TABLE DES MATIERES

ANNEXE A : LA PROGRAMMATION SOUS WINDOWS

ANNEXE B : NOTION SUR LA CRYPTOGRAPHIE

LISTES DES TABLEAUX

LISTES DES FIGURES

LISTES DES ABREVIATIONS

AM :	Amplitude Modulation
ANSI :	American National Standard Institute
ANSI X3.92 :	Norme pour le DES
API :	Application Programming Interface
BF:	Basse Fréquence
CAN :	Convertisseur Analogique Numérique
CNA :	Convertisseur Numérique Analogique
dB:	Décibels
DES:	Data Encryptions Standard
DEA:	Data Encryption Algorithm
FFT:	Fast Fourier Transform
GDI :	Graphic Device Interface
HF :	High Frequency
IBM :	International Business Machines
NIST:	National Institute of Standards and Technology
NSA:	National Security Agency
NBS:	National Bureau of Standards
RC4 :	Rivest Cipher 4
SNR :	Signal-to-noise ratio
RSA :	Rivest Shamir Adleman
TF:	Transformé de Fourier

INTRODUCTION

Dans les quelques dernières années, un besoin est apparu pour la protection des droits d'auteurs des médias électroniques. En effet, de notre jour, l'ordinateur a rendu possibles aux utilisateurs les capacités de créer et de copier facilement les contenus multimédia, et l'Internet a permis de diffuser cette information illégalement et à très faible coût.

Le cryptage audio est considéré comme étant une autre solution pour résoudre le problème de la protection des médias numériques (ou audio numériques), car il permet aux propriétaires de garder secret ses données aux mondes externe qui n'ont pas l'autorisation d'accès à ses biens.

Le but de cet article est de présenter comment on peut manipuler les données audio numériques avec la science de la cryptographie. L'analyse sera organisée comme suit. Dans la première partie, nous allons présenter la nature vibratoire du son, suivi de la méthode de numérisation des signaux aux deuxième partie, puis c'est dans la troisième partie qu'on va traiter comment on peut appliquer la cryptographie dans le domaine de l'audio numérique. Enfin, la dernière partie explique le logiciel « ***CryptSoundBuilder*** » ainsi créé après une analyse du projet.

CHAPITRE 1 : la nature vibratoire du Son

1.1. Qu'est-ce qu'un son ?

Le son est une onde produite par la vibration d'un support fluide ou solide se propageant sous forme d'ondes longitudinales [1]. Dans le cas plus simple, cette onde est une simple sinusoïde, c'est-à-dire une vibration d'équation :

$$x(t) = A.\sin(2\pi f.t - \varphi)$$

A est appelé l'*amplitude* (plus précisément amplitude maximale) du son, f sa fréquence et ϕ le déphasage de la vibration par rapport à l'origine des temps.

Un tel son est appelé un son « pur ». Dans l'air, l'amplitude correspond aux variations de pression qui caractérisent l'onde.

Les sons les plus souvent rencontrés sont rarement purs. Ils sont la somme de plusieurs sons purs, c'est-à-dire de plusieurs sinusoïdes, plus couramment appelées « harmoniques ».

On dit qu'un son est « riche » quand il contient de nombreux harmoniques (comme la parole) et « pauvre » quand il n'a que peu d'harmoniques (comme le son d'une flûte).

1.2. La propagation du son

Comme nous l'avons vu, le son est une onde, une vibration. Le son ne se propage pas dans le vide. Si une sonnerie est placée dans une cloche de verre et que le vide y est fait petit à petit, on remarquera que plus l'air se raréfie, plus le son s'atténuera jusqu'à son extinction totale.

Quand une source émet un son, la vibration se propage de particule en particule jusqu'à notre tympan qui vibre à son tour. Plus le milieu est dense plus le son se propage vite. Dans l'air le son se propage à environ 350 m/s, dans l'eau à environ 1500 m/s et à environ 5050 m/s dans l'acier. La vitesse de propagation du son dépend de la température, de la pression et surtout de la densité du milieu.

1.3. La perception humaine du son

Une vibration mécanique de la matière et de l'air qui fait vibrer notre tympan ne constitue pas en elle-même le son. C'est dans notre cerveau que le son naît et se forme. Le son n'existe pas en dehors de notre cerveau.

Entre l'arrivée des signaux vibratoires aux oreilles et la sensation de son dans le cerveau a lieu le phénomène de traitement des signaux par le système nerveux. Cela signifie que la vibration physique de l'air ne parvient pas de façon brute au cerveau. Elle est transformée.

La gamme des vibrations perceptibles est tronquée, c'est-à-dire que nous n'entendons pas les sons ni trop bas (de fréquences faibles) ni trop hauts (de fréquences élevées) même si leurs vibrations parviennent à notre oreille. Le système nerveux ne peut traiter que des vibrations dont la fréquence est comprise entre 20 Hz et 20 kHz [2]. Les sons de fréquences inférieures à 20 Hz sont appelés infrasons et ceux de fréquences supérieures à 20 kHz ultrasons.

Tout être vivant doté d'une ouïe ne peut percevoir qu'une partie du spectre sonore qui dépend de l'espèce concernée. Par exemple, le chat peut percevoir les sons de fréquence allant jusqu'à 25 kHz, le chien perçoit les sons allant jusqu'à 35 kHz, la chauve-souris et le dauphin les sons de fréquence jusqu'à 100 kHz.

En outre, l'ouïe est capable de traiter les signaux sonores de façon à n'en extraire que les informations nécessaires à notre perception de l'environnement. Par exemple, dans un environnement bruyant, un homme est capable d'extraire de façon automatique les sons qui ont un sens pour lui, comme les paroles de quelqu'un avec qui il parle. L'homme est également capable de reconnaître des formes sonores, tels que ceux produits par des instruments de musique.

1.4. Caractéristiques d'un son

a. L'intensité

L'intensité d'un son dépend directement de son amplitude. Elle caractérise ce que l'on entend par un son fort (c'est-à-dire qui tend à assourdir) ou faible (c'est-à-dire qui est presque inaudible).

L'intensité I en un point donné diminue en fonction de la distance r qui sépare ce point de la source. Elle est liée à la puissance P de l'émetteur par la formule :

$$I = \frac{P}{4\pi r^2}$$

L'intensité sonore exprime en effet la puissance de la vibration sonore reçue par unité de surface à l'endroit où l'on se trouve. Son unité est donc le W/m².

En acoustique, l'intensité est cependant exprimée en décibels (dB) pour les raisons citées ci-dessous :

- Ils permettent de travailler avec des valeurs facilement manipulables (ni « trop grandes », ni « trop petites »).
- L'oreille humaine perçoit l'intensité sonore de façon logarithmique.

Dans le standard international, on a pris comme intensité de référence I0 le seuil d'audibilité de l'oreille humaine pour un son de fréquence 1000 Hz et sa valeur est 10-12 W/m2. L'intensité acoustique (qui s'exprime en dB) est définie par :

$$L = 10 \log \frac{I}{I_0}$$

Le seuil d'audition de notre oreille se situe à 0 dB et le seuil de douleur à 120 dB. Le tableau 1.1 nous montre l'échelle de bruit pour l'oreille humaine :

Tableau 1.1 : Echelle de bruit pour l'oreille humaine

Classe du bruit	Intensité	Exemple
Sans danger pour l'audition	0 dB	Silence total
	15 dB	Bruissement des feuilles
	20 dB	Chuchotement / jardin paisible
	25 dB	Conversation à voix basse
	30 dB	Appartement dans une rue tranquille
	35 dB	Bateau à voile / Tic-tac d'une montre
	40 dB	Rue résidentielle
	50 dB	Bruit d'une voiture au ralenti
	60 dB	Grand Magasin / Sonnerie de téléphone
	70 dB	Restaurant Bruyant
	85 dB	Radio Volume à fond
Facteur de troubles auditifs	90 dB	Rue au trafic intense
	95 dB	Train passant à la gare
Pénible à entendre	100 dB	Marteau piqueur / Baladeur à fond
	105 dB	Discothèque / Concert
	110 dB	Atelier de chaudronnerie
Seuil de douleur	120 dB	Moteur d'un Boeing 747
	130 dB	Décollage d'un Boeing 747
Exige une protection auditive	140 dB	Turbo réacteur
	180 dB	Décollage d'une fusée

b. La hauteur

La hauteur d'un son est le paramètre qui permet de distinguer un son bas (ou grave) d'un son élevé (ou aigu). Elle dépend directement de la fréquence. Plus la fréquence augmente, plus le son est aigu et inversement, plus la fréquence diminue, plus le son est grave.

c. Le timbre

Le timbre est l'empreinte vocale qui permet de reconnaître la voix d'une personne, ou d'un instrument. Il est caractérisé par la fréquence des harmoniques, leur nombre, leur amplitude.

CHAPITRE 2 : Numérisation d'un signal

2.1. Introduction

L'objectif de cette partie est de mettre en place les outils mathématiques permettant de modéliser l'acquisition numérique de signaux analogiques.

Le but est de comprendre :

- Le choix de Te, période d'échantillonnage.
- Le Choix de n, nombre de bit de code.
- L'influence de l'échantillonnage sur les propriétés d'un signal.

Nous devrons garder à l'esprit le fait que l'acquisition numérique ne doit pas détériorer le signal. On doit conserver au travers de la numérisation l'information utile :

Voix : [0 ; 20kHz] ; Vidéo [0 ; 6MHz]

De plus, il faut limiter l'espace mémoire nécessaire au stockage. En effet, il faut stocker « n*Fe » bits par seconde. On s'attachera dans une chaîne d'acquisition à minimiser cette valeur tout en ne détériorant pas le signal.

2.2. Théorie de l'échantillonnage

a. Acquisition des Signaux

Pour transformer un signal analogique en un signal numérique, il faut le discrétiser. On va donc prélever régulièrement des échantillons du signal analogique pour le rendre discret et permettre ainsi sa numérisation :

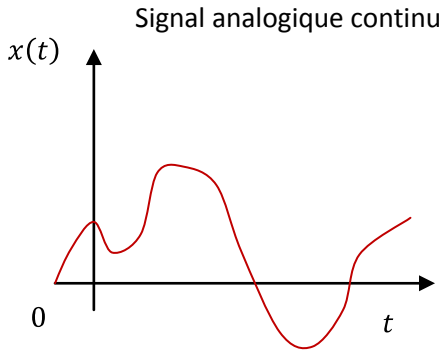


Figure 2.1 : Allure d'un signal analogique continu $x(t)$

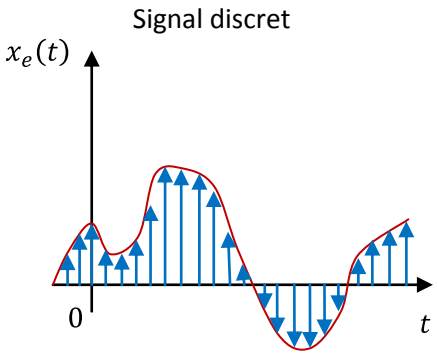


Figure 2.2 : Allure d'un signal échantillonné $x_e(t)$

On prend ainsi des valeurs de e(t) à des intervalles de temps régulier (tous les Te, période d'échantillonnage) à une fréquence Fe dite fréquence d'échantillonnage, que l'on déterminera par la suite. Suite à cet échantillonnage, on quantifie chaque échantillon par une valeur binaire pour la stocker sur un support numérique.

b. Modélisation de l'échantillonnage

L'opération mathématique associée à cette discrétisation revient à multiplier le signal e(t) par un peigne de Dirac $\delta_{Te}(t)$:

$$e^*(t) = e(t).\delta_{Te}(t) = e(t).\sum \delta(t - nTe)$$

On peut ainsi calculer la transformée de Fourier du signal échantillonné en utilisant les propriétés liant une multiplication temporelle qui dans l'espace fréquentiel devient un produit de convolution :

$$E^*(f) = TF\big(e(t).P_{Te}(t)\big) \rightarrow E^*(f) = \frac{1}{Te}E(f) * \delta_{fe=\frac{1}{Te}}(f)$$

Soit

$$E^*(f) = \frac{1}{Te}\sum_{k=-\infty}^{+\infty} E(f - k.f_e)$$

Échantillonner le signal e(t) dans le domaine temporel, revient donc à recopier dans le domaine fréquentiel son spectre E(f) tous les Fe.

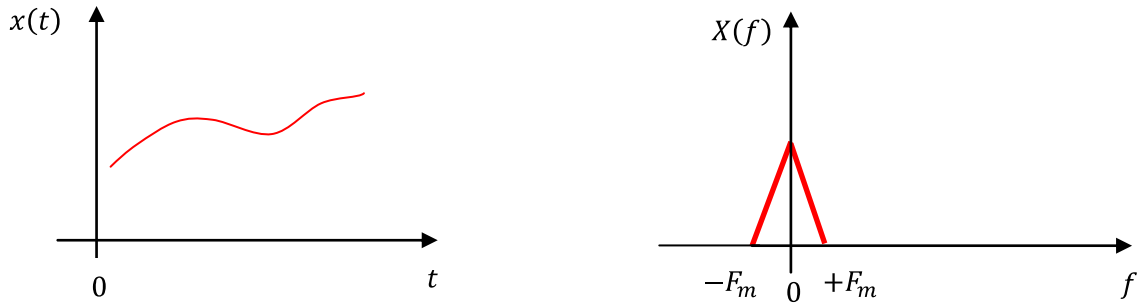


Figure 2.3 : Propriétés temporelles et fréquentielles du signal d'entrée

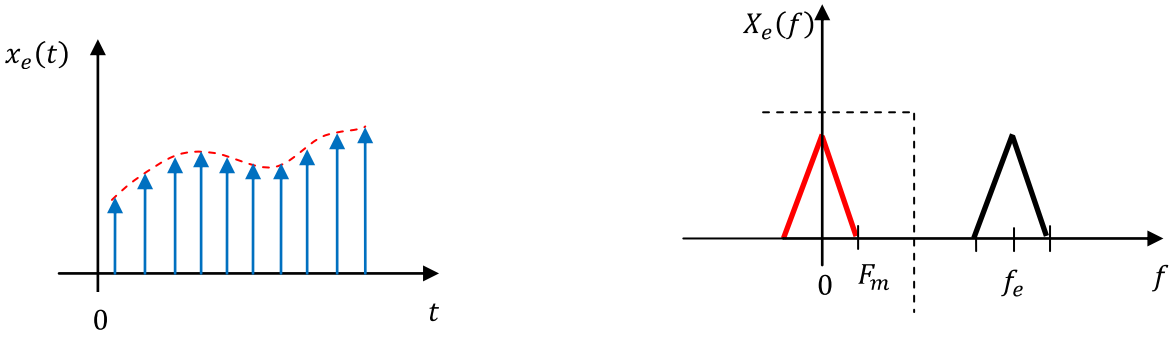


Figure 2.4 : Propriétés temporelles et fréquentielles du signal échantillonné

c. Notion de repliement de spectre

On remarquera que si le spectre du signal d'origine à une largeur supérieur à $2F_e$ on a ce qu'on appelle ***un repliement de spectre.*** Et le signal échantillonné devient comme illustré dans la Fig. 3.4.

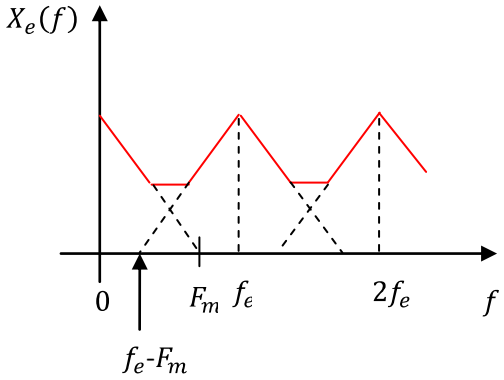


Figure 2.5 : Echantillonnage provoquant le repliement de spectre

S'il y a repliement de spectre, il n'est plus possible de retrouver le spectre du signal d'origine. Dans ce cas, l'opération d'échantillonnage modifie les caractéristiques du signal d'entrée.

Ainsi, si l'on ne veut pas perdre d'informations par rapport au signal que l'on échantillonne, on devra toujours respecter la condition : ($Fe \geq 2F_m$). Condition plus connue par le théorème de Shannon.

d. Théorème de Shannon

On ne peut échantillonner un signal sans pertes d'informations que si : $F_e > 2F_m$

Note : *Rôle du filtre d'entrée*

Dans le cas d'un spectre de largeur infinie (la réalité), il y a donc toujours repliement de spectre. Il est donc nécessaire de filtrer le signal d'origine afin de limiter cet effet de repliement.

Par exemple, dans le cadre de l'audio, on ne va garder que les fréquences que l'oreille est capable d'entendre. Les caractéristiques internes de l'oreille induisent une sensibilité fréquentielle pouvant aller de 20hz à 20khz. C'est pour cette raison que l'on a pris comme fréquence d'échantillonnage F_e =44,1 kHz dans le cas du CD.

Ainsi, avant d'échantillonner le signal, on place en amont un filtre qui a pour but d'éliminer toutes les fréquences supérieures à 20khz. C'est un filtre passe bas.

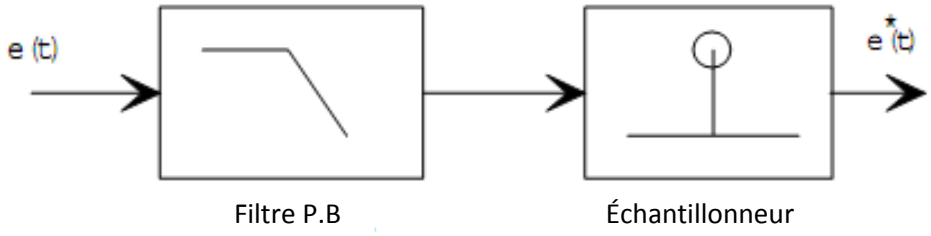


Figure 2.6: Utilisation du filtre en amont de l'échantillonneur

e. L'échantillonnage blocage

Une fois le signal filtré et échantillonné, il reste à le quantifier. Pour pouvoir réaliser cette fonction, on doit maintenir constant la valeur à quantifier afin de permettre au CAN de traiter l'échantillon et de le numériser. On appelle cette opération, le blocage. Ce blocage doit être d'une durée supérieure au temps de conversion :

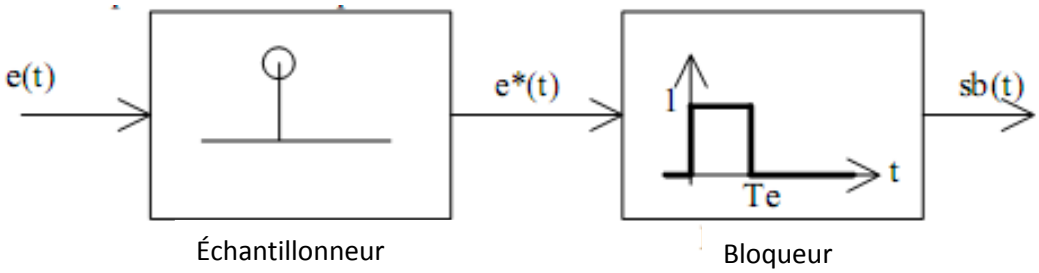


Figure 2.7: Association d'un bloqueur à l'échantillonneur

f. Modélisation de l'échantillonneur bloqueur

On suppose le blocage d'une durée θT_e où $\theta \in]0 ; 1]$. L'opération mathématique associée est la convolution du signal échantillonné e*(t) avec un rectangle de durée T_e:

$$S_b(t) = e^*(t) * Rect_{\theta Te}(t)$$

Ce qui alors pour le spectre, revient à le multiplier par un sinus cardinal :

$$S_b(f) = \lambda E^*(f).sinc(\theta \frac{f}{f_e})$$

On note que dans le cas d'un blocage de faible durée (θ<1), le sinus-cardinal atténue eu les premières recopies de spectre. Un filtre passe-bas avec une fréquence de coupure à Fe/2 permettrait de récupérer de manière parfaite le signal d'entrée.

Dans le cas d'un signal bloqué sur toute la période d'échantillonnage (ce qui correspond en fait au signal restitué en sortie d'un CNA), le sinus-cardinal écrase les fréquences proches de la fréquence d'échantillonnage et vient donc modifier les propriétés du spectre du signal d'entrée qui ne peut plus être restitué de manière parfaite à l'aide d'un simple filtre. Par contre, il présente l'avantage d'éliminer les recopies de spectre et donc d'alléger le contenu spectral du signal.

g. Nécessité du filtre d'anti-repliement

i) Caractéristiques idéales

Avant de réaliser l'échantillonnage du signal, nous avons vu la nécessité de filtrer ce dernier afin d'éviter ce que l'on appelle le repliement de spectre, plus connu sous la forme du théorème de Shannon.

Idéalisé, il doit avoir un gain de 1 sur une bande de fréquence Fe, centrée en zéro. Son rôle va être de limiter le contenu spectral du signal à la partie utile. Il va participer aussi à limiter l'influence du bruit éventuellement présent sur le signal à numériser.

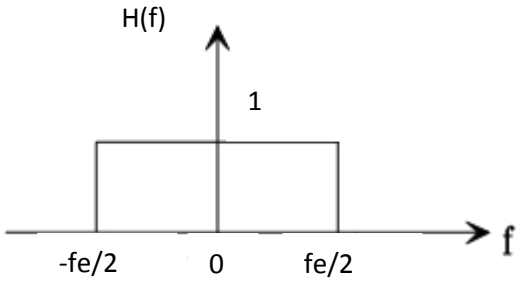


Figure 2.8: Gabarit idéal du filtre anti-repliement

ii) Filtre réel

De manière idéale, un filtre passe bas aura un gain constant dans la bande passante, et présentera une coupure infinie au-delà de sa fréquence Fc de coupure.

De manière réelle, on est amené à réaliser la synthèse d'un filtre en définissant sa fréquence de coupure à -3db ainsi qu'une atténuation minimum au-delà d'une certaine fréquence.

On fait en général appel, dans le cadre d'un filtre anti-repliement, à un filtre du type Butterworth. :

$$H(f) = \frac{1}{\sqrt{(1 + \frac{f^{2n}}{f_c})}}$$

Ce type de filtre présente l'avantage de répondre au critère de maximum de platitude dans la bande passante et de présenter un retard de groupe constant jusqu'à fc/2.

Le choix de l'ordre du filtre s'effectue de manière à limiter la puissance du signal dû aux recopies de spectre. On limite donc le recouvrement de spectre en termes de puissance ramenée par rapport à la puissance du signal :

$$\frac{P_{recouvrement}}{P_{signal}} \leq X\%$$

Si l'on suppose un signal à spectre constant, et que l'on admet une puissance ramenée d'au plus 1%, nous pouvons établir en fonction de n, la valeur de la fréquence de coupure du filtre :

Tableau 2.1 : la valeur de la fréquence de coupure en fonction de n

n	1	2	4	6
Fc	Fe/127	Fe/6	Fe/3	Fe/2

Remarque : Problème lié au retard de groupe

Tout filtre introduit un déphasage qui peut introduire une distorsion dans le cadre d'un signal multifréquence (cas de l'audio) :

$$V_e = Vcos(\omega t) \Longrightarrow V_s = Vcos(\omega t - \varphi) = Vcos(\omega(t - \frac{\varphi}{\omega}))$$

Ainsi un signal en sortie d'un filtre ressort avec un retard Tr : $T_r = \frac{\varphi}{\omega}$

Si ce retard n'est pas constant pour toutes les fréquences (déphasage linéaire avec la fréquence), on obtient alors une distorsion. Deux signaux synchrones, en entrée du filtre, ressortent désynchronisés :

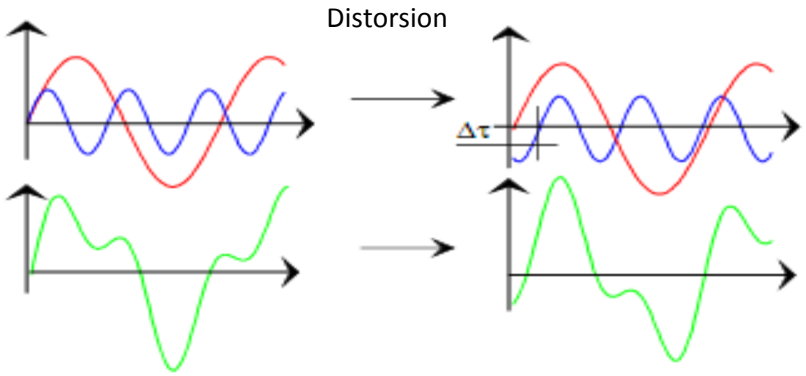


Figure 2.9: Distorsion introduite par un filtre à déphasage non linéaire

On peut très brièvement résumer la propriété de certains filtres :

- Un filtre dit de Bessel assurera un temps retard de groupe constant dans la bande passante, mais une atténuation lente.
- Un filtre de Tchebychev donnera une atténuation rapide mais par contre on aura de l'ondulation dans la bande passante et un très mauvais retard de groupe.
- Un filtre de Butterworth est un très bon compromis, il assure une réponse plate avec un retard de groupe constant pour les fréquences inférieures à $\frac{f_c}{2}$.

2.3. Théorie de la quantification

Le signal échantillonné - bloqué peut à ce stade être converti sous forme binaire (numérique) pour être stocké. Ce codage s'appelle la quantification.

Le rôle de la quantification est de donner une image binaire d'un signal analogique :

Passage Analogique –> Numérique

Signal Continu –> Signal discret

Tension –> chiffre

a. Principe

A chaque niveau de tension est associée une valeur binaire codée sur n bits:

N bits vont permettre de distinguer 2n niveaux de tension répartis de $-V_m$ à$+V_m$. On a ainsi un pas de quantification : q = $\frac{2V_m}{2^n}$

Ainsi un signal de ±5V codé sur 8 bits donnera un pas de quantification q=39mv.

La caractéristique d'entrée – sortie d'un CAN est une caractéristique en marche d'escalier. Chaque palier a une largeur d'un pas de quantification q. Le passage d'un palier à un autre correspond à une variation de « 1 » du code.

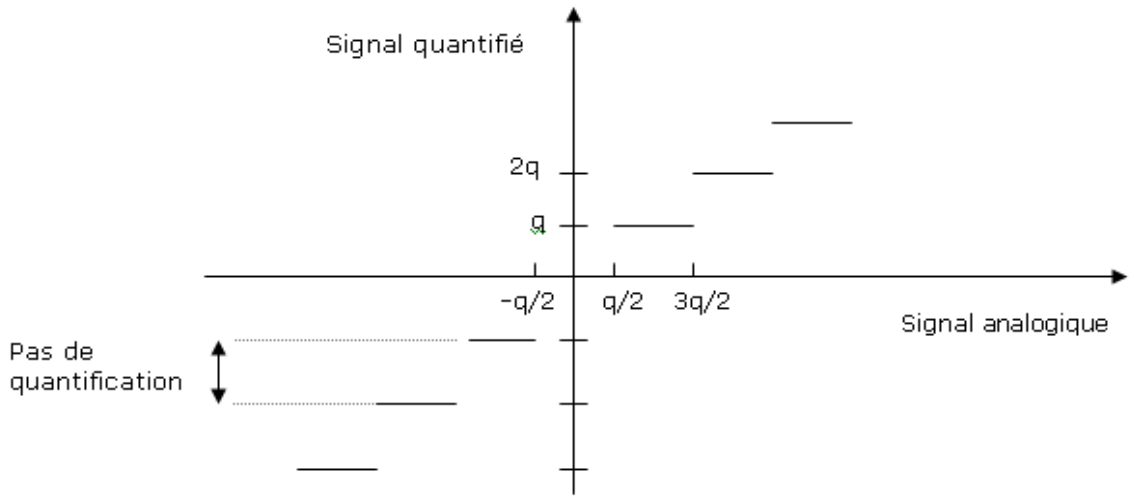


Figure 2.10: Caractéristique entrée-sortie d'un CAN

Le pas de quantification est aussi appelé **quantum**. Il correspond à la résolution du convertisseur. Le quantum est la plus petite variation de tension que le convertisseur peut coder.

b. Bruit de quantification

On a donc, lors de la quantification, une erreur de codage entre le signal échantillonné et la valeur du code correspondant à un niveau de tension (ce niveau de tension étant la moyenne des tensions correspondant à ce code).

Gamme de tension -> Code unique

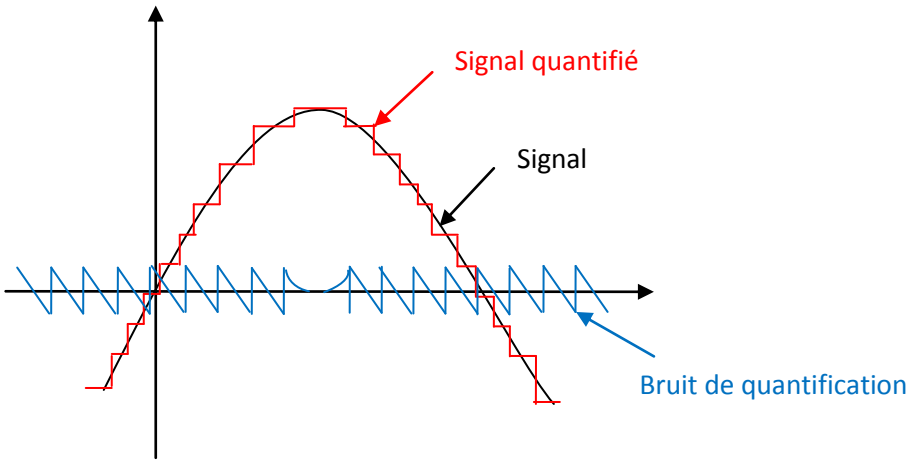


Figure 2.11 : Allure du bruit de quantification

L'évolution du bruit de quantification est une évolution en dent de scie avec une amplitude égale au quantum. En fonction du principe de quantification utilisé, les caractéristiques du bruit de quantification varient.

c. Caractéristiques du bruit de quantification

i) Quantification linéaire par défaut

Dans ce cas, le signal variant de 0 à E, on code les 2^n niveaux de tension avec un pas de quantification : $q = \frac{E}{2^n}$ On obtient une codification du signal d'entrée telle que :

$$[nq\ ;\ (n+1)q] \rightarrow nq$$

L'erreur de quantification évolue alors entre 0 et q.

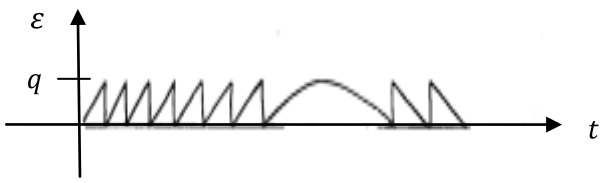


Figure 2.12: Erreur de quantification par défaut

Si l'on suppose que le signal est de répartition continue, avec un écart type supérieur à quelques quantum, on peut admettre que l'évolution du bruit de quantification ε est en dent de scie. On peut ainsi calculer sa puissance en termes de moyenne quadratique (elle correspond à la puissance du signal dans une résistance de 1Ω) :

$$P_\varepsilon = \langle \varepsilon^2(t) \rangle = \frac{q^2}{3}$$

ii) Quantification linéaire centrée

Dans la pratique, on préfère effectuer une quantification centrée. Dans ce cas, le bruit de quantification évolue entre ± q/2 :

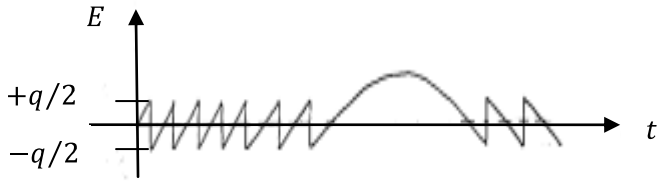


Figure 2.13: Erreur de quantification linéaire centrée

Le signal d'entrée est donc codé tel que :

$$\left[(2n-1)\frac{q}{2}; (2n+1)\frac{q}{2}\right] \longrightarrow nq$$

La puissance du bruit de quantification que l'on obtient est :

$$P_\varepsilon = \langle \varepsilon^2(t) \rangle = \frac{q^2}{12}$$

On notera l'avantage du codage centré par rapport à un codage par défaut. Le bruit de quantification est plus faible.

iii) Quantification non linéaire

L'intérêt des techniques de quantification non linéaires réside dans le fait qu'elles permettent de coder de manière plus précise les valeurs qui apparaissent plus souvent. Dans la plupart des applications, on utilise la technique de quantification linéaire centrée mais dans certains cas particuliers, il peut s'avérer intéressant d'opter pour une technique plus adaptée (donc forcément non linéaire).

d. Choix du nombre de bits de quantification

i) Choix classique

Dans le cadre d'une simple acquisition, on peut se contenter de choisir 'n' vis à vis de la résolution souhaitée :

Tableau 2.2: Un extrait de tableau de résolution en fonction de n

n	Résolution	1 quantum (%)
8	1/256	0.391
10	1/1024	0.0977
12	1/4096	0.0244
14	1/16384	0.0061

ii) Prise en compte du rapport signal sur bruit

Dans le cadre d'une acquisition - restitution, ce qui est le cas pour l'audio numérique, on va choisir le nombre de bits de codage par rapport au rapport signal sur bruit :

$$SNR_{dB} = 10\log\frac{\mathrm{P_{signal}}}{\mathrm{P_{bruit}}}$$

Les puissances sont ici calculées vis à vis d'une charge de 1Ω. Elles correspondent à la moyenne quadratique du signal :

$$P_v = \langle v^2(t) \rangle$$

Ainsi, dans le cas d'un signal sinusoïdal parcourant la pleine échelle du convertisseur, nous obtenons avec une quantification linéaire centrée un rapport signal sur bruit :

$$SNR_{dB} = 6n + 1.76\ db$$

Ce qui signifie qu'un bit de code rajoute 6dB de rapport signal sur bruit.

Dans le cadre du Compact Disc, la prise en compte de la physiologie de l'oreille fait apparaître un masquage sonore entre deux sons s'ils sont espacés de plus de 40dB. De plus, les dynamiques musicales (Type Opéra) sont d'environ 40 dB. Il faut donc un SNR d'au moins 80dB pour effectuer un enregistrement Haute Fidélité.

Un codage sur 14 bits suffit (85.76 dB de SNR). On a utilisé un code sur 16 car cela représente 2 octets, ce qui d'un point de vue informatique est plus simple à gérer. On a donc pour le C.D. un enregistrement qui est effectué avec un SNR de 96dB.

Remarque : pour un signal sinusoïdal d'amplitude V_{sin} (inférieur à la pleine amplitude E), le calcul du SNR donne :

$$SNR_{dB} = 6n + 1.76_{db} + 20\log\left(\frac{Vsin}{E}\right)$$

CHAPITRE 3 : Application de la Cryptographie dans le domaine de l'Audio Numérique

3.1. Principe du codage audio

a. Synoptique du cryptage

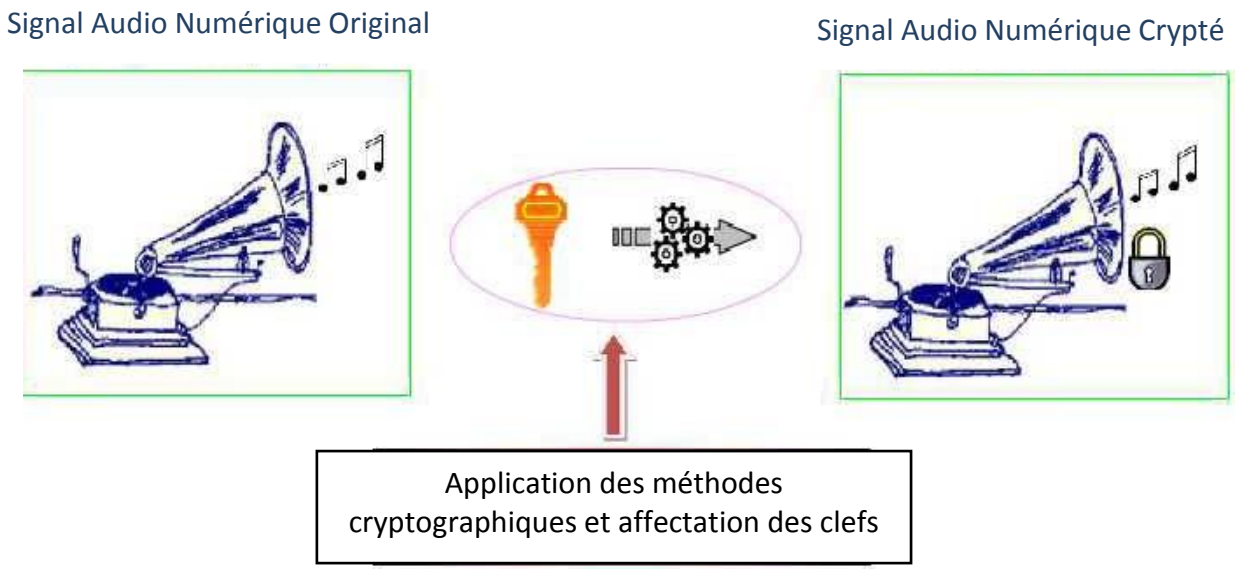


Figure 3.1 : Chaîne du cryptage Audio Numérique

b. Principes

i) Choix de la méthode de cryptage

Malgré les divers types de méthodes cryptographiques utilisés dans le domaine de la cryptographie pour le chiffrement ou le déchiffrement des informations [3], la méthode de cryptage utilisée dans le domaine Audio Numérique représentée dans cet ouvrage est classée à part. En effet, ici il n'est pas nécessaire d'utiliser les méthodes de chiffrement par bloc, par chaîne ou autre [3], car de la Transformée de Fourier apportée par la théorie du Traitement numérique de signal [4] on peut visualiser le spectre d'un Signal Audio, avec des notions complémentaires de programmation quelques que soient les langages que vous connaissez.

Ainsi, on peut baser notre recherche de la méthode de cryptage audio [5] en étudiant directement le spectre du signal. D'où la méthode qu'on a choisie ici consiste en

une inversion de spectre du signal audio par le principe de *Modulation d'Amplitude* au tour de 12.8khz.

ii) Clef utilisée pour le chiffrement

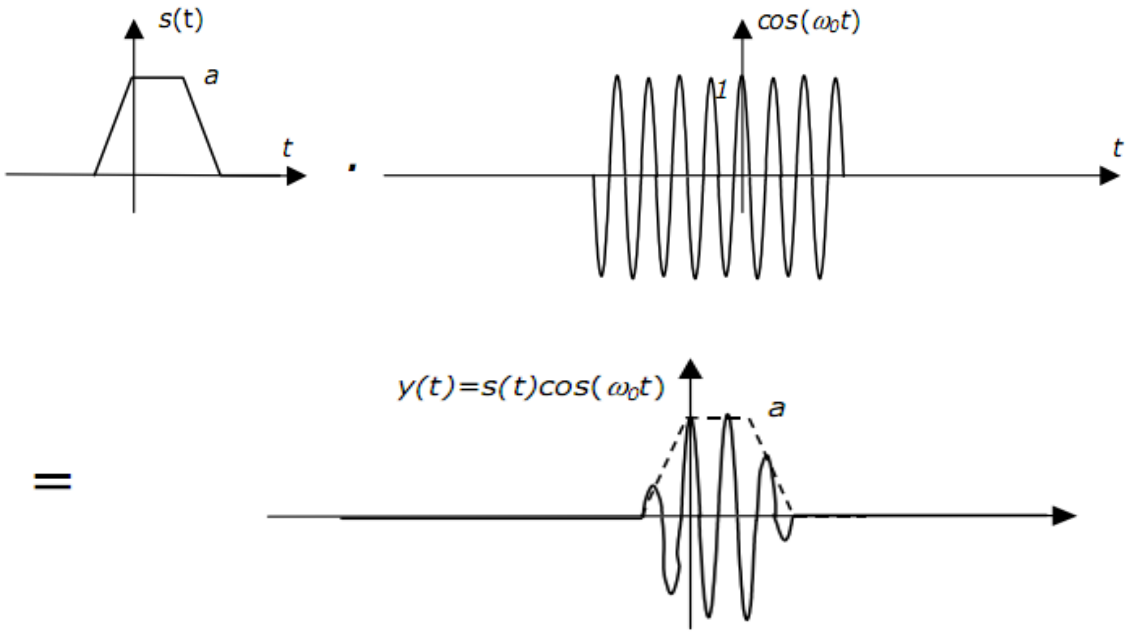
Après le choix de la méthode du cryptage, il est nécessaire de spécifier la clef de chiffrement. Pour en savoir de plus, intéressons nous à l'étude approfondie de la MA.

Rappelons par un exemple théorique ce qu'est un signal modulé en Amplitude. Soit s(t) un signal réel, de spectre limité par f_{bb} et $f_{bh,}$ sans composante continue, et $\cos(\omega_0 t)$ une onde périodique réelle de pulsation fondamentale ω_0 ($\omega_0 = 2\pi f_0$).

Le spectre du signal $s(t)\cos(\omega_0 t)$ résulte de la translation, sur l'axe des pulsations, de $\pm\omega_0$ du spectre de s(t) comme indique la figure 3.2.

Analyse temporel

Signal s(t) multiplié par le signal cos (ω_0t)



Convolution de S(f) avec TF [cos (ω_0t)]

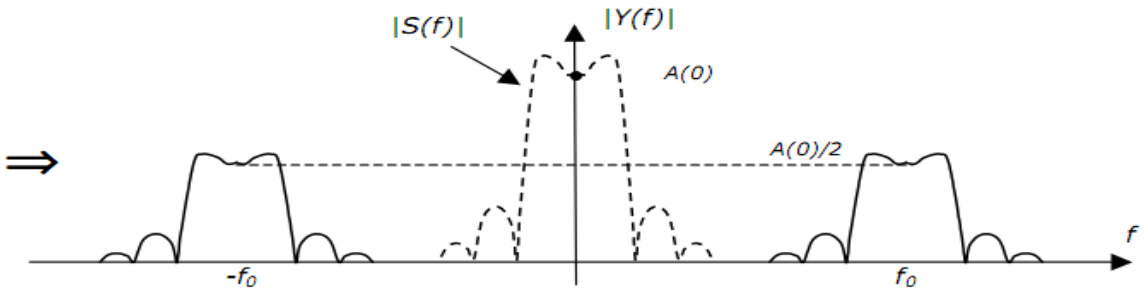


Figure 3.2 : Modulation d'amplitude

En effet le développement en séries de Fourier de cos (ω_0t) comprend deux termes d'amplitude ½ (figure 3.3):

$$\cos(\omega_0 t) \overset{F}{\Leftrightarrow} \frac{1}{2}[\delta(f - f_0) + \delta(f + f_0)]$$

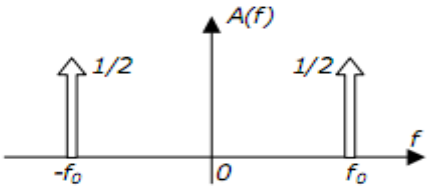


Figure 3.3 : TF d'un cosinus

Le produit s(t).cos (ω_0t) correspond à la convolution des transformés de Fourier de s(t) et cos (ω_0t) [8], et que la convolution d'un signal par une impulsion de Dirac correspond au déplacement de ce signal au droit de l'impulsion, et à une multiplication par son poids.

Ainsi, la clef du chiffrement utilisée pour le cryptage audio est un signal de type bcos(2π12800t + Φ) dont le fonctionnement est décrit ci après avec la méthode de la Modulation d'Amplitude .

Le principe de la Modulation d'Amplitude pour le cryptage audio consiste ici à multiplier le signal non codé en bande de base par un signal de type

« bcos(2π12800t + Φ) »

Dans l'exemple théorique donnée à la section précédente, les spectres translatés ne se recouvrent pas car on a supposé que $f_0 > f_{bh}$. Cette condition n'est visiblement pas vérifiée dans le cas du codage utilisé pour l'Audio Numérique. En effet le spectre d'un signal audible s'étend jusqu'à 20 kHz. Pour qu'il n'y ait pas de recouvrement des 2 spectres translatés, on procède d'abord à un filtrage passe bas (dont la fréquence de coupure est choisie précisément à 12.8 kHz) du signal non codé en bande de base. On procède ensuite à la modulation.

iii) Visualisation spectrale du mécanisme

Ici nous allons appliquer la méthode de la Modulation d'Amplitude pour le cryptage d'un signal réel s(t) qui sera filtré puis modulé.

Processus du cryptage vu spectral

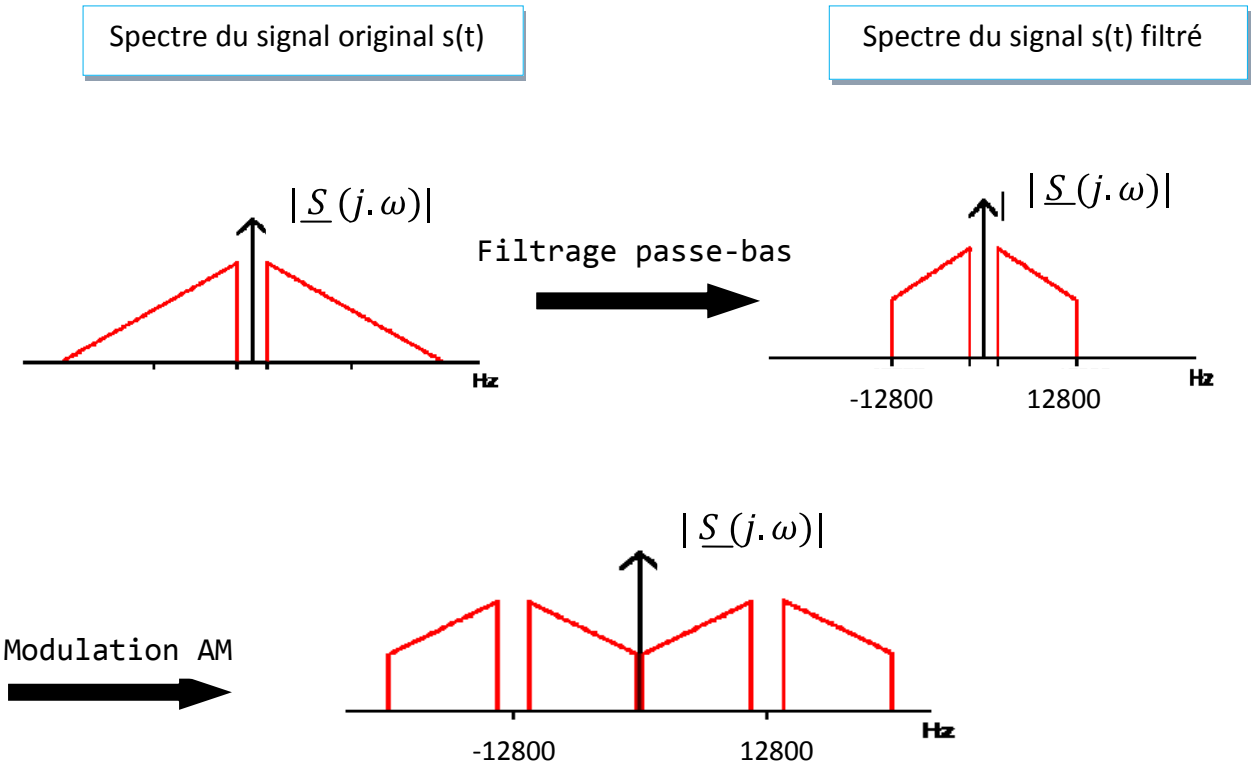


Figure 3.4 : Processus de cryptage d'un signal s(t) par Modulation d'Amplitude

iv) Résultat du cryptage avec un Audio Numérique

Le résultat qu'on devrait avoir pour le cryptage Audio est le même que ce obtenu avec le cryptage d'un signal réel quelconque étudié ci-dessus car le signal Audio Numérique est un signal réel.

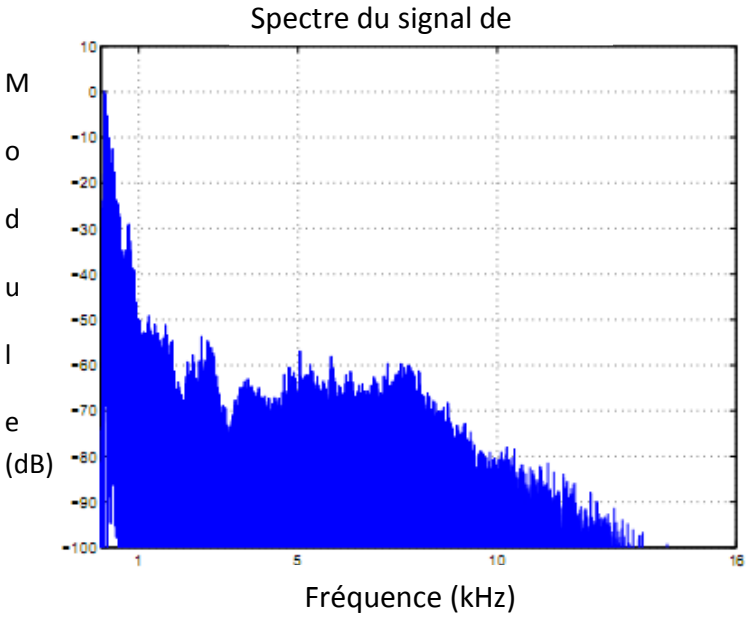


Figure 3.5 : Spectre d'un des extraits Audio

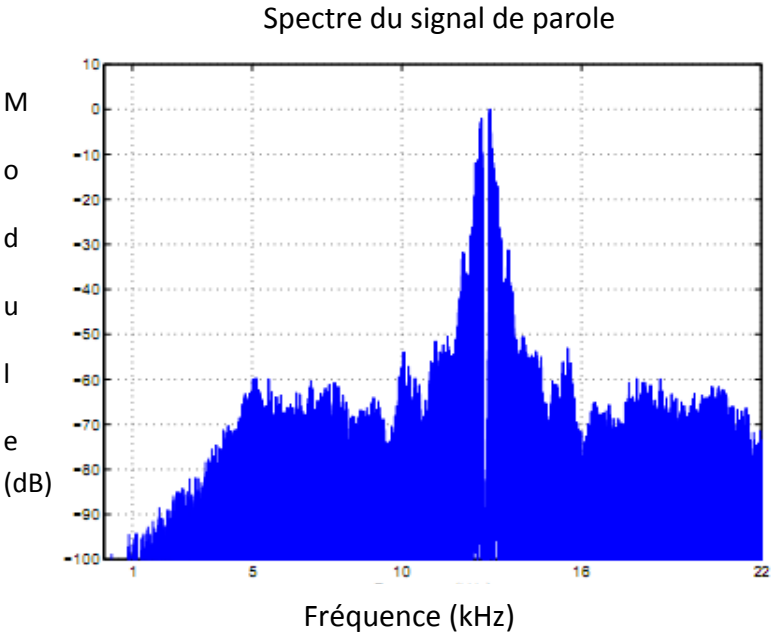


Figure 3.6 : Spectre de l'extrait Modulé

3.2. Principe du décodage audio

a. Synoptique du décryptage

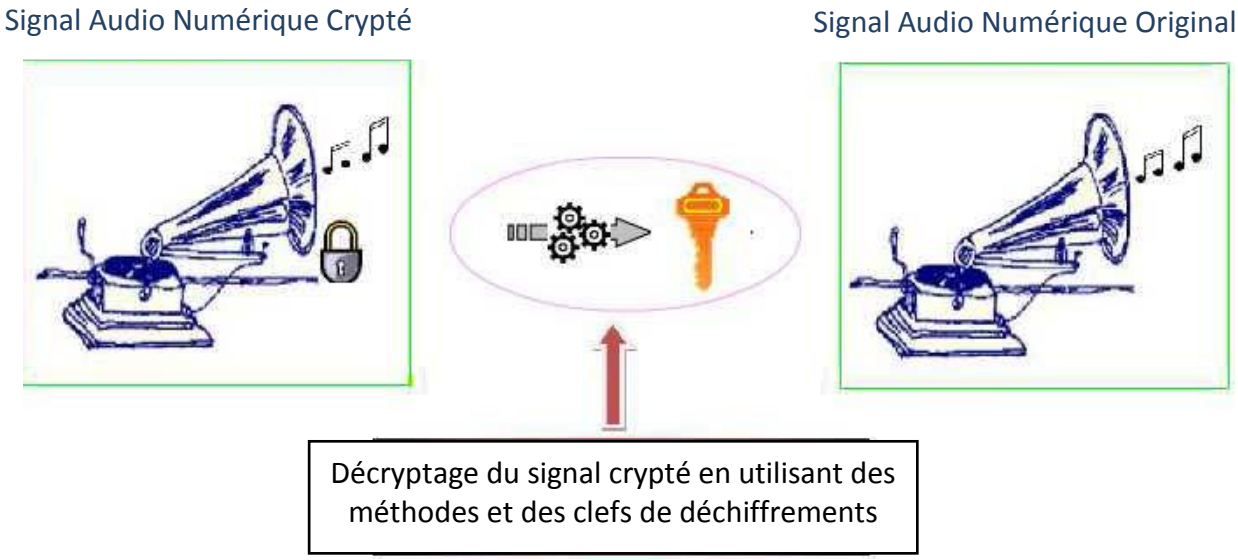


Figure 3.7 : Chaîne du décryptage Audio Numérique

b . Principes

i) Méthode de décryptage

Ici le décryptage consiste à démoduler le signal crypté pour retrouver le signal initial. La modulation est alors appliquée une nouvelle fois pour inverser le spectre, mais avant toute chose, on doit filtrer passe-bas le signal ainsi obtenu pour ne conserver que la partie nécessaire du spectre comme on peut le voir sur le spectre du signal modulé de la *figure 3.8.*

Après avoir démodulé le signal, on doit appliquer un nouveau filtrage passe-bas [5] pour ne conserver que la partie du spectre de fréquence inferieure à la fréquence de modulation de 12800Hz.

ii) Clef de déchiffrement

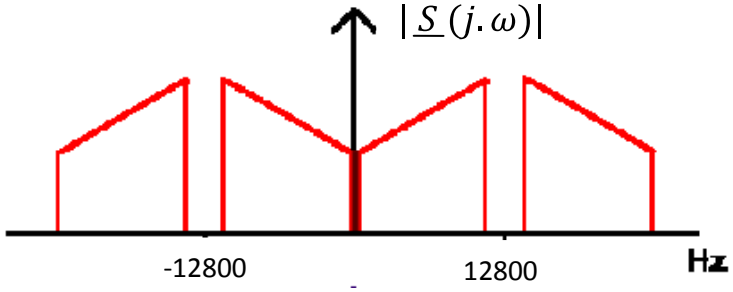
Comme la méthode de déchiffrement utilisée est la Modulation d' Amplitude, alors la clef du décryptage [3] est la même que celle utilisée pour le chiffrement. Le signal crypté qui est réel, est ainsi multiplié par un signal sinusoïdal de fréquence 12800Hz.

iii) Spectral du résultat

Voyons d'abord le principe de ce décryptage avec un signal réel crypté quelconque s(t) avant de l'appliquer au signal Audio Numérique.

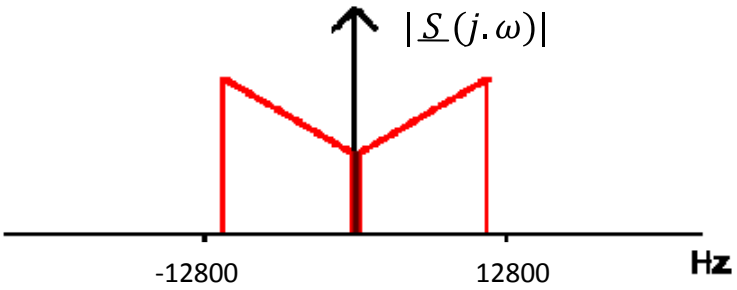
Spectre du signal s(t) Crypté

Cryptage par MA



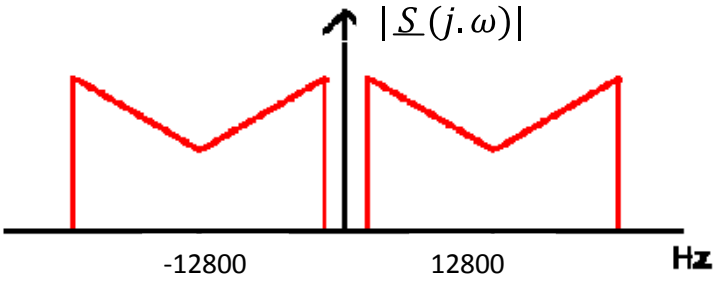
Spectre du signal s(t) Filtré

Filtrage passe-bas



Spectre du signal s(t) Modulé en Amplitude

Modulation AM



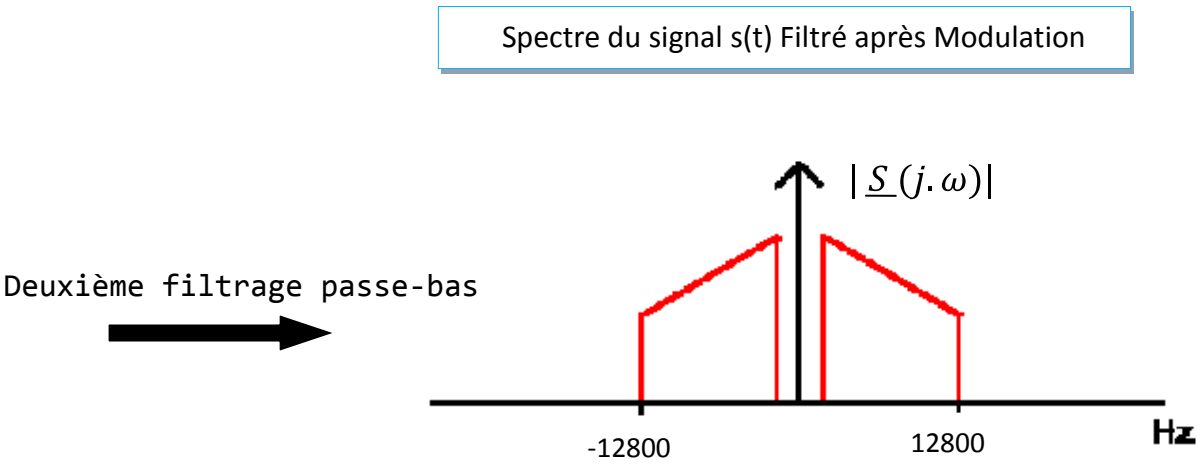


Figure 3.8 : Processus de décryptage d'un signal crypté s(t)

Cette méthode appliquée au décodage d'un signal Audio Numérique nous donne le résultat spectral ci-dessous.

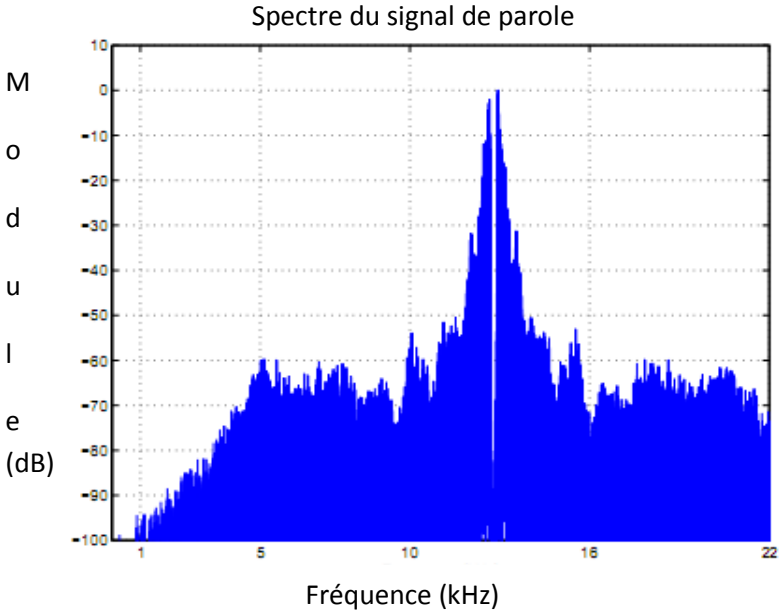


Figure 3.9 : Spectre d'un extrait audio crypté

Apres le filtrage passe-bas, une Modulation d'Amplitude et un deuxième filtrage passe-bas [2], on obtient le signal Audio Numérique original présenté par la figure 3.10.

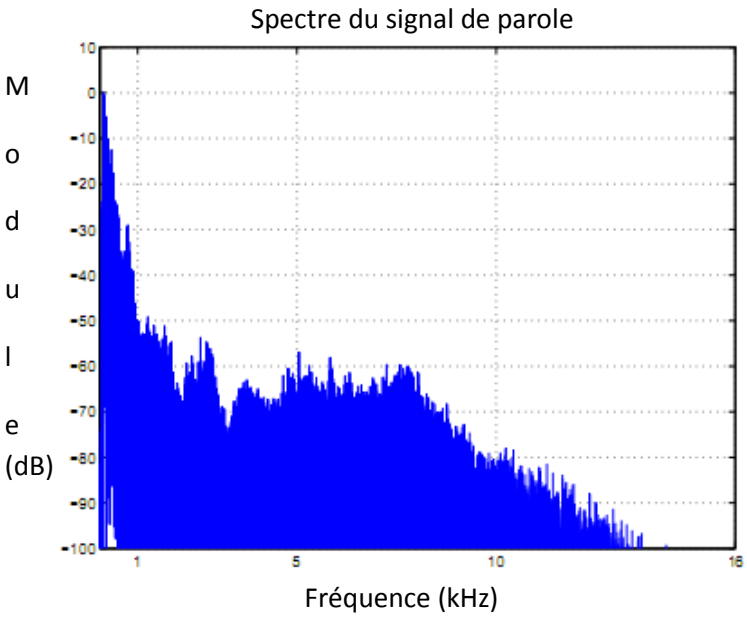


Figure 3.10 : Spectre de l'extrait du signal Audio Numérique original

CHAPITRE 4 : REALISATION

4.1. Point Visé

Le logiciel *« **CryptSoundBuilder** »* a été développé dans des buts éducatifs. En effet, il a pu associer trois domaines les plus en vogues de notre époque, telles que la cryptographie, la théorie du signal et le traitement sonore. Ainsi, non seulement on peut avoir touts les contrôles sur les informations multimédias mais on a résolu aussi le système de sécurisation des données audio numériques lors de ses passage dans un réseau (LAN ou Internet) qui n'est pas à l'abri des pirates voulant enrichir ses poches.

4.2. Le Cœur du « CryptSoundBuilder »

a. Présentation

*« **CryptSoundBuilder** »* hérite quelques applications offertes par *« **SoundBuilder** »* [6]. C'est un éditeur de sons au format wave et les fonctions de lecture et d'enregistrement des signaux audio (*entrée et sortie*) restent inchangées. Seulement il est équipé d'autres applications plus évolutives qui seront adaptés à d'autres types de format des fichiers multimédia.

Cette version intègre :

- Une application de chargement d'un fichier audio outre que le fichier wave (mp3, wma) avec visualisation spectrale.
- Des fonctions permettants de *crypter* et de *décrypter* des fichiers wave
- Une fonction de filtrage audio
- Une application permettant d'éditer les caractéristiques d'un fichier wave

Enfin, un équaliseur audio.

b. Programmation

« CryptSound Builder » a été écrit en langage C avec Visual Studio 2005. Il utilise :

- La GDI (cf. Annexe A) pour l'interface utilisateur
- Windows MultiMedia pour l'interface avec les périphériques audio

- L'algorithme de Cooley -Tukey pour le calcul de la FFT

Il est aussi réalisé avec quelques fonctions du flash macromedia qui a permis la réalisation de son interface.

c. Présentation Générale

i) Interface

L'interface du « ***CryptSound Builder*** » a été réalisée avec l'outil du *Macromedia Flash player.* Elle contient 2 boutons ***Designer*** et ***CryptSound*** qui affiche respectivement le designer du logiciel et l'application qui permet de crypter et de décrypter un fichier audio (figure 3.2 et 3.3).

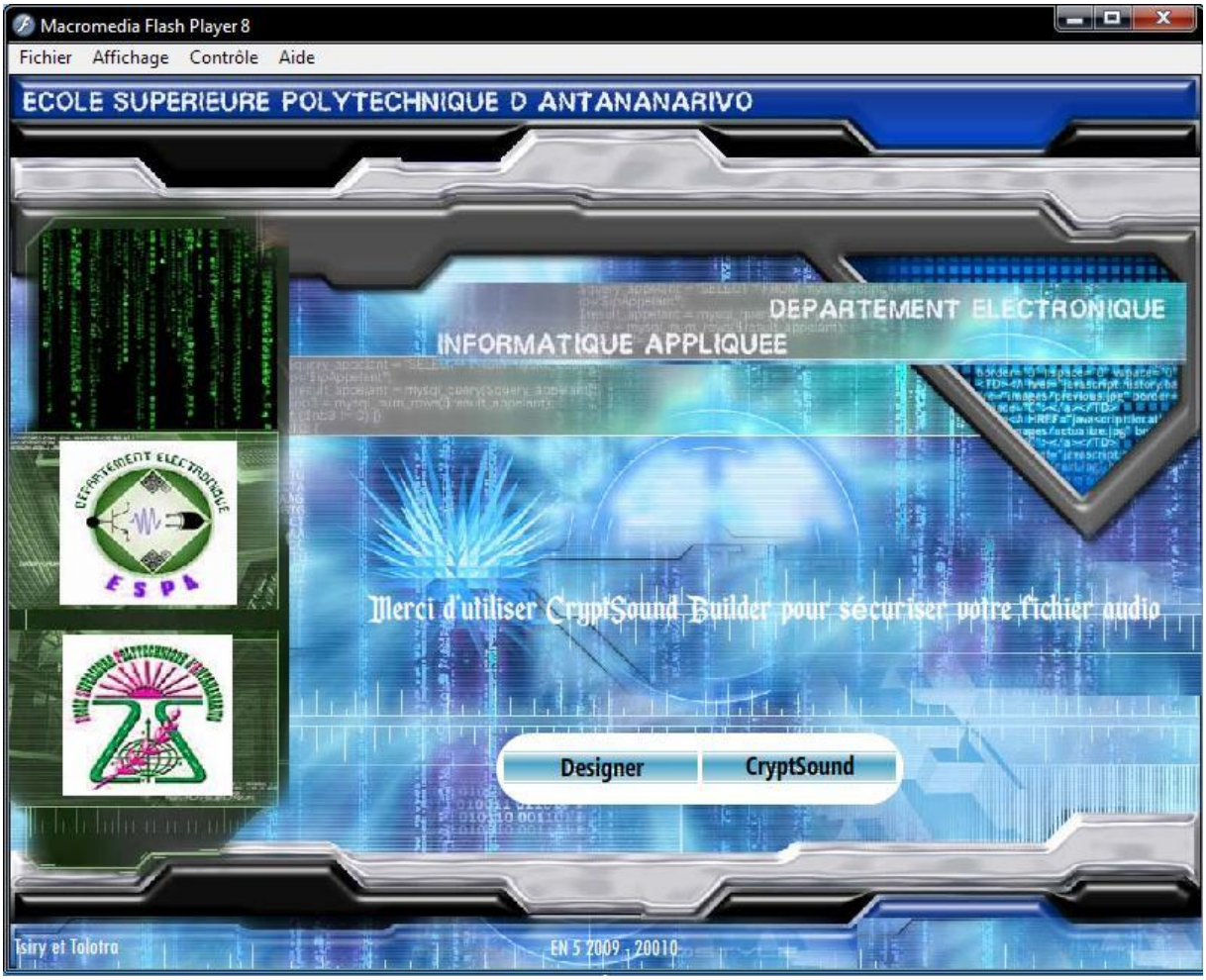


Figure 4.1: Interface initiale du *"CryptSound Builder"*

Figure 4.2: Effet du bouton « Designer »

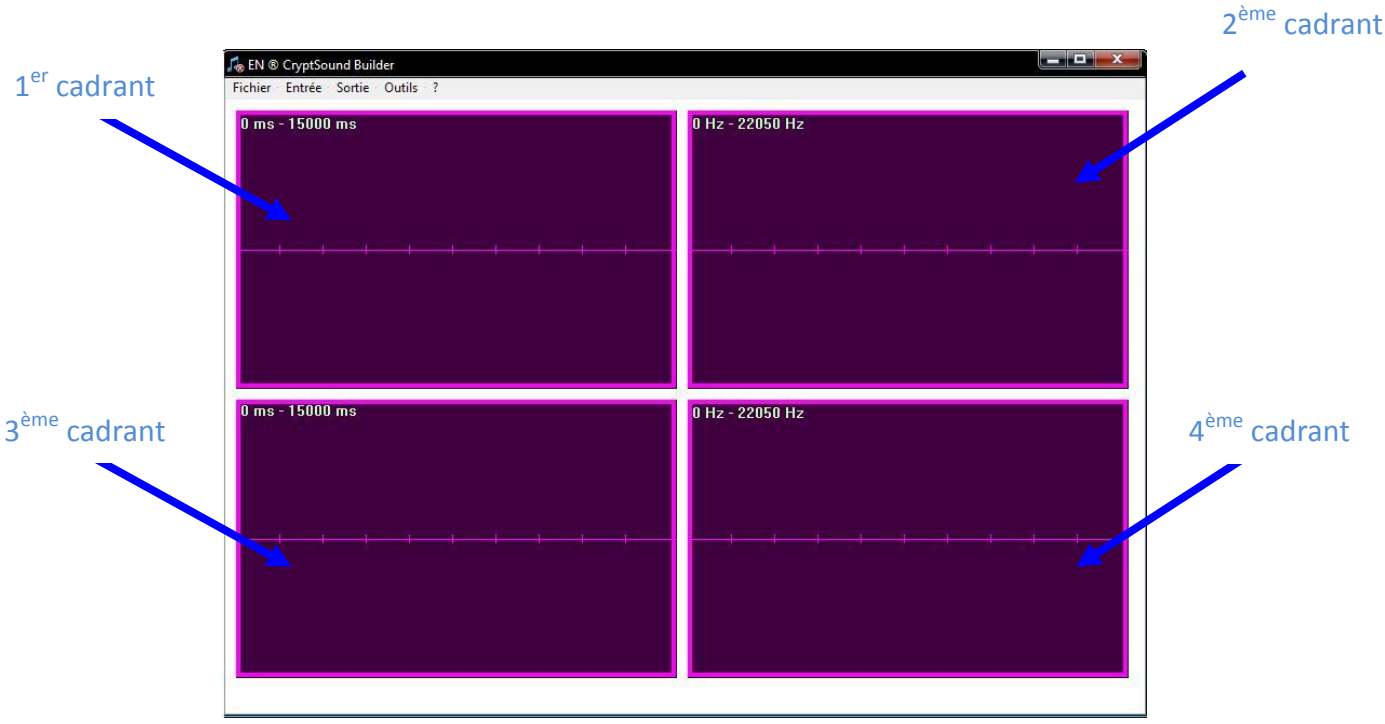


Figure 4.3: Effet du bouton « CryptSound »

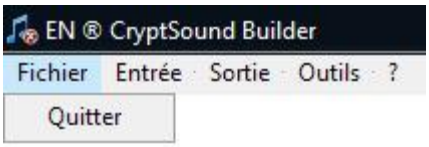
ii) Menus de navigations

Cette version du logiciel possède aussi les mêmes menus que ceux du SoundBuilder. On y trouve les menus : *Fichier, Entrée, Sortie, Outils et ?.*

Figure 4.4 : Les menus du « *CryptSound Builder* »

Ainsi chaque menu contient des sous menus suivants :

Menu *Fichier*



Un sous menu ***Quitter***, pour terminer et quitter l'application

Figure 4.5: Le sous menu *Quitter*

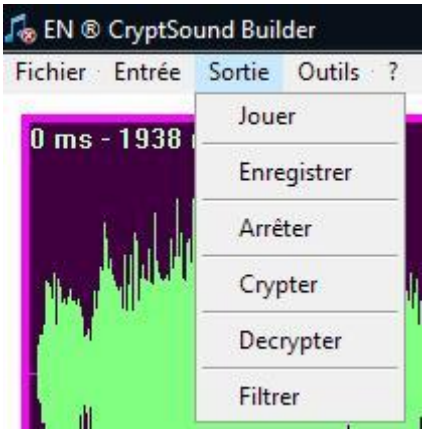
Menu *Entrée*

Trois sous menus :

- ***Choisir*** : pour l'ouverture d'un fichier wave
- ***Jouer*** : pour la lecture du signal audio (entrée)
- ***Arrêter*** : pour stopper la lecture

Figure 4.6 : Les sous menus *Choisir – Jouer - Arrêter*

Menu *Sortie*

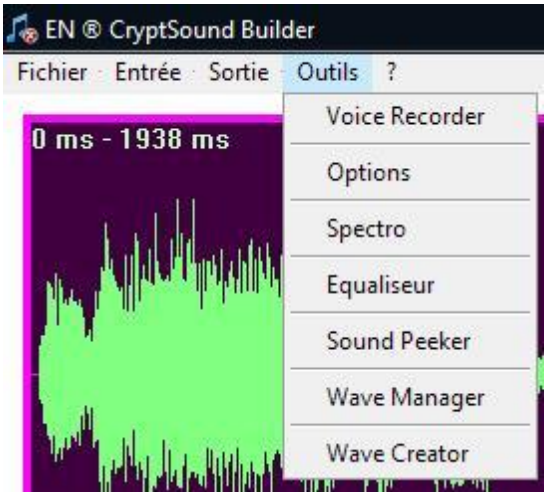


Six sous menus :

- ***Jouer*** : pour la lecture du signal audio (sortie)
- ***Enregistrer*** : pour enregistrer un fichier traité
- ***Arrêter*** : pour stopper la lecture
- ***Crypter*** : pour le cryptage d'un signal audio
- ***Décryptage*** : pour le décryptage d'un signal audio
- ***Filtrer*** : pour le filtrage d'un signal audio

Figure 4.7: Les sous menus *Jouer – Enregistre – Arrêter – Crypter – Décrypter – Filtrer*

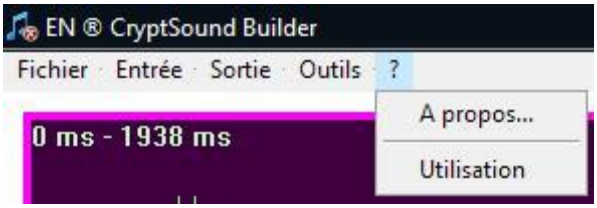
Menu *Outils*



Sept sous menus dont les détails vont être expliqués dans le sous paragraphe ***Menu Outils.***

Figure 4.8: Les sous menus *du menu Outils*

Menu *?*



Deux sous menus :

- A propos : Signature du concepteur
- Utilisation : mode d'emploi

Figure 4.9: Les sous menus du *A propos – Utilisation*

iii) Menu Outils

Le menu *Outils* possède les applications spécifiques du « *CryptSound Builder* » autre que les fonctions de *cryptage, décryptage* et *filtrage* ajoutées dans le menu *Sortie*. On distingue :

- Le sous menus *Voice Recorder* qui permet d'enregistrer les flux audio entrants et faisant apparaitre l'application suivante :

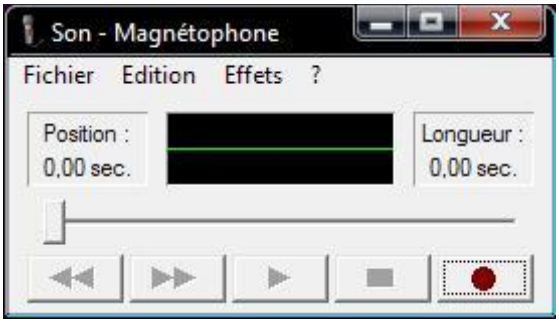


Figure 4.10 : Enregistreur des flux audio entrants

- Le sous menus *Options* qui affiche une palette de couleur configurant les couleurs du signal ou spectre et le fond de l'interface du logiciel.

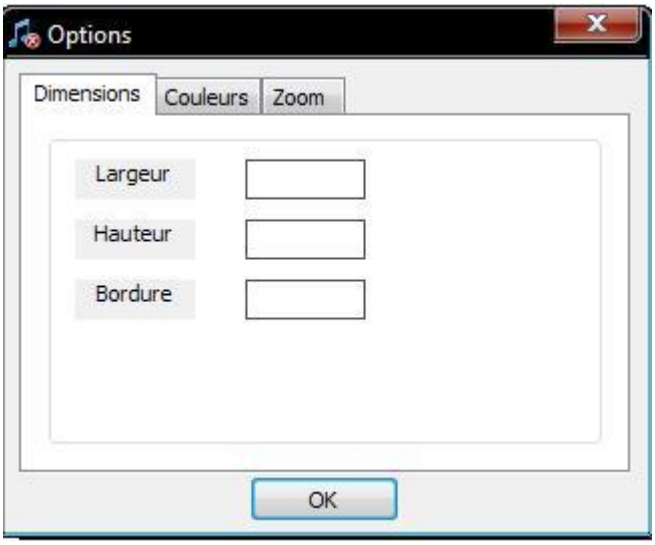


Figure 4.11 : Palette de configuration

- Le sous menus *Spectro* qui est un lecteur audio pour les fichiers au format wav, mp3 et wma. Le spectre du signal s'affiche dynamiquement pendant la lecture.

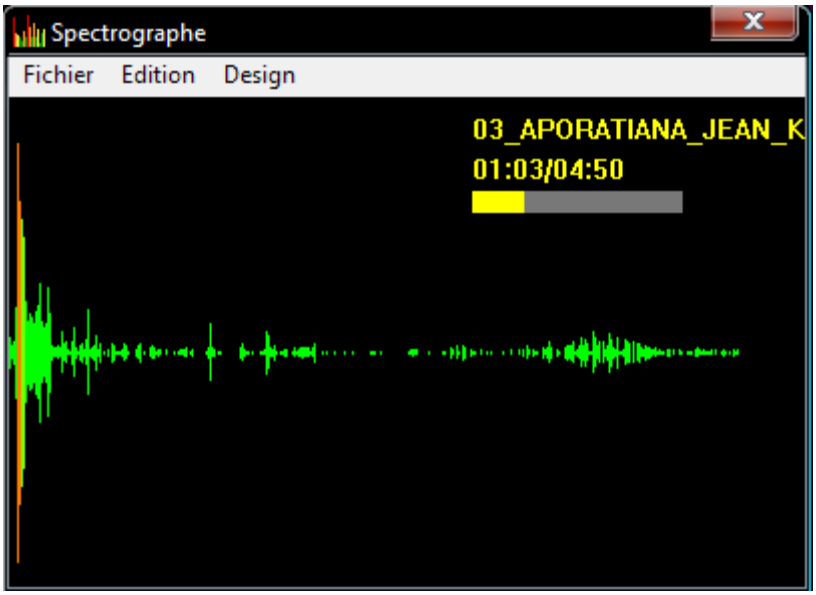


Figure 4.12 : lecteur de fichier wav, mp3 et wma avec visualisation spectrale

- Le sous menus *Equaliseur* qui est un équaliseur audio

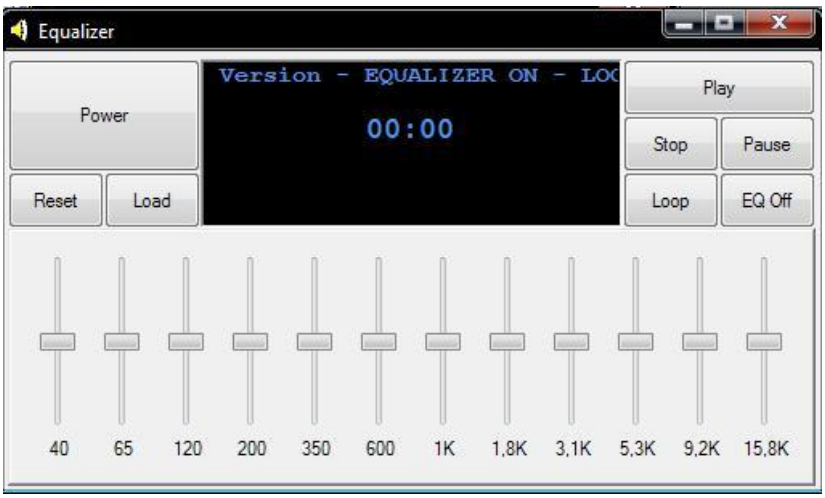


Figure 4.13 : Equaliseur audio

- Le sous menus *Sound Peeker* qui est aussi un lecteur audio au format wav, mp3 et wma mais avec une représentation de spectre

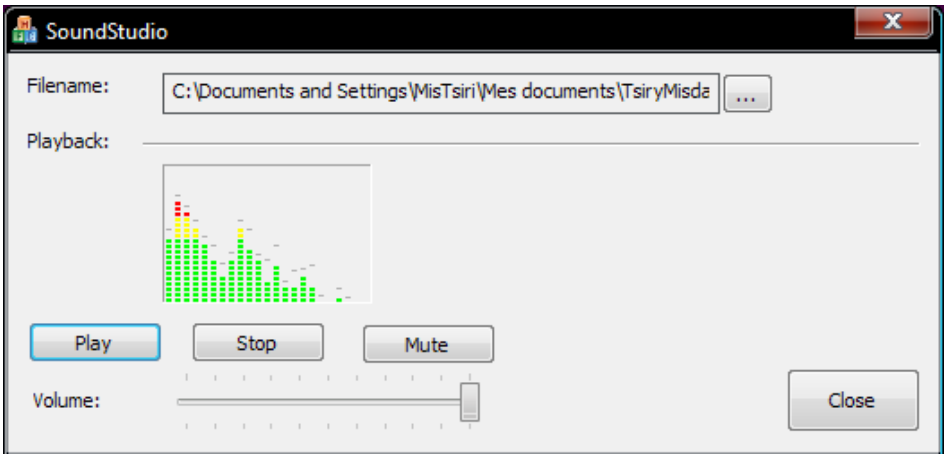


Figure 4.14: Lecteur audio avec représentation spectrale en Pic

- Le sous menu *WaveManager* qui permet de visualiser le signal et le spectre d'un fichier wave avec des coordonnées des échantillons. Il affiche aussi les caractéristiques d'un fichier wave telles que son nom, sa taille, mode de codage, compression, canaux, et le fréquence d'échantillonnage.

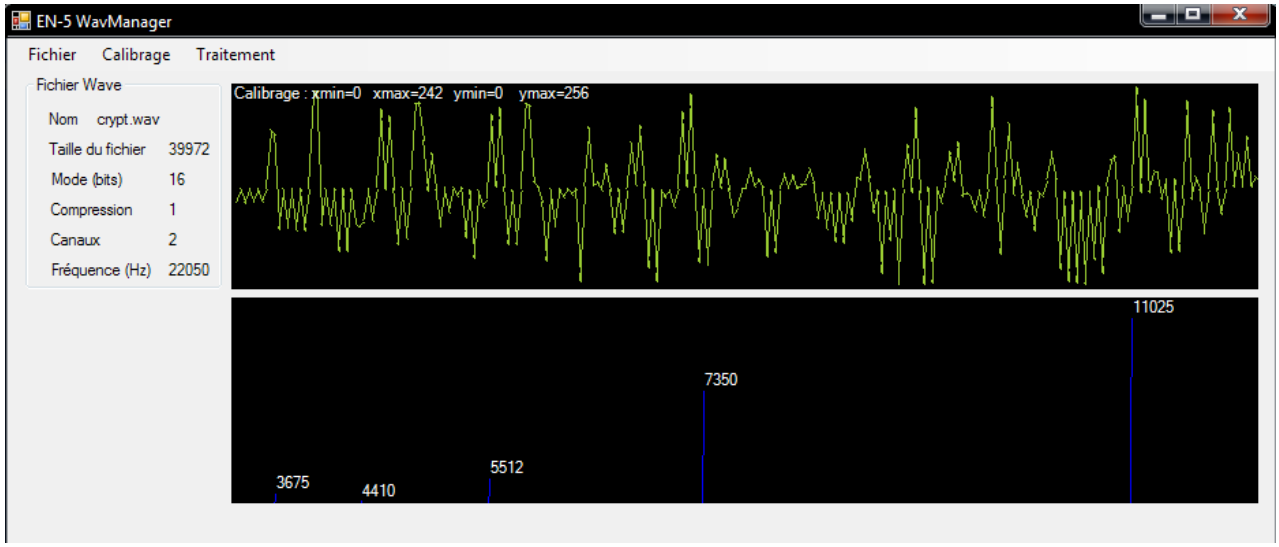


Figure 4.15 : visualisation des coordonnées pour le spectre d'un signal audio

- Le sous menu *Wave Creator* qui est un éditeur de fichier wave et qui nous permet de changer les caractéristiques d'un fichier audio de type wav.

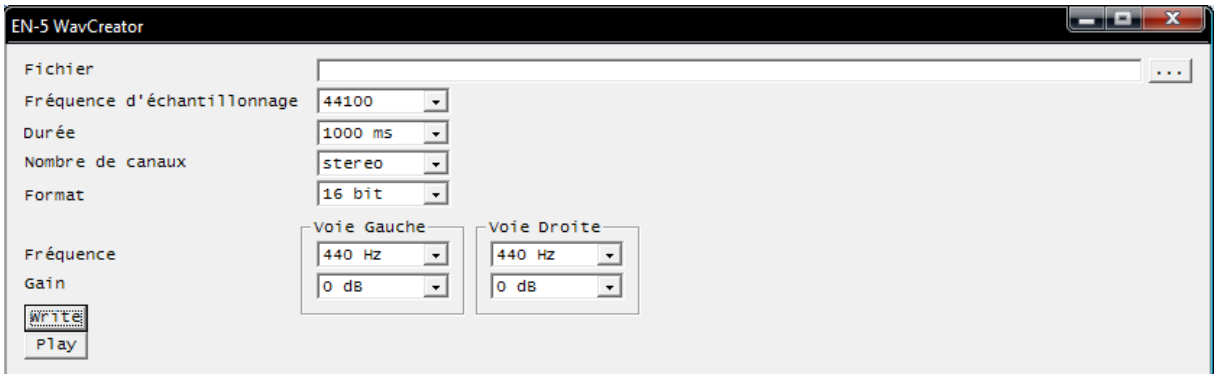


Figure 4.16 : Editeur numérique d'un fichier audio (wav)

d. Procédure Cryptage - Décryptage

i) Chargement audio

Avant d'entamer le chiffrement (cryptage) d'un fichier audio (uniquement un fichier audio d'extension wav), il faut d'abord charger le fichier. Pour ce faire, choisir le menu « Entrée » et cliquer sur le sous-menu « Choisir », une boite de dialogue va s'ouvrir et on choisit un fichier de type wav après, et cliquer sur « Ouvrir » pour terminer.

ii) Lecture des signaux

On peut écouter le fichier qu'on a chargé tout à l'heure pour vérifier que c'est le bon fichier qu'on va chiffrer. On sélectionne le menu « Entrée » et puis le sous-menu « Jouer ». Si la lecture est trop longue on peut l'arrêter en cliquant le sous menu « Arrêter ».

Quand on est sûr que c'est le bon fichier audio qu'on a chargé, on peut procéder maintenant au chiffrement du fichier audio.

iii) Spécification de la sortie : Cryptage - Décryptage - Filtrage

Pour chiffrer le fichier audio, entrer dans le menu « Sortie » et ensuite on clique sur le sous-menu « Crypter ». Les résultats s'affichent sur le 3ème et le 4ème cadrant du logiciel respectivement le signal et le spectre du fichier crypté. Et pour l'écouter, dans ce menu même, choisir le sous-menu « Jouer ». Il faut enregistrer le fichier crypté pour l'exporter à l'usage.

Quand on veut déchiffrer ce fichier, on charge une nouvelle fois dans le 1ère et le 2ème cadrant, pour cela, référer au paragraphe précédent (« *i) Chargement audio »*). Quand le fichier audio crypté est chargé, cliquer sur les menus suivants : « Sortie » et puis « Décrypter », on peut visualiser le fichier décrypté dans le 3ème et 4ème cadrant et on peut l'écouter après et aussi l'enregistrer respectivement dans les menus « Sortie » puis « Jouer » et « Sortie » puis « Enregistrer ».

Quand on veut avoir un son à peu près comme l'original on le filtre pour éliminer les hautes fréquences supérieures à la fréquence de modulation. On charge alors le fichier décrypté dans le premier et le second cadrant en choisissant les menus : « Entrée » puis « Choisir » après avoir l'enregistré. Puis, pour le filtrer, choisir les menus : « Sortir » puis « Filtrer » et le fichier filtrer s'affiche en bas dans le troisième et quatrième cadrant.

CONCLUSION

La cryptographie, étant toujours considérée comme un moyen de sécurisation des messages textuels, a trouvé un autre domaine d'utilisation c'est-à-dire son application avec la science du traitement des signaux. « ***CryptSoundBuilder*** » en est un témoin à cette fusion de deux sciences différentes donnant naissance à un moyen robuste de protection des données audio numériques pour celui qui veut lutter contre les piratages.

Mais la limite de ce logiciel nous a lancé dans la recherche d'autre amélioration future comme l'ajout d'autre format audio que « ***CryptSoundBuilder*** » peut crypter ou l'étude approfondie de la cryptographie et le tatouage audionumérique.

ANNEXE

ANNEXE A : LA PROGRAMMATION SOUS WINDOWS

A.1 Les bases de la programmation sous Windows

a. Introduction

La programmation sous Windows nécessite l'utilisation des fonctions propres à Windows connues communément sous le nom de **« API Windows »** ou **« Win32 API »**.

b. Hello, World !

Ce programme affiche la boîte de dialogue de la Fig. A.1

```
#include <windows.h>

int WINAPI WinMain(
       HINSTANCE hInstance,
       HINSTANCE hPrevInstance,
       LPSTR lpCmdLine,
       int nCmdShow)
{
    MessageBox(NULL, "Hello, world !", "Hello", MB_OK);
    return 0;
}
```

Figure A.1 : Boîte de dialogue « Hello, World ! »

La fonction **MessageBox** sert généralement à afficher un petit message à l'utilisateur comme pour l'informer du succès ou de l'échec d'une opération par exemple ou pour demander une action à effectuer lorsque le programme ne peut prendre de décision tout seul.

```
int MessageBox(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, UINT uType);
```

Paramètres

* hWnd : *handle* d'une fenêtre à laquelle la boîte de dialogue se rapporte ou tout simplement NULL si on ne veut lui associer aucune fenêtre.
* lpText : le texte à afficher
* lpCaption : le titre de la boîte de dialogue
* uType : type de la boîte de dialogue

Un **handle** est un numéro qui identifie de manière unique un objet quelconque.

c. La fonction WinMain

Le point d'entrée d'une application Windows est la fonction WinMain définie par :

```
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine,
       int nCmdShow)
```

Le mot-clé **WINAPI** indique que l'application ne peut être lancée que dans un environnement Windows.

Paramètres

* hInstance : handle de l'instance de l'application
* hPrevInstance : handle de l'instance précédente. Vaut toujours NULL. Ce paramètre fut utilisé dans les premières versions de Windows et est uniquement gardé pour des raisons de compatibilité.
* lpCmdLine : pointeur les arguments de la ligne de commande de l'application. Pour obtenir la ligne de commande dans son intégralité, utilisez la fonction GetCommandLine. Cette fonction ne nécessite aucun argument et retourne un pointeur sur la ligne de commande toute entière
* nCmdShow : indique comment l'utilisateur désire t-il que la fenêtre principale soit affichée : avec sa taille normale, agrandie ou réduite ? Rien n'oblige cependant le programme à considérer ce choix.

La fonction doit retourner le **code d'erreur** de l'application.

d. Unicode

Unicode est un jeu de caractères dérivé de l'ASCII utilisant 16 bits pour représenter chaque caractère. Il est utilisé en interne par Windows mais les applications peuvent également utiliser le jeu de caractères ANSI (8 bits) pour communiquer avec le système grâce à un jeu de conversions totalement opaques pour l'utilisateur. Par contre lorsqu'on développe des programmes qui doivent dialoguer directement avec le noyau du système (comme les pilotes de périphériques par exemple), on est obligé d'utiliser des chaînes en Unicode uniquement.

A.2 Les fenêtres

a. Introduction

Les étapes à suivre pour créer une fenêtre sont les suivantes :

- Enregistrer une classe (ou modèle) de fenêtre
- Créer une fenêtre (... à partir d'un modèle existant)
- L'afficher (en effet, la fenêtre est initialement invisible)
- Intercepter tous les messages (souris, clavier, etc.) puis les passer à la procédure de fenêtre.

Une procédure de fenêtre est une fonction chargée de traiter les messages reçus.

b. Les messages

Les moyens de communication principaux dont Windows et les programmes écrits pour Windows disposent sont les **messages**. Quand une opération ou action se produit, Windows répond à cette opération ou action en envoyant un message, soit à lui-même, soit à une autre application.

Les messages sont placés dans une file appelée **queue des messages** et sont traités les uns après les autres en fonction de leur arrivée dans la file.

Par exemple, quand l'utilisateur utilise la souris dans la fenêtre d'un programme, Windows intercepte cette action et envoie un message à l'application lui notifiant l'usage de la souris dans telle fenêtre du programme et à quel endroit.

Ensuite l'application commence son traitement en fonction du message ou l'ignore si le message n'a pas d'importance.

c. Enregistrer une classe de fenêtre

L'enregistrement d'une nouvelle classe de fenêtre peut se faire avec la fonction **RegisterClass**. Cette fonction nécessite comme paramètre l'adresse d'une structure de type **WNDCLASS**.

Par exemple :

```
WNDCLASS wc;

wc.cbClsExtra     = 0;
wc.cbWndExtra     = 0;
wc.hbrBackground  = (HBRUSH)(COLOR_WINDOW + 1);
wc.hCursor        = LoadCursor(NULL, IDC_ARROW);
wc.hIcon          = LoadIcon(NULL, IDI_APPLICATION);
wc.hInstance      = <Instance de notre application>;
wc.lpfnWndProc    = <Adresse d'une procédure de fenêtre>;
wc.lpszClassName  = "Classe 1";
wc.lpszMenuName   = NULL;
wc.style          = CS_HREDRAW | CS_VREDRAW;

RegisterClass(&wc);
```

d. Créer la fenêtre et l'afficher

Après avoir enregistré la classe de la fenêtre, on peut désormais créer une fenêtre.

```
HWND hWnd;

hWnd = CreateWindow("Classe 1", /* Classe de la fenêtre */
                    "Notre première fenêtre", /* Titre de la fenêtre */
                    WS_OVERLAPPEDWINDOW, /* Style de la fenêtre */
                    100, /* Abscisse du coin supérieur gauche */
                    100, /* Ordonnée du coin supérieur gauche */
                    600, /* Largeur de la fenêtre */
                    300, /* Hauteur de la fenêtre */
                    NULL, /* Fenêtre parent */
                    NULL, /* Menu */
                    <Instance de notre application>,

                    NULL /* Paramètres additionnels */);
```

où hWnd est le handle de la fenêtre qui vient d'être créée.

Ensuite on affiche la fenêtre :

```
ShowWindow(hWnd, <ShowCmd>);
```

où <ShowCmd> indique la façon d'afficher la fenêtre : SW_MINIMIZE pour une fenêtre réduite, SW_MAXIMIZE pour une fenêtre agrandie, etc.. D'habitude, on lui passe le paramètre **nCmdShow** de la fonction **WinMain** afin que la fenêtre s'affiche tout comme l'utilisateur l'a demandé.

e. Intercepter les messages

L'interception des messages se fait de la manière suivante :

```
MSG msg;

while (GetMessage(&msg, NULL, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

La fonction **GetMessage** retourne TRUE tant qu'elle n'a pas reçu le message WM_QUIT. Une application doit donc envoyer ce message pour quitter la boucle. Une fois qu'on a quitté la boucle, on termine le programme.

f. La procédure de fenêtre

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch(message)
    {
    case WM_DESTROY:
        PostQuitMessage(0);
        break;

    default:
        return DefWindowProc(hwnd, message, wParam, lParam);
    }

    return 0L;
}
```

Le message **WM_DESTROY** est envoyé par Windows lorsque la fenêtre est sur le point d'être détruite (après que l'utilisateur l'a fermée par exemple). A ce moment, nous

devons alors poster le message **WM_QUIT**. C'est ce qu'on a fait avec la fonction **PostQuitMessage**.

Le 0 passé en argument à **PostQuitMessage** indique une fin normale. C'est un entier qui indique la raison pour laquelle on a posté le message. Et enfin, il ne faut jamais ignorer un message. Si le message ne nous intéresse pas, on laisse Windows s'en occuper, en appelant tout simplement la fonction **DefWindowProc**.

A.3 Le graphisme

a. Introduction

La **GDI** ou Interface des Périphériques Graphiques est une API permettant de dessiner sur n'importe quel périphérique graphique (écran, imprimante, …) de manière standard c'est-à-dire sans avoir à communiquer directement avec le pilote.

b. Zone invalide

Une partie de la zone **cliente** d'une fenêtre est dite **invalide** lorsqu'elle doit être dessinée ou redessinée, ce qui se produit lorsqu'elle vient tout juste d'apparaître alors qu'elle était auparavant cachée (par une autre fenêtre par exemple).

Windows informe une fenêtre qu'une partie de sa zone cliente est invalide en lui envoyant le message WM_PAINT avec bien sûr en paramètre les informations supplémentaires, parmi lesquels un pointeur vers une structure appelée **Paint information Structure** contenant les coordonnées du plus petit rectangle contenant la région invalide, appelé **rectangle invalide**.

On peut invalider un rectangle avec la fonction **InvalidateRect**. Si le rectangle invalide est validé avant même qu'on ait eu le temps de traiter le message WM_PAINT, le message WM_PAINT sera supprimé de la queue des messages. On peut à tout moment connaître les coordonnées du rectangle invalide à l'aide de la fonction **GetUpdateRect** puis valider un rectangle avec la fonction **ValidateRect**.

c. Tracer des lignes et des points

Voici quelques fonctions permettant de tracer des lignes et des points :

```
COLORREF SetPixel(HDC hdc, int x, int y, COLORREF crColor); /* Modifie la couleur du pixel de coordonnées (x, y) */

COLORREF GetPixel(HDC hdc, int x, int y); /* Retourne la couleur du pixel de coordonnées (x, y) */

BOOL LineTo(HDC hdc, int x, int y); /* Trace une ligne de la position courante au point de coordonnées (x, y) */


BOOL Polyline(HDC hdc, CONST POINT *lppt, int cPoints); /* Trace des segments de droite reliant les points stockés dans la structure lppt */

BOOL PolylineTo(HDC hdc, CONST POINT *lppt, int cCount); /* Trace des lignes verticales passant par les points stockés dans la structure lppt */
```

ANNEXE B : NOTION SUR LA CRYPTOGRAPHIE

B.1. Introduction

La *Cryptologie* est une science mathématique qui comporte deux branches : la cryptographie et la cryptanalyse.

Le mot **cryptographie** vient du grec « kruptos » (qui signifie "caché") et « graphein » (pour "écrire") **[9]**

La cryptanalyse est la science analysant les cryptogrammes en vue de les décrypter. **[9]**

B.2. Vocabulaires

a. Le chiffrement

Le *chiffrement* est en cryptographie le procédé grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (de)chiffrement **[10]**

b. Le déchiffrement

C'est l'action de déchiffrer qui consiste à retrouver le texte original (aussi appelé texte clair) d'un message chiffré dont on possède la clé de chiffrement **[11]**

c. Crypter

C'est le synonyme du mot chiffrer.

d. Cryptolecte

C'est un jargon réservé à un groupe restreint de personnes désirant dissimuler leur communication **[9].**

e. Décrypter

C'est l'action de retrouver un message clair correspondant à un message chiffré sans posséder la clé de déchiffrement (terme que ne possèdent pas les anglophones, qui eux « cassent » des codes secrets) **[9]**

f. Cryptogramme

Le terme est parfois utilisé en cryptologie comme un synonyme de texte chiffré ou de « message chiffré ».

g. Clés

C'est une valeur utilisée dans un algorithme de cryptographie afin de générer un texte chiffré ou de le déchiffrer. Les clés sont en réalité des nombres extrêmement important (mesure en bits) [12].

h. Stéganographie

La stéganographie est une forme de cryptographie qui consiste à camoufler un message dans un texte, une image, ou n'importe quel support [13]

B.3. Les différents types de cryptanalyse

[14] Un attaquant est donc une personne qui tente de décrypter des messages, c'est-à-dire de retrouver des claires à partir de chiffrés sans connaître la clé. On réserve généralement le verbe « déchiffrer » à l'action du destinataire légitime qui effectue l'opération inverse du chiffrement.

La cryptanalyse d'un système cryptographique peut être :

- **Une cryptanalyse partielle** : l'attaquant découvre alors le texte clair correspondant à un ou plusieurs messages chiffrés interceptés.
- **Une cryptanalyse totale** : L'attaquant découvre un moyen de déchiffrer tous les messages, aussi bien ceux qu'il a interceptés que ceux à venir, par exemple en découvrant la clé utilisée.

Selon les moyens dont dispose l'attaquant, on distingue plusieurs types d'attaques. Par ordre de moyens croissants, on a :

- **Attaque à messages chiffrés (seulement) :** L'attaquant a seulement la possibilité d'intercepter un ou plusieurs messages chiffrés.
- **Attaque à messages clairs :** L'attaquant dispose d'un ou plusieurs messages clairs avec les messages chiffrés correspondants.
- **Attaque à messages clairs choisis :** L'attaquant a la possibilité d'obtenir la version chiffrée de messages clairs de son choix. On distingue alors deux sous-types d'attaque, suivant que l'attaquant est contraint de choisir les clairs en une seule

fois, ou au contraire peut faire évoluer ses choix au fur et à mesure des résultats obtenus. Dans le deuxième cas, on parle d'attaque adaptive à messages clairs choisis.

- **Attaque à messages chiffrés choisis :** L'attaquant a temporairement l'opportunité de déchiffrer les messages de son choix (en ayant accès par exemple à une machine déchiffrant). Il tente alors d'en profiter pour obtenir des informations lui permettant de décrypter ensuite d'autres messages par ses propres moyens. Comme dans le point précédent, on peut distinguer deux sous-types : attaque adaptive ou non.

B.4. Modes opératoires

a. ECB (Electronic Code Book)

Chaque bloque de texte en claire est chiffré indépendamment, on n'est pas obligé de chiffrer un fichier linéairement c'est-à-dire on peut d'abord chiffrer les 10 blocs de milieu ensuite de la fin puis ceux de début.

La plupart des messages ne se divisent pas exactement au même longueur de blocs. Il y a d'habitude un bloc court à la fin (ECB 64 bits). Il faut donc effectuer un remplissage. Le remplissage doit se faire suivant un schéma régulier des suites des 0 et de 1 (0101…01) pour compléter les blocs.

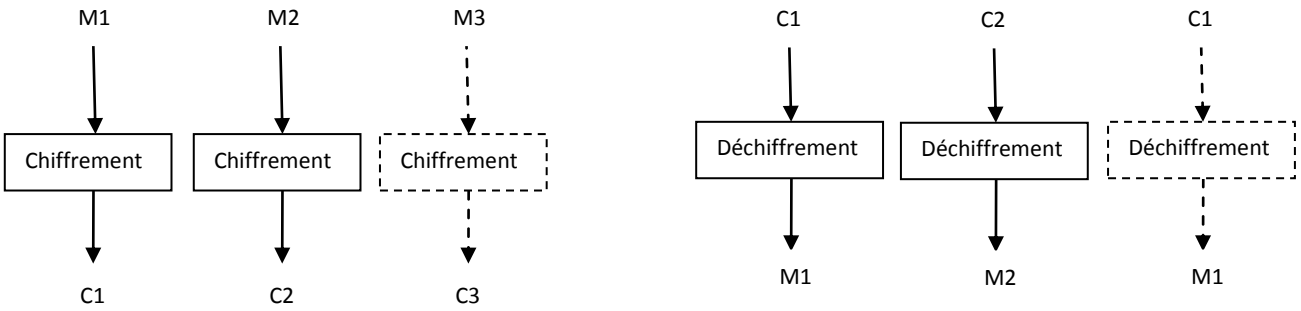


Figure B.1 : schéma synoptique de chiffrement et de déchiffrement en mode ECB

b. CBC (Cipher Bloc Chaining)

Le chaînage utilise une méthode de rétroaction car les résultats du chiffrement du bloc précédent sont réutilisés comme entrée pour le chiffrement du bloc courant. Chaque bloc chiffré dépend non seulement de bloc de texte qui l'engendre mais aussi de tous les blocs de texte en clair qui le précède. Le texte en clair est combiné par « ou exclusif » avec

les blocs chiffré précédent avant d'être chiffré. Après qu'un bloc de texte en claire a été chiffré, le texte chiffré correspondant est stockés dans un registre de rétroaction.

Avant que le bloc suivant de texte en claire soit chiffré, il est combiné par « ou exclusif » avec le registre de rétroaction pour devenir la nouvelle entrée de l'algorithme de chiffrement.

Equation:

$C_i = E_k(M_i \otimes C_{i-1})$

$M_i = C_{i-1} \otimes D_k(C_i)$

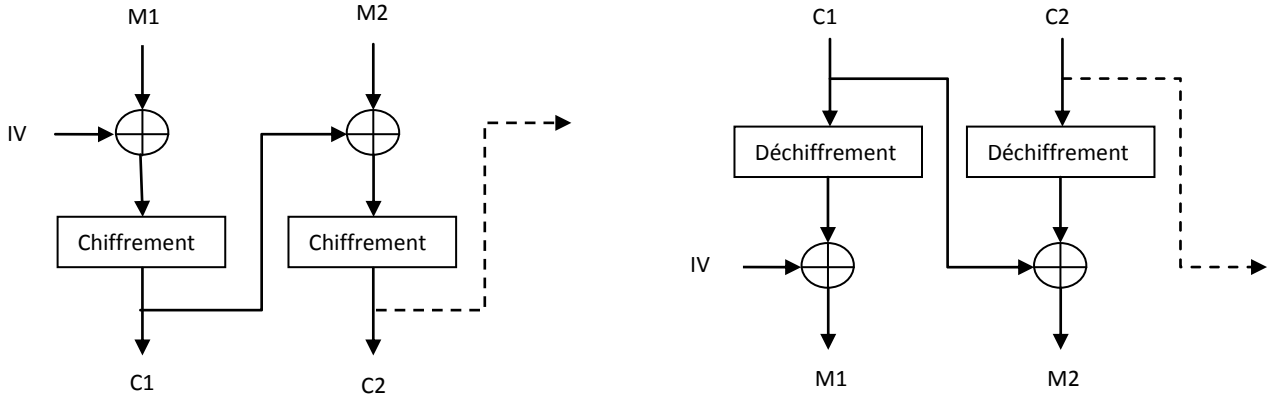


Figure B.2 : schéma synoptique de chiffrement et de déchiffrement en mode CBC

c. OFB (Output Feedback)

C'est une méthode similaire au mode CFB sauf que n bits de blocs de sortie précédent sont mis dans la position le plus à droite de la file d'attente.

Equation:

$C_i = M_i \otimes S_i$

$M_i = C_i \otimes S_i$ Avec $S_i = E_k\,(S_{i-1})$

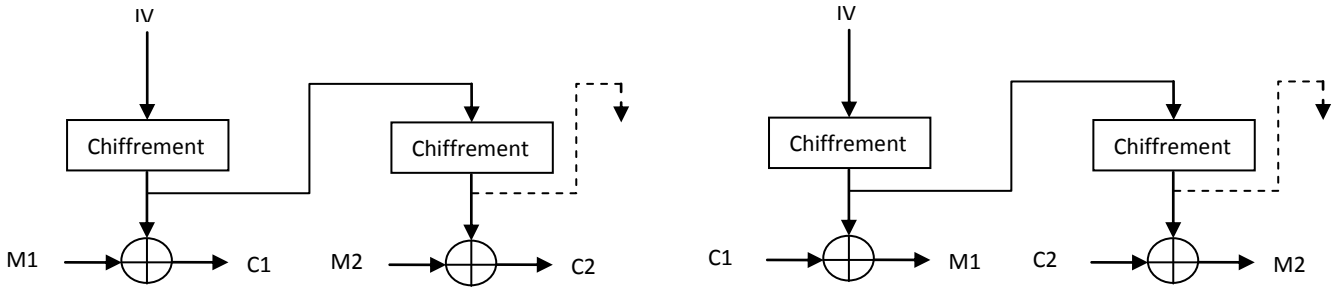


Figure B.3 : schéma synoptique de chiffrement et de déchiffrement en mode OFB

d. CFB (Cipher Feedback)

Le mode CFB manipule un file d'attente de la taille d'un bloc d'entré initialement, la file d'attente est chiffré et les 8 bits le plus à gauche de résultat (MSB) sont combinées par « ou exclusif » avec le 1er caractère de 8 bits du texte en clair pour devenir les 8 1er bits du texte chiffré. Ce caractère peut être maintenant transmis. Les même 8 bits sont placés dans les 8 bits le plus à droite de la file d'attente et tous les autre 8 bits sont ignorés. Le caractère suivant est alors chiffré de la même façon. Le déchiffrement est le processus inverse.

En mode CFB, on peut utiliser un nombre n quelconque de bits inferieur ou égal à 64 bits (talle de bloc).

Equation : CFB à n bits

$C_i = M_i \otimes E_k (C_{i-1})$

$M_i = C_i \otimes E_k (C_{i-1})$

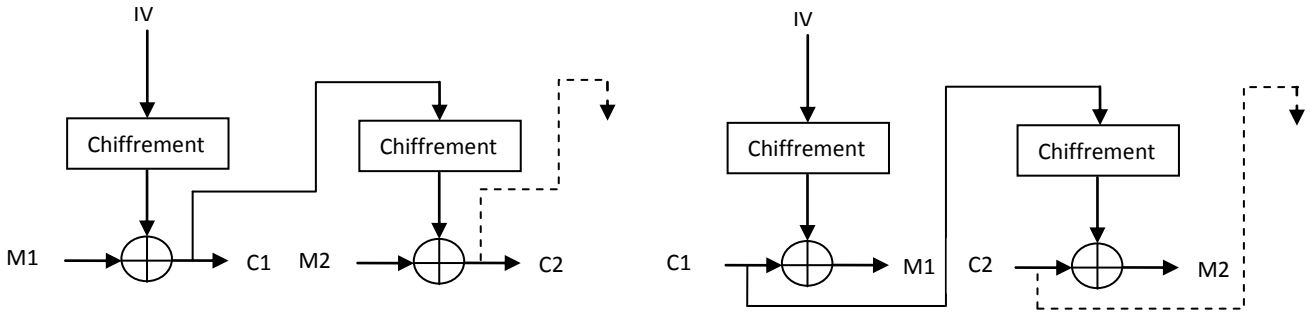


Figure B.4 : schéma synoptique de chiffrement et de déchiffrement en mode CFB

Avec :

M_i : i^{eme} bit du message clair et C_i le message chiffré correspondant

C_{i-1} : le message chiffré de $(i-1)^{eme}$ bit

E_k : fonction de chiffrement

S_i : état indépendant au texte chiffré et du texte en claire

IV : vecteur d'initialisation

B.5. Les deux grandes catégories de Chiffrement

a. Les chiffrements symétriques

i) Introduction

La cryptographie symétrique, également dite à clé secrète (par opposition à la cryptographie à clé publique), est la plus ancienne forme de chiffrement. On a des traces de son utilisation par les Égyptiens vers 2000 av. J.-C. Plus proche de nous, on peut citer le chiffre de Jules César, dont le ROT13 est une variante.

ii) Définition

On dit que le chiffrement est symétrique quand il utilise la même clé pour le chiffrement et le déchiffrement.

b. Les chiffrements asymétriques

i) Historiques

[15**]** Le concept de cryptographie à clé publique, autre nom de la cryptographie asymétrique, est dû à Whitfield Diffie et à Martin Hellman. Il fut présenté pour la première fois à la National Computer Conference en 1976, puis publié quelques mois plus tard dans New Directions in Cryptography.

En réalité, James Ellis, qui travaillait au service du chiffrement britannique (GCHQ, Gouvernement Communications Headquarters), avait eu cette idée peu avant. En 1973, C.C. Cocks décrivit (pour le même service du chiffre) ce qu'on a appelé l'algorithme RSA. Enfin, en 1974, M. J. Williamson invente un protocole d'échange de clé très proche de celui de Diffie et de Hellman. Ces découvertes n'ont été rendues publiques qu'en 1997 par le GCHQ.

Dans leur article de 1976, W. Diffie et M. Hellman n'avaient pas pu donner l'exemple d'un système à clé publique, n'en ayant pas trouvé. Il fallut attendre 1978 pour avoir un premier exemple, dans l'article A Method for Obtaining Digital Signatures and Public-key Cryptosystems de Ronald Rivest, Adi Shamir et Leonard Adleman, le RSA, abréviation tirée des trois noms de ses auteurs. C'est du moins la version académique. Les trois hommes fondèrent aussi la société RSA Security.

ii) Définition

La cryptographie asymétrique permet à tous d'envoyer un message chiffré à une personne de sorte à que celle-ci seule puisse le décoder, sans qu'elle n'ait besoin de divulguer la clé privée servant à déchiffrer.

iii) Principes

[16**]** La **cryptographie asymétrique** est fondée sur l'existence de fonctions à sens unique, une fois la fonction appliquée à un message, il est extrêmement difficile de retrouver le message original.

En réalité, on utilise en cryptographie asymétrique des fonctions à sens unique et à brèche secrète. Une telle fonction est difficile à inverser, à moins de posséder une information particulière, tenue secrète, nommée clé privée.

À partir d'une telle fonction, voici comment se déroulent les choses : Alice souhaite pouvoir recevoir des messages chiffrés de n'importe qui.

1. Alice génère deux clés. La clé publique verte qu'elle envoie à Bob et la clé privée rouge qu'elle conserve précieusement sans la divulguer à quiconque.

2. Bob chiffre le message avec la clé publique d'Alice et envoie le texte chiffre.

3. Alice déchiffre le message grâce à sa clé privée.

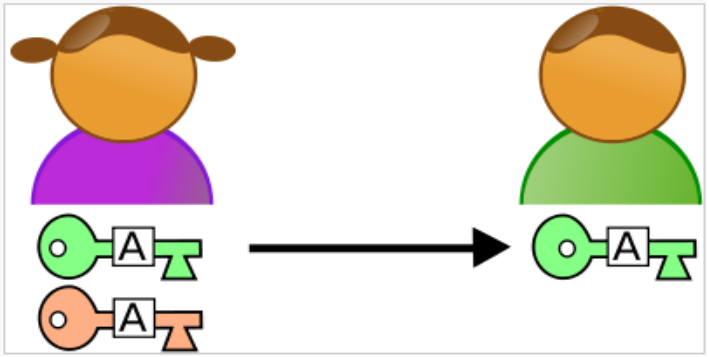


Figure B.5: 1ère étape du principe d'authentification

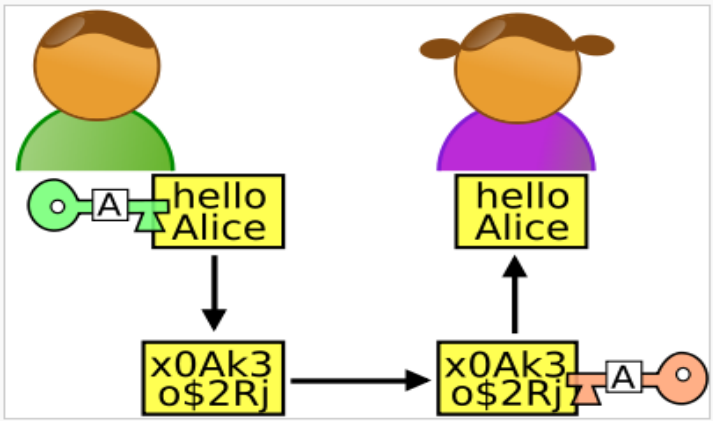


Figure B.6: 2ème et 3ème étape du principe d'authentification

iv) RSA

Rivest Shamir Adleman ou RSA est un algorithme asymétrique de cryptographie à clé publique, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été décrit en 1977 par Ron Rivest, Adi Shamir et Len Adleman, d'où le sigle RSA. RSA a été breveté par le MIT en 1983 aux États-Unis. Le brevet a expiré le 21 septembre 2000.

En 2008, c'est le système à clef publique le plus utilisé (carte bancaire française, de nombreux sites web commerciaux…).

- *Chiffrement et déchiffrement*

Fonction de chiffrement E (partie publique) :

La clé publique comporte deux entiers: $k = (e, n)$.Le chiffrement est effectué par bloc de M. On élève M à la puissance e modulo n:

$$E_k(M) = M^e (mod\ n)$$

Fonction de déchiffrement D (partie secrète) :

La clé secrète est un couple d'entiers: $k' = (d, n)$.On procède à un déchiffrement d'un bloc chiffré C sur b bits. Et on élève C à la puissance d modulo n: $D_{k'}(C) = C^d (mod\ n)$

Remarque: Les entiers n, e, d doivent être choisis selon des règles précises.

- ***Détermination de clés***

Détermination de n: valeur de modulo

- Choisir deux entiers premiers p et q (différents, grands et aléatoires).
- Calculer $n = p * q$.

La sécurité de RSA repose sur la difficulté de factoriser un entier n en deux entiers premiers p et q, n (grand) peut avoir les valeurs suivantes : 320 bits, 512 bits, 1024 bits …. La taille de n conditionne la vitesse de chiffrement et de déchiffrement.

Détermination de la clé publique (***e, n***)

- Calculer $z = (p - 1)(q - 1)$.
- Choisir un entier ***e*** premier avec ***z*** (***1< e < z***, ***e*** et ***z*** n'ont pas de diviseurs communs).

Détermination de la clé privée (***d, n***)

Choisir un entier ***d*** tel que :

$e^d = 1(mod\ z)$, ***d*** est un inverse de ***e*** dans l'arithmétique modulo ***z***

(Soit encore $e^d = k(p-1)(q-1)+1$).

Remarque :

La contrainte de RSA est que les blocs à chiffrer M sont des entiers inférieurs à l'entier ***n***

- ***M*** plus petit que ***n*** => les calculs sont conduits modulo ***n***.
- Exemple : si on choisit ***n*** sur ***d+1*** bits, on chiffre des messages ***M*** de ***d*** bits (taille inférieure) alors les messages chiffrés sont sur ***d+1*** bits (restes modulo ***n***).
- Autre aspect: utilisation d'une méthode de bourrage des messages en clair courts pour ne pas chiffrer de trop petites valeurs.

REFERENCES

[1] Zo Manankasina et Jessy Edouard, Livre de mémoire de fin d'étude intitulé : « Réalisation d'une suite complète d'acquisition, de génération et de traitement du son »

[2] http://aboudet.chez-alice.fr/doc_musique/son-nature.html
http://tcts.fpms.ac.be/cours/1005-07-08/speech/parole.pdf

[3] http://www.hsc.fr/ressources/cours/crypto/crypto.pdf

[4] http://www-prima.imag.fr/jlc/Courses/2000/ENSI2.TS/ENSI2.TS.S2.pdf

[5] MESANNEX\bruit.pdf

[6] http://lpce.cnrs-orleans.fr/~ddwit/enseignement/cours-signaux.pdf

[7] http://www-prima.imag.fr/jlc/Courses/2000/ENSI2.TS/ENSI2.TS.S2.pdf

[8] E341, Cours Théorie du signal, 3ème Année, Département Electronique, ESPA, 2008

[9] http://fr.wikipedia.org/wiki/Cryptographie

[10] http://en.wikipedia.org/wiki/Cryptography

[11] http://fr.academic.ru/dic.nsf/frwiki/548120

[12] http://fr.wikipedia.org/wiki/Cl%C3%A9_de_chiffrement

[13] http://fr.wikipedia.org/wiki/St%C3%A9ganographie

[14] http://fr.wikipedia.org/wiki/Cryptanalyse

[15] http://www.bibmath.net/crypto/moderne/clepub.php3

[16] http://fr.wikipedia.org/wiki/Cryptographie_asym%C3%A9trique