est donc le même algorithme qui était utilisé aussi bien pour le chiffrement que pour le déchiffrement (un premier décalage de treize lettres pour chiffrer suivi d'un autre décalage de treize lettres pour déchiffrer suffisait à redonner le texte en clair).

3.3. Le code de Vigenère :

Le code de Vigenère est présenté pour la première fois par le diplomate Blaise de Vigenère au courant du 16ème siècle.

Il reprend pour cela le principe du code de César mais il le complexifie en introduisant la notion de décalage variable en fonction d'une clé. Le code de Vigenère repose d'abord sur le carré de Vigenère. Il s'agit de l'alphabet qui est répété sur 26 lignes mais décalé d'une lettre à chaque fois.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

Il nécessite ensuite un mot clé seulement connu par l'émetteur et le destinataire du message. C'est le premier algorithme à introduire la notion de clé. Ce mot clé est répété autant de fois que nécessaire afin d'avoir autant de lettres (ou plus) que le texte en clair à chiffrer. Ensuite l'alphabet utilisé pour chiffrer une lettre est celui correspondant à la lettre du mot clé.

Si le texte à chiffrer est "rendons a vigenere ce qui est a vigenere", avec le mot clé "RAYMOND", le texte chiffré est "IELPCAV R VGSSAHIE AQ EHL VSR M JVJVNCDS".

```
rendons a vigenere ce qui est a vigenere
RAYMOND R AYMONDRA YM OND RAY M ONDRAYMO
```

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z A B C D E F G H I J K L M N O P Q R S T U V W X Y
```

Dans cet exemple on voit que les deux mots vigenere ne sont plus codés de la même manière (VGSSAHIE et JVJVNCDS) de même pour la lettre a qui est codée en R et M.

Ce code ne sera cassé qu'en 1854 (soit 200 ans plus tard) par le mathématicien Charles Babbage.

3.4. Conclusion :

Bien sûr, il y a eu de nombreux autres codes utilisant des variations sur ces codes ou alors de nouveaux concepts. Mais la plupart des procédés cryptographiques reposent essentiellement sur deux moyens :

- La **transposition**, l'ordre des lettres du message est modifié dans le message.
- La **substitution**, les lettres sont remplacées par d'autres lettres selon une certaine règle.

En ce qui concerne la substitution, il faut noter qu'il y en a plusieurs :

- Substitution mono-alphabétique : chaque lettre du message est remplacée par une autre lettre de l'alphabet. Le code de César utilise cette technique.
- Substitution poly-alphabétique : chaque lettre du message est remplacée par une autre lettre d'un alphabet indiqué par une lettre de la clé. Le code de Vigenère utilise cette technique.
- Substitution homophonique : chaque lettre du message est remplacée par une des lettres de substitution possibles. Le nombre de lettres de substitution possibles est fonction de la fréquence de la lettre dans le langage concerné. Cette méthode de substitution a pour but d'empêcher le déchiffrage du message par une analyse fréquentielle.
- Substitution polygrammes : chaque groupe de caractères est remplacé par un autre groupe de caractères.

4. La stéganographie : [Web01]

La stéganographie n'est pas vraiment une méthode cryptographique. Elle repose sur le fait que l'information à cacher est mélangée avec de l'information banale de manière à passer inaperçue.

4.1. Le texte caché :

Enfin	voici	le	printemps
Un	oiseau	de	passage
Chante	un	message	emprunt
D’	un	doux	secret

Enfin	**voici**	le	printemps
Un	oiseau	de	passage
Chante	un	**message**	emprunt
D’	un	doux	**secret**

Clé : BACD

Le texte caché est :

Voici un message secret

4.2. La stéganographie dans les fichiers images :

De la même manière que pour le texte, il est possible de cacher de l'information dans des fichiers images. En modifiant ou altérant quelques bits du fichier, il est ainsi possible de cacher un copyright ou alors un message de son choix sans que cela se voit. En effet, cette altération sera peu ou pas visible car l'œil humain n'est pas capable de discerner des petites aberrations sur une grande image à condition que le ratio taille du message / taille de l'image ne soit pas trop grand.

4.3. Conclusion :

En conclusion, il faut retenir que la stéganographie est une méthode de cryptographie faible : elle repose uniquement sur le fait que personne ne remarquera le canal caché. Dès lors que ce canal est connu, il n'y a plus aucune protection.

5. La cryptographie moderne : [Web01]

La cryptographie moderne repose maintenant uniquement sur les mathématiques. De plus, les règles de bases sont :

- L'algorithme utilisé n'est pas secret. Il peut être diffusé librement, cela ne doit avoir aucun impact sur la facilité ou non à déchiffrer le message.
- La clé de chiffrage utilisée est secrète.

Un protocole cryptographique basé sur la non-divulgation de l'algorithme mathématique utilisé n'est pas fiable. Tôt ou tard, l'algorithme utilisé sera connu et le protocole deviendra faible. Au contraire, diffuser l'algorithme mathématique utilisé permet à la communauté de valider et de tester la robustesse de cet algorithme.

5.1. Buts de la cryptographie :

Globalement, la cryptographie permet de résoudre quatre problèmes différents :

- La confidentialité. Le texte chiffré ne doit être lisible que par les destinataires légitimes. Il ne doit pas pouvoir être lu par un intrus.
- L'authentification. Le destinataire d'un message doit pouvoir s'assurer de son origine. Un intrus ne doit pas être capable de se faire passer pour quelqu'un d'autre.
- L'intégrité. Le destinataire d'un message doit pouvoir vérifier que celui-ci n'a pas été modifié en chemin. Un intrus ne doit pas être capable de faire passer un faux message pour légitime.
- La non répudiation. Un expéditeur ne doit pas pouvoir, par la suite, nier à tort avoir envoyé un message.

5.2. Quelques nombres pour fixer les idées :

Nombre de secondes dans un jour	86 400 secondes
Nombre de secondes dans une année	31 536 000 secondes
Nombre de secondes écoulées depuis la création de l'univers (13,7 milliard d'années)	432 043 200 000 000 000 secondes ou $432 * 10^{15}$ secondes

Le but de ces quelques nombres est de montrer l'inutilité des attaques dites force brute. En effet, un cryptanalyste en herbe pourrait se dire "Ce message est chiffré avec l'algorithme AES-256, il y a 2^{256} clés possibles (soient environ 10^{76} clés), testons les toutes pour déchiffrer le message". A raison de 100 milliards de tentatives par seconde (ce qui est énorme), il faudrait 10^{65} secondes pour tester toutes les clés. Ce temps de calcul nécessaire est beaucoup plus long que l'âge de l'univers.

5.3. Les différents modes de chiffrement :

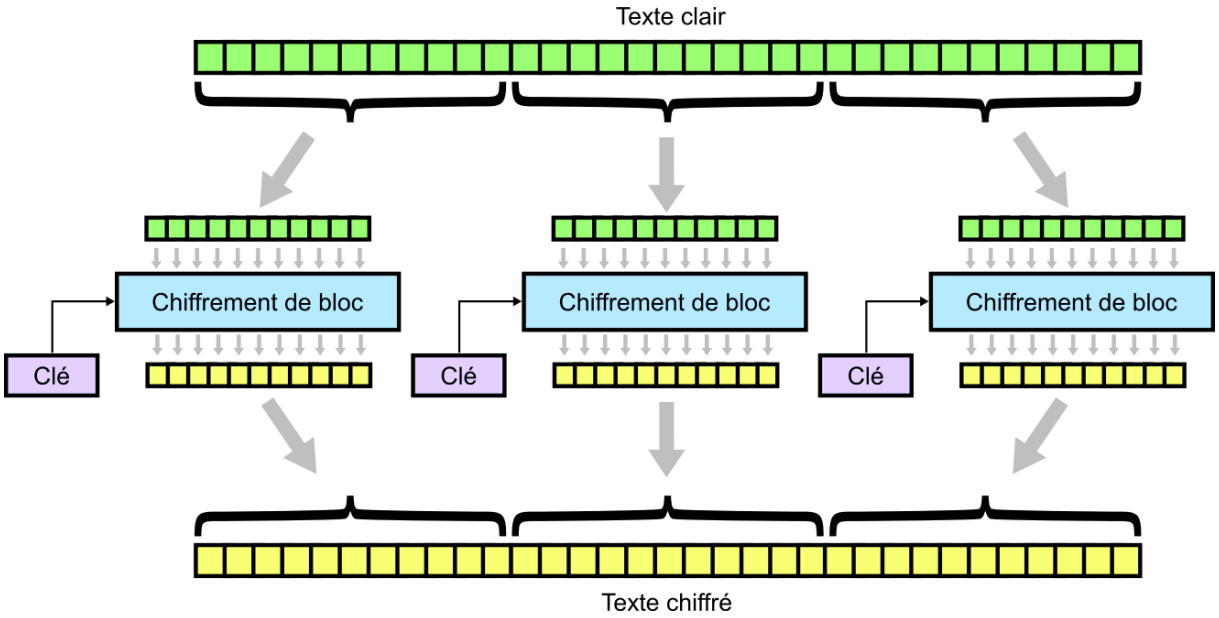
Le mode de chiffrement correspond à la manière dont on va utiliser un algorithme de chiffrement donné. Ce mode de chiffrement consiste par exemple à rajouter de la contre-réaction entre l'entrée et la sortie de l'algorithme afin de lui rajouter des caractéristiques bien précises. Les différentes modes de chiffrement utilisés sont les suivants :

- Le mode ECB pour **Electronic Code Book**
- Le mode CBC pour **Cipher Block Chaining**
- Le mode **chiffrement en continu**
- Le mode CTAK pour **Cipher Text Auto Key**
- Le mode CFB pour **Cipher Feed Back**
- Le mode KAK pour **Key Auto Key**
- Le mode OFB pour **Output Feed Back**
- Le mode CTR pour **CounTeR**
- Le mode BC pour **Block Chaining**
- Le mode PCBC pour **Propagating Cipher Block Chaining**
- Le mode CBCC pour **Ciher Block Chaining with Checksum**
- Le mode OFBNLF pour **Output Feed Back mode with a Non Linear Function**
- Le mode PBC pour **Plaintext Block Chaining**
- Le mode PFB pour **Plaintext Feed Back**
- Le mode CBCPD pour **Cipher Block Chaining of Plaintext Difference**
- Le mode CTS pour **Cipher Text Stealing**

Dans les paragraphes suivants nous allons regarder un peu plus en détail les modes ECB, CBC, CFB et OFB. Les autres modes sont nommés à des fins d'exhaustivité mais sont rarement utilisés.

5.3.1. Le mode ECB : [Web02]

Ce mode est le plus simple : un même bloc est toujours codé de la même manière. Il n'y a pas de rétroaction de l'entrée ou de la sortie sur la fonction de chiffrement.



Le mode ECB

Les avantages de ce mode sont les suivants :

- Le travail de chiffrement ou de déchiffrement peut être parallélisé. Plusieurs machines ou CPU peuvent travailler simultanément sur des parties différentes du message.
- Une erreur de transmission d'un bit affecte uniquement le décodage du bloc courant.

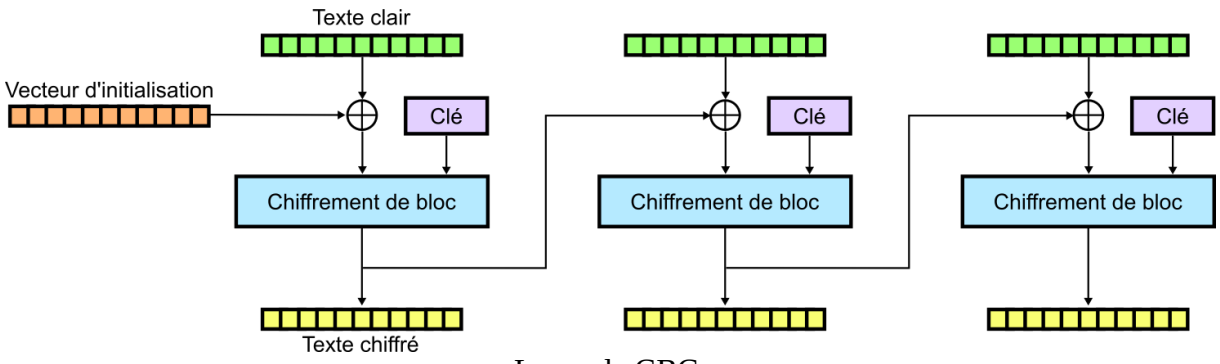
Par contre, ce mode a les désavantages suivants :

- Les répétitions du texte en clair ne sont pas masquées et se retrouvent sous la forme de répétitions de textes chiffrés.
- Des portions complètes du message peuvent être modifiées, répétées ou remplacées sans difficulté.

5.3.2. Le mode CBC : [Web02]

Dans ce mode de chiffrement, chaque bloc de texte en clair est d'abord combiné par un ou exclusif avec le dernier bloc du texte chiffré. La sortie de ce ou exclusif est ensuite appliquée à la fonction de chiffrement.

Ce mode de chiffrement dispose en plus d'un vecteur d'initialisation appelée IV (pour Initialisation Vector) qui permet d'initialiser le processus quand aucun bloc n'a encore été chiffré.



Le mode CBC

Les avantages de ce mode sont les suivants :

- Les répétitions de texte en clair sont masquées dans le texte chiffré.
- La valeur du vecteur d'initialisation IV n'a pas besoin d'être secrète.

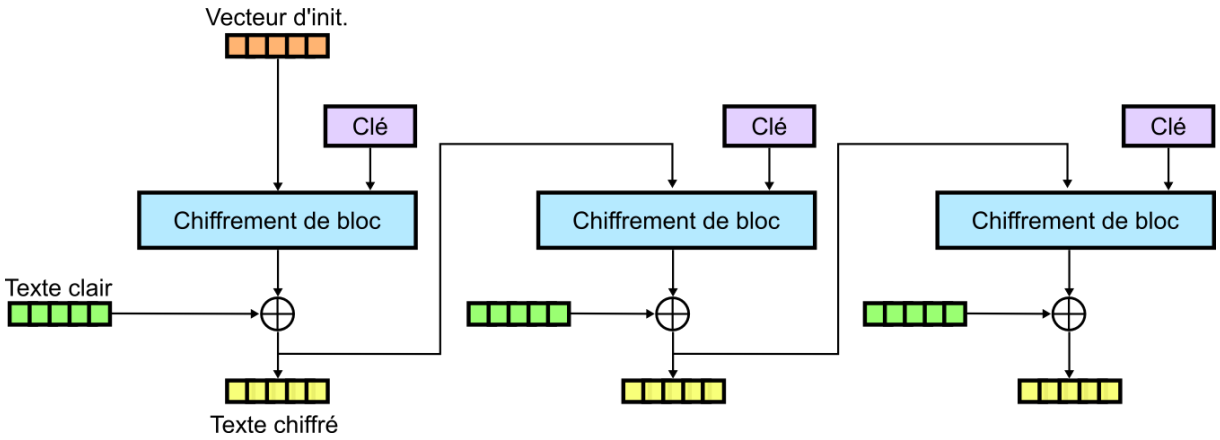
Par contre, ce mode a les désavantages suivants :

- Deux textes en clair commençant pareil auront le même début de texte chiffré.
- Une erreur de transmission d'un bit affecte uniquement le décodage du bloc courant ainsi que le décodage du même bit dans le bloc suivant.

5.3.3. Le mode CFB : [Web02]

Les modes ECB et CBC travaillent sur des blocs de texte en clair (64 bits par exemple). Ces modes ne sont pas utilisables que lorsqu'un bloc est complet. Sur des applications réseau, cela peut poser des problèmes car les valeurs à chiffrer arrivent sous forme d'octets et doivent être transmises immédiatement.

Ce mode dispose d'un vecteur d'initialisation qui a la même fonctionnalité que celui du CBC. L'octet du texte chiffré est combiné par un ou exclusif avec l'octet de texte en clair. Le résultat de cette opération est alors transmis en même temps qu'il est injecté dans la fonction de chiffrement.



Le mode CFB

Les avantages de ce mode sont les suivants :

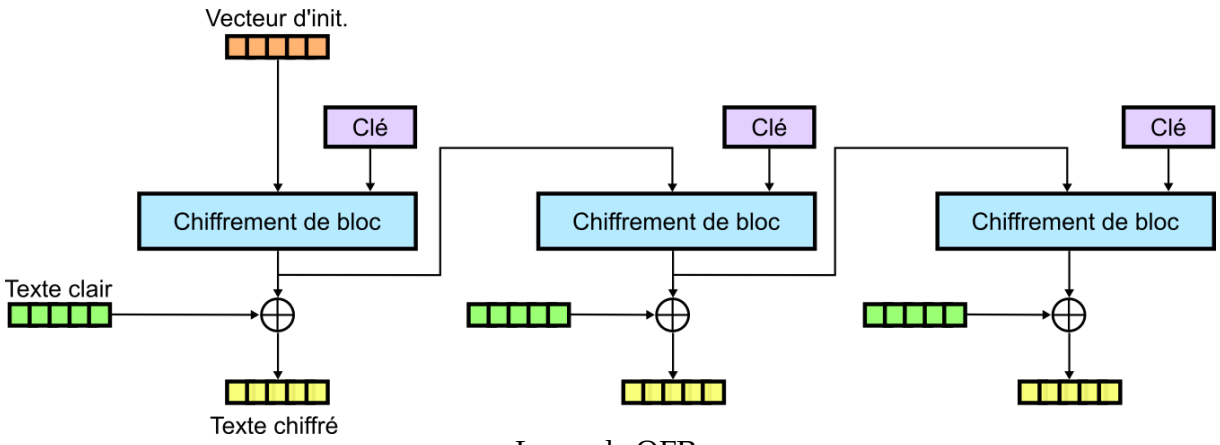
- Il est possible de chiffrer un flot de valeurs plus petites que la taille standard du bloc géré par l'algorithme.
- Les répétitions de texte en clair sont masquées dans le texte chiffré.
- La valeur du vecteur d'initialisation IV n'a pas besoin d'être secrète.

Par contre, ce mode a le désavantage suivant :

- Une erreur de transmission d'un bit affecte uniquement le décodage du bloc courant ainsi que le décodage du même bit dans le bloc suivant.

5.3.4. Le mode OFB : [Web02]

Le mode OFB ressemble au mode CFB. La seule différence est que l'octet injecté dans la fonction de chiffrement vient du chiffrement successif du vecteur d'initialisation.



Le mode OFB

Les avantages de ce mode sont les suivants :

- Les répétitions de texte en clair sont masquées dans le texte chiffré.
- La valeur du vecteur d'initialisation IV n'a pas besoin d'être secrète.
- Ce mode n'amplifie pas les erreurs. Une erreur de transmission d'un bit affecte uniquement ce bit lors du décodage.

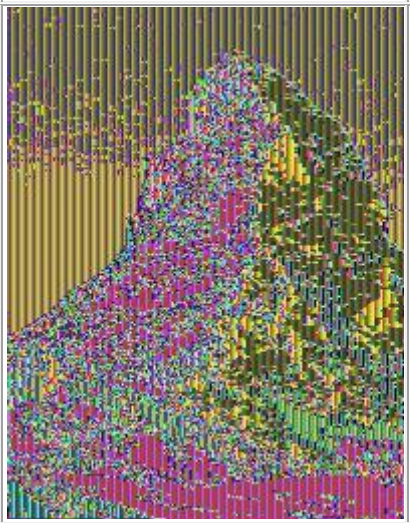
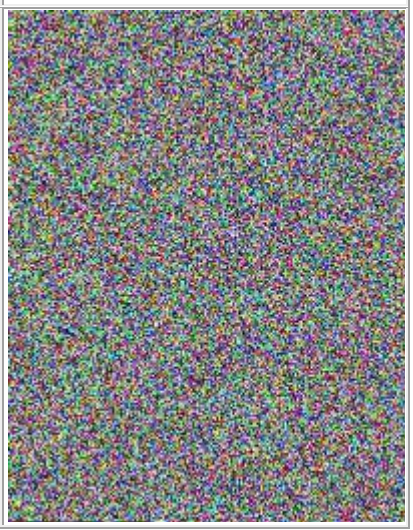
Par contre, ce mode a le désavantage suivant :

- Ce mode est très fragile vis-à-vis d'une attaque au clair. (l'attaquant possède des textes clairs ainsi que leurs versions chiffrées et est libre de les utiliser pour révéler d'autres informations secrètes comme la clé de chiffrement).

5.4. Quel mode choisir ?:

Les quatre modes présentés ainsi que tous les autres ont chacun leurs avantages et leurs faiblesses. Il conviendra de choisir le mode en fonction de ce que l'on veut faire.

Pour finir, voici trois photos du mont Cervin dans les Alpes (source Wikipédia). La première photo est le texte en clair, la seconde est le texte chiffré de la photo utilisant un algorithme en mode ECB et enfin la dernière photo utilise le même algorithme en mode CBC.

Non chiffré	Chiffré en mode ECB	Chiffré en mode CBC
		

Ces photos montrent clairement que le mode ECB n'est pas adapté au chiffrement de photographies. Ceci ne veut toutefois pas dire que le mode ECB ne doit pas être utilisé, il a d'autres domaines d'utilisation.

Après le parcours des différents modes de chiffrement utilisables, nous allons maintenant nous intéresser aux différents algorithmes de chiffrement. Il existe en tout quatre grandes familles d'algorithmes utilisées par la cryptographie. Ces familles sont :

- Les algorithmes de calcul d'empreinte
- Les algorithmes symétriques
- Les algorithmes asymétriques
- Les méthodes de génération de nombres aléatoires.

On s'intéressera ici aux algorithmes symétriques et asymétriques, pour les autres on fera une brève définition.

6. Les algorithmes de calcul d'empreinte : [Web01]

Un algorithme de calcul d'empreinte est un algorithme qui transforme un texte en clair en un nombre (une empreinte numérique) qui soit représentatif de ce texte en clair. Le but de ces algorithmes est qu'une altération aussi minime soit elle du texte en clair puisse être détectée par le fait que la nouvelle empreinte est différente et même très différente de la précédente.

En général, ce type d'algorithme fournit comme résultat un nombre dont la taille est fixe et ce quelle que soit la taille du texte en clair en entrée.

7. Les méthodes de génération de nombres aléatoires : [Web01]

Les algorithmes cryptographiques demandent souvent des nombres aléatoires. Ces nombres aléatoires peuvent être utilisés pour générer une clé aléatoire ou encore pour générer une clé publique et une clé privée pour certains algorithmes asymétriques.

Mais d'abord, quelles sont les caractéristiques d'une suite de nombres aléatoire ? Les mathématiciens aujourd'hui ne savent pas bien définir la notion d'aléatoire. Toutefois, on retient les caractéristiques suivantes :

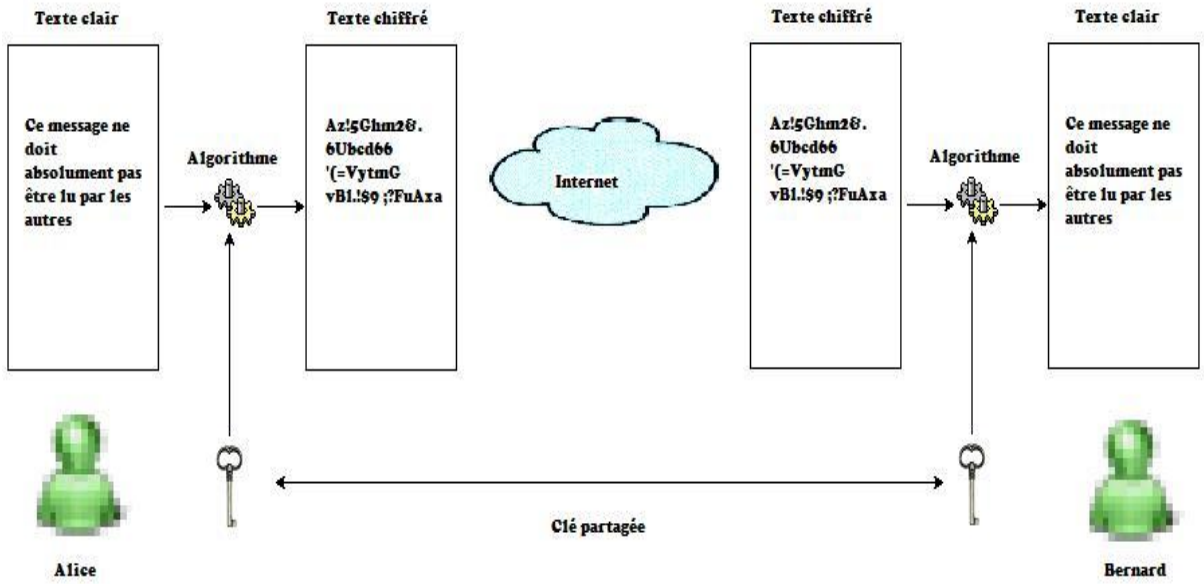
- La répartition des nombres de la suite est équilibrée, c'est-à-dire que statistiquement, il y a autant de 0 que de 1.
- La suite de nombres ne possède pas de séquences plus ou moins longues de nombres qui se répètent.

8. Les algorithmes symétriques : [Web01]

Un algorithme symétrique est un algorithme qui permet de transformer un texte en clair en texte chiffré en utilisant une clé et de retransformer le texte chiffré en texte en clair en utilisant la même clé.

Le secret de la communication est uniquement assuré par la clé qui est utilisée lors de la phase de chiffrement et de déchiffrement. L'algorithme utilisé ne fait pas partie du secret.

On parle d'algorithmes symétriques car c'est la même clé qui sert à la fois au chiffrement et au déchiffrement du message.



Principe de l'algorithme symétrique

Dans ce paragraphe nous allons plus particulièrement nous intéresser aux algorithmes DES, AES et IDEA.

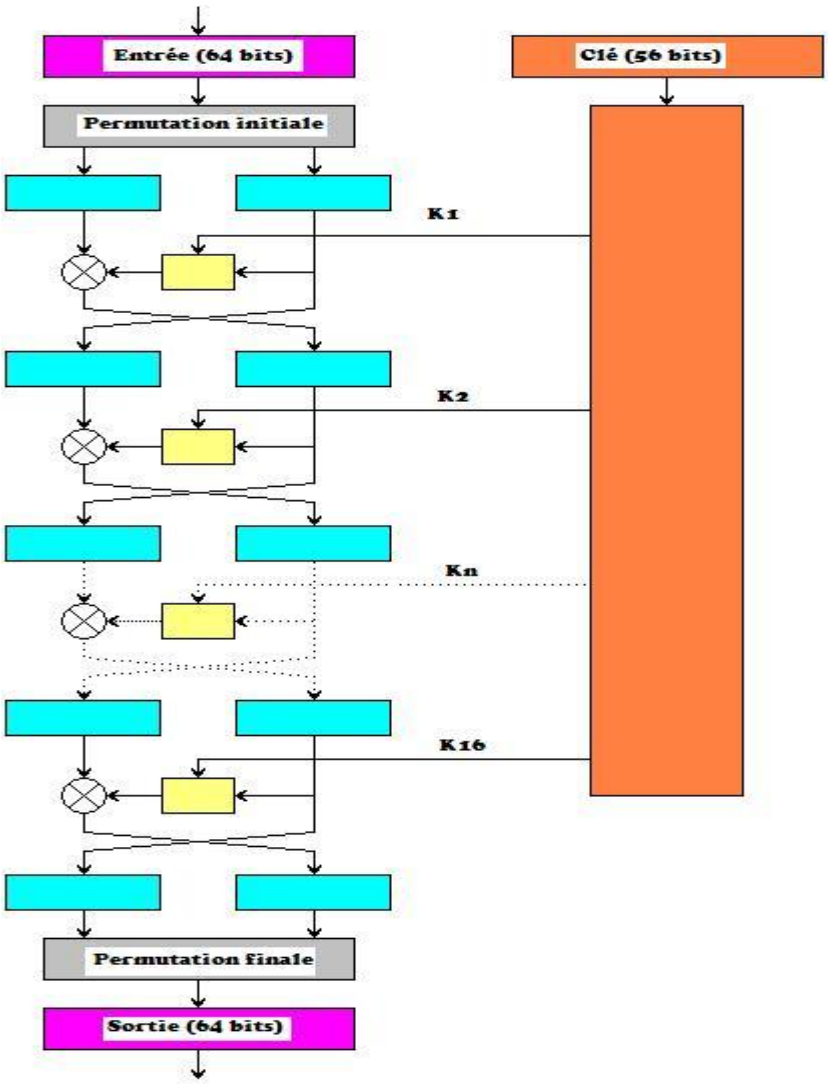
8.1. L'algorithme DES :

L'algorithme DES (Data Encryption Standard) est né dans les années 1970 sous l'impulsion du National Bureau of Standards américain. Celui-ci voulait promouvoir un système de chiffrement qui permettrait à différents interlocuteurs de communiquer en toute sécurité tout en utilisant un algorithme fiable et commun.

L'algorithme Lucifer (précurseur direct de DES) est l'algorithme, conçu par IBM, qui fut retenu. La NSA (National Security Agency) imposa toutefois quelques modifications dans l'algorithme. Ces modifications imposées furent les causes de beaucoup de rumeurs.

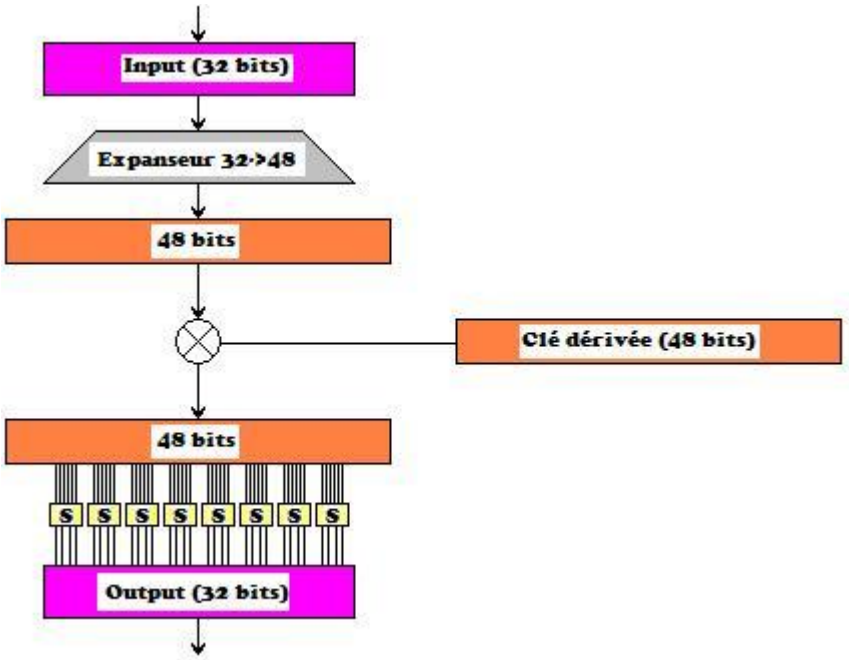
L'algorithme DES fut officiellement adopté le 23 novembre 1976.

L'algorithme DES est un algorithme symétrique de chiffrement par bloc de 64 bits (soient 8 octets) fonctionnant avec une clé de 56 bits. Il fonctionne sur 16 rondes, avant la première ronde le bloc de 64 bits est découpé en 2 blocs de 32 bits et lors de chacune des rondes ces derniers subissent un chiffrement.



Vue général du DES

Le bloc de droite subit des transformations à l'aide d'une des clés de 48 bits dérivées de la clé principale de 56 bits.



Opération sur le bloc de 32 bits de droite

Les 32 bits de droite sont d'abord étendus à 48 bits à l'aide d'une opération particulière. Ensuite, une opération de ou exclusif est effectuée entre ces 48 bits et les 48 bits d'une des clés dérivées de la clé principale. Et enfin, les 48 bits du résultat sont ensuite condensés sur 32 bits à l'aide des boîtes S1 à S8. Ces huit boîtes ont été imposées par la NSA et ont été la cause de toutes les rumeurs concernant cet algorithme. Chacune de ces boîtes transforme 6 bits en 4 bits selon un schéma encore une fois très particulier.

Comme on peut le voir, cet algorithme n'est pas très facile à transformer en programme compte tenu de toutes les permutations et compressions de bits. Par contre, c'est un algorithme très facile à implémenter en électronique. Il ne faut pas oublier qu'en 1970 on faisait plus de choses plus vite en électronique qu'en informatique.

Beaucoup d'applications encore de nos jours utilisent le DES. C'est par exemple l'algorithme qui a été utilisé pendant très longtemps (et encore maintenant pour certains) sur les ordinateurs de type Unix pour chiffrer les mots de passe des utilisateurs.

Cet algorithme ne devrait maintenant plus être utilisé du fait de son petit nombre de clés (2^{56}) qui paraissait énorme en 1970 mais qui est maintenant petit comparé à d'autres algorithmes et à la puissance de calcul disponible (le calcul distribué permet maintenant de trouver une clé en 24 heures). De plus, de nombreuses attaques cryptographiques ont permit de découvrir des petites faiblesses dans le DES.

8.2. L'algorithme 3-DES :

L'algorithme 3-DES (*Triple DES*) utilise l'algorithme DES avec 2 ou 3 clé différentes. L'algorithme va d'abord chiffrer avec une clé, déchiffrer avec la deuxième clé est enfin chiffrer encore avec la troisième clé.

$$C = E_{DES}^{k3}\left(D_{DES}^{k2}\left(E_{DES}^{k1}(M)\right)\right)$$

C : texte chiffré
M : texte clair
$K_{i\ (i = 1, 2, 3)}$: clés de chiffrement
E : fonction de chiffrement DES
D : fonction de déchiffrement DES

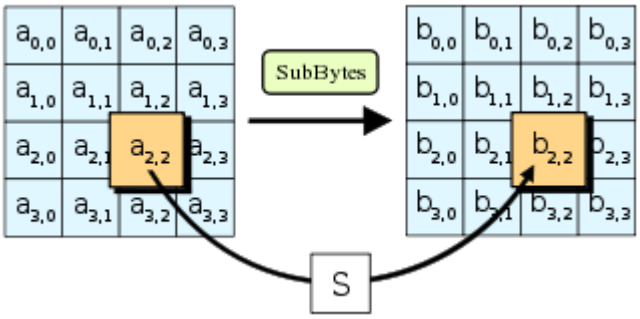
8.3. L'algorithme AES : [Web04]

Advanced Encryption Standard ou AES (soit « standard de chiffrement avancé » en français), aussi connu sous le nom de Rijndael, est un algorithme de chiffrement symétrique. Il remporta en octobre 2000 le concours AES, lancé en 1997 par le NIST (National Institute of Standards and Technology) et devint le nouveau standard de chiffrement pour les organisations du gouvernement des États-Unis. Il a été également approuvé par la NSA (National Security Agency) pour les informations top secrètes.

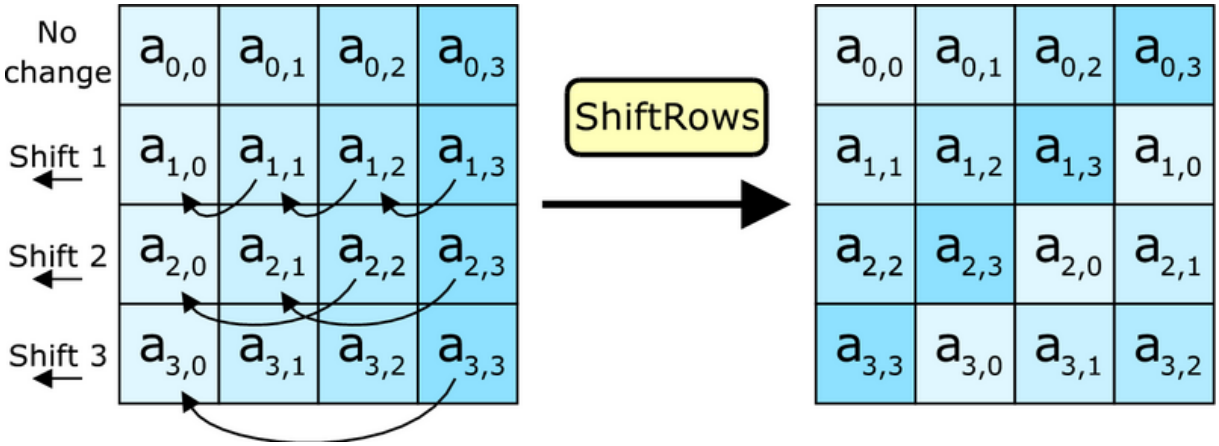
L'AES opère sur des blocs de 128 bits (plaintext P) qu'il transforme en blocs cryptés de 128 bits (C) par une séquence de Nombre d'opérations ou "rounds", à partir d'une clé de 128, 192 ou 256 bits. Suivant la taille de celle-ci, le nombre de rounds diffère : respectivement 10, 12 et 14 rounds.

Le schéma suivant décrit le déroulement du chiffrement :

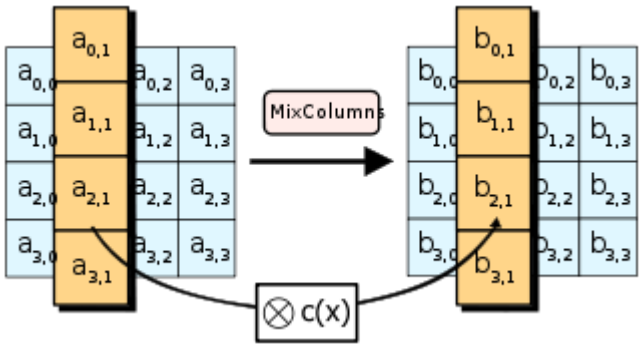
- BYTE_SUB (Byte Substitution) est une fonction opérant indépendamment sur chaque bloc à partir d'une table dite de substitution.



- SHIFT_ROW est une fonction opérant des décalages (typiquement elle prend l'entrée en 4 morceaux de 4 octets et opère des décalages vers la gauche de 0, 1, 2 et 3 octets pour les morceaux 1, 2, 3 et 4 respectivement).

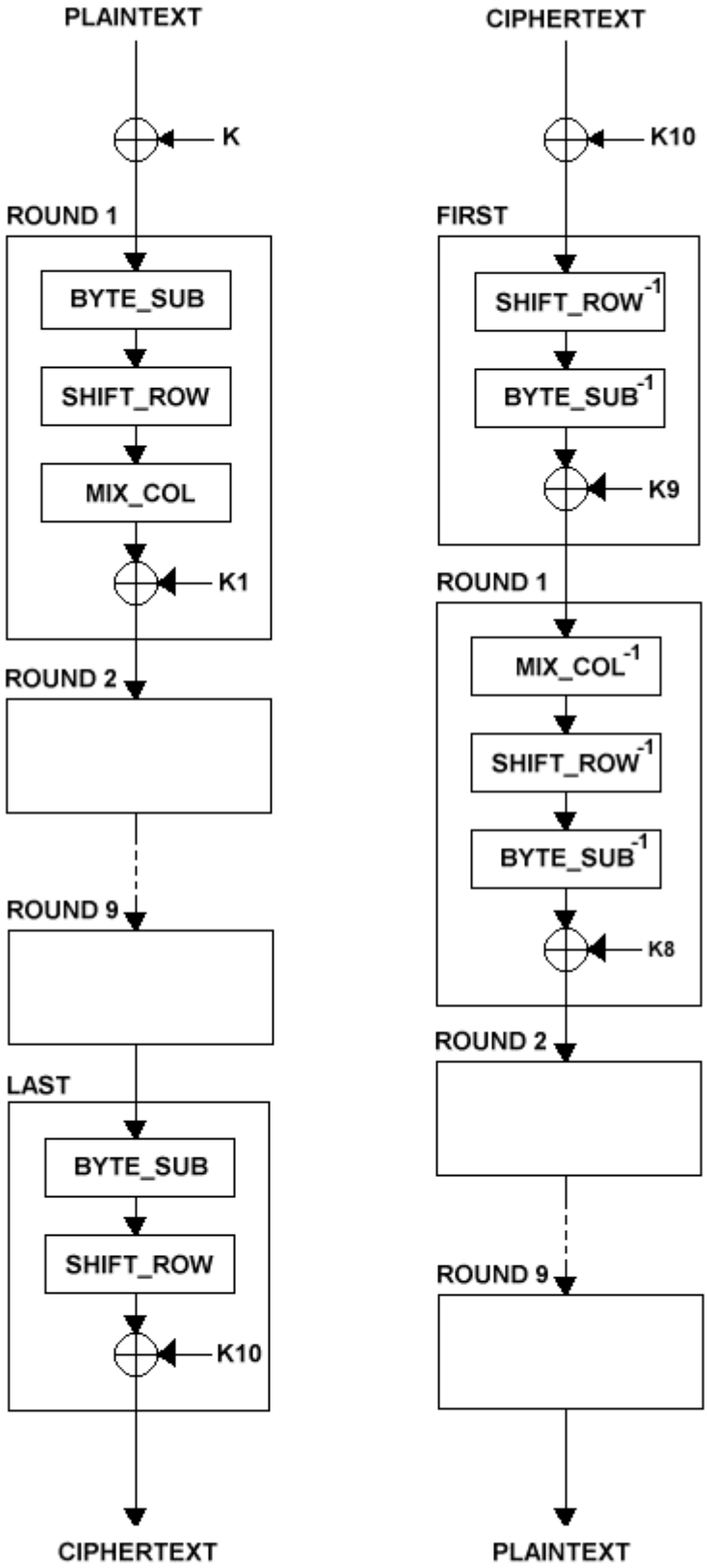


- MIX_COL est une transformation linéaire qui est appliquée sur la matrice, elle consiste en la multiplication binaire de chaque élément de la matrice avec des polynômes issus d'une matrice auxiliaire c(x).



- le + entouré d'un cercle désigne l'opération de OU exclusif (XOR).
- Ki est la ième sous-clé calculée par un algorithme à partir de la clé principale K.

Le déchiffrement consiste à appliquer les opérations inverses, dans l'ordre inverse et avec des sous-clés également dans l'ordre inverse.



Vue général de l'AES

8.4. L'algorithme IDEA : [Web03]

IDEA, pour *International Data Encryption Algorithm*, est un algorithme de chiffrement symétrique conçu par Xuejia Lai et James Massey, présenté pour la première fois en 1991

L'algorithme *IDEA* a été breveté par la société suisse Mediacrypt ; le brevet a expiré en 2011 en Europe, et en janvier 2012 aux États-Unis et au Japon. Mediacrypt met en avant depuis mai 2005 son nouveau chiffrement nommé « *IDEA NX*T » qui est en fait *FOX*.

Le bloc de données de 64 bits est divisé en 4 sous-blocs de 16 bits : X_1, X_2, X_3 et X_4. Ces quatre sous-blocs deviennent les entrées de la première ronde de l'algorithme. Il y a huit rondes au total. À chaque ronde, les 4 sous-blocs sont combinés par OU exclusif, additionnés, multipliés entre eux et avec 6 sous-blocs de 16 bits dérivés de la clé (128 bits). Entre chaque ronde, le deuxième et le troisième sous-bloc sont échangés. Enfin, les quatre sous-blocs sont combinés avec les quatre sous-clés dans une transformation finale.

À chaque ronde, la séquence d'évènements est la suivante :

1. multipliez X_1 et la première sous-clé
2. additionnez X_2 et la deuxième sous-clé
3. additionnez X_3 et la troisième sous-clé
4. multipliez X_4 et la quatrième sous-clé
5. combinez par OU exclusif les résultats des étapes (1) et (3)
6. combinez par OU exclusif les résultats des étapes (2) et (4)
7. multipliez le résultat de l'étape (5) avec la cinquième sous-clé
8. additionnez les résultats des étapes (6) et (7)
9. multipliez le résultat de l'étape (8) par la sixième sous-clé
10. additionnez les résultats des étapes (7) et (9)
11. combinez par OU exclusif les résultats des étapes (1) et (9)
12. combinez par OU exclusif les résultats des étapes (3) et (9)
13. combinez par OU exclusif les résultats des étapes (2) et (10)
14. combinez par OU exclusif les résultats des étapes (4) et (10)

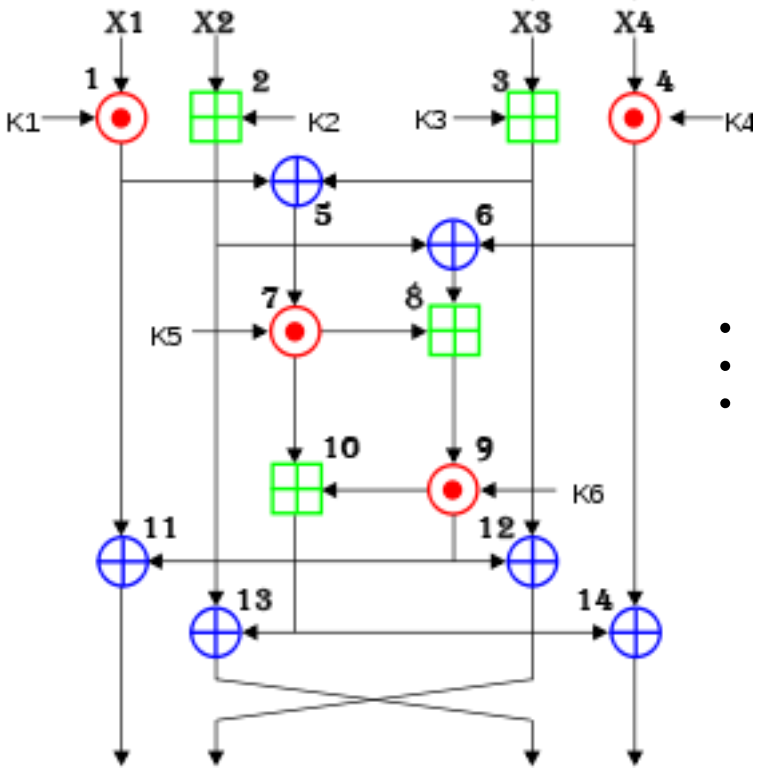
La sortie de la ronde est constituée des 4 sous-blocs produits par les étapes (11), (13), (12) et (14). Changez les deux blocs intérieurs (sauf lors de la dernière ronde) et cela donne l'entrée de la ronde suivante.

Après la huitième ronde, il y a une transformation finale :

1. multipliez X_1 et la première sous-clé ;
2. additionnez X_2 et la deuxième sous-clé ;
3. additionnez X_3 et la troisième sous-clé ;
4. multipliez X_4 et la quatrième sous-clé.

Enfin les 4 sous-blocs sont réassemblés pour former le texte chiffré.

Chaque ronde utilise 6 sous-clés de 16 bits et la transformation finale utilise 4 sous-clés, un total de 52 sous-clés est utilisé, les premières 8 sous-clés sont extraites directement à partir de la clé, des groupes de 8 clés sont extraits par rotation à gauche de 25 bits

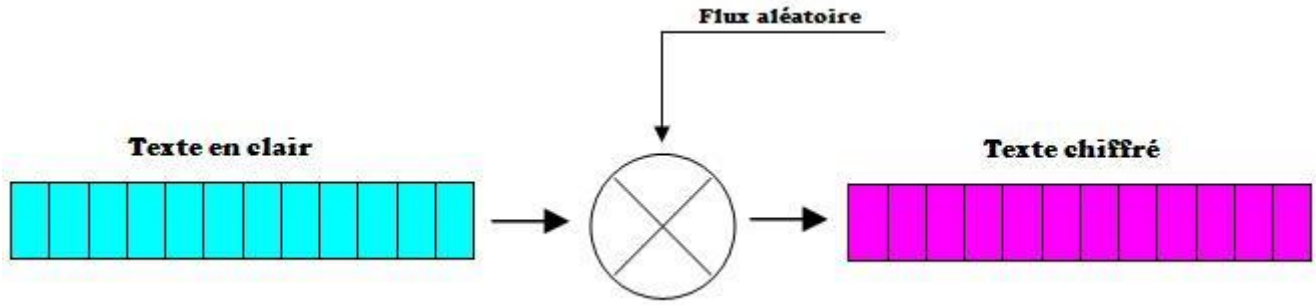


- OU exclusif (représenté par un ⊕)
- addition (représenté par un ⊞)
- multiplication (représenté par un ⊙)

Vue d'une ronde de l'IDEA

8.5. L'algorithme du masque jetable :

L'algorithme du masque jetable consiste à utiliser un flux de nombres aléatoires qui va être combiné par une opération ou exclusif sur le texte en clair pour générer le texte chiffré.



Algorithme du masque jetable

Les contraintes imposées par ce système sont :

- Le flux aléatoire doit être vraiment aléatoire.
- Le destinataire du message doit recevoir par un moyen fiable une copie du flux aléatoire utilisé pour chiffrer le texte en clair (CD-ROM convoyé par porteur spécial, etc.).
- Un flux aléatoire déjà utilisé ne doit jamais être réutilisé.

8.6. D'autres algorithmes :

Il existe encore d'autres algorithmes symétriques parmi lesquels on peut citer :

- Blowfish
- Fox
- Frog
- Misty1
- MMB
- 3-Way
- RC2, RC4, RC5
- REDOC
- SHACAL

9. Les algorithmes asymétriques : [Web01]

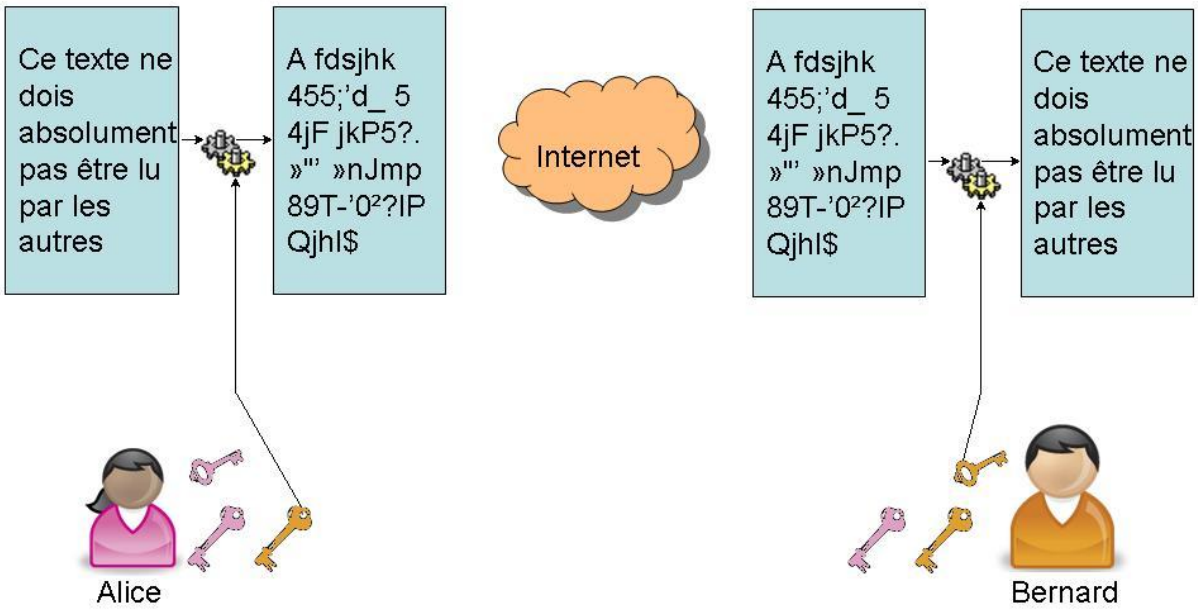
Les algorithmes symétriques vus précédemment sont tous fiables mais ils posent un problème, c'est celui de l'échange de la clé : comment transmettre de manière fiable à mon interlocuteur la clé de chiffrement utilisée pour chiffrer le message que je lui envoie ? Il y a bien sûr le téléphone, mais il y a aussi les écoutes téléphoniques.

Les algorithmes asymétriques ont été inventés pour pallier précisément le problème de transmission sécurisée de la clé.

On parle d'algorithmes asymétriques car ce n'est pas la même clé qui sert au chiffrement et au déchiffrement. Dans le cas de ces algorithmes, on parlera alors de clé privée et de clé publique. Ces deux clés, sont intimement liées par une fonction mathématique complexe.

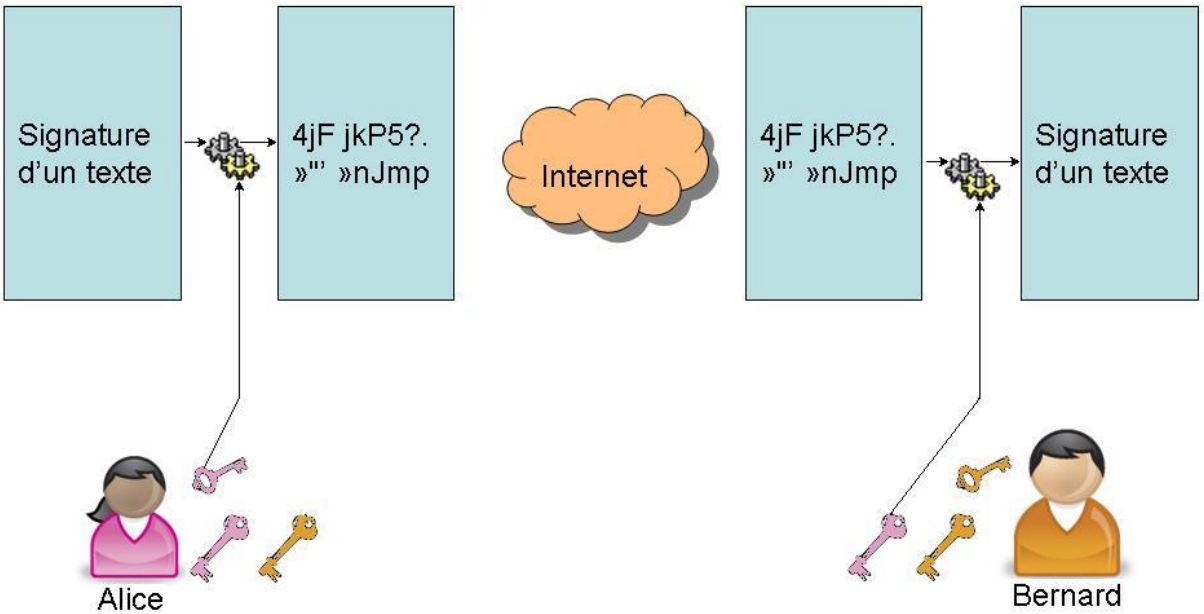
Les algorithmes asymétriques possèdent 2 modes de fonctionnement ;

- Le mode chiffrement dans lequel l'émetteur chiffre un fichier avec la clé publique du destinataire. Le destinataire utilise sa clé privée pour déchiffrer le fichier. Dans ce mode, l'émetteur est sûr que seul le destinataire peut déchiffrer le fichier.



Chiffrement avec l'algorithme asymétrique

- Le mode signature dans lequel l'émetteur signe un fichier avec sa propre clé privée. Le destinataire utilise la clé publique de l'émetteur pour vérifier la signature du fichier. Dans ce mode, le destinataire est sûr que c'est bien l'émetteur qui a envoyé le fichier.



Signature avec l'algorithme asymétrique

Donc pour résumer :

- L'émetteur chiffre avec la clé publique du destinataire, le destinataire déchiffre avec sa clé privée.
- L'émetteur signe avec sa clé privée, le destinataire vérifie la signature avec la clé publique de l'émetteur.

9.1. L'algorithme DH :

Whitfield Diffie et Martin Hellman ont travaillé ensemble en 1974 sur ce problème. Comment deux personnes sur Internet peuvent elle communiquer de manière sécurisée et donc pour cela s'échanger de manière fiable et sécurisée une clé de chiffrement permettant l'usage d'un des protocoles symétriques.

Ils ont tourné leurs recherches vers les fonctions non réversibles, c'est-à-dire les fonctions de la forme y = f(x) où quand on connaît x, il est très facile de calculer y mais quand on connaît y, il est impossible de recalculer x.

Par exemple, la fonction y = x * x est une fonction réversible. Connaissant x, il est facile de calculer y et connaissant y, il est facile de calculer x. La fonction réversible de y = x * x est x = racine(y).

Les fonctions qu'ils ont utilisées sont les fonctions de l'arithmétique modulo car elles ne sont pas réversibles.

Par exemple y = x (modulo 7) n'est pas une fonction réversible. En effet, pour x = 11, on obtient y = 4 mais pour y = 4, on obtient x = 11 ou x = 18 ou encore x = 25 et une infinité d'autres solutions encore.

Finalement, en 1976, ils découvrirent une fonction du type $Y = Z^X$ (modulo P) qui permet cet échange sécurisé.

Un exemple d'un échange sécurisé de clé est schématisé dans le tableau suivant. Alice veut communiquer avec Bernard et Estelle peut écouter les échanges.

Alice	Estelle	Bernard
Nous utiliserons la fonction $y = 11^x$ modulo 13	Ha ! Ils vont utiliser la fonction $y = 11^x$ modulo 13	OK, utilisons la fonction $y = 11^x$ modulo 13
Alice choisit un nombre secret A = 5	Estelle ne sait pas les nombres secrets A et B choisis par Alice et Bernard.	Bernard choisit un nombre secret B = 8
Alice calcule le nombre $A' = 11^A$ modulo 13. A' = 7 et envoie ce nombre A' = 7 à Bernard de manière quelconque	Ha ha ! Le nombre A' d'Alice est 7 et le nombre B' de Bernard est 9	Bernard calcule son nombre $B' = 11^B$ modulo 13. B' = 9 et envoie ce nombre B' = 9 de manière quelconque à Alice

Alice prend le nombre B' = 9 de Bernard et calcule le résultat de : B'^{A} modulo 13 et obtient : 3	Estelle ne sait pas calculer ces nombres car elle ne dispose ni du nombre secret initial d'Alice A, ni du nombre secret initial de Bernard B.	Bernard prend le nombre A' = 7 d'Alice et calcule le résultat de : A'^{B} modulo 13 et obtient : 3

En final, Alice et Bernard obtiennent un même nombre secret 3 qu'ils peuvent utiliser comme clé pour un algorithme symétrique choisi. Le calcul de cette clé s'est effectué en s'échangeant 2 valeurs presque quelconques. Estelle, qui pourtant a la possibilité d'écouter les échanges, ne sait rien de cette clé et n'a pas la possibilité de la calculer.

Bien sûr, dans la pratique, les nombres choisis seront plus grands mais le principe reste le même.

Avec cet algorithme, Whitfield Diffie et Martin Hellman ont prouvé que l'échange des clés de manière fiable et sécurisée était possible sans avoir besoin de se rencontrer.

Cet algorithme, bien que simple souffre d'un défaut. Il faut que Alice et Bernard se synchronisent afin de pouvoir s'échanger un minimum d'information : les paramètres de la fonction utilisée ainsi que les valeurs A' et B'. Cela signifie qu'Alice doit attendre que Bernard soit réveillé pour lui envoyer un message.

L'échange sécurisé des clés étant réglé, il restait le problème de synchronisation obligatoire à supprimer.

9.2. L'algorithme RSA :

Suites aux travaux réalisés par Whitfield Diffie et Martin Hellman, Ronald Rivest, Adi Shamir et Leonard Adleman ont eux aussi travaillé tous les trois sur les algorithmes asymétriques à partir de 1975.

L'algorithme qu'ils ont mis au point repose aussi sur une fonction a sens unique ou plutôt une fonction très difficilement réversible. Cette fonction difficilement réversible est la factorisation d'un nombre en produit de facteurs premiers. En effet, autant, il est très facile de choisir deux nombres (48 et 52 par exemple) et de calculer leur produit (2496), autant il est long et pénible d'extraire les facteurs premiers du nombre 2397 qui sont 47 et 51.

Le principe de l'algorithme RSA est donc basé sur la difficulté de factoriser un nombre surtout quand ce nombre est très grand et qu'il est le produit de deux nombres premiers très grands eux aussi.

L'algorithme imaginé est le suivant :

- Alice choisit au hasard 2 nombre premiers p et q (13 et 17 par exemple).
- Elle construit le produit de ces deux nombres N (221 dans notre exemple).
- Elle choisit un nombre e. Ce nombre ne doit pas être un diviseur de (p-1) * (q-1) (5 par exemple).

- La clé publique d'Alice est composée de la paire de nombre e et N (5 et 221 dans notre exemple). Cette clé publique peut être publiée dans un annuaire.
- Alice calcule ensuite le nombre d donné par la formule e*d ≡ 1 [(p-1)*(q-1)] (77 dans notre exemple). Ce nombre représente la clé privée d'Alice. Pour calculer plus facilement d on utilise la formule d = ((p-1)*(q-1)*k+1)/e, trouver un K (K=1, 2, 3…) pour que le résultat de la fraction soit un entier.
- Bernard veut envoyer le message bonjour à Alice. Pour cela, il convertit le mot en valeurs (nous prendrons les valeurs issues du code ASCII dans notre exemple). La conversion donne les valeurs 98, 111, 110, 106, 111, 117, 114.
- La formule utilisée pour chiffrer les valeurs est la suivante : $c_i = m_i^e (\mathrm{mod}\, N)$ dans lequel ci représente la valeur chiffrée de mi, e et N représentants les paramètres publics de la clé d'Alice.
- Avec le texte en clair 98, 111, 110, 106, 111, 117, 114, et en appliquant la formule pour chacune des valeurs on obtient le texte chiffré suivant : 115, 76, 145, 123, 76, 104, 173.
- Pour déchiffrer le message, Alice utilise la formule suivante : $m_i = c_i^d (\mathrm{mod} N)$ dans lequel mi représente la valeur en clair de ci, d et N représentants les paramètres publics et privés de la clé d'Alice.
- Avec le texte chiffré 115, 76, 145, 123, 76, 104, 173, et en appliquant la formule pour chacune des valeurs on obtient à nouveau le texte en clair d'origine : 98, 111, 110, 106, 111, 117, 114.

Cet exemple montre que les paramètres publics e et N suffisent à chiffrer un message et que les paramètres privés et publics d et N suffisent pour déchiffrer le message. Bien sûr, dans la pratique, les nombres choisis seront plus grands mais le principe reste le même.

Alice peut donc diffuser librement sa clé publique (les paramètres e et N). Toute personne qui veut communiquer avec Alice peut récupérer la clé publique d'Alice et lui envoyer un message chiffré. Alice est la seule à pouvoir déchiffrer ce message puisqu'elle est la seule à détenir le paramètre d de sa clé privée.

De plus Alice et Bernard n'ont plus besoin de se synchroniser pour commencer l'échange puisque la clé publique est disponible dans un annuaire public et que n'importe qui peut aller la chercher à tout moment pour l'utiliser.

La force de l'algorithme repose sur la difficulté à factoriser N. Si la factorisation était facile à réaliser, Estelle pourrait recalculer les nombres p et q et, connaissant e, elle pourrait alors calculer d et donc elle pourrait déchiffrer le message destiné à Alice.

Pour résumer :

- L'émetteur chiffre un message avec la clé publique du destinataire de façon à ce que seul le destinataire puisse déchiffrer le message avec sa clé privée.
- L'émetteur signe un message avec sa clé privée de façon à ce que le destinataire puisse vérifier la signature du message avec la clé publique de l'émetteur.

9.3. D'autres algorithmes :

Il existe encore d'autres algorithmes asymétriques parmi lesquels, on peut citer :

- Cryptographie sur les courbes elliptiques
- Cryptographie sur les courbes hyper elliptiques
- Crypto système de Blum-Goldwasser
- Crypto système de ElGamal
- Crypto système de Goldwasser-Micali
- Algorithme de Guillou-Quisquater
- Crypto système de McEliece
- Crypto système de Paillier
- Crypto système de Rabin
- Algorithme LLL

10. Conclusion :

Obtenir un niveau de sécurité informatique suffisant pour prévenir les risques technologique et informationnels est primordial pour un bon fonctionnement des organismes. Il est important de pouvoir les identifier correctement pour circonscrire les périmètres à mettre en place et protéger efficacement les valeurs qui doivent l'être. Ceci implique une approche pluridisciplinaire et systémique de la sécurité globale de cet organisme.

1. Introduction :

Apres avoir vu, dans le chapitre précédent les différents concepts nécessaires à l'accomplissement de notre travail, nous passons maintenant à la partie conception.

Dans ce chapitre nous nous intéresserons à la modélisation de notre algorithme qui se fera en deux étapes :

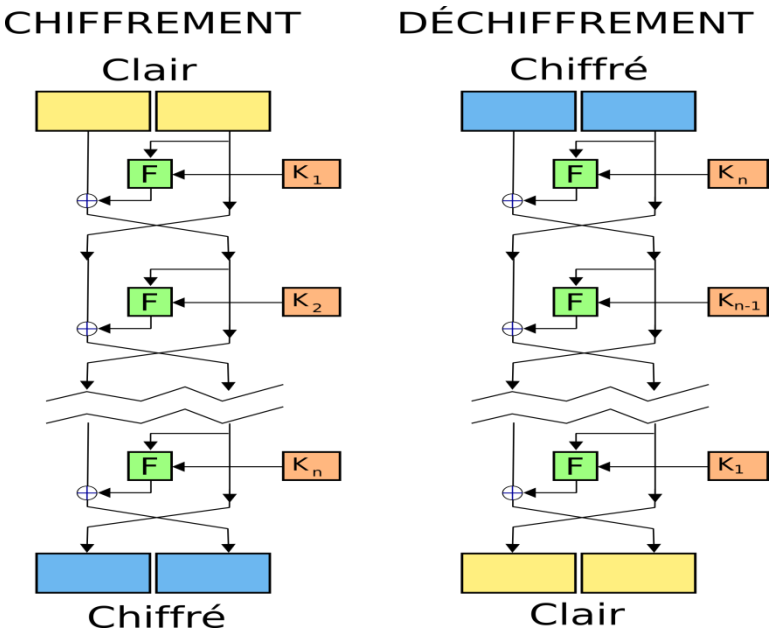
- Définition du concept des réseaux de Feistel.
- Modélisation de l'algorithme avec un réseau de Feistel en mode ECB et CBC.

2. Réseau de Feistel : [Web05]

Un réseau de Feistel est une construction utilisée dans les algorithmes de chiffrement par bloc, nommée d'après le cryptologue d'IBM, Horst Feistel. Elle a été utilisée pour la première fois dans Lucifer et DES. Cette structure offre plusieurs avantages, le chiffrement et le déchiffrement ont une architecture similaire voire identique dans certains cas. Un réseau de Feistel repose sur des principes simples dont des permutations, des substitutions, des échanges de blocs de données et une fonction prenant en entrée une clé intermédiaire à chaque étage.

2.1. Structure :

Un réseau de Feistel est subdivisé en plusieurs tours ou étages. Le réseau traite les données en deux parties de taille identique. À chaque tour, les deux blocs sont échangés puis un des blocs est combiné avec une version transformée de l'autre bloc. Pour simplifier, la moitié des données sont encodées avec la clef, puis le résultat de cette opération est ajouté grâce à un xor (ou exclusif) à l'autre moitié des données. Puis au tour suivant, on inverse : c'est au tour de la dernière moitié d'être chiffrée puis d'être ajoutée avec un xor à la première moitié, Le schéma ci-contre montre le cheminement des données (le "plus" entouré représente le xor). Chaque tour utilise une clé intermédiaire, en général tirée de la clé principale via. Les opérations effectuées pendant l'encryptage avec ces clefs intermédiaires sont spécifiques à chaque algorithme.



Réseau de Feistel à n tours utilisant l'opérateur XOR

2.2. Exemple :

Soit à crypter les deux blocs suivants :

B1=0100

B2=1001

Avec les deux clefs suivantes :

K1=0001

K2=0010

La fonction de chiffrement est l'addition d'un bloc avec une clef K_i. F=bloc+K_i (i=1, 2)

En utilisant un réseau de Feistel de 2 tours on aura le schéma de chiffrement/déchiffrement suivant :

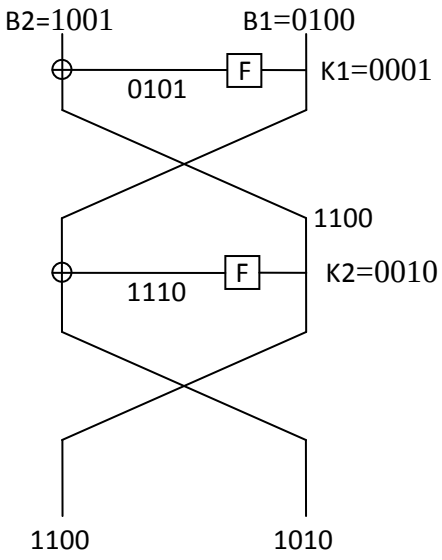


Schéma de chiffrement

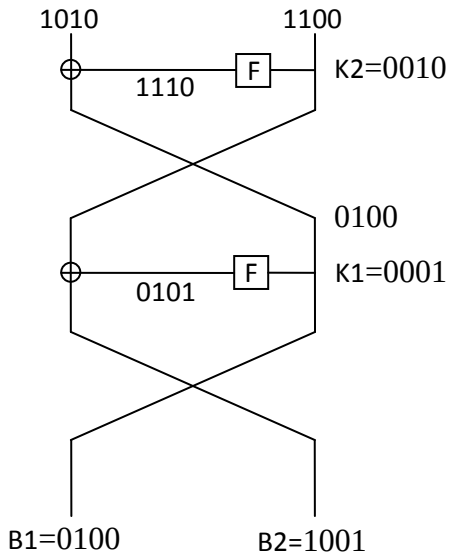


Schéma de déchiffrement

Remarque :

En cas de débordement le système prend les bits de poids faible correspondant à la taille du bloc.

2.3. Composition des tours :

Chaque tour applique plusieurs transformations sur les données provenant du tour précédent :

- permutation des bits
- substitution des bits
- mixage en utilisant la fonction XOR
- application de la clé du tour (intégrée dans une fonction ou via un XOR)

2.4. Algorithmes :

Un grand nombre d'algorithmes utilise des réseaux de Feistel, avec des variantes. Voici une liste de ces différents algorithmes :

- DES
- Blowfish
- Twofish
- Camellia
- SEED
- RC5
- OAEP

3. Présentation du projet :

Notre projet consiste à la création d'une application de chiffrement/déchiffrement de données basée sur un algorithme de cryptage par bloc à clé symétrique.

Cet algorithme repose sur un réseau de Feistel de 128 tours (64 cycles) prenant en entrée un bloc de données clair de 128 bits et une clé de 512 bits.

A l'entrée du réseau, le bloc de données est divisé en deux sous bloc de 64 bits chacun. Dans chaque cycle (1 cycle = 2 tours) les 2 sous blocs subissent les opérations suivantes :

1ere tour du cycle :

Le 2eme sous bloc subit les modifications suivantes :

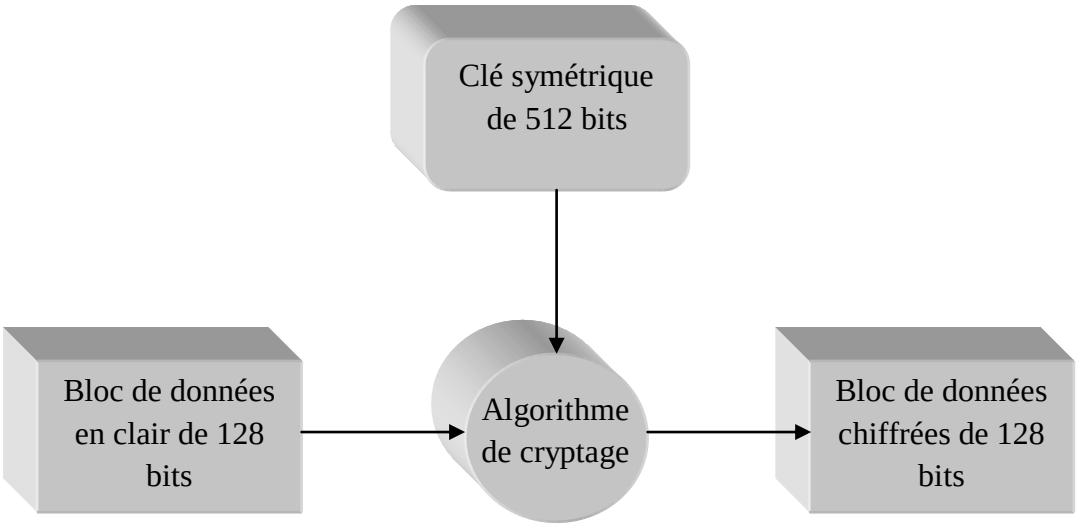
- Des décalages de bits (à droite et à gauche)
- Des additions avec les 4 premières sous clés de 64 bits chacune (dérivées de la clé principale) et un nombre spécifique de 64 bits (dans notre cas le nombre d'or = 9E3779B97F4A7C15 qui est la partie fractionnaire en hexa)
- Le tout sera combiné avec un XOR (ou exclusif) et le résultat de cette opération sera additionné au 1er sous bloc

2eme tour du cycle :

- Les 2 sous blocs seront permutés, puis les opérations précédentes seront appliquées à une différence prés, les additions se feront avec les 4 dernières sous clés.

A la fin de chaque cycle le nombre d'or sera incrémenté par lui-même et les 8 sous clés subirons des rotations de bits.

Dans le décryptage, les 8 sous clés subirons des rotations jusqu'à atteindre la 64eme rotation qui est la dernière atteinte dans le processus de cryptage afin d'entrer dans le décryptage. Le nombre d'or sera initialisé à 8DDE6E5FD29F0540 qui est la 64eme incrémentation de ce nombre par lui-même et qui sera décrémenté dans chaque cycle de décryptage.



Vue globale du système de cryptage

4. Présentation de l'algorithme :

4.1. Algorithme de cryptage :

Fonction crypter (tableau bloc_a_crypter, tableau cle)

Debut

b1 = bloc_a_crypter[0]

b2 = bloc_a_crypter[1]

nb_or = 0x9E3779B97F4A7C15

som = 0

Pour i=0 jusqu'a i=64

Debut

som = som + nb_or

Décalage de b2 de 2 bits à gauche + cle[0] (1)

Décalage de b2 de 5 bits à droite + cle[1] (2)

b2 + som (3)

Décalage de b2 de 7 bits à gauche + cle[2] (4)

Décalage de b2 de 3 bits à droite + cle[3] (5)

b1 = b1 + [(1) XOR (2) XOR (3) XOR (4) XOR (5)]

Décalage de b1 de 2 bits à gauche + cle[4] (6)

Décalage de b1 de 5 bits à droite + cle[5] (7)

b1 + som (8)

Décalage de b1 de 7 bits à gauche + cle[6] (9)

Décalage de b1 de 3 bits à droite + cle[7] (10)

b2 = b2 + [(6) XOR (7) XOR (8) XOR (9) XOR (10)]

Rotation à gauche de 3 bits pour la cle[0]

Rotation à gauche de 5 bits pour la cle[1]

Rotation à gauche de 7 bits pour la cle[2]

Rotation à gauche de 11 bits pour la cle[3]

Rotation à gauche de 3 bits pour la cle[4]

Rotation à gauche de 5 bits pour la cle[5]

Rotation à gauche de 7 bits pour la cle[6]

Rotation à gauche de 11 bits pour la cle[7]

Fin boucle

Retourne (b1, b2)

Fin fonction

4.2. Algorithme de décryptage :

Fonction decrypter (tableau bloc_a_decrypter, tableau cle)

Debut

b1 = bloc_a_decrypter[0]

b2 = bloc_a_decrypter[1]

nb_or = 0x9E3779B97F4A7C15

som = 0x8DDE6E5FD29F0540

Pour i=0 jusqu'a i=64

Debut

Rotation à droite de 3 bits pour la cle[0]

Rotation à droite de 5 bits pour la cle[1]

Rotation à droite de 7 bits pour la cle[2]

Rotation à droite de 11 bits pour la cle[3]

Rotation à droite de 3 bits pour la cle[4]

Rotation à droite de 5 bits pour la cle[5]

Rotation à droite de 7 bits pour la cle[6]

Rotation à droite de 11 bits pour la cle[7]

Décalage de b1 de 2 bits à gauche + cle[4] (1)

Décalage de b1 de 5 bits à droite + cle[5] (2)

b1 + som (3)

Décalage de b1 de 7 bits à gauche + cle[6] (4)

Décalage de b1 de 3 bits à droite + cle[7] (5)

b2 = b2 - [(1) XOR (2) XOR (3) XOR (4) XOR (5)]

Décalage de b2 de 2 bits à gauche + cle[0] (6)

Décalage de b2 de 5 bits à droite + cle[1] (7)

b2 + som (8)

Décalage de b2 de 7 bits à gauche + cle[2] (9)

Décalage de b2 de 3 bits à droite + cle[3] (10)

b1 = b1 - [(6) XOR (7) XOR (8) XOR (9) XOR (10)]

som = som - nb_or

Fin boucle

Retourne (b1, b2)

Fin fonction

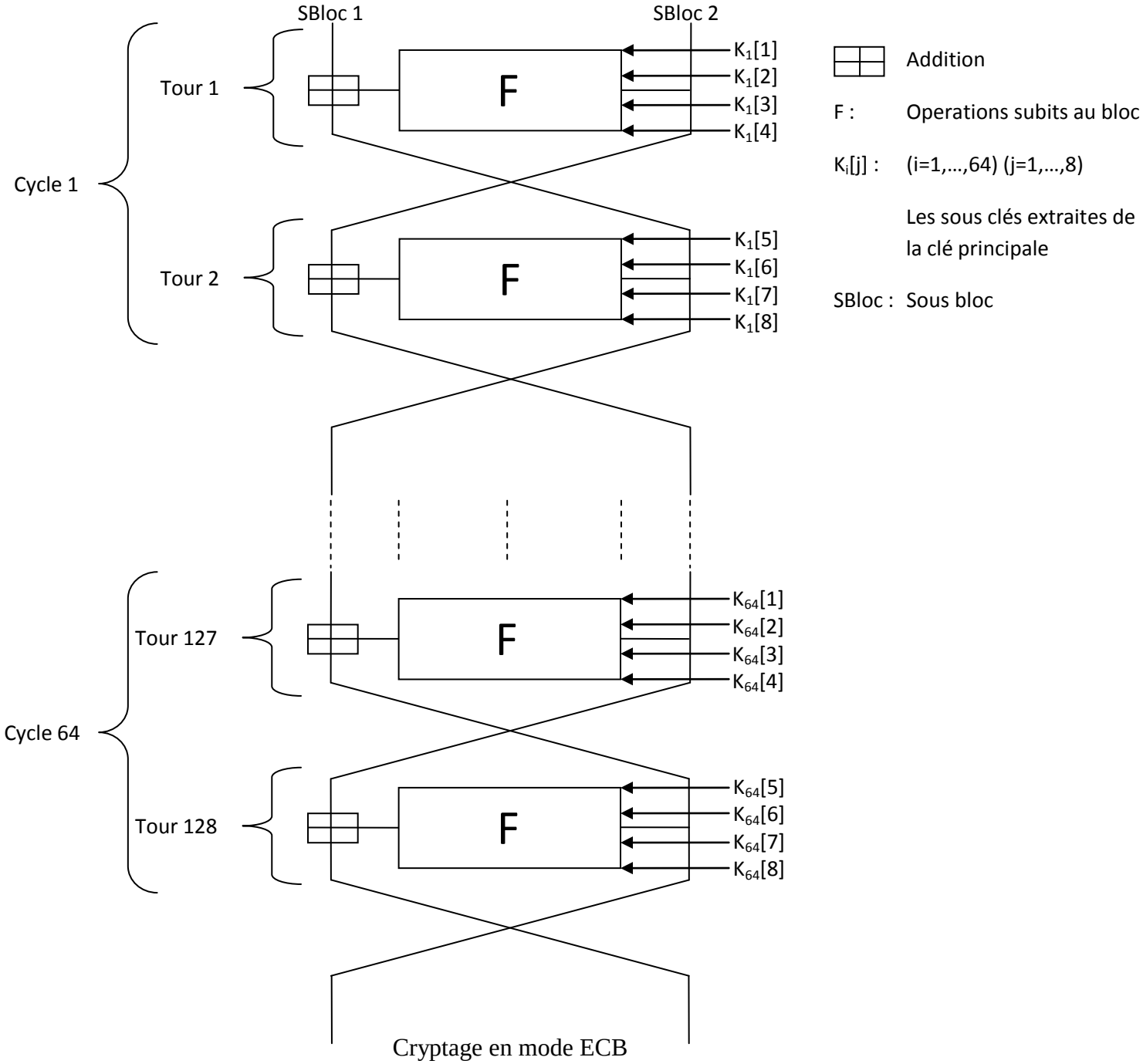
5. Fonctionnement de l'algorithme :

Dans ce qui suit nous allons détailler les 2 modes de fonctionnement de notre algorithme.

5.1. Fonctionnement en mode ECB :

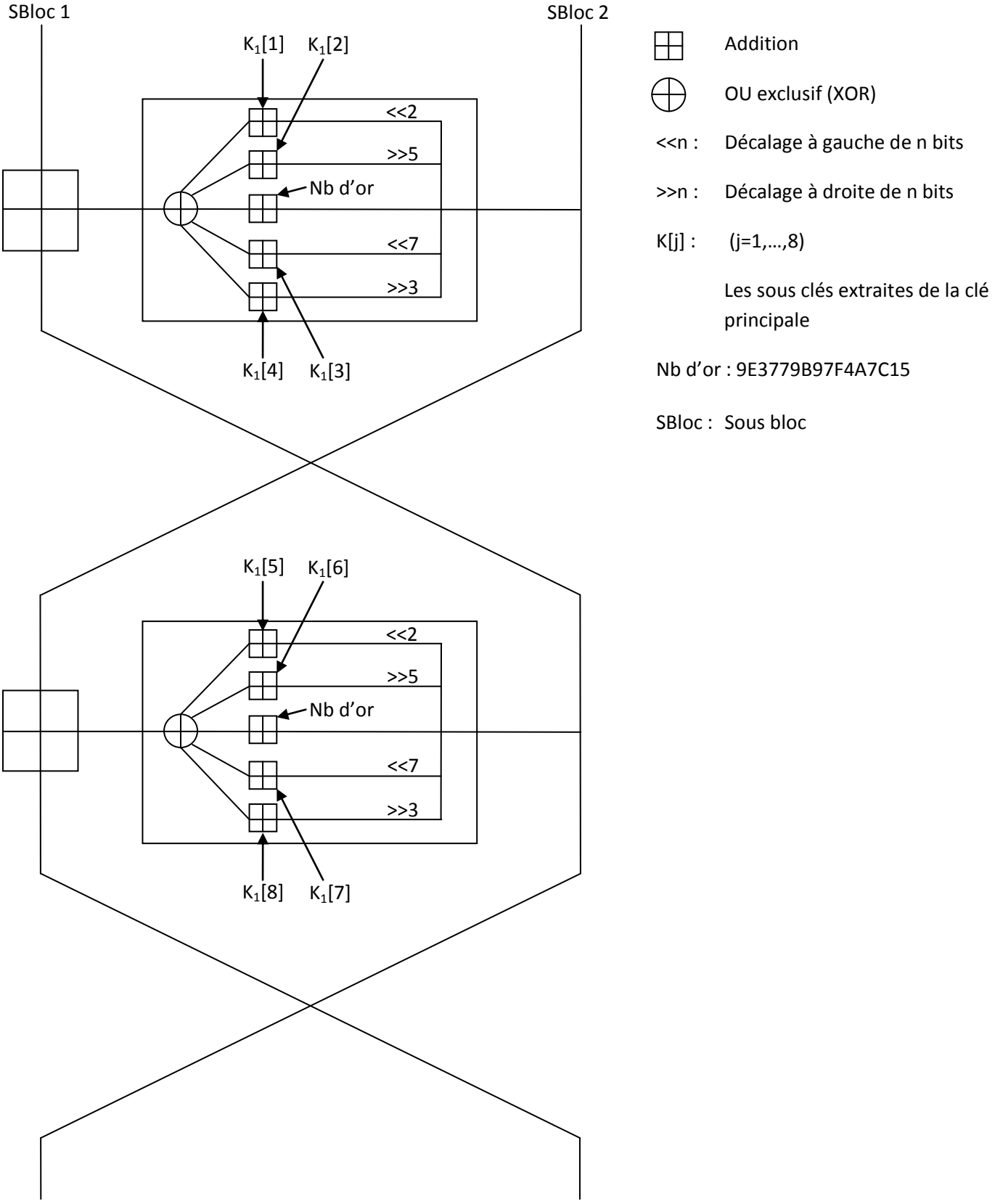
5.1.1. Cryptage :

Le schéma ci-dessous représente le réseau de Feistel de notre algorithme avec chiffrement en mode ECB (Electronic Code Book) :



Cryptage en mode ECB

Pour mieux expliquer le déroulement de l'algorithme de chiffrement on prendra seulement un cycle, plus précisément le premier cycle et la fonction (F) sera détaillée comme suit :



Vue détaillée d'un cycle de chiffrement en mode ECB

Exemple :

Cet exemple va illustrer le fonctionnement du schéma précédent en prenant en entrée un bloc de 16 bits et une clé de 64 bits au lieu d'un bloc de 128 bits et d'une clé de 512 bits comme vu précédemment. Ceci afin de faciliter l'exemple mais le principe reste le même.

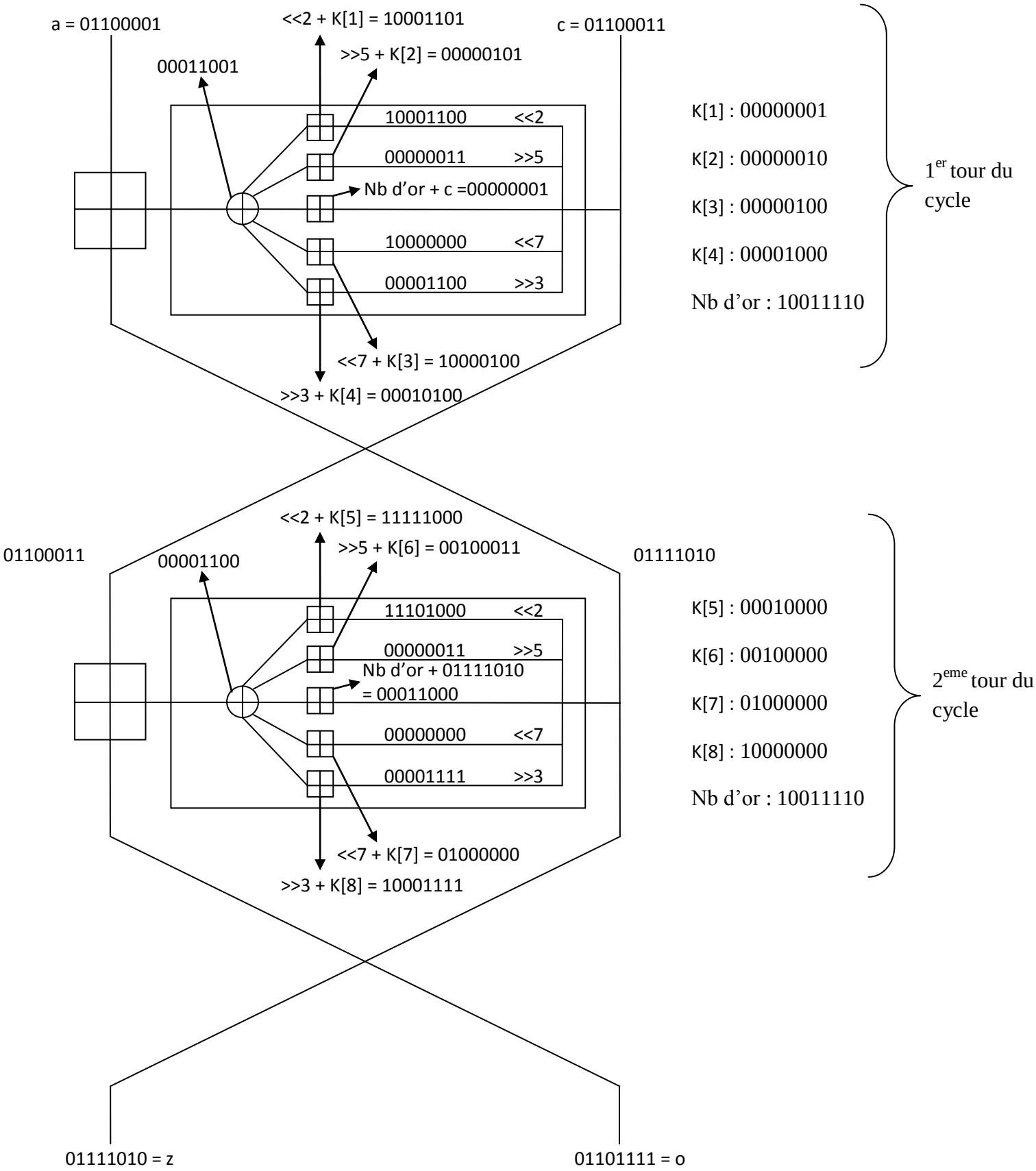
On prend un bloc de 16 bits qui est un mot 2 caractères alphabétiques (ac) :

Caractère	Code ascii décimal	Code ascii binaire
A	97	01100001
C	99	01100011

On prend une clé de 64 bits divisée en 8 sous clés de 8 bits chacune :

Numéro de la clé	Code binaire de la clé
K[1]	00000001
K[2]	00000010
K[3]	00000100
K[4]	00001000
K[5]	00010000
K[6]	00100000
K[7]	01000000
K[8]	10000000

Le schéma suivant montre les différentes étapes du processus de cryptage du mot (ac) dans un seul cycle :


a = 01100001
<<2 + K[1] = 10001101
c = 01100011
>>5 + K[2] = 00000101
00011001
10001100 <<2
00000011 >>5
Nb d'or + c =00000001
10000000 <<7
00001100 >>3
<<7 + K[3] = 10000100
>>3 + K[4] = 00010100
K[1] : 00000001
K[2] : 00000010
K[3] : 00000100
K[4] : 00001000
Nb d'or : 10011110
1er tour du cycle
<<2 + K[5] = 11111000
01100011
>>5 + K[6] = 00100011
00001100
01111010
11101000 <<2
00000011 >>5
Nb d'or + 01111010 = 00011000
00000000 <<7
00001111 >>3
<<7 + K[7] = 01000000
>>3 + K[8] = 10001111
K[5] : 00010000
K[6] : 00100000
K[7] : 01000000
K[8] : 10000000
Nb d'or : 10011110
2eme tour du cycle
01111010 = z
01101111 = o

1er tour du cycle :

- La lettre (c) subit des décalages de 2 bits a gauche, 5 bits a droite, 7 bits a gauche et 3 bits a droite.
- Ces derniers décalages sont additionnés respectivement avec les sous clés K[i] (i=1, 2, 3, 4).
- La lettre (c) est additionnée avec le nombre d'or.
- Le résultat de toutes ces additions est combiné avec un XOR.
- Le résultat du XOR est additionné avec la lettre (a).
- La lettre (a = 01100001) devient maintenant (z = 01111010) elle est donc cryptée.

2eme tour du cycle :

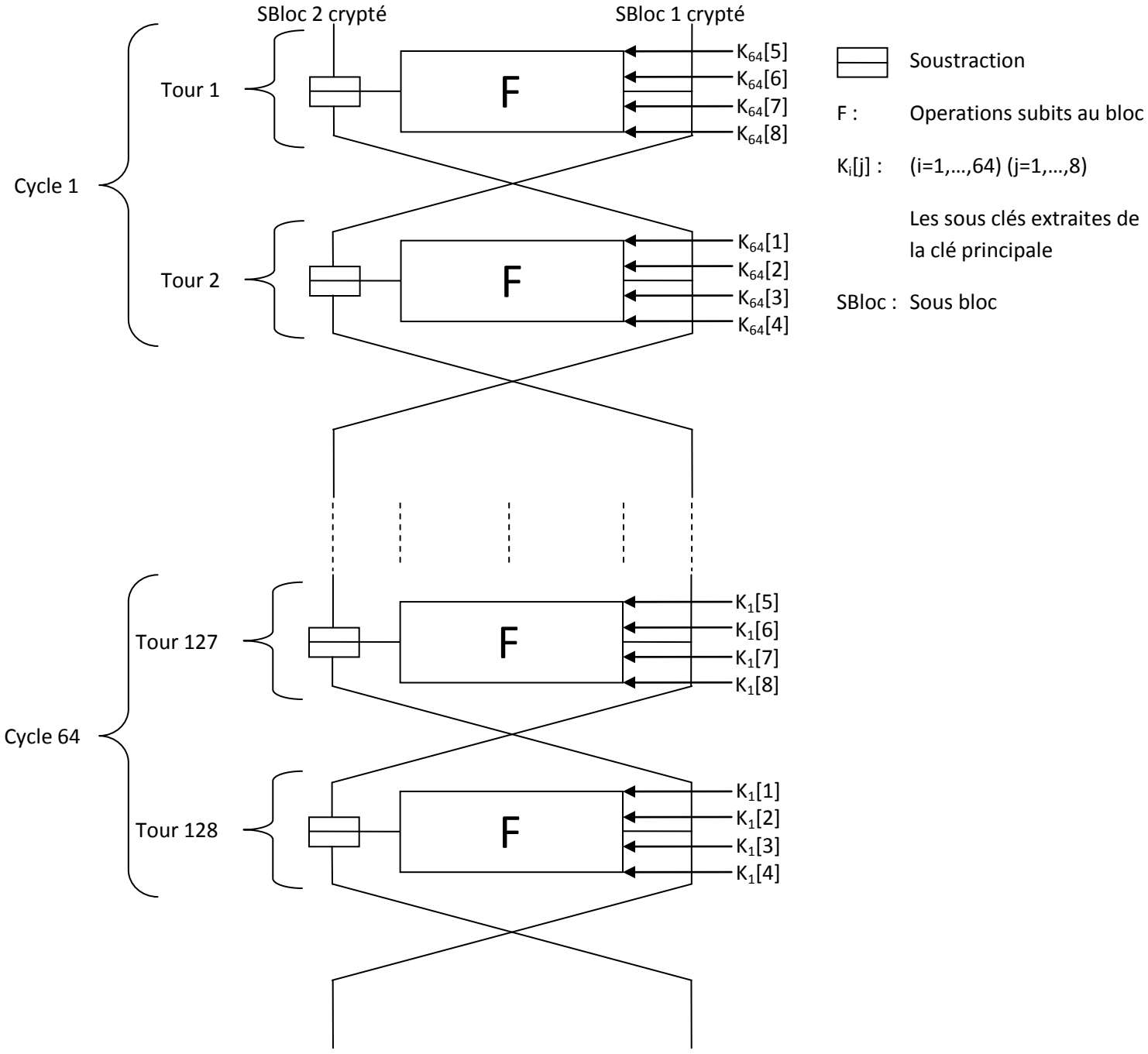
- Au second tour les 2 sous bloc sont inversés.
- La lettre (c) reste intacte.
- La lettre (z) subit des décalages de 2 bits a gauche, 5 bits a droite, 7 bits a gauche et 3 bits a droite.
- Ces derniers décalages sont additionnés respectivement avec les sous clés K[i] (i=5, 6, 7, 8).
- La lettre (z) est additionnée avec le nombre d'or.
- Le résultat de toutes ces additions est combiné avec un XOR.
- Le résultat du XOR est additionné avec la lettre (c).
- La lettre (c = 01100011) devient maintenant (o = 01101111) elle est donc cryptée.

Fin du cycle :

- Les 2 blocs seront encore permutés.
- Le mot (ac) est crypté en le mot (zo).
- A la fin du cycle dans notre algorithme, le nombre d'or est incrémenté par lui-même et les clés subissent des rotations de bits avant d'entrer dans le prochain cycle. Ceci n'est pas fait dans l'exemple afin de le faciliter.

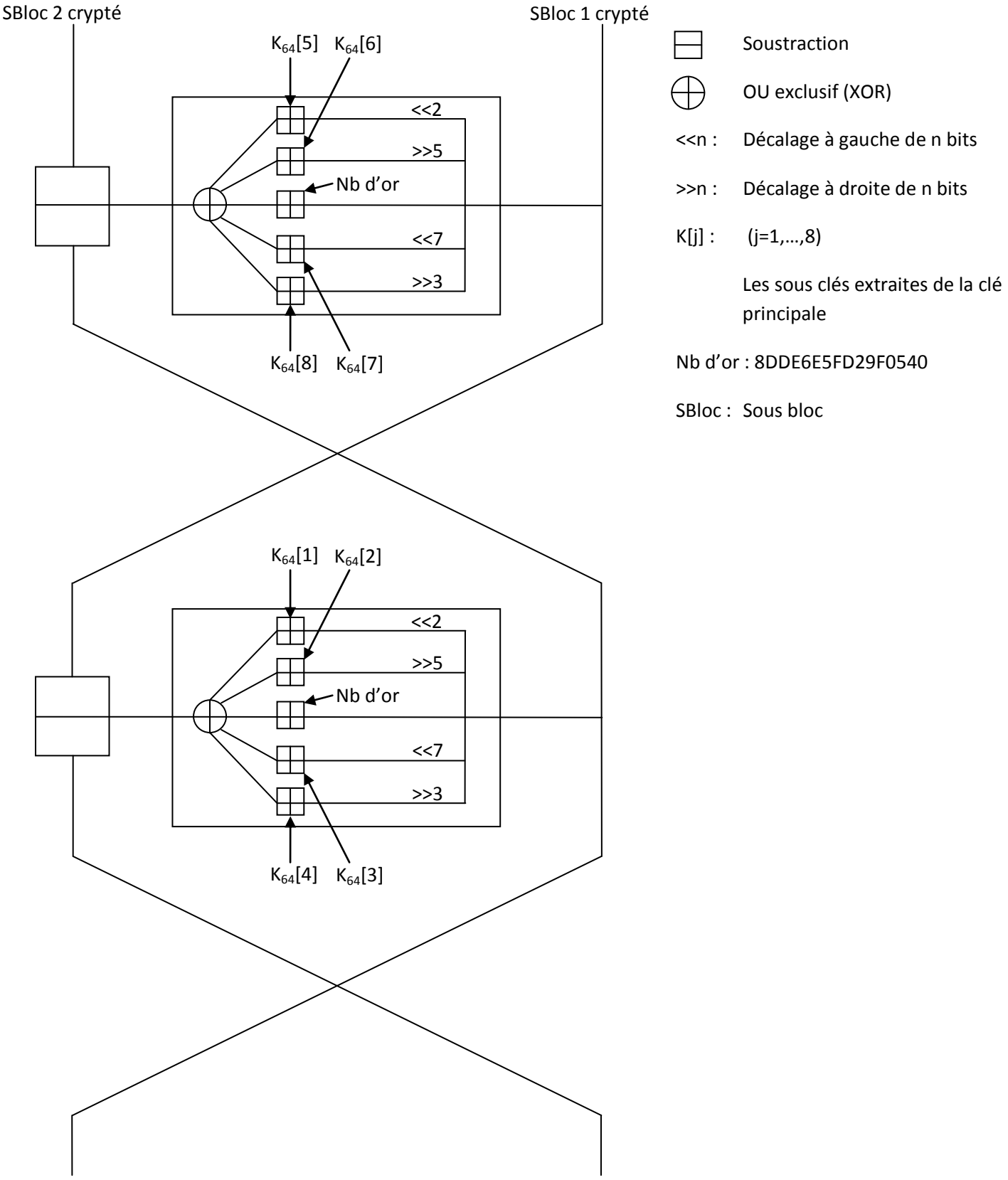
5.1.2. Décryptage :

Le schéma ci-dessous représente le réseau de Feistel de notre algorithme avec déchiffrement en mode ECB (Electronic Code Book) :



Décryptage en mode ECB

Pour mieux expliquer le déroulement de l'algorithme de déchiffrement on fera comme vu précédemment en prenant seulement le premier cycle :



Vue détaillée d'un cycle de déchiffrement en mode ECB

Exemple :

Cet exemple va illustrer le fonctionnement du schéma précédent en prenant en entrée le bloc chiffré précédemment qui est le mot (zo) avec la meme clé. A la fin de cet exemple on devrait retrouver le mot déchiffré (ac).

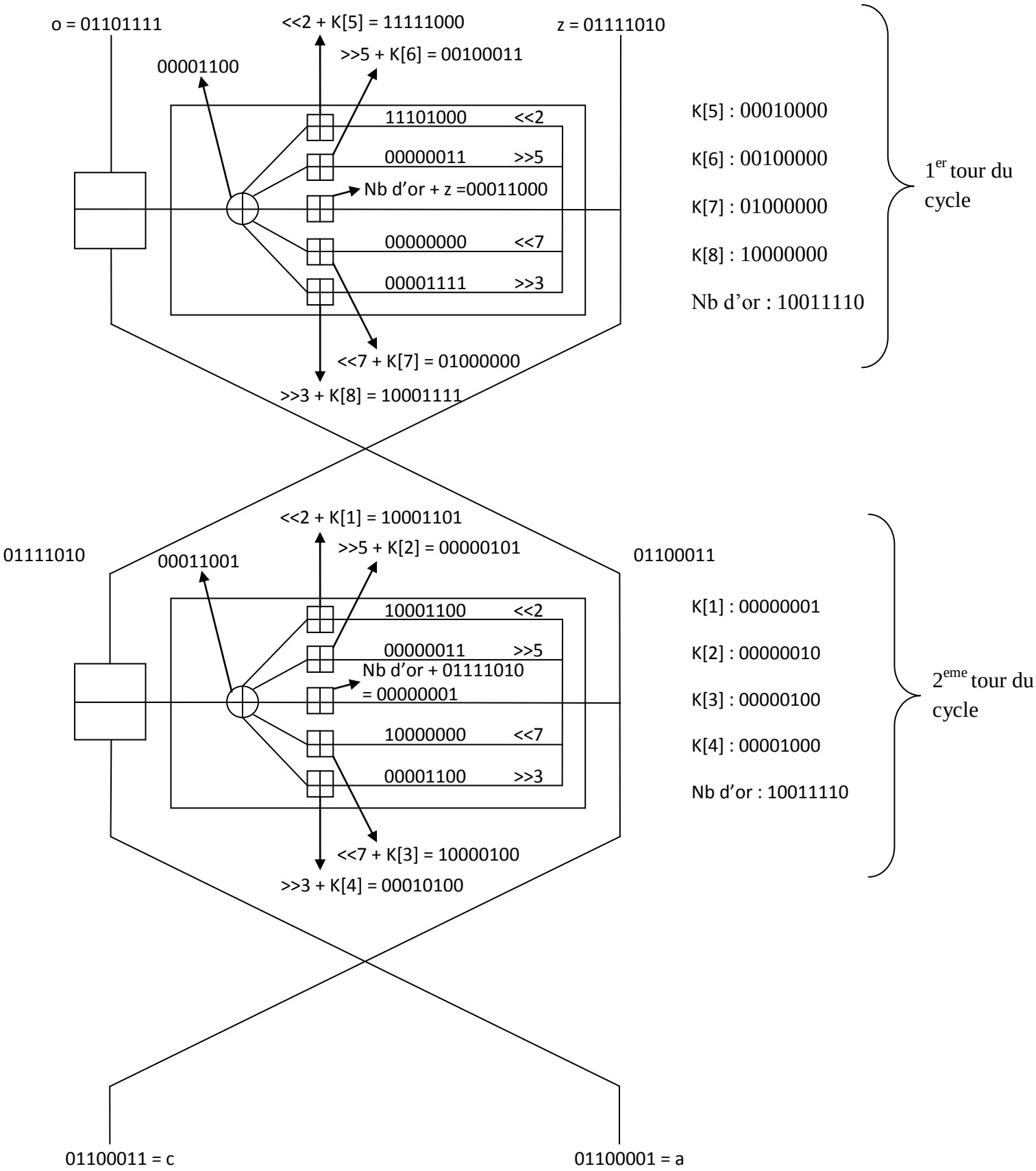
On prend le bloc de 16 bits qui est un mot 2 caractères alphabétiques (zo) :

Caractère	Code ascii décimal	Code ascii binaire
z	122	01111010
o	111	01101111

On prend la même clé précédente de 64 bits divisée en 8 sous clés de 8 bits chacune :

Numéro de la clé	Code binaire de la clé
K[1]	00000001
K[2]	00000010
K[3]	00000100
K[4]	00001000
K[5]	00010000
K[6]	00100000
K[7]	01000000
K[8]	10000000

Le schéma suivant montre les différentes étapes du processus de décryptage du mot (zo) dans un seul cycle :


o = 01101111
<<2 + K[5] = 11111000
z = 01111010
>>5 + K[6] = 00100011
00001100
11101000 <<2
00000011 >>5
Nb d'or + z =00011000
00000000 <<7
00001111 >>3
<<7 + K[7] = 01000000
>>3 + K[8] = 10001111
K[5] : 00010000
K[6] : 00100000
K[7] : 01000000
K[8] : 10000000
Nb d'or : 10011110
1er tour du cycle
<<2 + K[1] = 10001101
01111010
>>5 + K[2] = 00000101
00011001
01100011
10001100 <<2
00000011 >>5
Nb d'or + 01111010 = 00000001
10000000 <<7
00001100 >>3
<<7 + K[3] = 10000100
>>3 + K[4] = 00010100
K[1] : 00000001
K[2] : 00000010
K[3] : 00000100
K[4] : 00001000
Nb d'or : 10011110
2eme tour du cycle
01100011 = c
01100001 = a

1er tour du cycle :

- La lettre (z) subit des décalages de 2 bits a gauche, 5 bits a droite, 7 bits a gauche et 3 bits a droite.
- Ces derniers décalages sont additionnés respectivement avec les sous clés K[i] (i=5, 6, 7, 8).
- La lettre (z) est additionnée avec le nombre d'or.
- Le résultat de toutes ces additions est combiné avec un XOR.
- Le résultat du XOR est soustrait de la lettre (o).
- La lettre (o = 01101111) devient maintenant (c = 01100011) elle est donc décryptée.

2eme tour du cycle :

- Au second tour les 2 sous bloc sont inversés.
- La lettre (z) reste intacte.
- La lettre (c) subit des décalages de 2 bits a gauche, 5 bits a droite, 7 bits a gauche et 3 bits a droite.
- Ces derniers décalages sont additionnés respectivement avec les sous clés K[i] (i=1, 2, 3, 4).
- La lettre (c) est additionnée avec le nombre d'or.
- Le résultat de toutes ces additions est combiné avec un XOR.
- Le résultat du XOR est soustrait de la lettre (z).
- La lettre (z = 01111010) devient maintenant (a = 01100001) elle est donc décryptée.

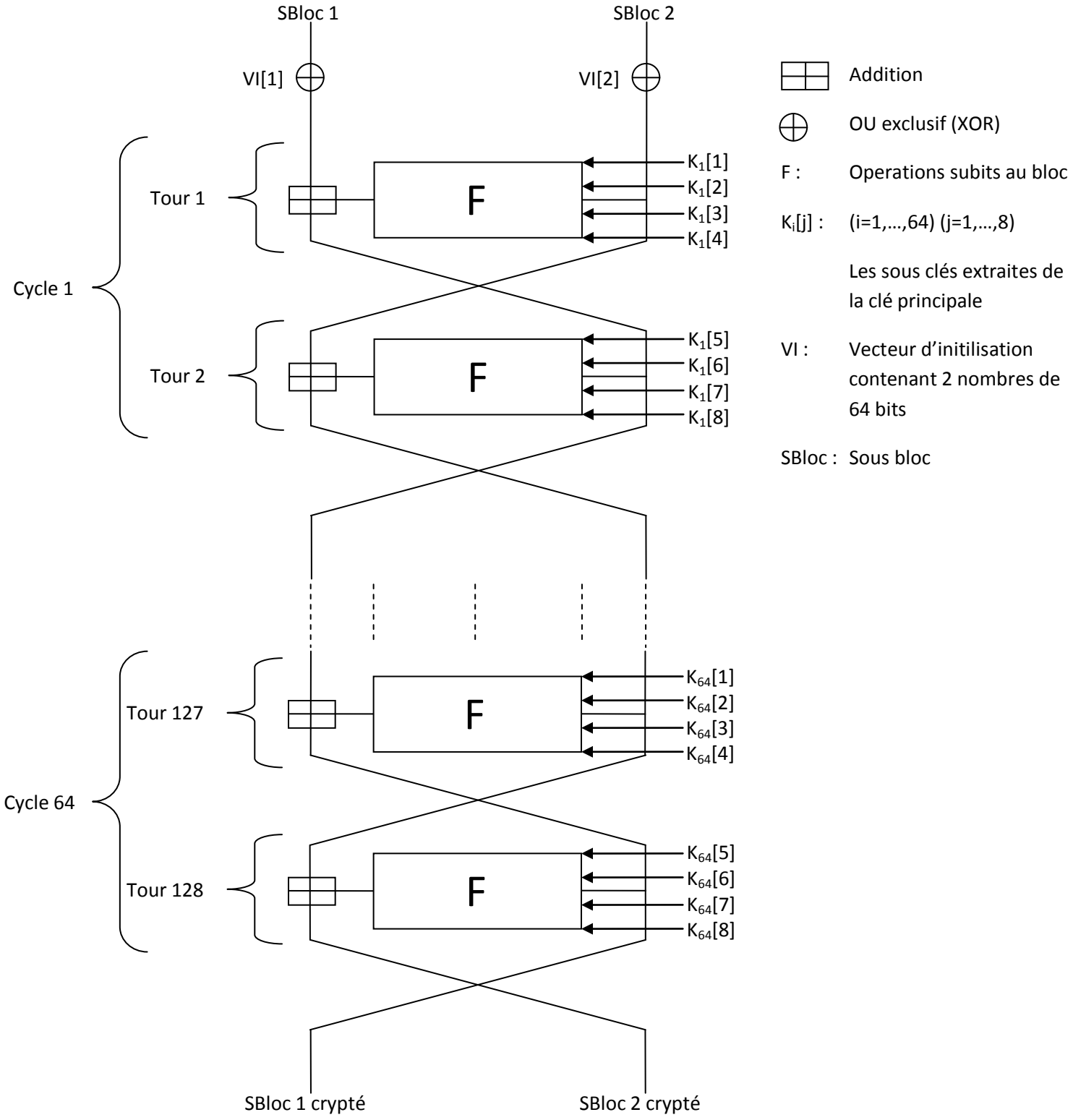
Fin du cycle :

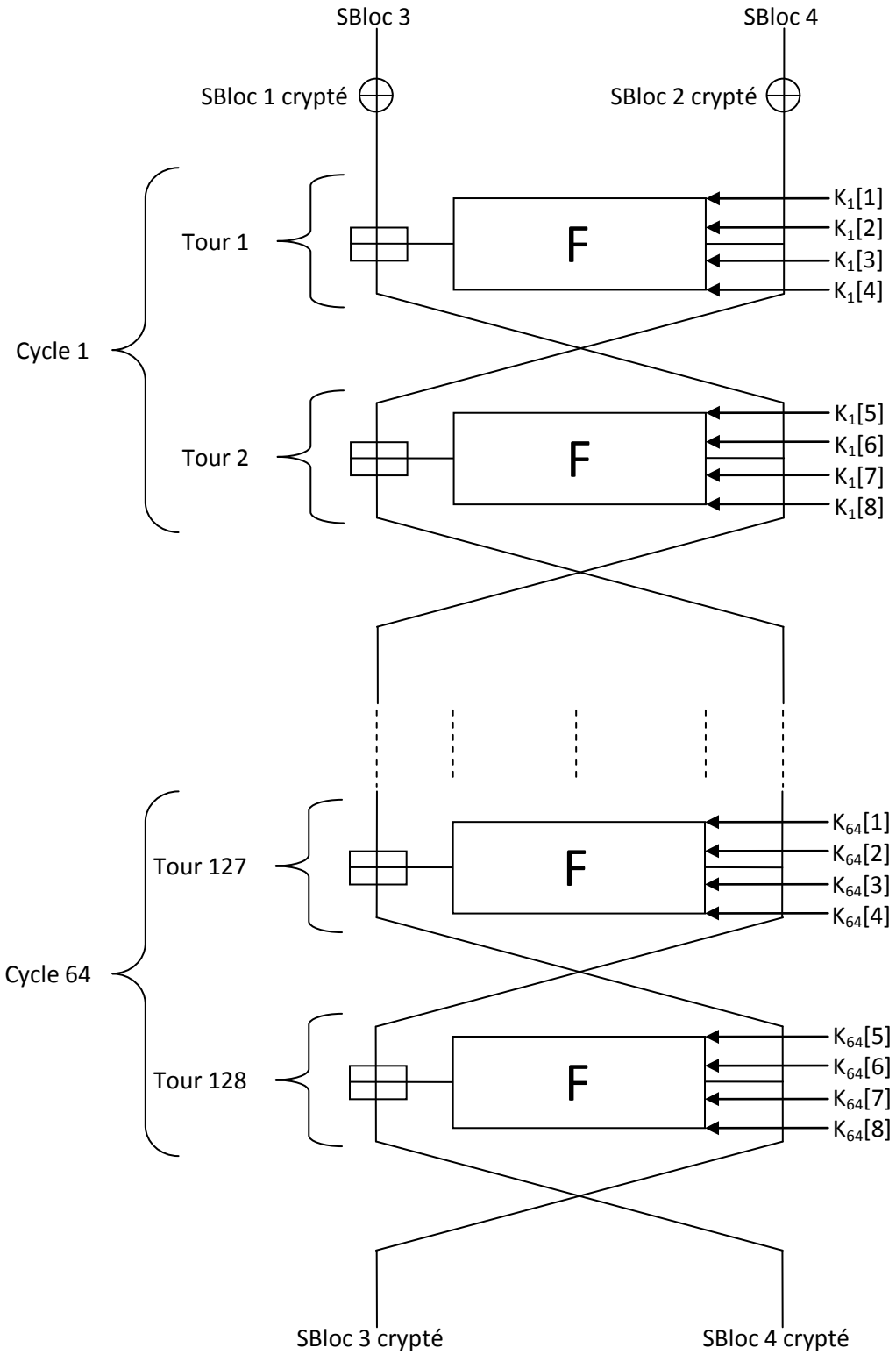
- Les 2 blocs seront encore permutés.
- Le mot (zo) est décrypté en le mot (ac).
- A la fin du cycle dans notre algorithme, le nombre d'or est décrémenté par lui-même et les clés subissent des rotations inverses de bits (par rapport au cryptage) avant d'entrer dans le prochain cycle. Ceci n'est pas fait dans l'exemple afin de le faciliter.

5.2. Fonctionnement en mode CBC :

5.2.1. Cryptage :

Le schéma ci-dessous représente le réseau de Feistel de notre algorithme avec chiffrement en mode CBC (Cipher Block Chaining) :



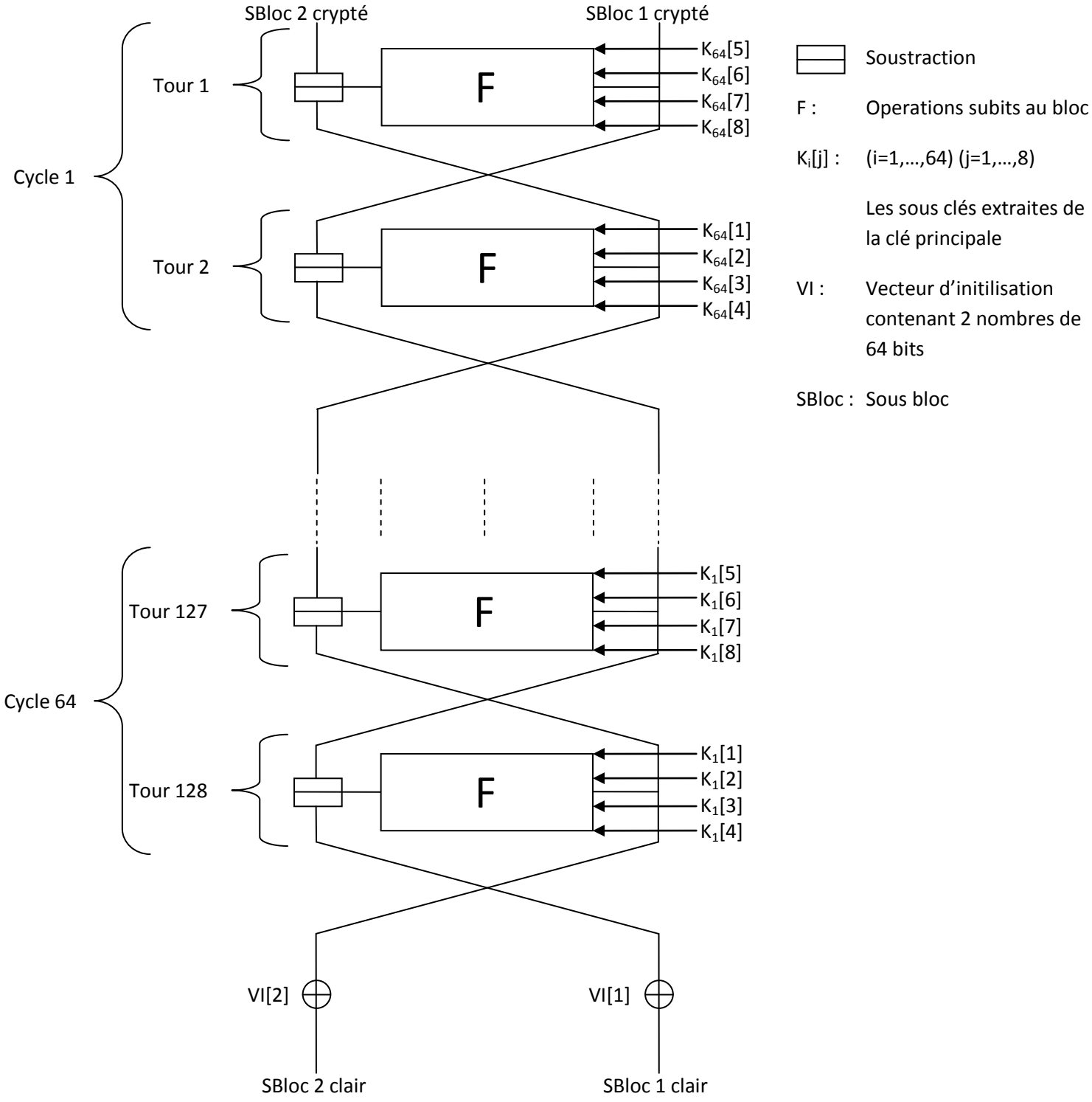


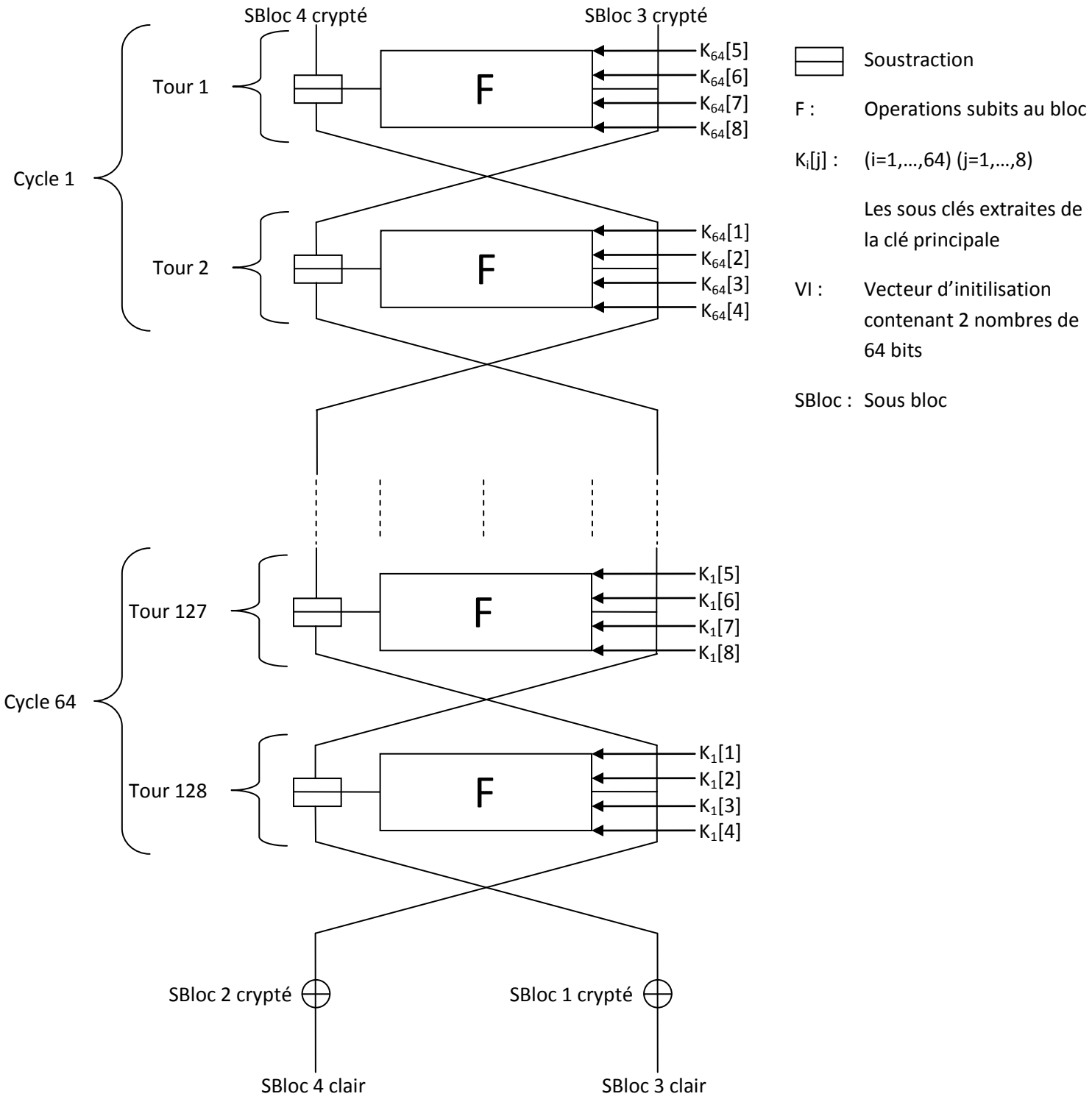
Cryptage en mode CBC

On peut voir que ce schéma est quasiment le même que celui du mode ECB à la différence que les précédents sous blocs chiffrés sont combinés avec les prochains sous blocs clairs en utilisant XOR.

5.2.2. Décryptage :

Le schéma ci-dessous représente le réseau de Feistel de notre algorithme avec déchiffrement en mode CBC (Cipher Block Chaining) :



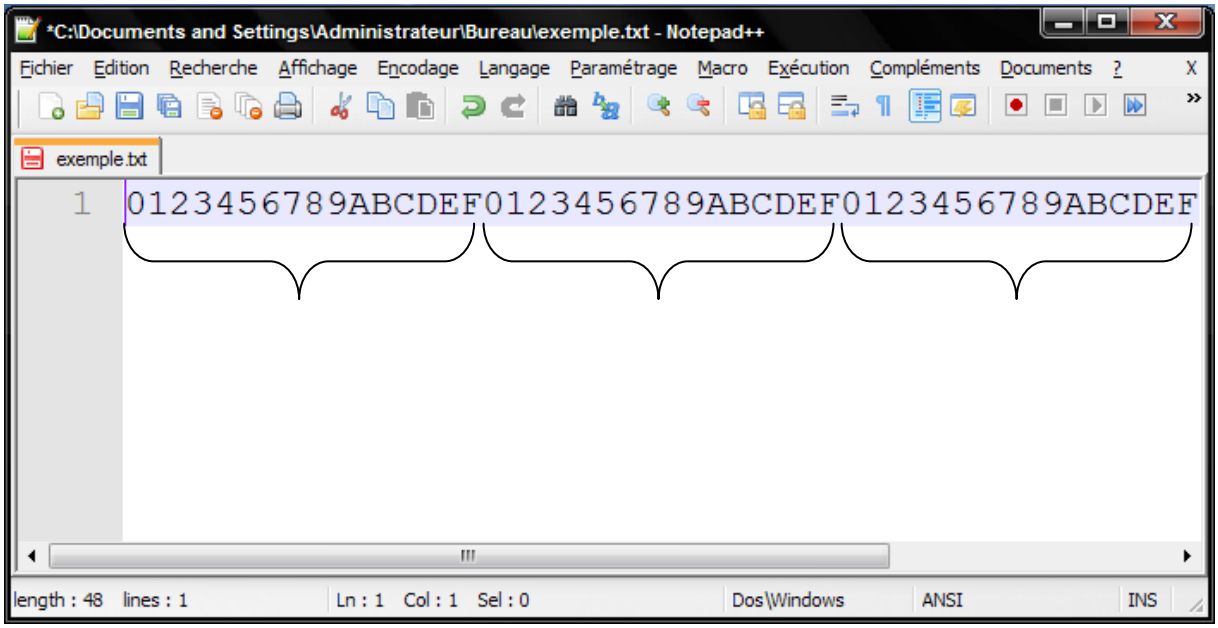


Décryptage en mode CBC

A la sortie des 64 cycles du réseau, les sous blocs sortants sont combinés avec un XOR aux sous blocs chiffrés précédents pour avoir le bloc clair.

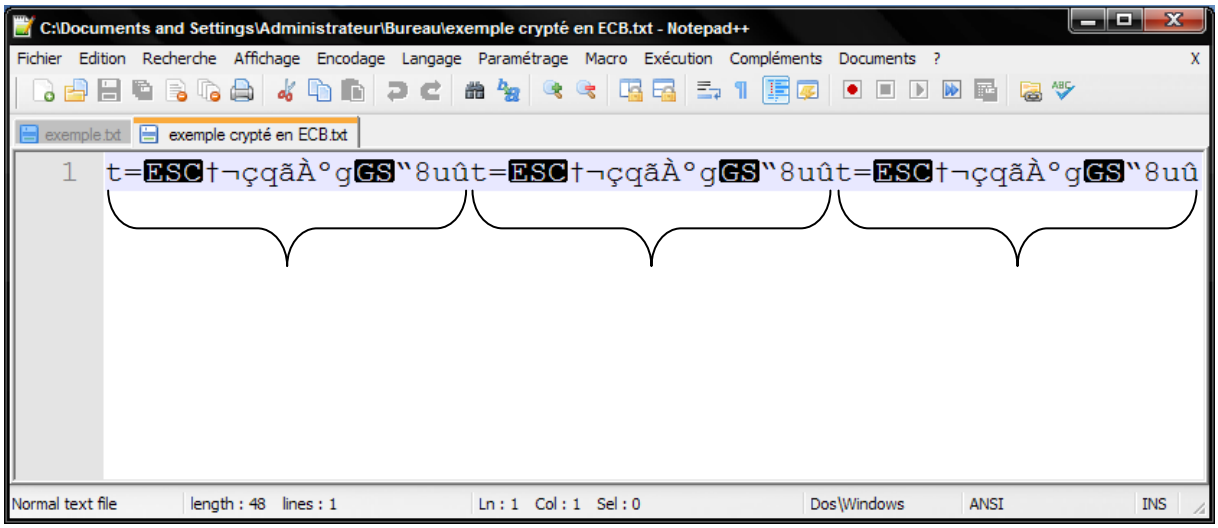
6. Différence de cryptage entre ECB et CBC :

Nous allons vous montrer ici par des screenshot, la différence des deux modes de cryptages précédents :



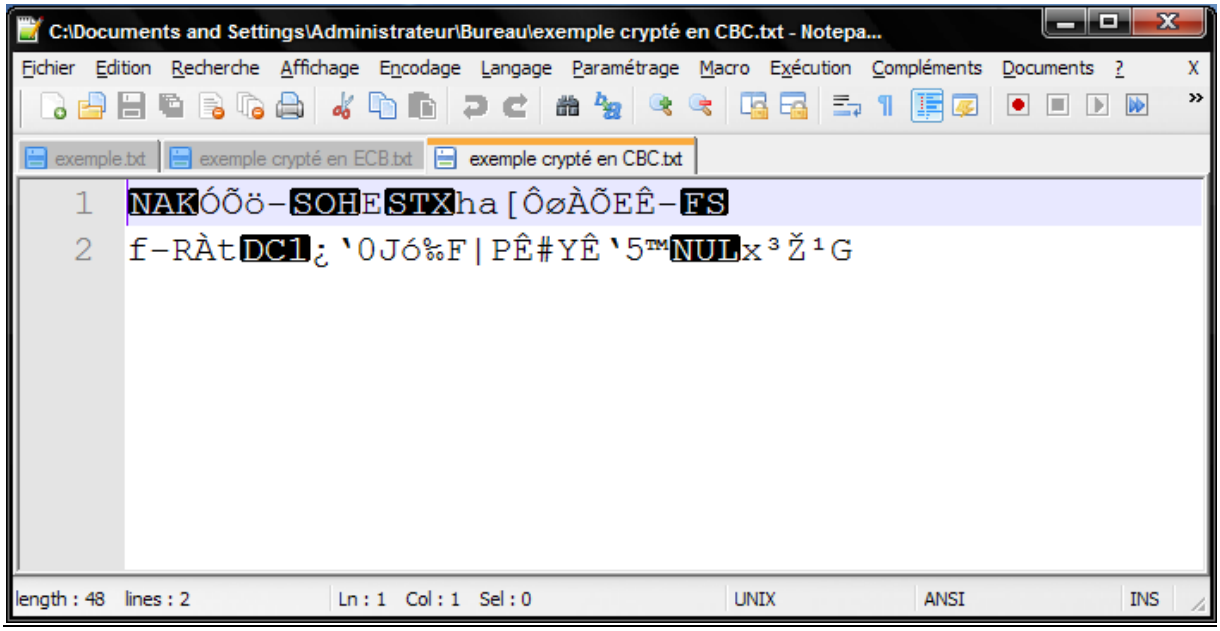
Cette image ci-dessus représente un fichier texte de 48ø contenant donc 3 blocs de 16ø. Chacun de ces blocs est une chaine de caractères (0123456789ABCDEF).

Cryptage en ECB :



On remarque ici que les 3 blocs sont chiffrés de la même manière car les 3 blocs en clair précédents sont identiques l'un en vers l'autre.

Cryptage en CBC :



Par contre ici on ne remarque aucune ressemblance entre les blocs.

7. Conclusion :

Dans ce chapitre, nous avons vu les étapes de modélisation qui nous ont conduites à la conception de notre algorithme. En effet ceci était nécessaire pour la réalisation de notre application.

1. Introduction :

Dans ce chapitre, on parlera dans un premier temps de l'environnement de développement de notre application ensuite nous allons voir des exemples de fonctionnement de cette dernière à travers des screenshots.

2. Environnement de développement :

2.1. Support machine utilisé :

Nous avons fait tourner notre application sous une machine ayant les caractéristiques suivantes :

- Processeur i3 à Fréquence d'horloge 2.5Gh$_z$
- 2Gø de RAM
- Carte graphique Intel 1Gø de mémoire

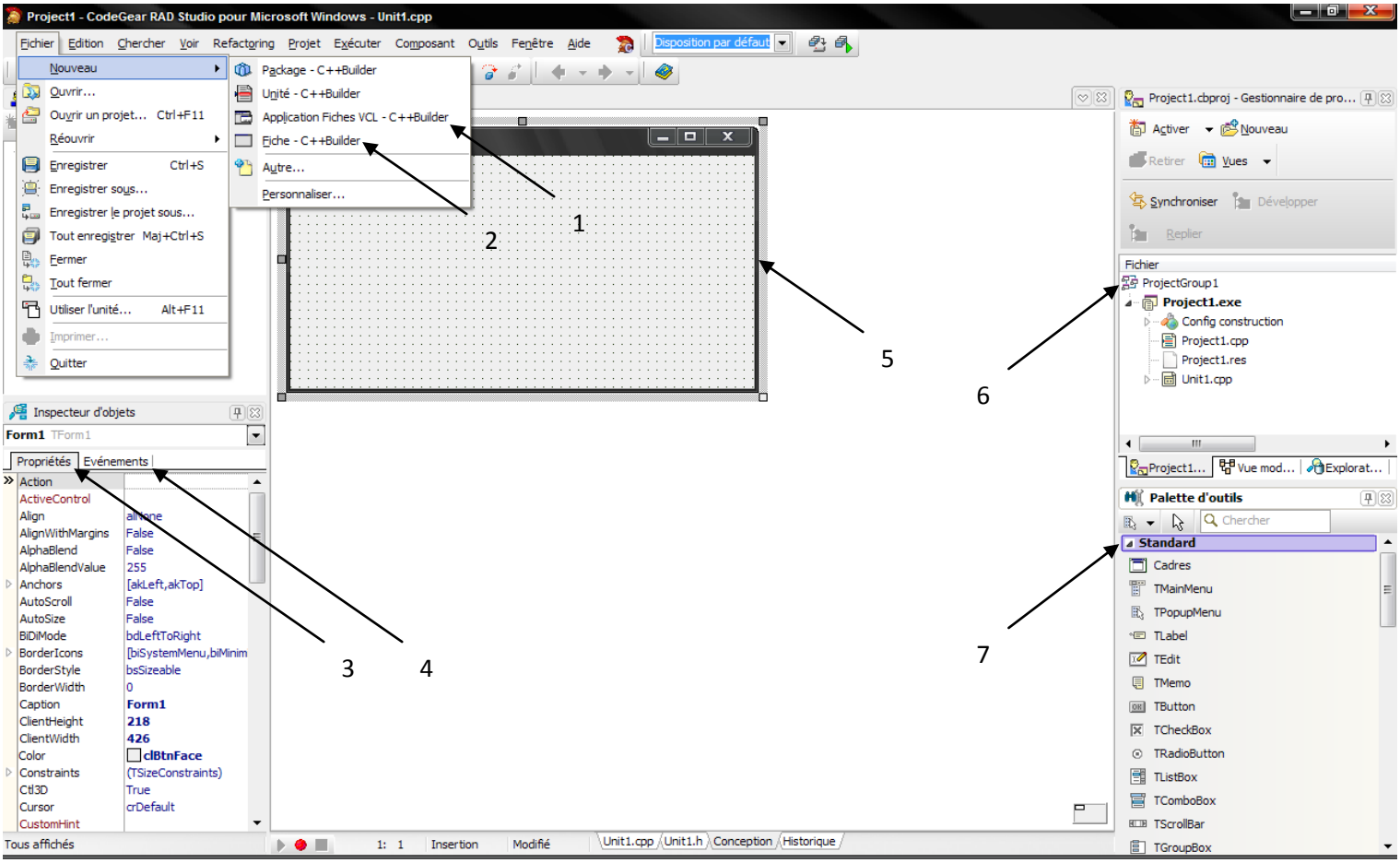
2.2. Langage de programmation utilisé : [Web06]

Dans ce projet, nous avons utilisé C++ pour programmer notre application avec l'EDI (environnement de développement intégré) C++Builder 2009.

C++Builder est un RAD (Rapid Application Development) conçu par Borland qui reprend les mêmes concepts, la même interface et la même bibliothèque que Delphi en utilisant le langage C++. Il permet de créer rapidement des applications Win32 ainsi qu'une interface graphique avec son éditeur de ressources.

Les causes qui nous ont poussé à choisir C++Builder sont :

- interface utilisateur plus agréable
- Présence de nombreux composants : on peut quasiment créer n'importe quelle application simplement en drag and droppant le composant adéquat sur la form principale
- code compréhensible
- portabilité vers Linux (si on se contente des objets CLX)



Vue générale d'un projet en C++Builder

- (1) : Création d'une nouvelle application.
- (2) : Création d'une fenêtre supplémentaire pour une application.
- (3) : Propriétés d'un composant.
- (4) : Evénement d'un composant.
- (5) : Fenêtre de l'application.
- (6) : Détail du projet contenant toutes les unités utilisées par ce dernier.
- (7) : Liste des composants standards.

2.3. Les outils utilisés : [Web07] [Web08]

Photoshop : c'est un logiciel de retouche, de traitement et de dessin assisté par ordinateur édité par Adobe. Il est principalement utilisé pour le traitement de photographies numériques, mais sert également à la création d'images.

Inno Setup : c'est un logiciel libre permettant de créer des installateurs pour Windows. Ceux-ci peuvent comporter des scripts programmés en Pascal.

3. Screenshot de l'application :

Dans ce qui suit, nous allons vous montrer quelques screenshots de notre application afin d'illustrer son fonctionnement.

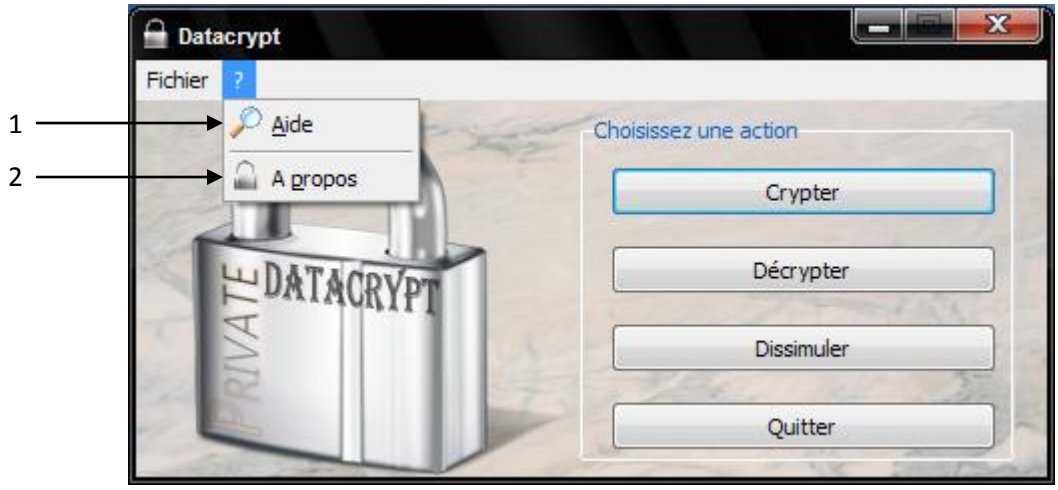
3.1. Menu principal :

Menu principal

- (1) : bouton pour accéder à la fenêtre de cryptage.
- (2) : bouton pour accéder à la fenêtre de décryptage.
- (3) : bouton pour accéder à la fenêtre de dissimulation de texte dans une image.
- (4) : bouton pour quitter l'application.
- (5) : Affiche le menu de Fichier.
- (6) : Affiche le menu d'aide (?).

Menu Fichier du menu principal

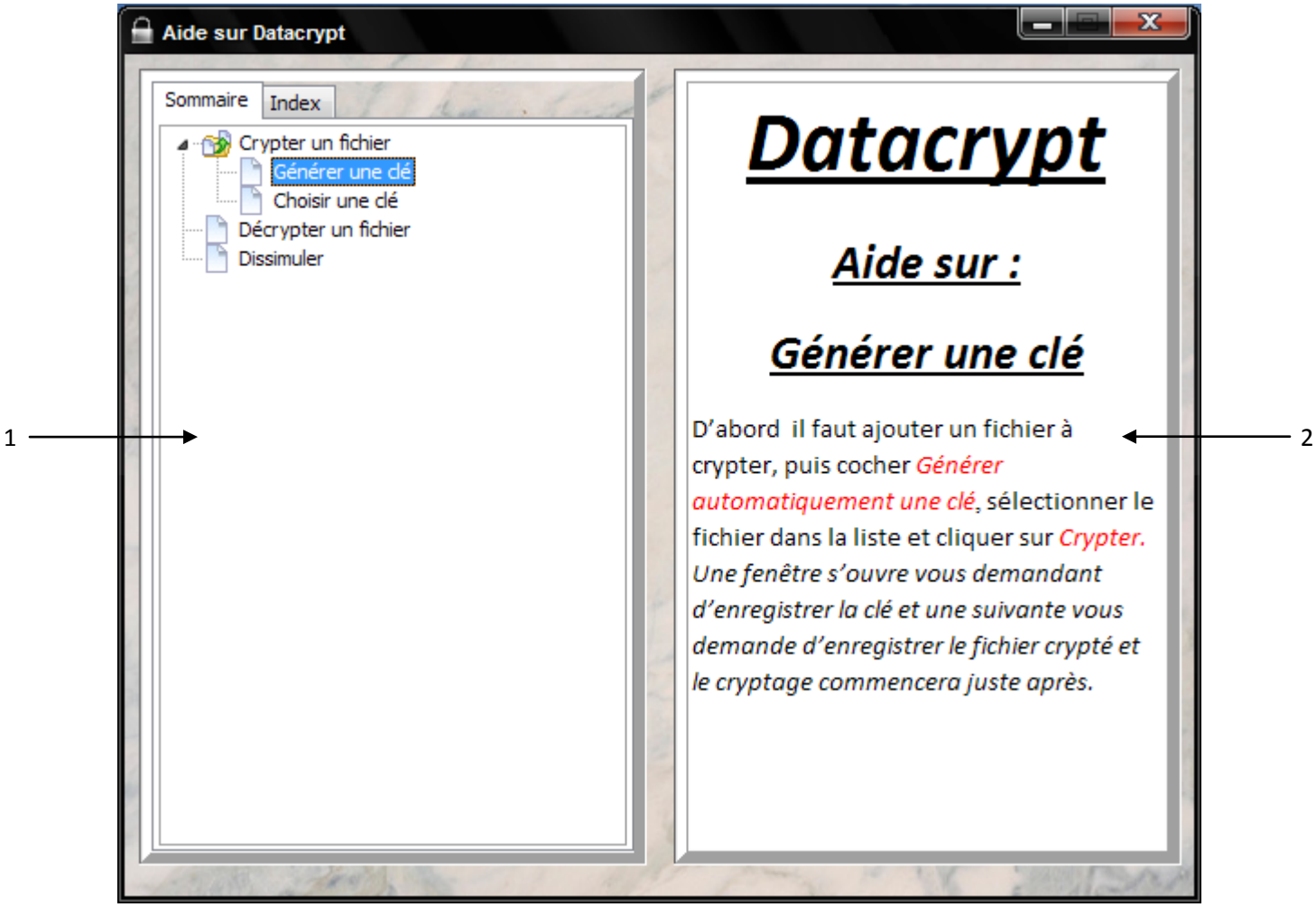
Ce menu est composé des mêmes boutons vus précédemment.



Menu Aide du menu principal

- (1) : Cette option permet d'accéder à la fenêtre d'aide.
- (2) : Cette option permet d'accéder à la fenêtre d'a propos.

3.2. Aide :



Sommaire d'aide

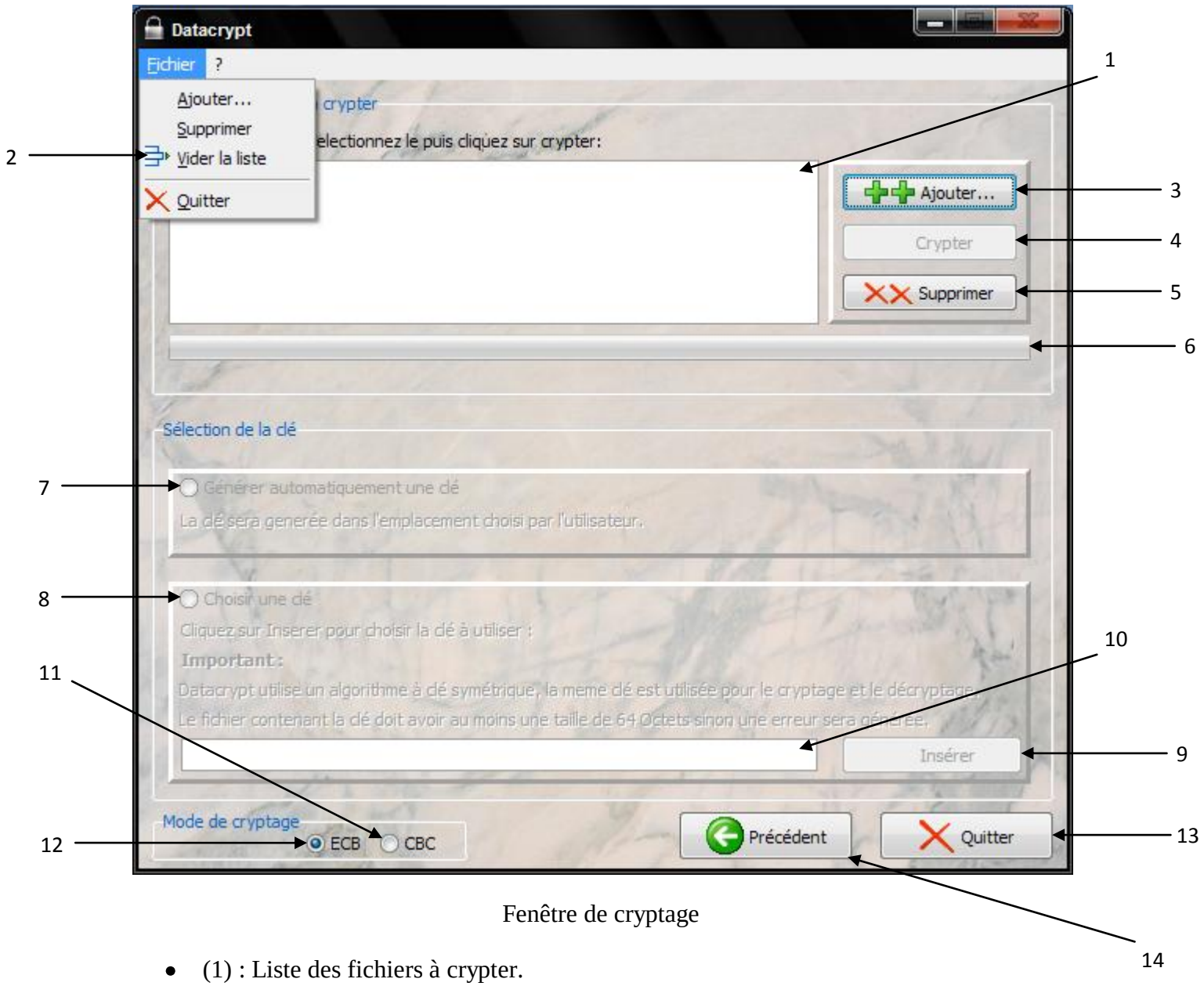
- (1) : Sommaire d'aide.
- (2) : Détail de la fonctionnalité sélectionnée dans le sommaire.

Index d'aide

- (1) : Index d'aide.
- (2) : Détail de la fonctionnalité sélectionnée dans l'index.

En gros, cette fenêtre sert à expliquer les différentes fonctionnalités de l'application afin de connaitre son usage. Elle explique aussi les différentes procédures réalisées par chaque bouton et les deux modes de cryptage/décryptage.

3.3. Cryptage :

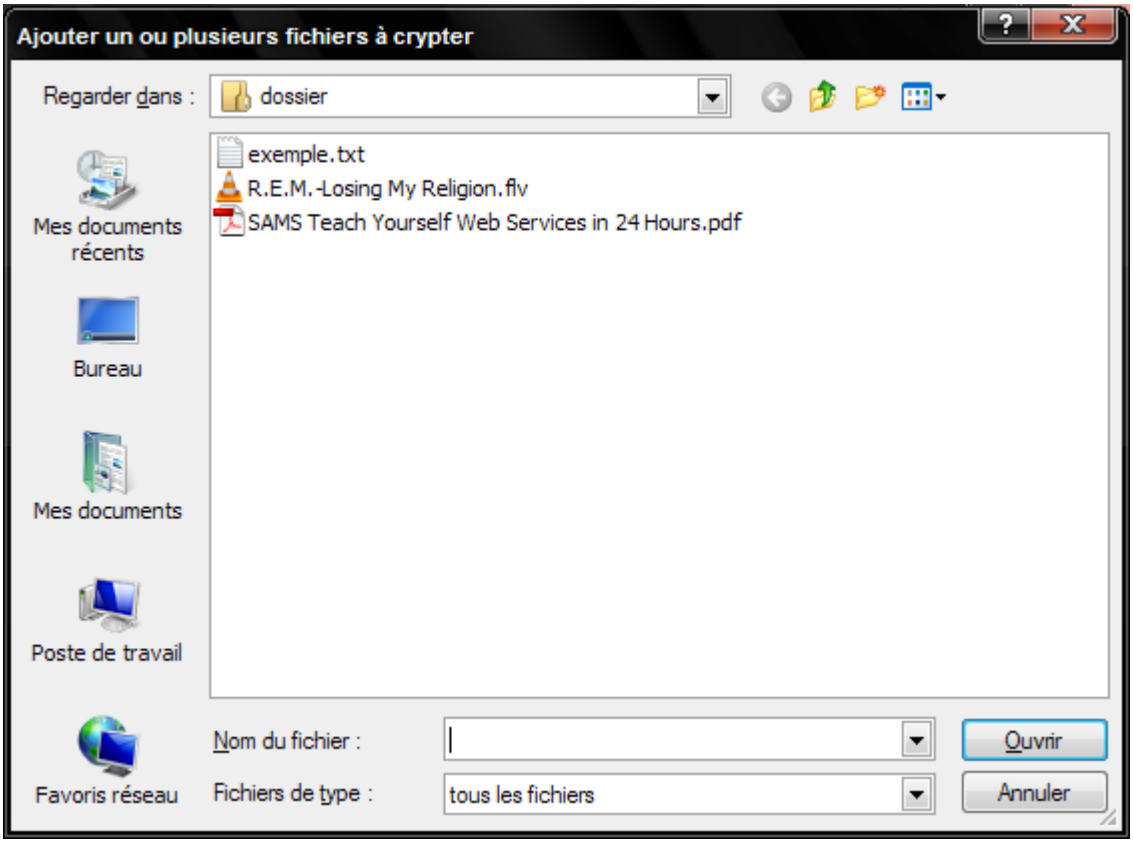


Fenêtre de cryptage

- (1) : Liste des fichiers à crypter.
- (2) : Vider la liste des fichiers à crypter.
- (3) : Ajouter un ou plusieurs fichiers à crypter dans la liste.
- (4) : Crypter le fichier sélectionné dans la liste.
- (5) : Supprimer le fichier sélectionné dans la liste.
- (6) : Barre de progression du cryptage.
- (7) : Cocher ce radio pour générer une clé automatiquement.
- (8) : Cocher ce radio pour sélectionner une clé.
- (9) : Insérer une clé de cryptage.
- (10) : Affiche le chemin de la clé sélectionnée.
- (11) : Cocher ce radio pour un cryptage en mode CBC.
- (12) : Cocher ce radio pour un cryptage en mode ECB.
- (13) : Quitter l'application.
- (14) : Revenir au menu principal.

Exemple illustrant comment crypter un fichier :

Tout d'abord, cliquez sur le bouton (Crypter) du menu principal pour accéder à la fenêtre de cryptage ensuite cliquez sur le bouton (Ajouter) et une boite de dialogue s'ouvre :

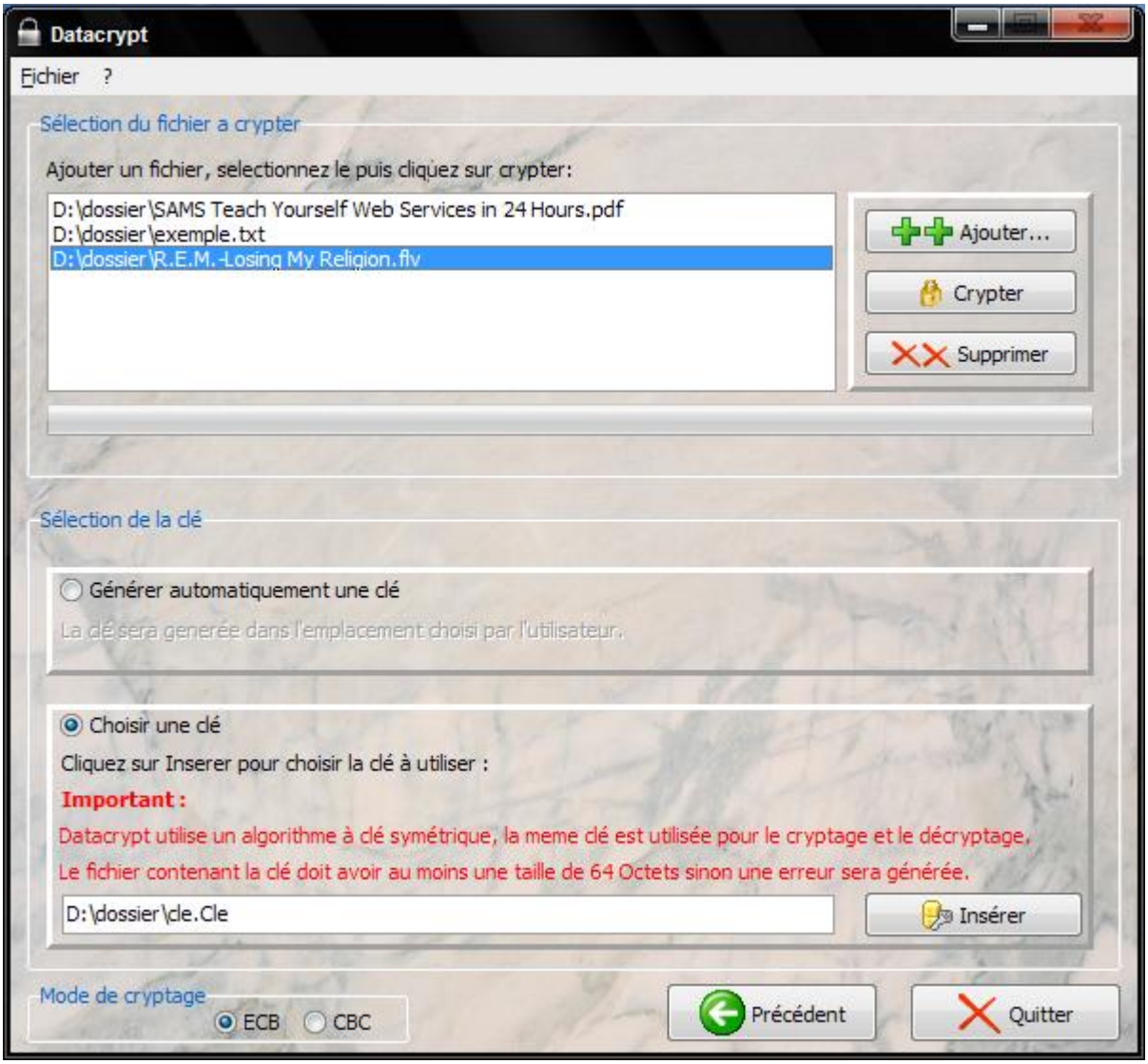


Boite de dialogue

Sélectionnez un ou plusieurs fichiers à crypter puis cliquez sur le bouton ouvrir. Les fichiers sélectionnés vont apparaitre dans la liste des fichiers à crypter dans la fenêtre de cryptage.

Choisissez un fichier à crypter dans la liste, sélectionnez la clé de cryptage en choisissant de la générer automatiquement ou de l'insérer manuellement. Le fichier représentant la clé doit avoir une taille minimum de 64ø.

Choisissez l'un des deux modes de cryptage proposés par l'application.

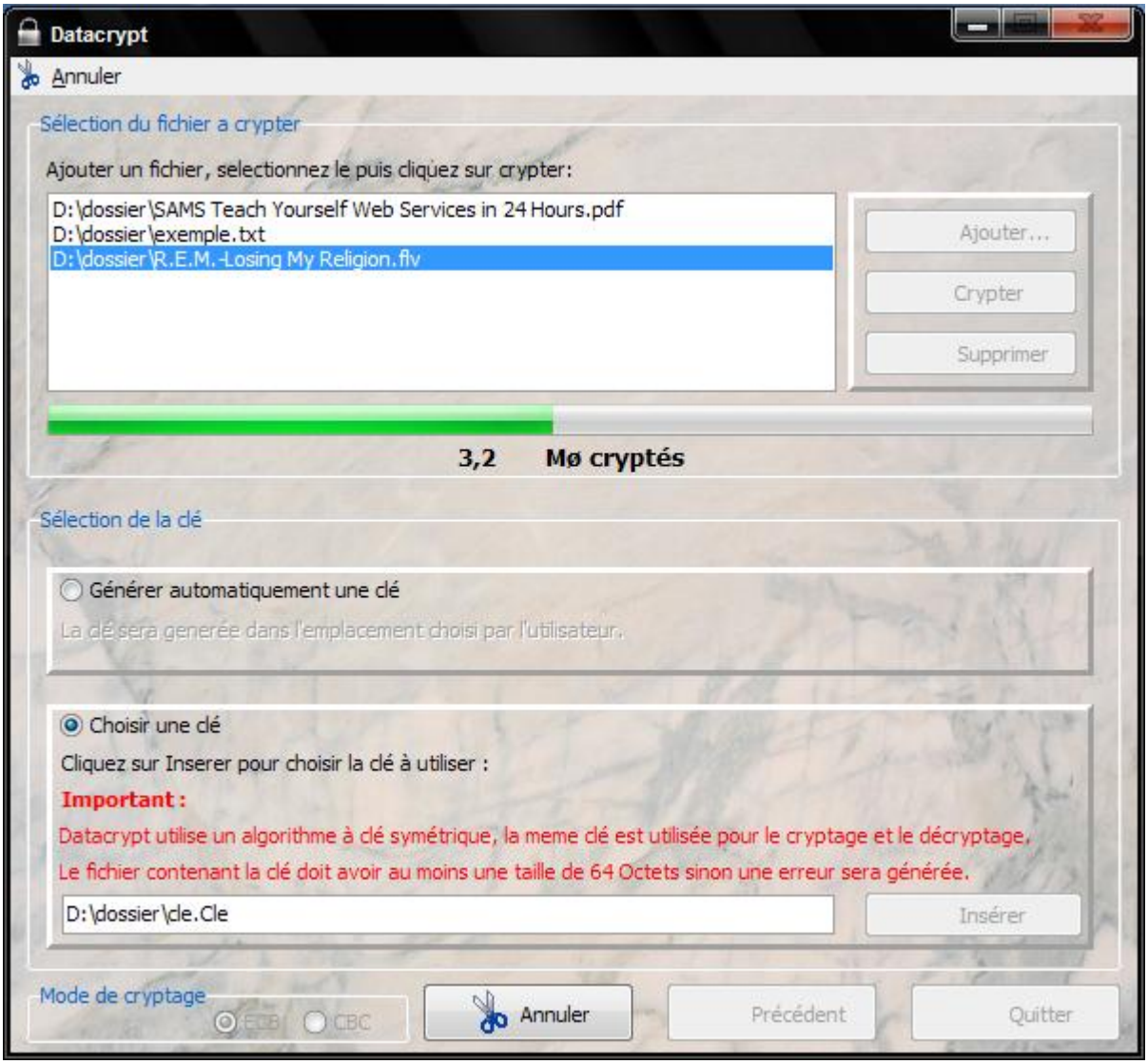


Fenêtre de cryptage après sélection du fichier à crypter

Cliquez sur le bouton (crypter) et une autre boite de dialogue va s'ouvrir, vous demandant de donner le nom du fichier crypté.

Donnez un nom à ce fichier puis cliquez sur enregistrer, la procédure de cryptage va commencer et vous pourrez voir la progression de ce dernier grâce à la barre de progression.

En cas d'erreur de votre part ou si vous voulez seulement annuler le cryptage, cliquez sur le bouton (Annuler) qui s'affiche seulement quand la procédure de cryptage est en cours.

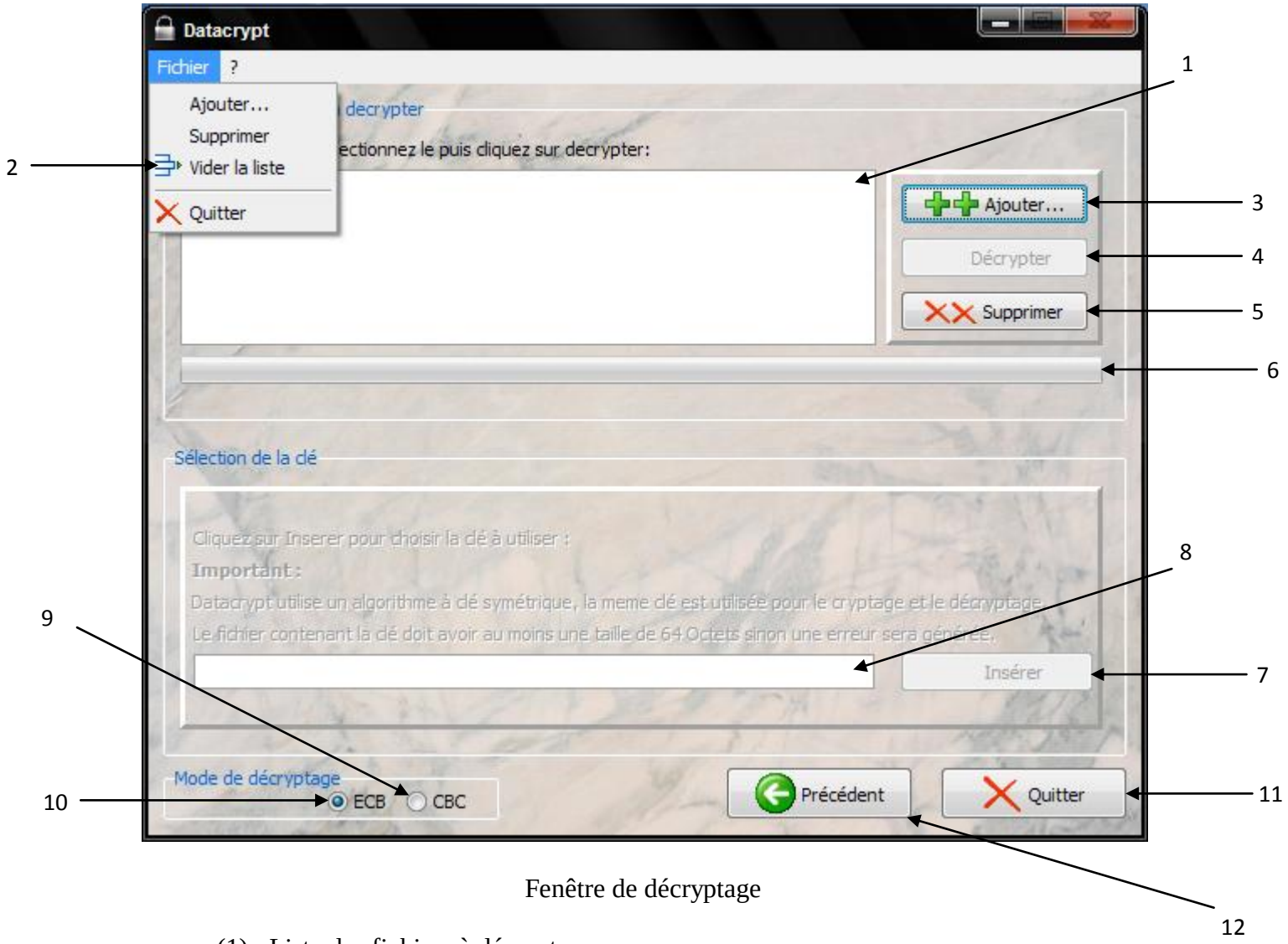


Cryptage d'un fichier en cours

A la fin du cryptage, un fichier (.crypt) est généré.

Pour décrypter ce fichier, il faut utiliser la même clé et le même mode employé précédemment dans le cryptage.

3.4. Décryptage :

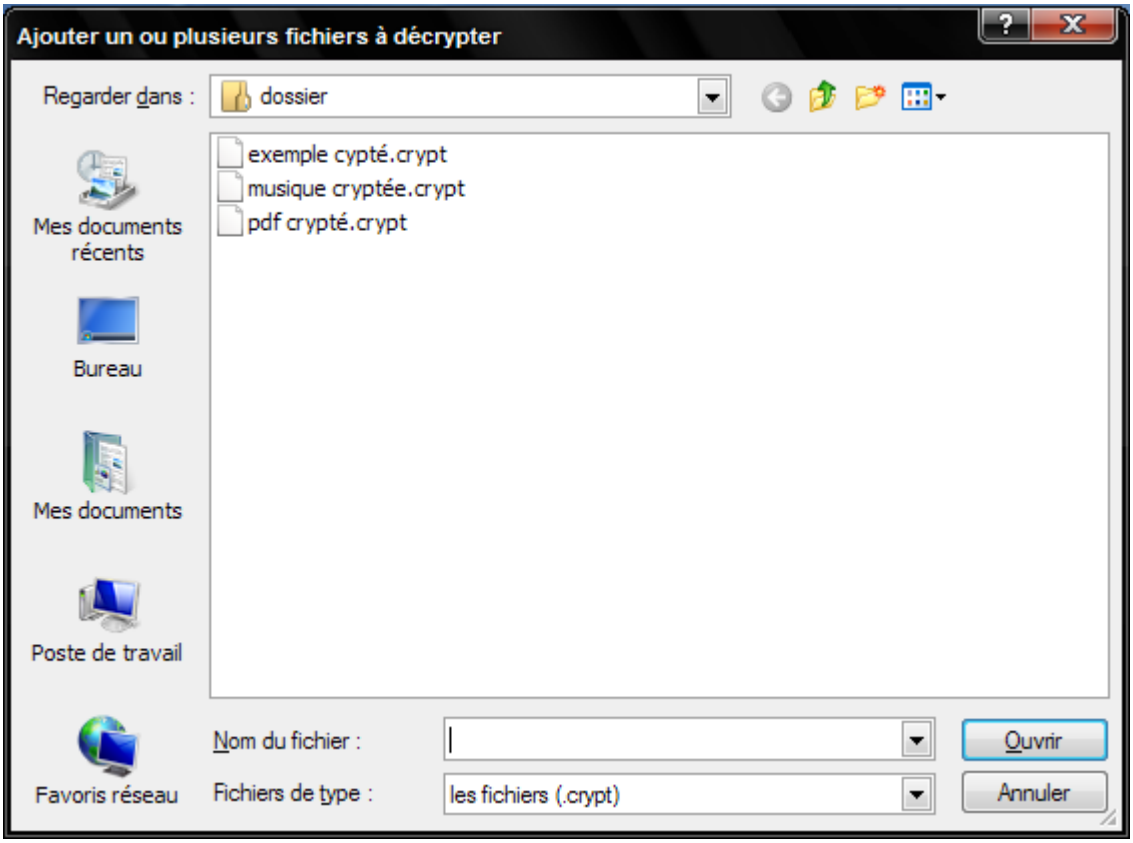


Fenêtre de décryptage

- (1) : Liste des fichiers à décrypter.
- (2) : Vider la liste des fichiers à décrypter.
- (3) : Ajouter un ou plusieurs fichiers à décrypter dans la liste.
- (4) : Décrypter le fichier sélectionné dans la liste.
- (5) : Supprimer le fichier sélectionné dans la liste.
- (6) : Barre de progression du décryptage.
- (7) : Insérer une clé de décryptage.
- (8) : Affiche le chemin de la clé sélectionnée.
- (9) : Cocher ce radio pour un décryptage en mode CBC.
- (10) : Cocher ce radio pour un décryptage en mode ECB.
- (11) : Quitter l'application.
- (12) : Revenir au menu principal.

Exemple illustrant comment décrypter un fichier :

Tout d'abord, cliquez sur le bouton (Décrypter) du menu principal pour accéder à la fenêtre de décryptage ensuite cliquez sur le bouton (Ajouter) et une boite de dialogue s'ouvre :

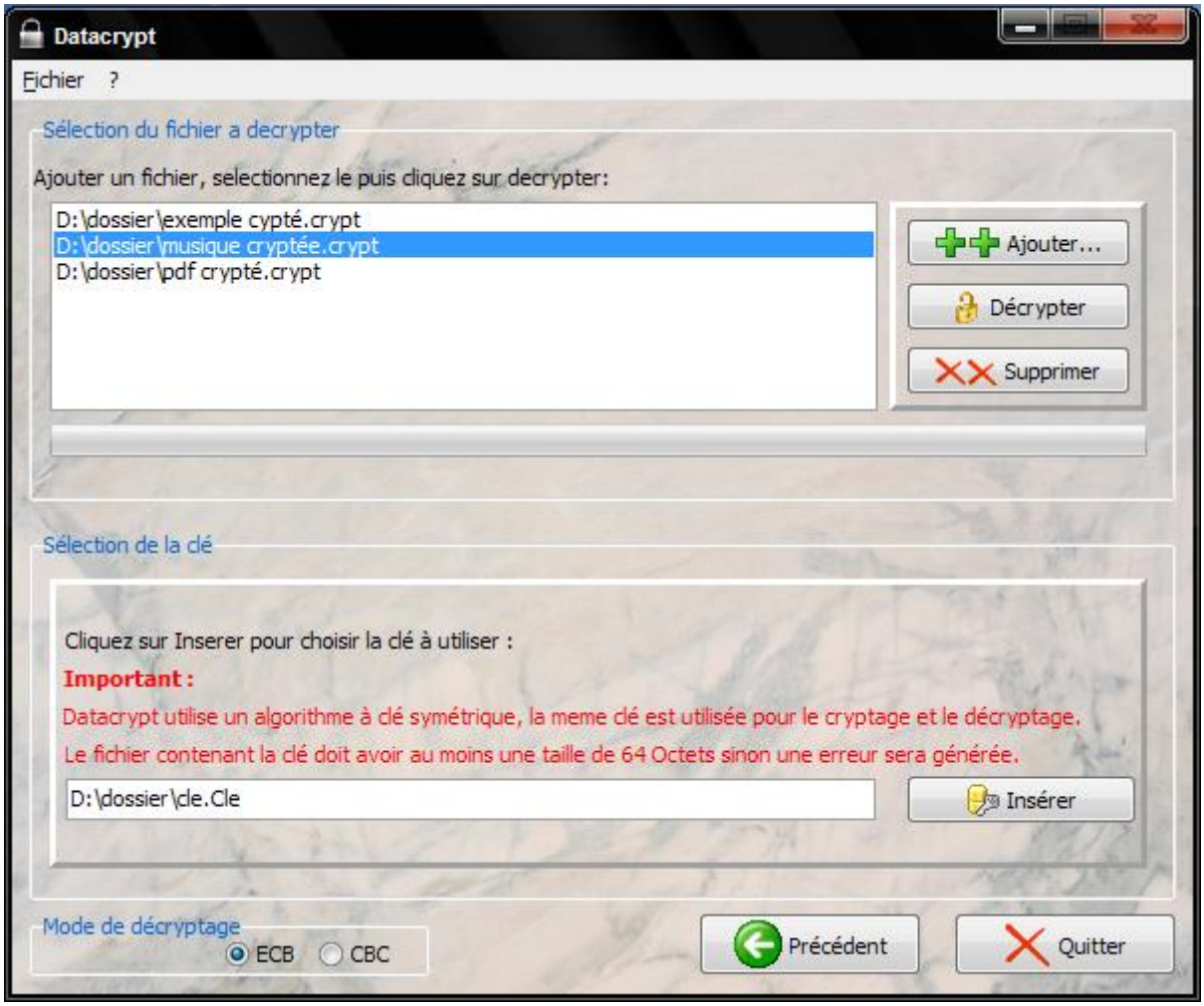


Boite de dialogue

Sélectionnez un ou plusieurs fichiers à décrypter puis cliquez sur le bouton ouvrir. Les fichiers sélectionnés vont apparaitre dans la liste des fichiers à décrypter dans la fenêtre de décryptage.

Choisissez un fichier à décrypter dans la liste, sélectionnez la clé de décryptage en cliquant sur le bouton (Insérer). Le fichier représentant la clé doit avoir une taille minimum de 64ø.

Choisissez l'un des deux modes de décryptage proposés par l'application. Si vous avez crypté le fichier avec un mode, il faudra le décrypter avec le même mode.



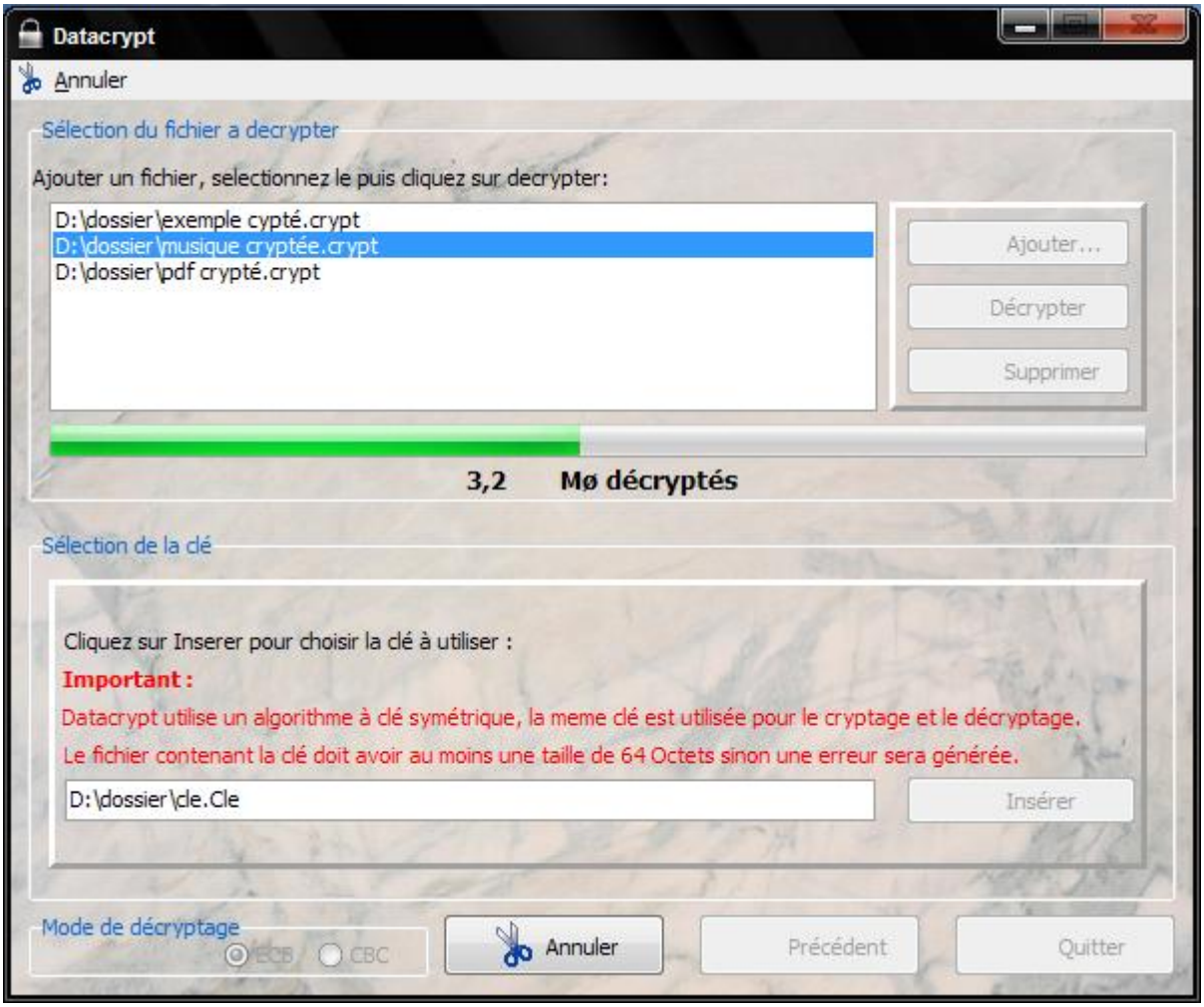
Fenêtre de décryptage après sélection du fichier à décrypter

Cliquez sur le bouton (décrypter) et une autre boite de dialogue va s'ouvrir, vous demandant de donner le nom du fichier décrypté.

Donnez un nom à ce fichier puis cliquez sur enregistrer, la procédure de décryptage va commencer et vous pourrez voir la progression de ce dernier grâce à la barre de progression.

L'extension du fichier décrypté sera détectée automatiquement par le logiciel, il est donc pas nécessaire de mettre une extension a ce dernier.

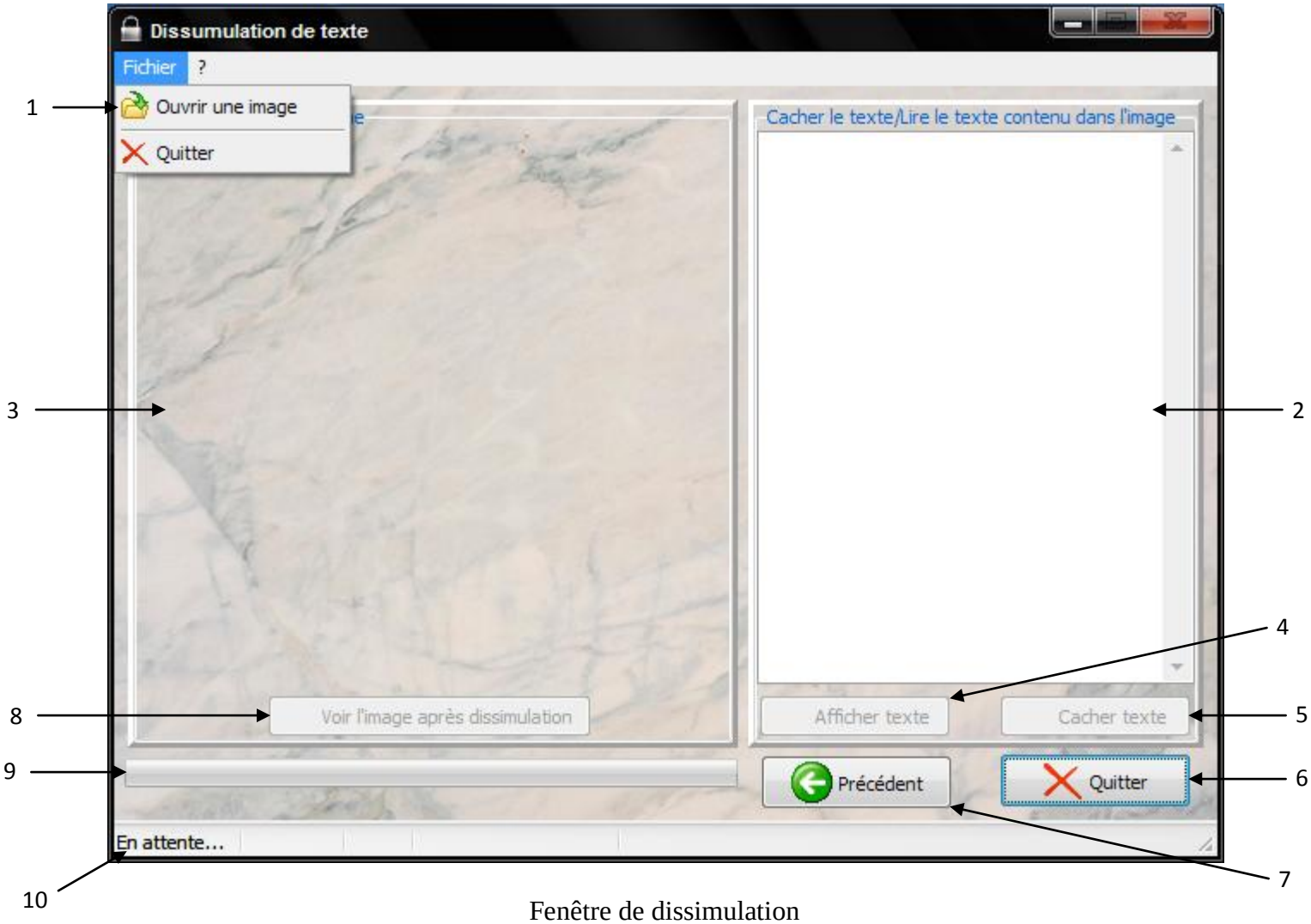
En cas d'erreur de votre part ou si vous voulez seulement annuler le décryptage, cliquez sur le bouton (Annuler) qui s'affiche seulement quand la procédure de décryptage est en cours.



Décryptage d'un fichier en cours

A la fin du décryptage, un fichier clair correspondant au fichier d'origine avant le cryptage est généré.

3.5. Stéganographie :



Fenêtre de dissimulation

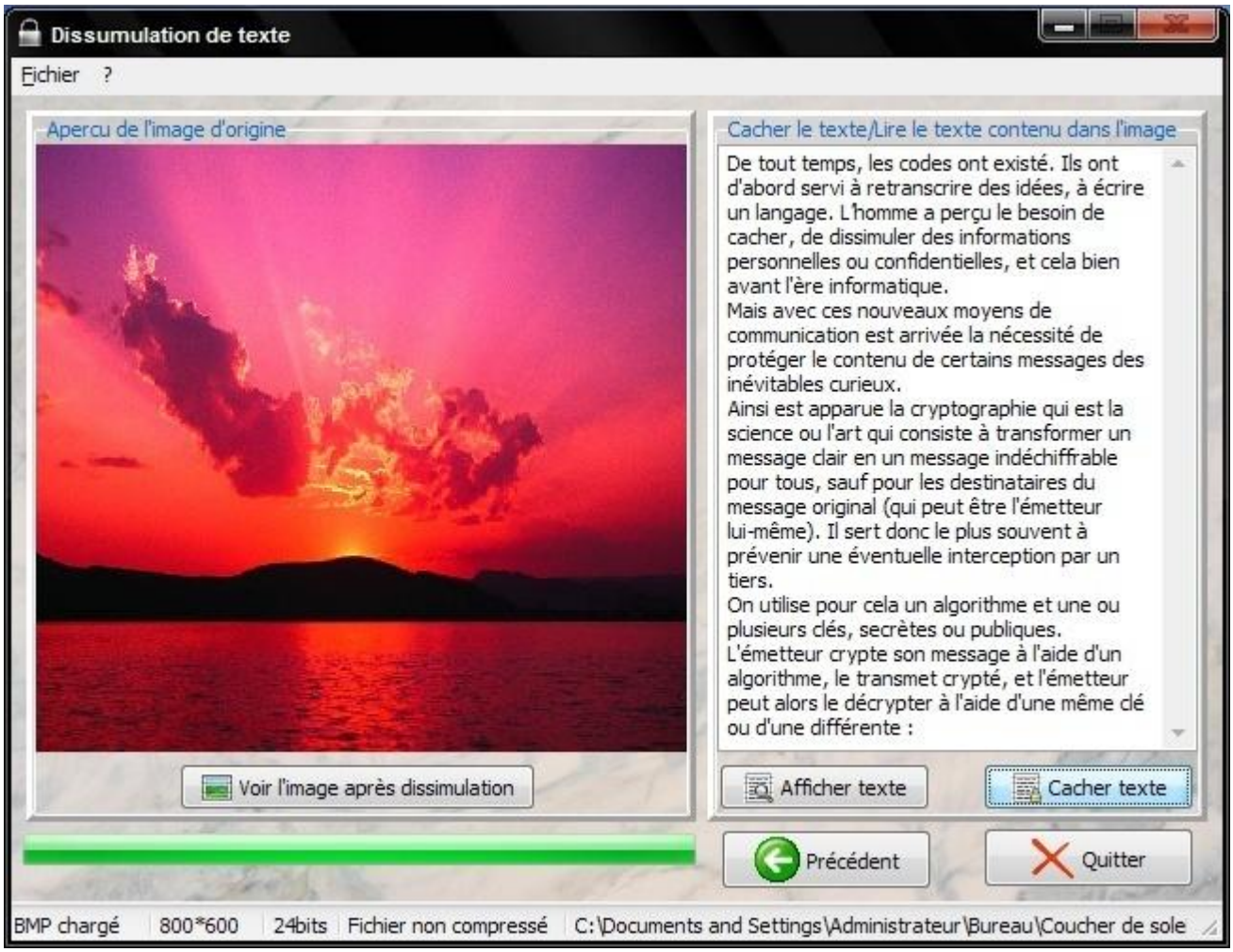
- (1) : Ouvrir une image (.bmp)
- (2) : Saisir le texte à cacher dans l'image ou affiche le texte caché dans cette dernière.
- (3) : Affiche l'image ouverte par la commande (1).
- (4) : Afficher le texte caché dans l'image.
- (5) : Cacher le texte saisi dans (2).
- (6) : Quitter l'application.
- (7) : Revenir au menu principal.
- (8) : Voir l'image après dissimulation du texte.
- (9) : Barre de progression.
- (10) : Barre d'état.

Exemple illustrant comment dissimuler un texte dans une image :

Tout d'abord, cliquez sur le bouton (Dissimuler) du menu principal pour accéder à la fenêtre de dissimulation ensuite cliquez sur Fichier => Ouvrir une image et une boite de dialogue s'ouvre.

Sélectionnez une image dans la quelle vous allez dissimuler votre texte, l'image sélectionnée va être affichée dans la fenêtre de dissimulation.

Saisissez votre texte dans le champ de saisi puis cliquez sur le bouton (Cacher texte).



Exemple de dissimulation d'un texte

Pour afficher le texte caché dans l'image, ouvrez cette dernière comme vu précédemment puis cliquez sur le bouton (Afficher texte).

Le texte caché sera affiché dans la zone de saisie.

4. Conclusion :

Dans ce chapitre, nous avons vu les différentes fonctionnalités de notre application qui consistent au cryptage et décryptage de fichiers ainsi qu'à la dissimulation de texte à travers d'exemples illustrés par des screenshots.

Nous avons aussi vu les causes qui nous ont poussés à choisir C++Builder comme EDI (environnement de développement intégré).

Conclusion générale

L'utilisation de réseaux spécialisés permet d'acheminer tout type d'information : son, texte, image, …etc.

On entre dans une ère d'évolution des réseaux dont les qualités du système informatique doivent être préservées ; mais on ne doit pas oublier l'existence des menaces, aussi nombreuses que variées, qui pèsent sur le réseau.

Le travaille que nous avons mené, nous a permis d'étudier quelques solutions pour contrer ces menaces et également de nous initier dans le domaine de la cryptographie et sur la programmation en C++Builder.

En effet, notre application s'est portée sur le chiffrement de données. Le logiciel que nous avons réalisé permet de chiffrer et déchiffrer tout type de fichier avec une clé secrète grâce à un algorithme symétrique. Notre application permet aussi de dissimuler du texte dans une image (Bitmap) et cette dernière pourra être cryptée à son tour. Les informations sont donc sécurisées et mises à l'abri des indiscrets.

Nous tenons à souligner que la cryptographie est large domaine de recherche et un domaine sensible car il concerne la sécurité des données. Tout cryptosystème conçu doit donc être testé plusieurs fois par des cryptographes pour prétendre à une bonne sécurisation de l'information.

Enfin, nous souhaitons que d'autres étudiants puissent tirer profit de ce travail pour concevoir et réaliser des cryptosystème assurant une bonne sécurité de l'information.

Webographie

[Web01]: http://ram-0000.developpez.com/tutoriels/cryptographie/

[Web02]: http://fr.wikipedia.org/wiki/Mode_d%27op%C3%A9ration_%28cryptographie%29

[Web03]: http://fr.wikipedia.org/wiki/International_Data_Encryption_Algorithm

[Web04]: http://www.securiteinfo.com/cryptographie/aes.shtml

[Web05]: http://fr.wikipedia.org/wiki/R%C3%A9seau_de_Feistel

[Web06]: http://fr.wikipedia.org/wiki/C%2B%2BBuilder

[Web07]: http://fr.wikipedia.org/wiki/Photoshop

[Web08]: http://fr.wikipedia.org/wiki/Inno_Setup