1. Introduction

Le format de fichier Wave est le format Windows de fichier natif pour stocker des données audio numériques. Il est devenu l'un des plus largement soutenue formats de fichiers audio numériques sur le PC grâce à la popularité de Windows et le grand nombre de programmes écrits pour la plateforme. Pratiquement tous les programmes modernes qui peut ouvrir et / ou enregistrer l'audio numérique prend en charge ce format de fichier, ce qui rend à la fois très utile et une exigence virtuel pour les développeurs de logiciels à comprendre.
Le format WAV est un fichier suivant la norme RIFF (Ressource Interchange File Format).

Dans ce chapitre, nous allons découvrir la structure du type de fichier .wav-pcm, car c'est celui-ci que nous utiliserons pour notre fichier son et qui sera par la suite crypté par les algorithmes de chiffrement

2. Le format Wav

Le format PCM : C'est le format de fichier "standard" pour les samples (les enregistrements de sons), car les données sont brutes, c'est-à-dire qu'elles ne sont ni modifiées, ni compressées. Le fichier possède une en-tête de 44 octets (en tout cas en général, voir ci-dessous), permettant de connaître le type du sample: Son format, sa fréquence, le nombre de voies, etc...
Cet en-tête (« header » en anglais) peut, dans certaines variantes du format PCM, avoir une taille supérieure à 44 octets. Mais voyons d'abord le format le plus standard :

Un fichier WAVE est composé de 3 blocs distincts : RIFF, fmt et data…

- Le **bloc RIFF** comporte les informations concernant le type d'en-tête, la taille du fichier et le format du fichier.
- Le **bloc fmt** comporte les spécifications audio : le format audio, le nombre de canaux, la fréquence, le byte rate, le nombre de bits par échantillons…
- Le **bloc data** contient la taille du bloc de données et les données. **[s25]**

2.1. Format (bloc RIFF)

• On trouve d'abord la mention « RIFF » dans les 4 premiers octets du fichier ($52, $49, $42, $42 en hexadécimal. Ce sont les codes ASCII des lettres R,I,F,F)

• On trouve ensuite la taille TOTALE du fichier codée sous la forme d'un entier long (4 octets, donc)

• On trouve ensuite la mention « WAVE» soit 4 caractères ce qui nous donne en hexadécimal : $57,$41,$56,$45

Cette première partie est suivie du. **[s25]**

2.2. Descriptif du son (bloc fmt)

Fréquence d'échantillonnage, nombre d'octets par échantillons, etc. Les données qu'il contient ainsi que leurs types et leur ordre correspondent exactement à la structure WAVEFORMAT (cette structure, définie par Microsoft, se retrouve dans pas mal de docs. Elle est aujourd'hui officiellement remplacée par la structure WAVEFORMATEX qui est exactement la même pour les 6 premiers champs, c'est-à-dire tout les champs que l'on retrouve dans l'en-tête du fichier WAVE/PCM).

• **Indicateur de zone** : « fmt » soit 4 caractères (le dernier caractère est un espace) ce qui nous donne en hexadécimal : $66,$6D,$74,$20. « fmt » est l'abréviation de « format ». Cet indicateur nous prévient que les informations qui vont suivre concernent le format du son.

• **Taille de la structure WAVEFORMAT** sous la forme d'un entier long (soit 4 octets). La structure WAVEFORMAT comporte 16 octets. On trouve donc, logiquement, la valeur 16 (ou $10 en hexadécimal) stockée à cet endroit.

• **wFormatTag.w** (2 octets) : ce champ contient un code correspondant au format exact de codage des données. Pour les fichiers de type PCM, ce champ contiendra la valeur 1.

• **nChannels.w** (2 octets) : nombre de canaux. On aura la valeur 1 pour les sons mono, 2 pour les sons stéréo, et éventuellement plus pour les sons moins standard.

• **nSamplesPerSec.l** (4 octets) : nombre d'échantillons par seconde.

• **nAvgBytesPerSec.l** (4 octets) : nombre d'octets par secondes. Cette valeur fait double emploi avec les autres valeurs enregistrées mais vous devez la compléter correctement pour être certains que votre fichier sera compatible avec tous les programmes de son. Le nombre d'octets par seconde dépend :

• Du nombre d'échantillons pas seconde

• Du nombre d'octets par échantillon

• Du nombre de canaux

Il se calcul comme suit :

nAvgBytesPerSec= nSamplesPerSec* nBitsPerSample/8* nChannels

(nBitsPerSample est le nombre de BITS par seconde, un octet comporte 8 bits)

• **nBlockAlign.w** (2 octets) : contient la taille totale (en octets) d'un échantillon. Cette valeur fait également double emploi avec les autres valeurs enregistrées mais vous devez aussi la compléter correctement pour être certains que votre fichier sera compatible avec tous les programmes de son. Elle dépend :

• Du nombre d'octets par échantillon

• Du nombre de canaux

Elle se calcul comme suit :

nBlockAlign = nBitsPerSample/8* nChannels*

(nBitsPerSample est le nombre de BITS par seconde, un octet comporte 8 bits)

• **nBitsPerSample.w** (2 octets) : contient le nombre de bits par échantillon (voir note concernant l'amplitude)

Pour finir, le fichier contient les données proprement dites, que l'on appelle, en anglais, les data. **[s25]**

2.3. DATA : partie données de fichier wav (bloc data)

• **Indicateur de zone** : «data» soit 4 caractères ce qui nous donne en hexadécimal : $64,$61,$74,$61. Cet indicateur nous prévient que les informations qui vont suivre sont les données proprement dites.

• **Taille des datas** sous la forme d'un entier long (soit 4 octets). Cette taille est le nombre total d'octets des datas. Elle nous permet, par exemple, de calculer la durée du sample en appliquant la formule :

Durée = Taille des datas/ nAvgBytesPerSec

La taille des données doit TOUJOURS être un multiple de nBlockAlign. Lorsqu'une modification d'un son amène une modification de la taille des données, on peut « arrondir » la taille à un multiple de nBlockAlign à l'aide des deux opérations suivantes :

NouvelleTaille = NouvelleTaille/ nBlockAlign

NouvelleTaille = NouvelleTaille* nBlockAlign [s25]

2.4. Structure générale d'un fichier wav

Offset (décimal)	offset (hexa)	nom	longueur (oct.)	**description**
0	00h	rID	4	contient "RIFF"
4	04h	rLen	4	longueur du fichier
8	08h	wID	4	contient "WAVE"

Le **Format** Chunk:

Offset (décimal)	offset (hexa)	nom	longueur (octet)	**description**
12	0Ch	fId	4	contient "fmt " ("fmt espace")
16	10h	fLen	4	Longueur du Chunck
20	14h	wFormatTag	2	**format** (1 = Microsoft Pulse Code Modulation PCM)
22	16h	nChannels	2	nombre de canaux (1=mono, 2=stéréo)
24	18h	nSamplesPerSec	4	fréquence d'échantillonage (en Hz)
28	1Ch	nAvgBytesPerSec	4	= nChannels * nSamplesPerSec * (nBitsPerSample/8)
32	20h	nBlockAlign	2	= nChannels * (nBitsPerSample / 8)
34	22h	nBitsPerSample	2	longueur d'un échantillon en bits (8, 16, 24 ou 32)

Le WAVE Data Chunk:

Offset (décimal)	offset (hexa)	nom	longueur (octet)	**description**
36	24h	dId	4	contient "data"
40	28h	dLen	4	longueur du chunck dData (en octets)
44 et plus	2Ch	dData	dLen	les données du son échantillonné

3. Caractéristique de fichier wav

3.1. L'amplitude

Un sample est composé d'une courbe continue ayant une valeur bi-polaire (signée) . Le 1er élément d'un son est l'amplitude: C'est le point le plus élevé (et le plus bas) de la courbe. Plus l'amplitude est élevée, plus le son est fort, bruyant. L'unité de grandeur de l'amplitude est le décibel (dB). "C'est une mesure logarithmique donnant le degré d'amplification d'une vibration." Nous n'irons pas plus loin dans la description du décibel, puisque ce n'est pas indispensable dans la programmation. L'amplitude est digitalisée avec l'ADC de la carte son. Par exemple, en 8 bits, l'amplitude possède une résolution de 256 valeurs. En 16 bits, 65536 valeurs, etc... Il existe aussi le 24 et 32 bits. Pour le moment, toutefois, SoudEditor ne supporte pas les données 24 bits qui sont très embétantes à manipuler et qui sont très rarement utilisées. Plus la résolution est élevée, plus l'échantillon est proche du son original. Dans la figure 4.1, l'amplitude digitalisée est illustrée en vert. En 8 bits, la valeur de l'amplitude est non signée, et en 16 bits, l'amplitude est signée. **[s26]**

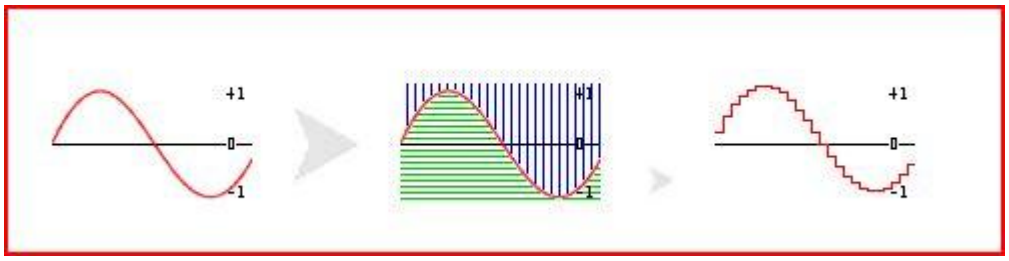

Figure 4.1. Amplitude, fréquence **[s26]**

3.2. La fréquence

En bleu (figure 4.1), c'est la fréquence d'échantillonnage, le nombre de valeurs définissant l'amplitude pour une seconde d'enregistrement. Ainsi 44100 Hz signifie 44100 échantillons pour une seconde de son mémorisé. Plus la fréquence d'échantillonnage est élevée, meilleure est la qualité du sample, plus les données digitales sont proches de l'original. **[s26]**

3.3. Le débit

On peut calculer le "débit" (ko/s) d'un sample avec ces paramètres: L'amplitude (format 8 ou 16 bits), le mode (mono ou stéréo) et la fréquence. Par exemple, en 8 bits mono 44100Hz, cela nous donne pour une seconde d'enregistrement: 44100 octets (1 octet=8 bits). En stéréo, c'est le double. En 16 bits stéréo, c'est le quadruple. Ainsi, avec la qualité du CD audio (16 bits stéréo 44,1 kHz), on obtient 176400 octets par seconde. A partir de la taille du fichier (donnée par fileSize), et les caractéristiques du sample, il est facile de calculer le temps total en secondes (ou ms) de lecture d'un sample. **[s26]**

3.4. L'ordre des données

Après les 44 octets de l'en-tête, viennent les données. Les données ont un ordre bien défini. Dans le cas d'un sample **8 bits mono,** c'est **1** seul octet par sample, donc les données se suivent normalement. Pour un sample **8 bits stéréo** , ce sont **2** octets par sample, l'octet de la voie de gauche, puis l'octet de la voie de droite: L,R, L,R, L,R, etc... Dans le cas d'un sample **16 bits mono** , ce sont **2** octets par sample, l'octet de poids faible, puis l'octet de poids fort. Par exemple pour une donnée qui vaudrait 15000, nous aurions: $98 $3A (les octets sont toujours inversés). Dans le cas d'un sample **16 bits stéréo** , ce sont **4** octets par sample; Deux

octets pour la voie de gauche, et deux pour la voie de droite: L,L,R,R, L,L,R,R, L,L,R,R, etc...

Le format des données : Le sample **8 bits** (mono ou stéréo) possède des données non-signées, c'est-à-dire que le point (l'amplitude) le plus bas vaut **zéro** , le point du milieu vaut **127** , et le point le plus haut vaut **255** .

Pour pouvoir travailler sur ces données 8 bits, (par exemple pour modifier le volume du sample, ou le mixer avec un autre, etc...) les données 8 bits doivent subir une petite manipulation afin d'être présentées de la même façon que les données 16 ou 32 bits. **[s26]**

Exemple détaillé de la structure des fichiers audio wav

Nom	Nombre d'octets	Type	Endian	Description
Bloc de déclaration d'un fichier au format WAVE				
Chunk ID	4	4 char	big	4 caractères constants RIFF : 0x52494646
Chunk Size	4	int 32	little	Taille du fichier - 8 octets des deux premiers blocs
Format	4	4 char	big	4 caractères constants WAVE : codé 0x57415645
Bloc décrivant le format audio				
SubChunk 1 ID	4	4 char	big	4 caractères constants "fmt " : codé 0x666d7420
SubChunk 1 Size	4	int 32	little	Taille du sous-bloc "SubCHunk 1" - 8 octets, vaut 16 pour le PCM
Audio Format	2	int 16	little	Type de compression audio, 1 pour PCM
Num Channels	2	int 16	little	Nombre de canaux (1 = Mono, 2 = Stéréo, ...)
Sample Rate (Frequence)	4	int 32	little	Fréquence d'échantillonnage (nombre d'échantillons par seconde)
Byte Rate (BytePerSec)	4	int 32	little	Nombre d'octets par seconde = Frequence * BitsPerSample/8 * NumChannels
Block Align (BytePerSample)	2	int 16	little	Nombre d'octets par échantillon (tous canaux confondus) = Nombre de canaux * Nombre d'octets par échantillons
Bits per Sample	2	int 16	little	Nombre de bits par échantillon
Bloc des données				
SubChunk 2 ID	4	4 char	big	4 caractères constants : "data" : 0x64617461
SubChunk 2 Size	4	int 32	little	Taille des données = NbCannaux * NbSamples * BitsPerSample/8 = taille du fichier - 44 octets (taille de l'en-tête, rappelez-vous-en !)
Datas (données)	FileSize - 44	...	little	Les données !

3.5. Fichier wav en hexadécimal

Voila une représentation détaillée d'un fichier wav avec les trois blocs (riff, fmt et data)

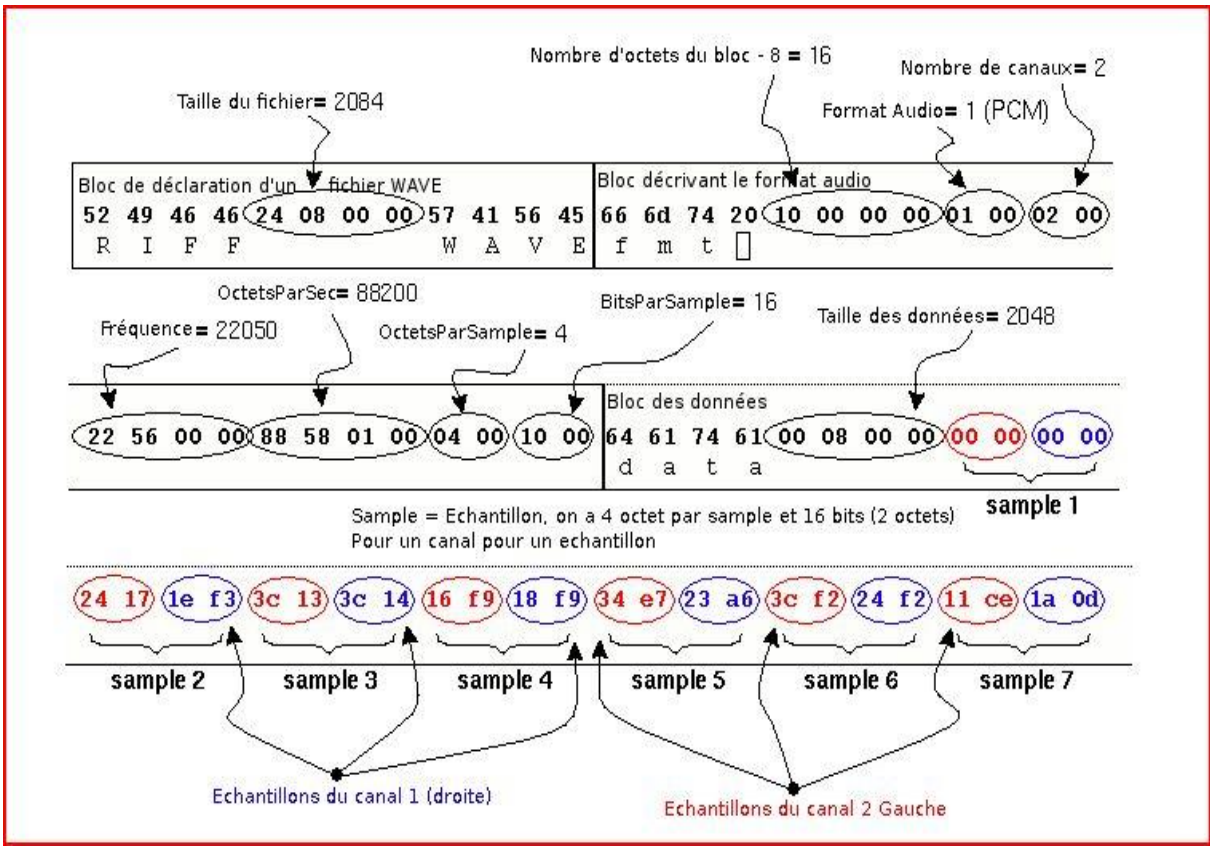

Figure 4.2. Fichier wav en hexadecimal **[s26]**

4. Conclusion

Dans ce chapitre, nous avons présenté une description détaillée sur la structure du fichier WAVE. L'étude et l'analyse le format des fichiers son de type wav sera exploité dans la phase de chiffrement par les différents algorithmes de cryptage qui sera notre objectif dans le prochaine chapitre.