Acknowledgment

First of all great thanks are due to Allah who helped me and gave me the ability to achieve this research from the first to the last step.

I would like to express my deep gratitude and sincere thanks to my supervisor ***Dr. Abeer M. Yousif*** *for guidance, assistance and encouragement during the course of this project.*

Grateful thanks for ***Dr. Loay E. George*** *the Head of Department of Computer Science of Baghdad University for the continuous support during the period of my studies.*

Deep gratitude and special thanks to my family: ***my parent, brothers, and Husband*** *for their encouragements and supporting to succeed in doing this work.*

Deep thanks to ***Dr. Haitham Abdul Lateef*** *and* ***Dr. Ban Nadeem Thannoon*** *for their support, interest and generosity.*

Special thanks to all my friends for giving me advises.

Sarah

Appendix A
RIFF WAVE (.WAV) File Format

A.1. Waveform Audio File Format (WAVE):

The canonical WAVE format starts with the RIFF header:

Offset	Field name	Length	Contents
0	"RIFF" file description header	4 bytes	Contains the letters "RIFF" in ASCII
4	Size of file	4 bytes	The file size which is less the size of the "RIFF" description (4 bytes) and the size of file description (4 bytes).this is usually: file size-8
8	"WAVE" description header	4 bytes	Contains the letters "WAVE"

Next to the RIFF header comes first chunk 'fmt chunk' which describes the sample format:

Offset	Field name	Length	Contents
12	"fmt" id	4 bytes	Contains the letters "fmt "
16	Size of file	4 bytes	The size of the WAVE type format (2bytes) + number of channels (2bytes) + sample rate (4 bytes) + Byterate (4bytes) + Block alignment (2bytes) + Bitspersample (2 bytes). This is usually 16
20	Audio Format	2 bytes	Type of WAVE format. if 1=PCM (Pulse Code Modulation), values other than 1 indicate some form of compression
22	Number of Channels	2 bytes	Channels :mono=1, stereo=2
24	SampleRate	4 bytes	Sample per Second e.g.8000, 44100
28	ByteRate	4 bytes	Byte per Second or ==SampleRate*number of channels*bitspersample/8
32	BlockAlign	2 bytes	Block alignment: the number of bytes for one sample including all channels. Or ==number of channels*bitspersample/8
34	BitsPerSample	2 bytes	Bits per sample 8 bits=8, 16 bits=16, etc.

Finally, the data chunk contains the sample data:

Offset	Field name	Length	Contents
36	"data" description header	4 bytes	Contains the letters "data"
40	Size of data chunk	4 bytes	Number of bytes of data is included in the data section == number of samples * number of channels * bitspersample/8
44	Data	Unspecified data buffer	The actual sound data

A.2 Data Packing for PCM WAVE Files:

In single channel WAVE files, samples are stored consecutively. For stereo WAVE files, channel 0 represents the left channel and channel 1 represents the right channel. In multiple channel WAVE files, samples are interleaved. The following diagrams show the data packing for some common WAVE file formats.

Data packing for 8-bit mono PCM:

Sample1	Sample2	Sample3	Sample4
Channel 0	Channel 0	Channel 0	Channel 0

Data packing for 8-bit stereo PCM:

Sample1		Sample2	
Channel 0 (left)	Channel 1 (right)	Channel 0 (left)	Channel 1 (right)

Data packing for 16-bit mono PCM:

Sample1		Sample2	
Channel 0 Low-order byte	Channel 1 High-order byte	Channel 0 Low-order byte	Channel 1 High-order byte

Data packing for 16-bit stereo PCM:

Sample1			
Channel 0 (left)	Channel 0 (left)	Channel 1 (right)	Channel 1 (right)
Low-order byte	High-order byte	Low-order byte	High-order byte

A.3 Data Format of the Samples:

Each audio sample is contained in an integer i. The size of i is the smallest number of bytes required to contain the specified sample size. The least significant byte is stored first. The data format and maximum and minimum values for PCM waveform samples of various sizes are shown in the following table:

Sample Size	Data Format	Maximum Value	Minimum Value
One to eight bits	Unsigned integer	255	0
Nine or more bits	Signed integer	32767	-32768

Appendix

الملخص

الحماية الفعالة والآمنة للمعلومات الحساسة هو مصدر القلق الأساسي في نظم الاتصالات أو أنظمة التخزين الشبكي، فمن المهم لأي معالجه معلومات هو ضمان عدم العبث بالبيانات.و لتحقيق السرية وسلامة المعلومات والوسائط المتعددة، تم وضع عدة مخططات الحصص السرية المختلفه (SSS).

تقدم هذه الرسالة (2,n) طريقه للتقاسم الحصصي وبسريه، والذي يعتبر نوع من أنواع الحصص السريه المختلفه ونوع خاص من نظام (k,n) الحصصي السري. في نموذج مخططنا سوف يقسم الملف السري بشكل حصص بين المشاركين عدد n، وأي اثنين من المشاركين ممكن أن يشتركو لإعادة بناء الملف سري، وإلا فلن يتمكن المشاركين من استرجاع اي شيء. تم تصميم الخطة المقترحة لحل مشكلة زيادة حجم الحصص إلى جانب ضمان درجه منخفضة من التعقيد .
النموذج المقترح يتكون من وحدتين: وحدة التشفير ووحده فك الترميز، ويكون الادخال عباره عن بيانات صوت من نوع (.WAV) في حين أن الناتج هو n من الحصص. تمر مرحله التشفير عبر مرحلتين: الضغط والترميز. ويتحقق ذلك عن طريق إجراء تحويل ضغط الجيب التمام المتقطع DCT، تثبيت القيم (Quantization)، تشغيل ترميز الطول Run Length) (Encoding، وتحويل مراحل الترقيم تسلسليا (Shifting Coding). في حين يتم الحصول على التشفير عن طريق نشر المعلومات واعاده توزيع مواقعها (Diffusing)، وتوليد معاملات عشوائية (Generate Random Coefficients) لتوليد معادلات تخصيص حصص المشاركه (Generate Share Functions) على التوالي ليتم توليد عدد n من الحصص.

تم اختبار أداء الخطة المقترحة، وذلك باستخدام بعض المعايير مثل متوسط الخطأ (MSE) و نسبة الضوضاء والذروة للاشاره (PSNR) لقياس معدل الخطأ في حين يتم استخدام نسبة الانضغاط (CR) لقياس كفاءة الضغط. وكانت نتيجة الاختبار نتائج مشجعه حيث تمكنا من ضغط الفايلات وحصلنا على حصص بسعه منخفضه عن الفايلات الاصليه وبصوت واضح وعلى درجه جيده جدا من الوضوح عند الاستماع للموجه بعد استرجاعها من اي حصتين عشوائيه.

جمهورية العراق
وزارة التعليم العالي و البحث العلمي
جامعة النهرين
كلية العلوم
قسم علوم الحاسوب

تشفير الصوت الحصصي باستخدام تحويل الجيب تمام المنقطع

رسالة
مقدمة الى كلية العلوم / جامعة النهرين
كجزء من متطلبات نيل شهادة الماجستير في علوم الحاسوب

من قبل

سارة سعد علي البغدادي

(بكالوريوس ٢٠٠٣)

إشراف
الاستاذة الدكتورة

عبير هتي يوسف

نيسان ٢٠١٤ جمادي الثاني ١٤٣٥

Chapter Five
Conclusions and Suggested Future Work

5.1 Conclusions

In the previous chapters, sharing audio files using discrete cosine transform were established and its performance was tested. Various tests were performed to study the effects of the involved coding parameters on performance.

The main goal of this work is to reduce the size of the shares for the purposes of saving storage media and to speed up transmit via low bandwidth network besides guaranteeing security at low complexity.

In the following points some remarks related to the behavior and performance of the suggested Share Audio Cryptography Using DCT transform is indicated and some derived conclusions are presented too:

1. The security of proposed system is acquired by several layers which add more strength. At the first layer, the attacker should guess *secret* coefficients, *a1* and *a2*, for each share at share reconstruction phase, which made recovery secret data is time consuming for him (i.e. computationally secure method). At the second layer, is the use of diffuser to prune the existing bits significance in compressed data makes shares unbiased toward local significance; this will avoid the occurrence of localization problem.
2. If any block or blocks of one share is corrupted, we still can fully recover the whole secret file from other shares.

3. Spatial correlation property of the secret audio is eliminated when compression is performed.
4. Since the linear function is unique to a specific share, even minor changes to that share result in a dramatically different function, thereby alerting a user to potential tampering.
5. The size of shares are significantly less than the size of the secret file.
6. Table (4.9) shows Best Results the PSNR values of the reconstructed secret audio range from 40.2457 to 39.9813 dB with high MSE values.

7. Better CR without high MSE were not enough to get best PSNR values as can be seen in comparison between table (4.9) and (4.10).

5.2 Suggested Future Work

The following suggestions are recommended for future work:

1. The proposed system could be implemented on image or video instead of audio wave file.
2. Instead of using wave file, other audio formats could be used in the future.
3. In Share Process Instead of (2,n) scheme make it (k,n) scheme where k<n, this will lead to make number of random coefficients will be change according to k, where needed number of coefficients and functions will be the same as k, in order to retrieve the original file by k authorized subsets.

Chapter Four

Experimental Results and System Evaluation

4.1 Introduction

This chapter is devoted to present definition for the proposed scheme with simple example and discuss the results of the conducted tests to test the performance of the established system. The test was conducted on wave files with 1 channel and 16 bits per sample.

4.2 Mathematical Definition for the Proposed SSS (2,n)

The Goal is to distribute a Secret Compressed File (F) into n shares **{$Share_1$, $Share_2$, …., $Share_n$}** in such a way that:

- Knowledge of any 2 shares makes secret file easily computable.
- None of the shares appear to reveal information about the secret file
- Shares size will be smaller than the secret file.

Each **$Share_n$** is encoding of F using the linear function

$$S_n = v_j a_{1n} + v_{j-1} a_{2n} .$$

a_{1n} and a_{2n} are Random Coefficients linearly independent.

j = 1, 2, …, m.

m is the size of compressed file.

n is the number of shares.

v represent bytes of compressed file.

Share n items will be : { $v_1a_{1n} + v_2\, a_{2n}$, $v_3a_{1n} + v_4\, a_{2n}$, $v_5a_{1n} + v_6\, a_{2n}$, $v_7a_{1n} + v_8\, a_{2n}$, … , $v_ma_{1n} + v_{m-1}\, a_{2n}$ }.

4.3 Sharing Example

We will take wave file (F) of size 15 MB as an example to illustrate the mechanism of the proposed scheme (2,n) for shares generation and n is the number of shares.

In the proposed scheme the wave file will be compressed first. After Compression diffusion will be made for the compressed file F to prune the significance of the transformed bits by reversing the bits for each byte.

Now the shares generation will start and for this example n=3. Tow random coefficients and linear function will be generated for each single share of n shares.

Each share will be generated using random coefficients (a_1, $a_2$which should be linearly independent) with arithmetic linear function to encode the compressed and diffused file by handling its contents as 2 bytes (v_1, v_2) at each mathematical computing process for the arithmetic function till the end of the file.

Table (4.1) (example of wave file size before and after compression)

	Secret File Size	Compressed File
File	15.0 MB	8.15 MB

After run the program the main menu will appear as in figure (4.1). Select wave file for reading will be made by select Reading wave button in figure (4.2).

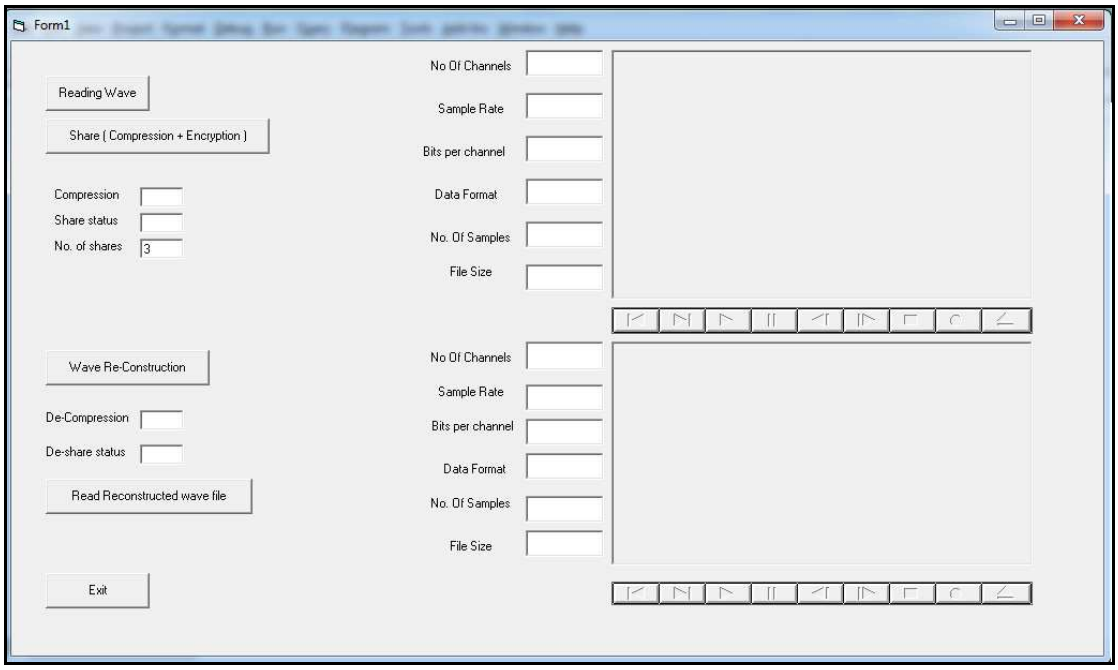


Figure 4.1 (Main menu for the proposed scheme)

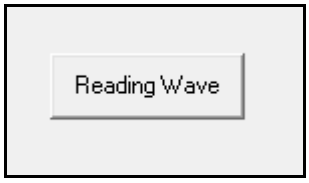


Figure 4.2 (Reading wave file button)

The figure (4.3) will appear to select wave file for reading. The data will be loaded for one dimensional array. We can play the wave file and view the shape of the signal as in figure (4.4).

After Reading the file the compression phase will began by press the button in figure (4.5).

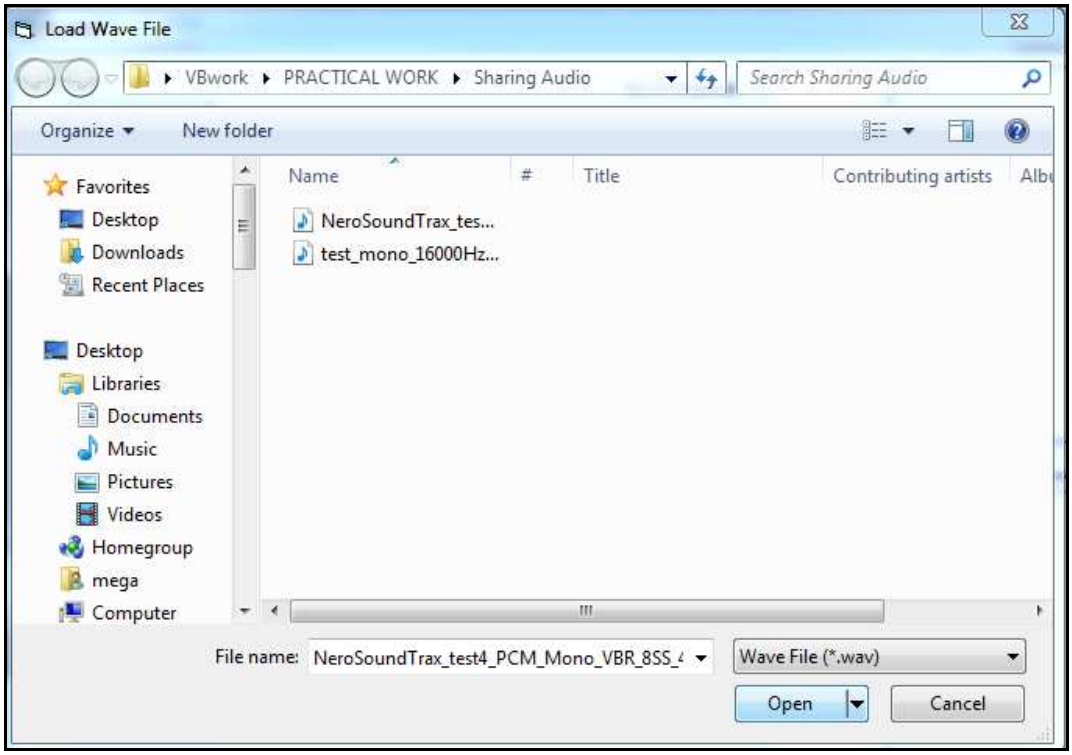


Figure 4.3 (Loading wave file)

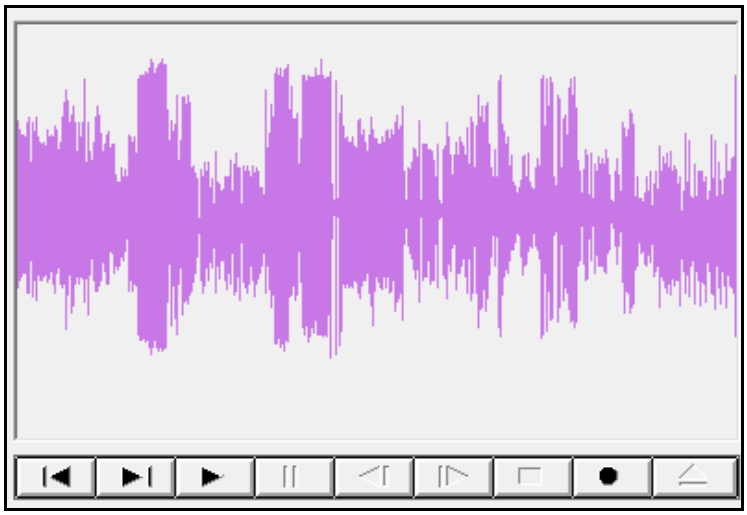


Figure 4.4 (Play the wave file)

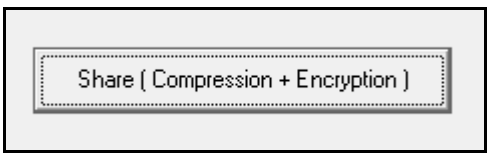


Figure 4.5 (Start (Sharing + Compression) Processing)

Now the proposed compression methods will be executed (DCT, Quantization, Zeros Run Length and Shifting Code). The result will be stream of bits and will be saved in a file.

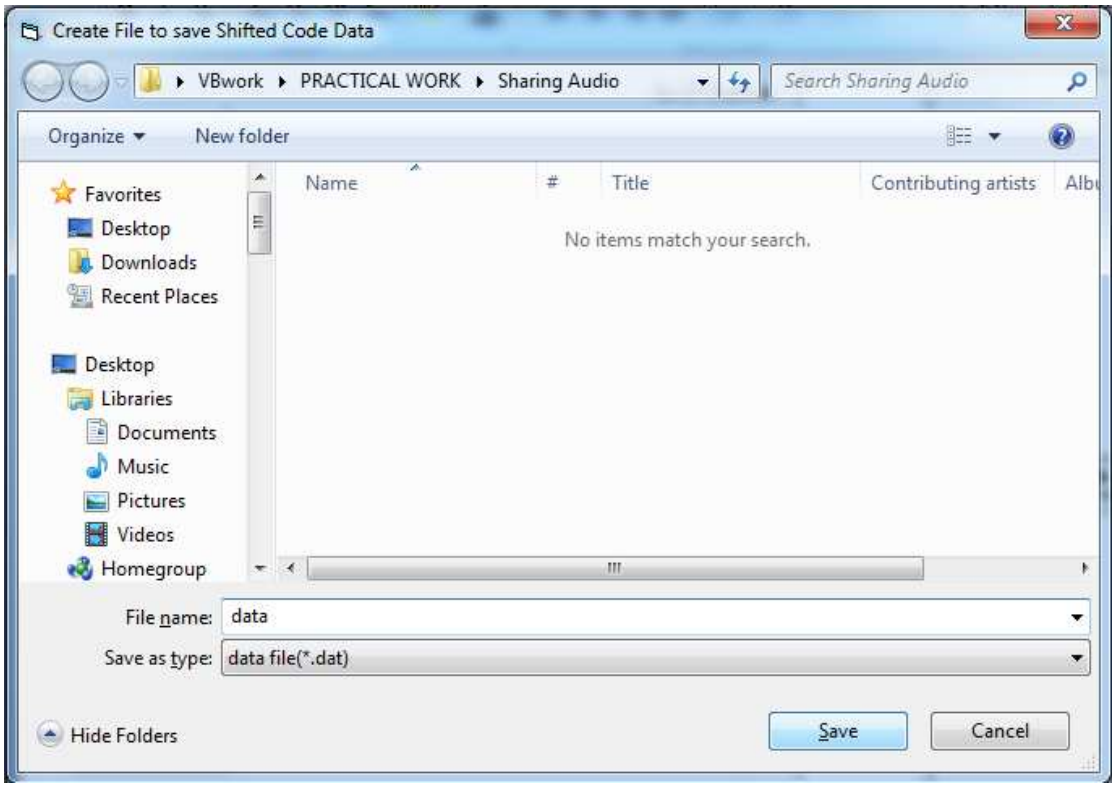


Figure 4.6 (Saving data in file data.dat after shifting Code)

"ok" will appear as in figure 4.7 to declare that compression phase are finished. After Compression the encoding will start with diffusion and then the share process will start by Creating file, generating tow random coefficients and linear function for each share.

Figure 4.7 (Sharing and compression status)

Table (4.2) (generation of linear functions with tow random coefficients for each share, n=3)

	$Share_1$	$Share_2$	$Share_3$
Function	$S_1 = v_j a_{11} + v_{j+1} a_{21}$	$S_2 = v_j a_{12} + v_{j+1} a_{22}$	$S_3 = v_j a_{13} + v_{j+1} a_{23}$
(a_1, a_2)	(48, 54)	(43, 31)	(29, 61)

Each file will represent one single share and will be filled By handling diffused and compressed data as 2 bytes (v_1, v_2) at each mathematical computing process for the arithmetic function till the end of the file. "ok" will appear as in figure 4.7 to declare that encoding process is completed (diffusion + sharing).

Now the size of the generated shares will be smaller than the original file size as noticed in table (4.3).

Table (4.3) (wave file size with share size after shares generation in the proposed scheme)

	Secret File Size	Share size
File	15.0 MB	8.15 MB

For Re-Construction process press the button wave reconstruction in figure (4.8) and choose any tow shares of the generated shares to start the process. First the de-share operation will start using arithmetic functions to produce items of the diffused and compressed data and then inverse the diffusion. "ok" will appear as in figure 4.9 to declare that de-share process are finished. De-Compression will start by inverse the methods shifting code, zeros run length, Quantization and DCT. "ok" will appear

as in figure 4.9 to declare that reconstruction of the original file are finished.

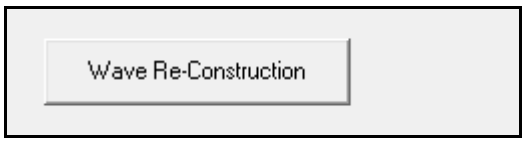


Figure 4.8 (Wave reconstruction button)

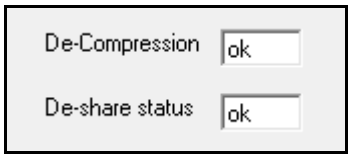


Figure 4.9 (Wave reconstruction status button)

Now press the button in figure 4.10 to play the reconstructed file and listen to the wave file.

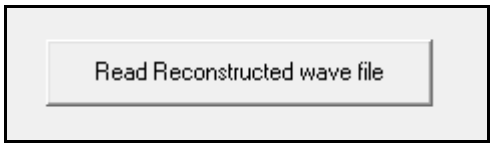


Figure 4.10 (Reading Re-Constructed wave file)

4.4 The Test Measures

Different key parameters were utilized in the literature to describe the performance of various compression methods. In this research the fidelity criteria (MSE and PSNR) in addition to the Compression Ratio were used to describe the performance of the established compressed stage at different coding conditions.

4.4.1 Fidelity Measures

There are several types of matching criteria, among which the Mean Square Error (MSE) and Mean Absolute Difference (MAD) are used most often. It is noted that the Sum of the squared Difference (SSD) or the Sum of the Squared Error (SSE) is essentially same as MSE. The Mean Absolute Difference is sometimes referred to as the Mean Absolute Error [Salo07].

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(s_i - s_i')^2 \qquad (4.1)$$

Where,

s_i is the i^{th} sample in the original wave data.

s_i' is the corresponding i^{th} sample in reconstructed wave data.

M is the total number of audio samples.

Developers of compression methods need a standard metric to measure the quality of reconstructed files compared with the original ones. Well reconstructed file resembles the original one, and the metric value should indicate this resemblance in proper way. Such a metric is a dimensionless number, and that number should not be very sensitive to small variations in the reconstructed file. The most common measure used for this purpose is the *Peak Signal to Noise Ratio*(PSNR). It is familiar to workers in the field, it is also simple to calculate, but it has only a limited, approximate relationship with the perceived errors noticed by the human. This is why higher PSNR values imply closer resemblance between the reconstructed and the original files, but they do not provide a

guarantee that viewers or listeners will like the reconstructed files[Salo07].

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \quad (4.2)$$

PSNR values are used only to compare the performance of different lossy compression methods, or to describe the effects of different parametric values on the performance of an algorithm. In this research the fidelity criteria used to assess the performance of the proposed system are MSE and PSNR. The performance tests were performed on the proposed system by using some selected audio files as test material.

4.4.2. Compression Compactness

A very logical way of measuring how well a compression algorithm compresses a given set of data is to look at the ratio of the number of bits required to represent the data before compression to the number of bits required to represent the data after compression. This ratio is called Compression ratio [Umba98].

CR= Uncompressed Data size / Compress file size (4.3)

CR measurement has been used as indicator for the compactness ability of the established compression scheme in this research project.The system will compress the file size during the compression phase and will produce coded minimized file size ready for encryption phase. During encryption and specifically in sharing process, any 2 files will be able to reconstruct the wave file with size equal to the original file size.

4.5 Test Parameters

In DCT method, the control parameters that have significant effects on the transformation and ratio of the correctly retrieved secret bits; are: **Block size** value and **quantization step**.

The effects of the involved control parameters (Block length and Quantization Step values) have been studied; specifically their effects on the compression process in order to minimize the shared file size to construct smaller shares than the original file size and to accomplish the sharing process.

4.5.1 Block Length Test

The conducted test had been applied on an audio data array, this array was partitioned into blocks whose size is BL (12, 16 and 20) samples. Table (4.4) shows the obtained test results.

Table (4.4) (The effect of the block size (BL) on the performance of Compression of wave files used)

Compressed File size in KB			Original File Size in KB	ID
BL = 20	BL = 16	BL = 12		
73.7	75.1	78.3	80.6	File 1
74.2	86.2	98.3	162	File 2
88.4	102	116	193	File 3
124	141	165	253	File 4
171	186	211	310	File 5
499	550	662	848	File 6

4.5.2 Quantization Tests

The main parameters used to manage quantization Process is Qstp and weight values in equation (3.1) as shown in tables (4.5) and (4.7). If

the value of *weight* is increased it causes an increase in the distortion level of signal.

a) In this test **Qstp** in equation (3.1) will set to 1 and **weight** is set to 0.1, with 2 test of **BL** 12 and 16.

Table (4.5) (The effect of the Quantization on the performance of Compression of wave files used, Qst ( ) Vector dimension at equation (3.1) is set to block length 12,Qstp =1 and Weight is set to 0.1)

PSNR	MSE	CR	compress file KB	original size KB	
51.418	0.46912	1.02937	78.3	80.6	File 1
52.0561	0.40501	1.64802	98.3	162	File 2
52.0274	0.4077	1.66379	116	193	File 3
51.4056	0.47046	1.53333	165	253	File 4
51.5307	0.4571	1.46919	211	310	File 5
51.2343	0.48938	1.28097	662	848	File 6

Table (4.6) (The effect of the Quantization on the performance of Compression of wave files used, Qst ( ) Vector dimension at equation (3.1) is set to block length 16 ,Qstp =1 and Weight is set to 0.1)

PSNR	MSE	CR	compress file KB	original size KB	
51.134	0.50082	1.07324	75.1	80.6	File 1
51.7902	0.43059	1.87935	86.2	162	File 2
51.8053	0.42909	1.89216	102	193	File 3
51.9653	0.41357	1.79433	141	253	File 4
51.773	0.4323	1.66667	186	310	File 5
51.7798	0.43161	1.54182	550	848	File 6

b) In this test **Qstp** in equation (3.1) will set to 1 and **weight** is set to 0.2, with 2 test of **BL** 12 and 16.

Table (4.7) (The effect of the Quantization on the performance of Compression of wave files used, Qst ( ) Vector dimension at equation (3.1) is set to block length 12,Qstp =1 and Weight is set to 0.2)

PSNR	MSE	CR	compress file KB	original size KB	
50.9766	**0.5193**	**1.25741**	64.1	80.6	File 1
52.906	0.33303	2.22222	72.9	162	File 2
52.8957	0.33382	2.22094	86.9	193	File 3
52.6946	0.34964	2.024	125	253	File 4
52.2664	0.38587	1.76136	176	310	File 5
51.9441	0.4156	1.60911	527	848	File 6

Table (4.7) gives better Compression Rate results than table (4.5) with weight value equal to 0.2 rather than 0.1under the same conditions, but the PSNR still in fifties with less MSE values except in case of File 1 although its CR is not big compared to other files.

Table (4.8) (The effect of the Quantization on the performance of Compression of wave files used, Qst ( ) Vector dimension at equation (3.1) is set to block length 16,Qstp =1 and Weight is set to 0.2)

PSNR	MSE	CR	compress file KB	original size KB	
50.3753	**0.59642**	**1.121**	71.9	80.6	File 1
53.0854	0.31955	2.62136	61.8	162	File 2
53.1116	0.31763	2.6584	72.6	193	File 3
52.5632	0.36038	2.34259	108	253	File 4
55.2394	0.1946	1.9375	160	310	File 5

51.6562	0.44408	1.696	500	848	File 6

c) In this test **Qstp** in equation (3.1) will set to 1 and **weight** is set to 0.5, with 2 test of **BL** 12 and 16.

Table (4.9) (The effect of the Quantization on the performance of Compression of wave files used, Qst ( ) Vector dimension at equation (3.1) is set to block length 12,Qstp =1.5 and Weight is set to 0.5)

PSNR	MSE	CR	compress file KB	original size KB	
40.0751	**6.390955**	**1.3021**	61.9	80.6	File 1
39.9813	**6.530509**	**2.7**	60	162	File 2
39.9822	**6.529173**	**2.73759**	70.5	193	File 3
39.9834	**6.527369**	**2.38679**	106	253	File 4
40.1959	**6.215679**	**1.97452**	157	310	File 5
40.2457	**6.144837**	**1.75207**	484	848	File 6

Table (4.10) (The effect of the Quantization on the performance of Compression of wave files used, Qst ( ) Vector dimension at equation (3.1) is set to block length 16,Qstp =1.5 and Weight is set to 0.5)

PSNR	MSE	CR	compress file KB	original size KB	
49.4129	0.74438	1.31485	61.3	80.6	File 1
51.9068	0.41918	3.15789	51.3	162	File 2
49.2973	0.76445	2.99225	64.5	193	File 3
51.5939	0.4505	3.04087	83.2	253	File 4
50.4903	0.58084	2.31343	134	310	File 5
50.1662	0.62583	2.03357	417	848	File 6

Table (4.9) gives better MSE and PSNR results than table (4.10) with BL equal to 12 rather than 16 under the same conditions, the PSNR values decreased to acceptable level under fifties and MSE values increased While in table (4.10) High CR are given with less MSE and high PSNR values.

4.6 The Effect of using Zeros Run Length.

Run-length encoding for a data sequence having frequent runs of *zeroes*, each time a zero is encountered in the input data, *two* values are written to the output file. The first of these values is a zero, a flag to indicate that run-length compression is beginning. The second value is the number of zeros in the run. Since our wave data will have large number of consecutive zeros the data size can be greatly reduced using RLE therefore this compression will be useful.

4.7 The Effect of using Shifting Code

The binary code words, produced by applying shift encoder on encoded data, are saved in the file. Algorithm (3.5) shows the steps to implement the shift encoding, taking into consideration the following remarks:

a) Before applying shift coding the data have been mapped into positive values, which is a necessary condition to conduct shift encoding.
b) Before a applying shift encoding the proper size of its code words should be determined, so in algorithm (3.5) a simple optimization technique is implemented, it is based on testing all possible codeword sizes to find out the best size that lead to lowest consumption in bits (i.e. lowest output size).

4.8 The Effect of Sharing Random Coefficients.

Random Coefficients will be used as a key for each share, proposed scheme in our research is (2,*n*) sharing scheme therefore to reconstruct the original file any two share files will be used to retrieve the original wave file as shown in algorithm (3.9).

Since each share will be generated using arithmetic function with coefficients belong to that share therefore to guarantee that each share will be distinctly different than the other, Random Coefficients which assigned for each share will be completely different. During share generation, the Compressed diffused data we use the following Functions:

$$Share_1 = v1a_{11} + v2\ a_{21}$$

$$Share_2 = v1\ a_{12} + v2\ a_{22}$$

Now to calculate v1 and v2 during the de-share process we could derive their values arithmetically by using the following equation (4.4) and (4.5):

$$v1 = (Share_1\ a_{22} - Share_2\ a_{21}) \setminus (a_{11}\ a_{22} - a_{21}\ a_{12}) \qquad (4.4)$$

$$v2 = (Share_2\ a_{11} - Share_1\ a_{12}) \setminus (a_{11}\ a_{22} - a_{21}\ a_{12}) \qquad (4.5)$$

As in equations (4.4) and (4.5), Two shares should be pooled in order to reconstruct the compressed diffused file and if attacker have one share the he should use N^2 Random choice for each $Share_2$, a_{21} *and* a_{22} to try to reveal the Second share.

4.9 Test Results

1. The increase in the block length parameter causes an increase in CR and a decrease in PSNR.
2. The increase in the value of MSE causes a little increase in Cr value, and a good decrease in PSNR values as shown in table (4.9).
3. The PSNR value decreased and the C.R value increased depend on decreasing the number of compressed bits.
4. Best Result of PSNR at BL 12, Qstp 1.5 and Weight 0.5 in table (4.9), While increasing the BL to 16 under the same conditions only gives best result in CR but less quality for the reconstructed file.

Chapter One
Introduction

1.1 Motivation

Due to the recent development of computers and computer networks, huge amount of digital data can easily be transmitted or stored. However, it is noted that transmitted data in networks or stored data in computers may easily be eavesdropped or substituted by enemies if the data are not enciphered by some cryptographic tools. Therefore, encryption methods are one of the popular approaches to ensure the integrity and secrecy of the protected information. However, one of the critical vulnerabilities of encryption techniques is single-point-failure, for example, the secret information cannot be recovered if the decryption key is lost or the encrypted content is corrupted during the transmission. In addition, the possession of an extremely sensitive key by an individual may not be advisable as the individual may not be fully trusted. To address these reliability problems, a Secret Sharing Scheme (SSS) is a better alternative to take care of such vulnerabilities.

Secret Sharing Scheme deals with problem, namely sharing a highly sensitive secret among a group of n users so that only when a sufficient number k of them come together, the secret can be reconstructed, that reduces the risk of losing the key but also makes compromising the key more difficult. [CHA13]

1.2 Concept of Secret Sharing Scheme (SSS)

A Secret Sharing Scheme (SSS) is a process or a protocol that divides a secret given as input into pieces called shares such that only specific subset of shares allow reconstruction of the original secret [Men96].

More formal, it is a pair of algorithms (**Share, Rec**)*, where:*

1. **Share**(*S,m*) *is a randomized sharing algorithm that on input a secret S outputs a set of m shares* $\{s_1,\ldots,s_m\}$;

2. **Rec**($s_{i1},\ldots,s_{it}$)*,* $t \leq m$ *is a deterministic reconstruction algorithm from shares that outputs S if* $\{s_{i1},\ldots,s_{it}\}$ *is authorized and halts otherwise.*

A set of shares that allows reconstruction is addressed as *authorized* or *qualified*. The set of all authorized sets of shares is called the *access structure* and it is denoted by ***AS***. Usually the shares are distributed to distinct participants and hence they also refer to *authorized* (or qualified) set of users as the ones able to reconstruct the secret by putting their shares together; no other set of parties should be able to disclose it. An access structure is called *monotone* if any set containing an authorized set is also authorized. Informally, this means that if a group of users can recover the secret, then no matter how many others members join, the extended group will also be able to determine it [Smi06].

1.3 Classification of Secret Sharing Scheme

Secret sharing schemes are characterized along multiple independent dimensions; these are [Bei96, Hof08, Guo13]:

A. **Quantity Dimension**. Depending on the "*quantity*" of the secret-information leaked to an unauthorized group, secret sharing schemes can be classified as *Perfect Secret Sharing* and *Computational secure secret sharing*:

i. ***Perfect Secret Sharing***. A secret sharing scheme is *perfect* if the shares corresponding to each unauthorized subset provide absolutely no information about the secret [Oli13]. More formally, a perfect SSS:

- permits perfect reconstruction for each authorized set, i.e. $H[S|A] = 0$, for all $A \in AS$;
- discloses no information about the secret for each unauthorized set, i.e. $H[S|B] = H[S]$, for all $B \notin AS$.

ii. ***Computational secure secret sharing***. A secret sharing scheme is *computational secure* if some information about the secret is leaked to the unauthorized groups, but the problem of finding the secret is intractable. SS scheme could be based on any assumption of computational difficulties like the factorization of integers or the calculation of discrete logarithms.

B. **Share Dimension**. Based on the *dimension of the shares*, SSS categorize into: *idea*l and *non-ideal* [Hof08].

i. A secret sharing scheme is *ideal* if it is perfect and its information rate equals 1, (the *information rate* of a user is defined as the size of the shared secret divided by the size of the user's share), in other words, the size of each share is equal to the size of the secret.

ii. In order to be practical, the amount of secret information distributed as shares should be as small as possible in *non-ideal* scheme. For perfect secret sharing, the size of any share is larger than the size of the shared secret. Therefore, the information ratio is upper bounded by 1, which becomes the optimal case.

C. **Access structure Dimension**. Considering the *access structure*, SSS classify into: Special SSS and General SSS [Guo13].

i. Formal definition of *general secret sharing* is: Let P = $\{p_1,p_2,\ldots,p_n\}$ be the set of participants, and let $\Gamma = (\Gamma 1; \Gamma 2; \ldots; \Gamma m)$ be an m-tuple of access structures on the set of P, where m

$\leq 2^{|P|-1}$. The secret data can be shared among these n participants, and each participant holds one piece of information relating to the secret data. Each qualified subset Γ , $1 \leq i \leq m$ of *P*, pre-determined according to the access structures $\Gamma = (\Gamma 1; \Gamma 2; \ldots ; \Gamma m)$, can cooperate to reconstruct the shared secret data. In General SSS, the access structure is not restricted in any way, except (usually) monotony, as required for the majority of practical schemes.

ii. Special SSS The access structure satisfies some specific properties. These are:

- *Threshold SSS*. The access structure contains all sets with the cardinal at least *t*, $1 \leq t \leq m$. Informally any *t* or more shares are enough to restore the secret, while few than *t* shares are not. We denote such a scheme by (t; m)-secret sharing;
- *Unanimous* (or *all-or-nothing*) SSS. The access structure contains only the set of all shares. Informally, all shares are required for secret reconstruction. Unanimous SSS is a (*m*; *m*)-threshold SSS.

D. **Type of the shared secret**. Depending on the *type of the shared secret*, SSS separate into:

i. Bit-String SSS. The shared secret is a sequence of bits, such as an element from a finite field or a text. These are the most popular SSS and present a high utility in practice.

ii. Visual SSS (VSSS). The shared secret is an image. A *visual secret sharing* (VSS) scheme, which originates from the *visual cryptography* proposed by Naor-Shamir [Nao94], may be one of the most well-known realization of SS schemes. The VSS scheme is a method to encode a secret image into

several shares, each of which does not reveal any information of the secret image. Each share is printed on a transparency, and is distributed to one of n participants. The secret image can easily be decrypted only by stacking the shares in an arbitrary order.

iii. Audio or Video SSS. The shared secret is an audio or video file. An Audio Secret Sharing (ASS) scheme is a special type of secret sharing scheme, where the shares of embedded messages and/or the secret are/is audio file(s). It is likely that the secret is not a bit string or visual and it deals with audio secrets. In these cases, it could be desired to reconstruct the audio secret without any computation. In a perfect ASSS with an audio secret the following conditions hold [Ehd08]:

- Every authorized subset could recognize the audio secret by pooling their shares together.
- Every unauthorized subset could not reach a nonrandom audio file by pooling their shares together. (i.e. the mutual information between the secret file and the obtained file by an unauthorized subset should be zero.)

1.4 Problem statement

One of the most important issues when designing secret sharing schemes is the **size** of the pieces. Secret sharing schemes usually have information theoretic security which implies that each share be as large as the secret [Des93]. So, in the case were the secret is N bits and is shared among K people who must all come together to recover the secret, each share must be at least N bits because it is required that information obtained must be zero any K-1 users working together. Any scheme overcome this bound must leak partial information of the secret. [Che12]

Satisfying the above condition becomes a problem, especially when the secret is multimedia content, even with the best known schemes. Because the parties will not have enough memory to store their pieces even in fairly small networks or are able to successfully transmit it through a low bandwidth network.

A second major problem with secret-sharing schemes is that the shares' size in the best known secret-sharing schemes realizing *general access structures* is exponential in the number of parties in the access structure. Thus, the known constructions for general access structures are impractical. On the other hand, the best known lower bounds on the shares' size for sharing a secret with respect to an access are far from the above upper bounds. The best lower bound was proved by [Csi96], proving that, for every n, there is an access structure with n parties such that sharing l-bit secrets requires shares of length $\Omega(ln/\log n)$. So, the question if there exist more efficient schemes handle space problem remains open [Bei96].

1.5 Previous Work

Recent works that deal with Audio Secret Sharing have used different methods and with different efficiencies. The following are some studies in this field:

- Yang [Yan02], proposed two methods (Construction 1, Construction 2) to construct the 2-out-of-n audio cryptography scheme with only one cover sound, and one method (Construction 3) to construct the 2-out-of-n optical cryptography scheme with only one cover image.
- Lin *et al*. **[Lin03]**, they proposed two new (2, n) ASS schemes, which carefully employ the technique of time division with only one cover sound. Comparing with the first scheme, the second scheme has the advantage of flexible improvement in relative contrast as needed. To test

the acoustic result, they implemented those two proposed (2, *n*) ASS schemes for small *n* using one wave-type cover sound and then obtained near expected results.

- Socek and Magliveras **[Soc05]**, propose and analyze a new type of cryptographic scheme, which extends principles of secret sharing to Morse code-like audio signals. The proposed "audio cryptography scheme" (ACS) is perfectly secure and easy to implement. It relies on the human auditory system for decoding. Audio sharing scheme (ASS) proposed earlier were based on disguising secret binary message with a cover sound. Moreover, only 2-out-of-n audio sharing schemes have ever been proposed. Their scheme correlates strongly, and is analogous to schemes in well-studied visual cryptography. Consequently، they were able to use the existing visual cryptography constructions and obtain not only *k*-out-of-*n* audio sharing schemes, but also the most general audio cryptography schemes for qualified subsets.
- Fugita *et al.* **[Fuj06]**, proposed Audio Secret sharing scheme, in this scheme to reconstruct secret or get the secret, a decoder is required. It is not possible directly playing all shares sound simultaneously.
- Kurihara *et al.* **[Kur08],** proposed a new fast (*k*, *n*)-threshold scheme which uses just EXCLUSIVE-OR(XOR) operations to make *n* shares and recover the secret from *k* shares. They prove that every combination of *k* or more participants can recover the secret, but every group of less than *k* participants cannot obtain any information about the secret in the proposed scheme. Moreover, the proposed scheme is an ideal secret sharing scheme similar to Shamir's scheme, in which every bit-size of shares equals that of the secret. They also evaluate the efficiency of the scheme.

- Ehdaie *et al*. **[Ehd08]**, a new audio secret sharing scheme which is secure and ideal is proposed. This scheme is (k, n) threshold for $k \geq 2$, where the previous schemes were (2, n). It is assumed that both ”shares” and ”secret‘” are audio files instead of a bit string secret. The audio secret is reconstructed without any computation that is only by playing audio shares simultaneously. Moreover, the simulation results show that the new scheme is not sensitive to audio noise.
- Nikam *et al*. **[Nik10]**, new (2, 2) Audio Cryptography Scheme is proposed, the proposed scheme hides a digital secret message into two specified cadence. The original secret wave file is perceived from both cadences playing simultaneously.
- Naskar *et al*. **[Nas11],** suggested a novel secret sharing scheme which employs simple graphical masking method, performed by simple ANDing for share generation and reconstruction can be done by performing simple ORing the qualified set of legitimate shares. Not only that, the generated shares are compressed and each share contains partial secret information that leads to added protection to the secret and reduced bandwidth requirement for transmission.
- Shaw **[Sha12]** a modern approach for secured transmission of audio files using fractal based chaos is proposed, which is based on audio secret sharing scheme and cryptography. This scheme uses a powerful encryption algorithm in the first level of security, which is very complex to break. In the second level it uses a more powerful secret sharing scheme to divide information into several pieces such that certain subsets of these pieces (shares) can be used to recover the information. If intruders want to get the information then several shares need to be theft. If intruders want to destroy the information then several shares need to be destroyed. This scheme have introduced a new robust and fractal based

cryptography with secret sharing scheme which provides low computational complexity during both sharing and reconstruction phases. It also reduces the bandwidth of transmission medium and provides strong protection of the secret file and the compression rates vary depending upon the threshold value.

- Chaudhuri **[Cha13]**, proposed a new robust and secured Secret Sharing scheme which is equally applicable for any file formats (e.g. Image, Audio, and Text etc.) and it provides low computational complexity during both sharing and reconstruction phases. The proposed scheme employs simple graphical masking method, done by simple ANDing for share generation and reconstruction can be done by simple ORing the qualified set of shares. Thus his scheme ensures minimal computational complexity compared to the earlier proposed schemes thus makes it effective for addressing energy saving distributed environment where battery driven low end processors are used and security is also a major challenge.

1.6 Aim of thesis

This project aims to establish a new secret sharing scheme capable of protecting Audio data using DCT transform, The proposed encryption scheme generates Shares by combining bit-level decomposition/stacking with a {2, n}-threshold sharing strategy. Perfect reconstruction is achieved by performing decryption through simple operations based on linear equations without the need for any post processing operations. The use of DCT domain will be useful to reduce the total size of shares due to its compactness capabilities. Since the low frequency coefficients of DCT transform hold a significant part of the audio signal energy; which in turn causes the significance localization problem, a set of hashing functions

will be used to prune the significance of the transform bits to overcome this problem.

1.7 Thesis Organization

Beside Chapter One, the remaining part of this thesis consists of four chapters:

Chapter 2:-*"Principles of Secret sharing Schemes and Compression"*

This chapter presents the background of the used threshold secret sharing schemes. The well-known and commonly used Stages and algorithms in data compression based on DCT are also demonstrated.

Chapter 3:-*"The Proposed Scheme"*

This chapter covers the details of the developed *Audio Secret Sharing Scheme*; their stages and steps. Also, a description for the implementation of each step is described. Also, some examples are given to illustrate the performance of the suggested methods to handle each system task.

Chapter 4:- *"Experimental Results and System Evaluation"*

This chapter presents the results of experimental analysis of some tests applied to define the compression performance of the scheme in addition to its secrecy capability.

Chapter 5:-*"Conclusions and Suggested Future work"*

This chapter holds a list of some conclusions after implementing the proposed scheme, and it gives some suggestions for future work to enhance the presented system.

Chapter Three
The Proposed Scheme

3.1 Introduction

Secret sharing is a method to distribute a secret between some participants such that particular subsets (i.e. authorized subsets) could obtain the secret, whereas unauthorized subsets could not.

An Audio Sharing Scheme is proposed in this thesis. This scheme is (2, *n*) threshold, 2 is the number of participants who can reconstruct the audio file together, and *n* is the total number of generated shares which is greater than 2.

This chapter is dedicated to present the design considerations, implementation requirements and the steps taken throughout the establishment of the proposed sharing scheme. The implementation steps are illustrated in details using diagrams and/or pseudo code.

3.2 Design Considerations

The main goal of the proposed encryption scheme is to reduce the size of secret data before the sharing process take place to preserve a sufficient level of security and confidentiality of audio data under real time constraints, the proposed sharing scheme should be perfect scheme.

3.3 Tools and Requirements

The proposed encryption scheme is implemented using Visual Basic 6 computer language. It is executed for testing purpose on computer with processor of 2.2 GHZ dual core, 3 Giga Byte of RAM under Microsoft Windows 7 operating system.

3.4 The proposed Scheme Structure

The proposed scheme prototype consists of two modules: *Encoder* and *Decoder*. The input to the former is plain audio data (.WAV) while the output is *n* shares. Both modules perform the same functions but in *opposite* way as shown in Figures (3.1), (3.2).

Encoder is passes through two stages: ***compression*** and ***encryption***. The main functions of *Encoder module* are listed below:

Stage A- Compression functions:

The main functions of *Compression module* are listed below and shown in Figure (3.3)

1- Loading and Reading a wave data.
2- Decompose wave data into Blocks
3- Transforming each block from spatial to frequency domain by adopting DCT.
4- Quantizing DCT coefficients to the nearest integer value.
5- Applying Run-Length Encoding (Spatial Encoder).
6- Performing Shifting Code (Entropy Encoder).

Stage B- Encryption Functions:

The main functions of *Encryption module* are listed below and shown in Figure (3.4)**.**

1- Diffusion.
2- Generating Random Coefficients.
3- Generating Share Functions.
4- Generating *n* shares.
5- Distributing *n* shares.

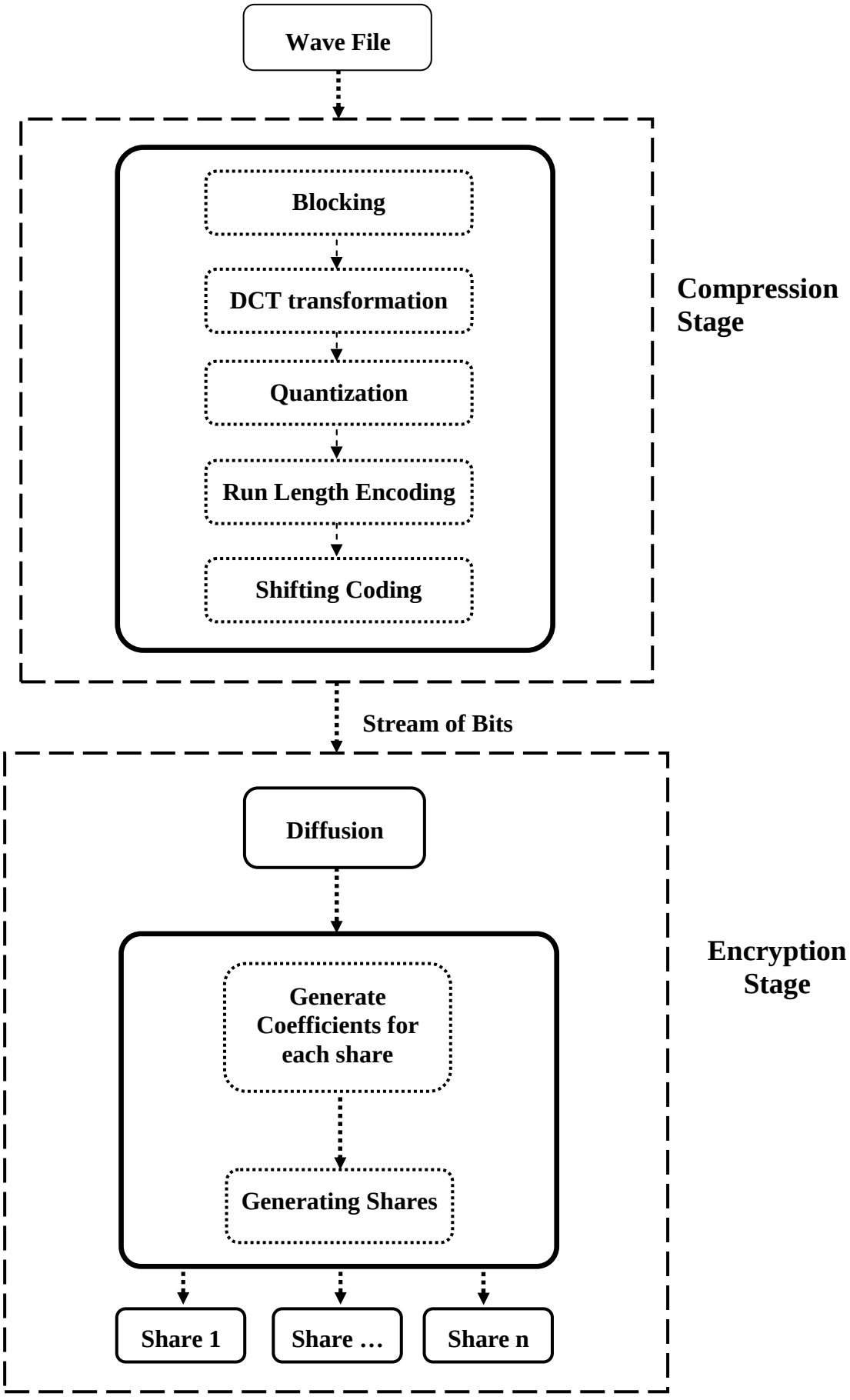


Figure (3.1) The Block Diagram of Encoding Module

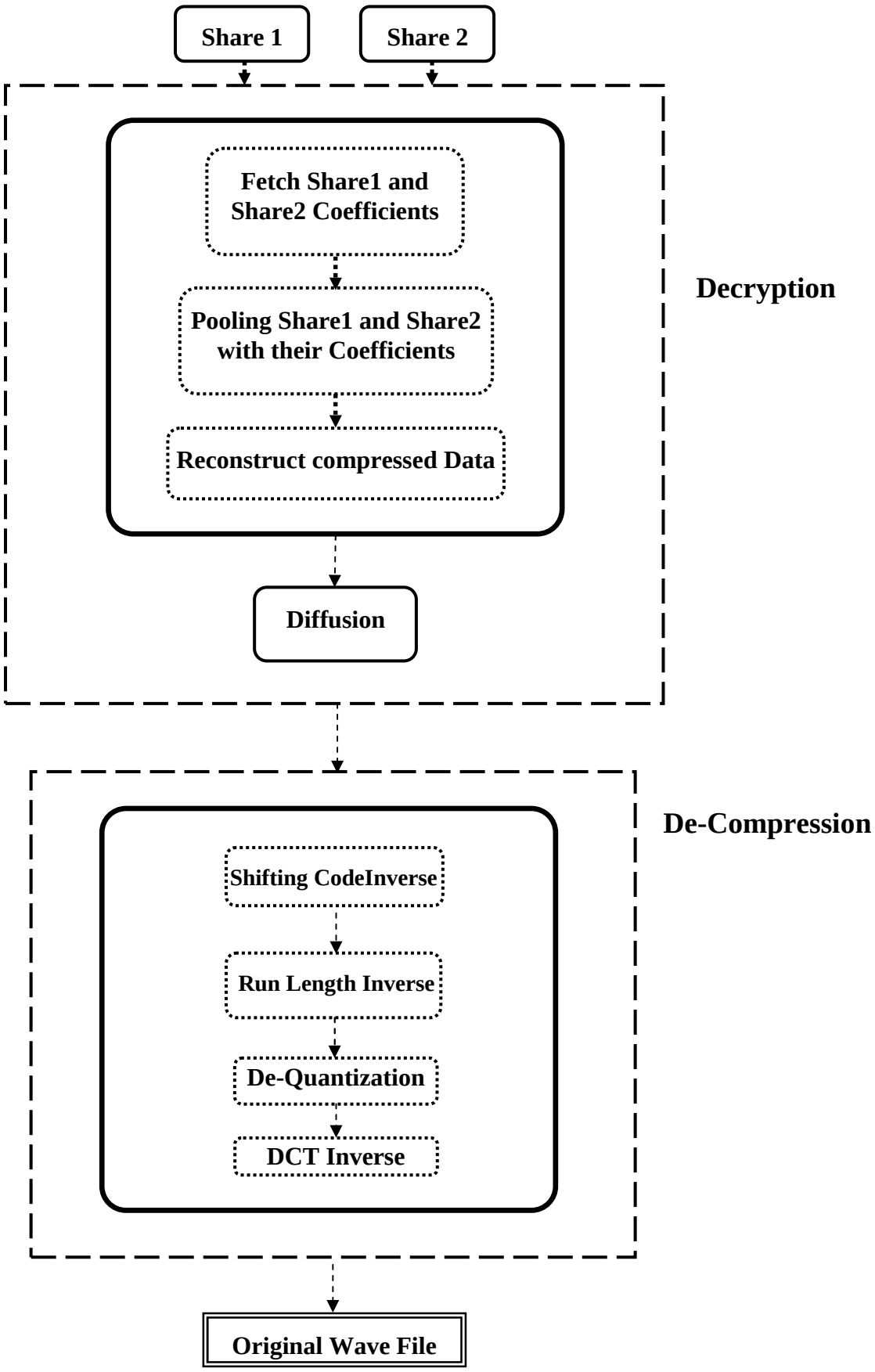


Figure (3.2) The Block diagram Of Decoding Module

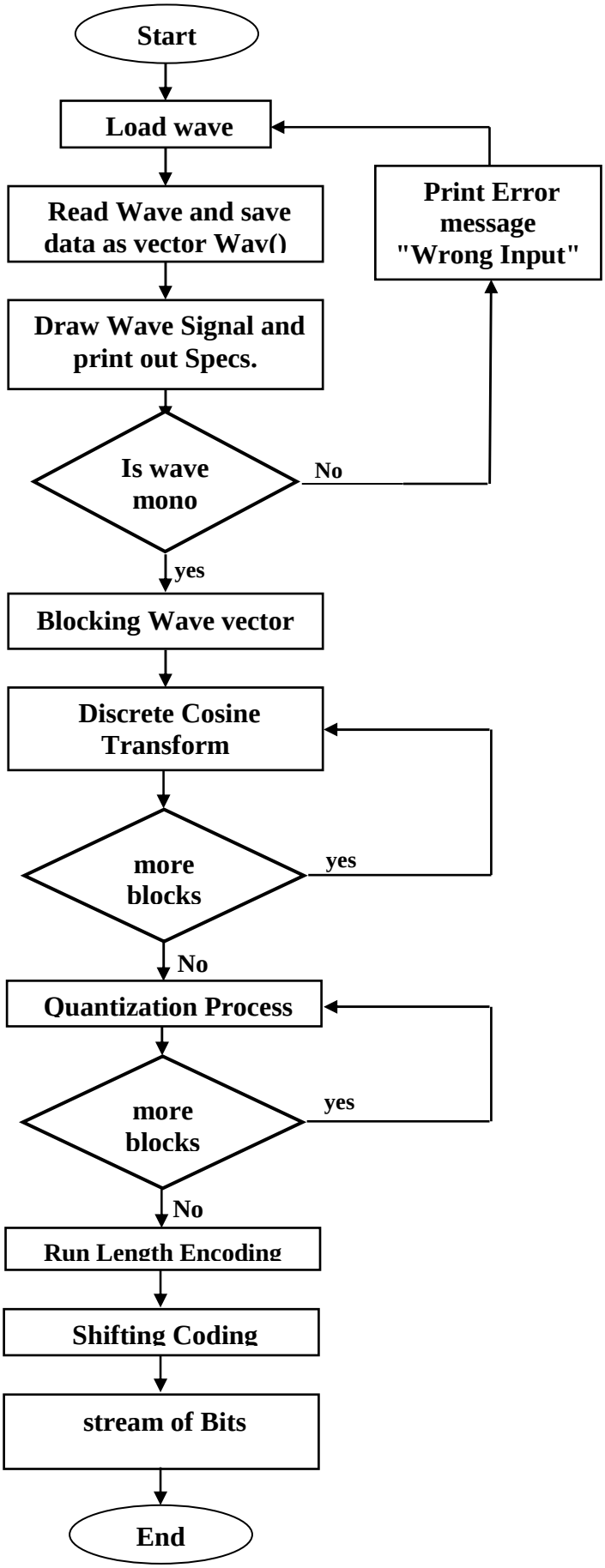


Figure (3.3) The Flowchart OF Compression

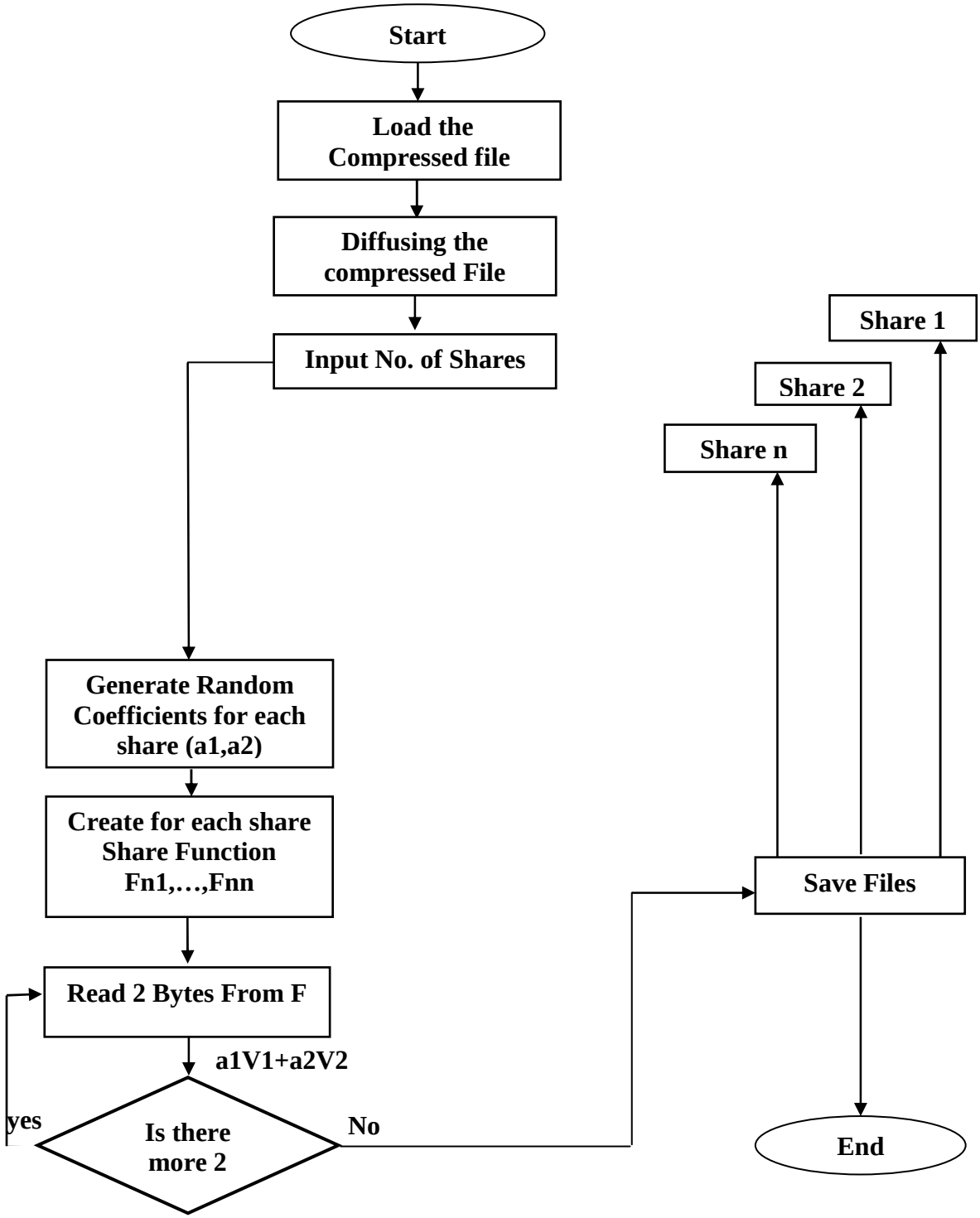


Figure (3.4) Encryption Flowchart

3.4.1 Wave Loading

The wave file (.WAV) is selected in the current work to understand its structure see Appendix A. This type of wave files is considered to be the simplest form of audio format; it could be simply converted to any other forms of audio files. To load the actual data, the following steps should occur:

- First File header is read;
- Second File header is analyzed;
- Third the actual raster data is extracted

In the recent work, the wave file is chosen to be "mono" and the number of bits per sample is equal to twelve, algorithm (3.1) presents this function.

Algorithm (3.1) Loading Original Wave File

Input: Original Wave file
Output: Wav ( ) ' Array of integers contains the data samples of the original wave file
Nm ' Number of samples

Procedure

Step1: **Read** the header of the wave file
Step2: **Read** Data format from the header
Check if data format = 1 ' means the format of the data is PCM"
Read Channel Number from the header
Read Sample Resolution from the header
Check if channel Number = 1 then
Check If Sample Resolution = 8 then ' mean the channel type is mono
Read the Chunk Size of fmt chunk from the header
Check If Chunk Size of fmt chunk > 16 then
Jump to location after (Chunk Size -17)
End if
Read Chunk ID
Read Data Size

Step3: **Set** L=47 ' jump to location after header types until data chunk
While Chunk ID <> " data"
Set L = L + Data size
Seek to L in file

```
                    Read Chunk ID
                    Read Data Size
                  End While
                Set Nm = Data Size
                Read original data from data chunk then store it in Wav ( ) array
              Else
                Printout " Wrong Sample Resolution: Choose another wave file"
              End if
            Else
              Printout " Wrong Channel type: Choose another wave file
          End if
        End if
Step4: Return (Wav , Nm)
End Procedure
```

3.4.2 Decompose Wave Data into Blocks

After loading the mono wave data into one dimension array, decomposing function will executed. The audio channel data will be divided into small blocks with block size equal twelve because each sub-block in the source encoder exploits some redundancy in the audio data in order to achieve better compression. The transformation sub-block decorrelates the data. Figure (3.5) shows the form of splitting a *frame A* into four blocks to show the merits of splitting into blocks over the whole channel.

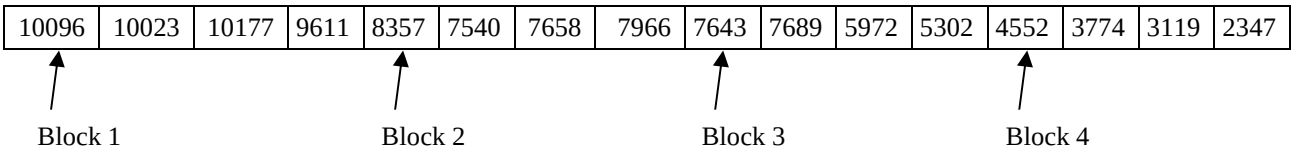


Figure 3.5 frame (A) is Split into 4 blocks

3.4.3 DCT Transformation

As common, DCT transformation will be applied on each block along in the entire wave data to be transformed from time domain into frequency domain. The result of applying DCT transformation to each

block of previous *frame A* is shown in Figure (3.6), while algorithm (3.2) presents the implementation steps of DCT transformation.

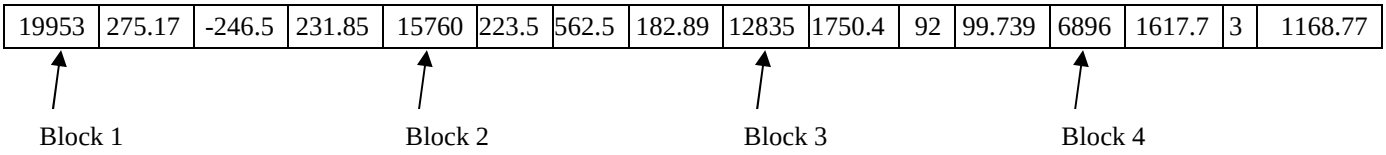


Figure (3.6) DCT transformation on *frame (A)*

Algorithm (3.2) Discrete Cosine Transform

Input: Wav () ' data samples of the original wave file
Nm ' Number of samples
BL ' Block length

Output: DCT.C () ' DCT coefficients

Procedure

Step1: **Split** Wav ( ) array into blocks of length BL

Step2: **'Compute** *DCT coefficients*

```
For U = 0 To (Nm - 1 - BL) Step BL
    Set I = U/ BL
     For J = 0 To BL - 1
  Set temp = 0
   For k = 0 To BL - 1
    temp = temp + Wav(k + U) * Cos(((((2 * k) + 1) * J * 3.14) / (2 *
BL)))
   End For k
   If J = 0 Then
      T2 = ((2 / BL) ^ 0.5) * (1 / 2 ^ 0.5)
  Else
     Set  T2 = ((2 / BL) ^ 0.5)
  End if
 Set DCT.C (I, J) = T2 * temp
 End For  J
End For U
```

Step3: **Return(**DCT.C ())

End Procedure

3.4.4 Quantization

To improve the compression gain, *quantization* is taken place. I.e. reducing the number of possible values of a quantity, thereby reducing

the number of bits needed to represent it. This is done by simply dividing each component in the frequency domain by a weight factor, and then rounding to the nearest integer. As a result of this, many of the higher frequency components are rounded to zero, and many of the rest of the components become small numbers.

In the current work, The DCT Coefficient will be quantized uniformly by dividing each coefficient by weight factor equal 0.1then the rounding reals into integers represented by three bits only, that allows more values and loses less information. The steps of Quantization are illustrated in algorithm (3.3).

In this process, The number of possible values of the quantity (and thus the number of bits needed to represent it) is reduced at the cost of losing information. A "finer" quantization, that allows more values and loses less information, can be obtained by Formula 3.1:

$$Q(DCT.C) = CInt (DCT.C / Qst(x)) \qquad (3.1)$$

$$Qst(U) = Qstep * (1 + weight * U)$$

And **Qstep = 1**

weight = 0.1

Qst ( ) Vector dimension is set to block length 12 elements.

DCT.C ( ) DCT Coefficients will be treated as blocks Qst( ) of Coefficients.

Qstep could be set to 1 or 1.5, **weight** could be set to 0.1or 0.2.

Algorithm (3.3) ***Quantization***

Input: DCT.C ( ) ' Array of DCT coefficients
Nm ' Represent Number of samples
BL ' Represent Block length

Output: Q ( ) ' Array of integers contains the Quantized coefficients

```
Procedure
Step1:      Set Q ( ) array dimensions  to Q(Nm / BL, BL)
            Set Qst ( ) Vector dimension to Qst ( BL )
            Set Qstep = 1
            Set Alpha = 0.1            ' Represent the weight to minimize the
                                         quantization error
Step2:      For U = 0 To BL - 1
                Qst( U ) = Qstep * (1 + Alpha * U)
            End For U

Step3:      For U = 0 To Nm - 1 - L Step BL
                For X = 0 To BL - 1
                    Q(U / BL, X) = CInt ( DCT.C ( U / BL,X) / Qst(X) )
                End For X
            End For U
Step4:      Return(Q)
End  Procedure
```

3.4.5 Run-Length Encoding

Run Length Encoding (RLE) is lossless compression that represent any group of data as pare of (item, run), this coding is useful when there is long stream of redundant items. In this research, a special run length encoder for treating zeroes is utilized as illustrated in algorithm (3.4).

The first 4 items of the DCT array contains a value that is always of a very high magnitude, it is called the DC coefficient, other coefficients represent increase in higher frequencies, they are called AC coefficients and they are become of lower magnitude as they move from left to right therefore the first Four coefficients are left as its. While the remaining coefficients will be encoded by RLE.

Algorithm (3.4) Spatial Encoding (Run Length of zeros)

```
Input:  Q ( )        ' DCT coefficients
        Nm           'Represent Number of samples
        BL           'Represent Block length

Output: RL ( )    ' Vector of  Run Length Values
        RLsize    '  Size of RL Vector ( )

Procedure

Step1:  'Copy  Quantization Vector into 1D temp vector
```

```
          Set U = 0    ' Counter
          For I = 0 To Nm / BL - BL
            For J = 0 To BL - 1
               Set  tempRL(U) = Q(I, J)
               Set U = U + 1
            End For J
          End For I

Step2:  ' save the first 4 items into the RL Vector
 and the
          Set J = 0
          For I = 0 To 3
              Set   (I) = tempRL( I )
           End For I
Step3:     'start Run Length for the 0s of    quantized values
           Set J = 4
           Set I = 4
           Set count = 0
           While I < Nm
              Set   count = 0
              While (tempRL(I) < 1) And I < Nm '(TempRL(I) > -1
                    Set  count = count + 1
                    Set  RL(J) = count
                     Set   I = I + 1
                End While
                 While tempRL(I) > 0 And I < Nm
                    Set RL(J) = tempRL( I )
                    Set    I = I + 1
                    Set   J = J + 1
                End While
            End While
          Set RLsize = J
Step4: Return(RL(),RLsize)
End Procedure
```

3.4.6 Shifting Code (Entropy Encoder)

The entropy encoder compresses the values resulted from Run length encoding as a step to result in better overall compression. It is accurately determines the probabilities for each value and produces an appropriate code based on it, so that the resultant output code stream (or Code words) will be smaller than the input stream. In the current system, shifting coding technique is adopted to encode the output of RLE, taking into consideration that there are long runs of zeros. Here, each item will

first mapped to positive value and then its histogram will be calculated. After that an optimizer is used to produce the required number of bits for representation.

The main idea of this coding method is to find the coefficient that is abnormally large between coefficients; the shifting coding technique partitions the large coefficient into smaller coefficients. Shifting Coding steps are illustrated in algorithm (3.5)

Algorithm (3.5) Entropy Encoding (Shifting Code)

Input: RL ( ) ' Run Length Values Vector
RLsize ' the Size of Vector RL ( )

Output: CodeFile contains data after shifting

Procedure

```
Step1:    ' Mapping RL ( ) vector values into Positive
          For I = 0 To RLsize
            If RL(I) < 0 Then
               Set RL(I) = -2 * RL(I) - 1
            Else
               Set RL(I) = 2 * RL(I)
            End If
          End For I
Step2:    ' Find the Max value from Run Length vector
           Set  Max = RL(0)
           For I = 0 To RLsize
              If Max < RL(I) Then
                 Set Max = RL(I)
              End If
           End For I
          Set      Nb = Int(Log(Max) / Log(2) + 0.9)
          If (2 ^ Nb - 1) < Max Then
             Set Nb = Nb + 1
          End If
Step3     ' Calculate the Histogram
         For I = 0 To RLsize
             Set J = RL(I)
             Set His(J) = His(J) + 1
         End For I
Step4:    ' Start Shift Coding Optimizer
          Set OptSize = Nb * (RLsize + 1)
          Set OptBt1 = Nb: OptBt2 = 1
          For Nbb = 2 To Nb - 1
               Set Rng = 2 ^ Nbb – 1
               Set Sm = RLsiz + 1
```

```
                Set Smm = 0
                For I = 0 To Rng
                      Set Smm = Smm + His(I)
                End For I
               Set  Sm = RLsize + 1 + Smm * Nbb
               Set Smm = 0
               Set  Rngg = Max – Rng
               Set Nbt = Int(Log(Rngg) / Log(2) + 0.9)
               If (2 ^ Nbt - 1) < Rngg Then Nbt = Nbt + 1
              For I = Rng + 1 To Max
                    Set   Smm = Smm + His(I)
             End For I
            Set  Sm = Sm + Smm * Nbt
            If OptSiz > Sm Then
                Set OptSize = Sm: OptBt1 = Nbb
                Set OptBt2 = Nbt
           End If
       End For Nbb

Step5:    ' Create file to store coded data
          Open  FileCode for writing
          Write  RLsize, 24
          Write (OptBt1), 4
          Write (OptBt2), 4
         Set Rng = 2 ^ OptBt1
         For I = 0 To RLsize
             If RL(I) < Rng Then
                Write Bit 0: WriteWord CLng(RL(I)), OptBt1
              Else
                Write Bit 1: WriteWord CLng(RL(I) - Rng), OptBt2
            End If
       End For I
  Close FileCode
End   procedure
```

3.4.7 Diffusion

The compressed stream will be then diffused by applying Simple diffuser to prune the bits significance in order to avoid localization problem.

The proposed diffuser in our scheme is shown in Figure (3.7), and illustrated in algorithm (3.6) which implies re-arrangement for bits in reversal for each byte in the encrypted file. This reversible diffuser is applied to compressed coefficients. It is required to provide the necessary diffusion. It is used to achieve better security properties. This approach

enables us to get better security per-instruction ratio for our implemented software than is possible for the existing ciphers.

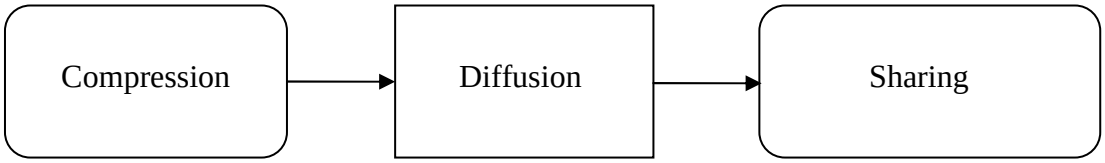


Figure 3.7 Diffusion for Compressed Stream of Bits

Algorithm (3.6) Diffusion for The Compressed File

Input: CodeFile of Compressed data

Output: FileS of Diffused Data

Procedure

Step1: open CodeFile to read byte of data, FileS to save Diffused Bytes
Step2: **Read Byte** of CodeFile
re-arrange bits in reverse
Save diffused byte in FileS
Step3: ***Repeat step2 While*** Not end of file CodeFile

Step4: Close CodeFile and FileS
End Prcedure

3.4.8 Generating Random Coefficients

The produced compressed and ciphered stream bytes of the secret audio file is then shared using (2, *n*) threshold, so any two participants from *n* users can reconstruct the audio file together. To satisfy the former condition, two random coefficients (a1 and a2) for each participant will be generated using arithmetic functions produces values linearly independent and fulfill the following conditions: Both Coefficients for all shares must be:

- For either two shares they should satisfy the conditions $a_{11}a_{22}$-$a_{21}a_{12}$ <> 0.

  Where a_{11} and a_{21} belong to Share1, a_{12} and a_{22} belong to Share2.

- The values of as' must be:
  a) Either three of them odd or the other is even.
  b) Or three of them are even and the other is odd.

Algorithm (3.7) Generate Random Coefficient for each share

```
Input:     Shares_no        ' Number of shares

Output:    randA (No. of Shares, 2)          ' array of Random Coefficient
           File RandA                        ' File to save generated random numbers

Procedure

Step1:     ' Initialization step
           If (No. of Shares) > 2 Then
              Randomize()
              Set    tmp = Int(255 * Rnd() + 1)
              Set   no(1) = tmp
              Set    k = 2
Step2:   ' Generate Random Numbers
           Do While k <= (No. of Shares * 2)
               Randomize()
              Set tmp = Int(255 * Rnd() + 1)
              For j = 1 To k
                    If Not tmp = no(j) Then     flg = True
                 Else
                     Set flg = False
                      Set  j = k
                  End If
               End For j
              If   flg = True Then
                  Set  no(k) = tmp
                  Set   k = k + 1
            End If
Step3:   ' Save non repeated values and set them to Random Coefficients
                For k = 1 To No. of Shares
                  If no(k) + no(k + No. of Shares) <= 127 Then
                        Set randA(k, 1) = no(k): randA(k, 2) = no(k + No. of Shares)
                End For k
Step4:     Return(randA())
End Prcedure
```

3.4.9 Generating Share Functions

After Generating two random Coefficients for each share, each share will be generated using the following formula:

$$S_1 = v1a_{11} + v2\ a_{21}$$

$$S_2 = v1\ a_{12} + v2\ a_{22}$$

…

$$Sn = v1a_{1n} + v2\ a_{2n} \qquad (3.2)$$

Where

n is the number of shares,

v represent bytes of compressed file.

Algorithm (3.8) Generate Share Function for each share

Input: Shares_no ' Number of shares
randA () ' array of Random coefficient

Output: S1() ' Array of functions, one for each share

Procedure

Step1: **For** j = 1 To Shares_no
Set S1(j) = (v1 * randA(j, 1)) + (v2 * randA(j, 2))
Save S1(j) in file j
End For
Step2: **Return**(S1())

End Procedure

3.4.10 Generating *n* shares

After the process of generating random coefficients and functions for each share were constructed, the generation of shares will be the next step using formula (3.2). Since the secret audio file will be shared among *n* users, then *n* shares will be generated and each one is saved in separate file. Algorithm (3.9) illustrates the steps of shares generation process.

Example:

Encrypted file F= { v1, v2, v3,v4,v5,…,vm}.
m is the length of file F.

To generate Share 1 we will use the Coefficients a1 and a2 that belong to Share1 and the function $S_1 = v1a_{11} + v2\ a_{21}$

Share 1 items will be :

{ $v1a_{11} + v2\ a_{21}$, $v3a_{11} + v4\ a_{21}$, $v5a_{11} + v6\ a_{21}$, $v7a_{11} + v8\ a_{21}$, … , $vm\text{-}1a_{11} + vm\ a_{21}$ }.

Share 2 items will be:

{ $v1a_{12} + v2\ a_{22}$, $v3a_{12} + v4\ a_{22}$, $v5a_{12} + v6\ a_{22}$, $v7a_{12} + v8\ a_{22}$, … , $vm\text{-}1a_{12} + vm\ a_{22}$ }.
Share n items will be:

{ $v1a_{1n} + v2\ a_{2n}$, $v3a_{1n} + v4\ a_{2n}$, $v5a_{1n} + v6\ a_{2n}$, $v7a_{1n} + v8\ a_{2n}$, … , $vm\text{-}1a_{1n} + vm\ a_{2n}$ }.

Calculate each S by taking 2 bytes form the compressed secret audio file and use the share coefficient to produce share items at each computing process.

Algorithm (3.9) Generate N Shares

Input: Shares_no ' Number of shares
randA() ' array of Random coefficients
FileS ' Represent File of Diffused data to be shared.

Output: N Number of files, each file will represent a separate share
N 'Represent No. of shares
File RandA ' File to save generated random numbers as array (shares_no,2)

Procedure

Step1: If (No. of Shares) > 2 Then
generateA (No. of Shares) ' procedure generate 2 different numbers for each share
Set randA(No. of Shares, 2) ' Save generated Numbers as array 2D in file
Shares_no = No. of Shares
Set S1(Shares_no) ' S1 array to compute values for each share
Else
MsgBox "Error, shares must be greater than 2"
Jump to Read (No. of Shares)
End If
Step 2: For j = 1 To Shares_no
Create File j ' Represent separate share
Next j

size = LOF(100) 'size of FileS
Nsize = (size) 'how many integer number contain
Set no(Nsize)

```
If LOF(100) > 0 Then     ' check if FileS contain a data & read them all
      Get no(Nsize) from data file            ' Save data in array no()
  For i = 0 To (Nsize - 1) Step 2
     Set v1 = no(i)            ' v1 represent First byte
     Set v2 = no(i + 1)        ' v2 represent 2nd byte
     For j = 1 To Shares_no
           S1(j) = (v1 * randA(j, 1)) + (v2 * randA(j, 2))    ' compute Shares Values of
FileS and store each share values in separate File'
           Save S1(j) in file j
     Next j
  Next i
End If
Close FileS

Step 3:      For j = 1 To Shares_no
                              Close file j
              Next j
End Procedure
```

3.5 The Designed Decoder

In Encoding Stage the encoded wave data is transformed into compressed stream of bits thereafter that stream will be encrypted by using arithmetic calculations to produce *n* different shares save in separated file. On the other hand, in decoding module any two shares are used to reconstruct the compressed data, thereafter decompression is implemented to obtain the original wave file as shown in algorithm (3.10).

Decoding process is considerably easier and faster than the encoding process because it involves little or less computations than encoding process. As shown in Figure (3.2) the designed decoder is Consists of the following steps:

1. Load two shares (two files).
2. The dealer will fetch the Generated Coefficients as a1, a2 for each share.
3. The dealer wills Fetch Generated Functions to reconstruct the original compressed bits.

4. Reconstruct the original Diffused Compressed file from Share1 and Share 2 data.
5. Inverse the diffusing for the compressed file
6. Now the output of the de-sharing will be a file of compressed stream of bits by Perform Shifting Code Inverse the output will be integer values.
7. Zeros Run Length Inverse will be performed to re-insert the compressed consecutive zeros.
8. De-Quantization will be performed now.
9. DCT Inverse will be performed on the de-quantized values
10. The output from DCT Inverse will be the saved to reconstruct finally the original wave data.

Algorithm (3.10) Decoding process

```
Input:    File RandA ' File to save generated random numbers as array (shares_no,2)
          2 share files
Output: File De-share of the reconstructed wave file

Procedure
' Start De-Share Process'
Read (Share1, Share2)
Fetch share1 coefficients a11,a21 from File RandA
Fetch share2 coefficients a12,a22 from File RandA
Create File D  ' to save data after de-share process
While Not End Of file Share1 and Share2
  Read S1 from Share1
  Read S2 from Share2
  Compute v1 = (S1*a22 – S2*a21) \ (a11*a22 – a21*a12)
  Compute v2 = (S2*a11 – S1*a12) \ (a11*a22 – a21*a12)
  Save v1 and v2 at File D
End While
Close File D
Perform Inverse Diffusing For Compressed diffused Data
'Start De-Shifting Process'
RLsiz = ReadWord  ' Set file size
OptBt1 = ReadWord(4)
OptBt2 = ReadWord(4)
ReDim rRL(RLsiz)
Rng = 2 ^ OptBt1
For ii = 0 To RLsiz
    If ReadBit = 0 Then
            rRL(ii) = ReadWord(OptBt1)
```

```
    Else
        rRL(ii) = ReadWord(OptBt2) + Rng
    End If
Next ii
CloseReadFile
' Inverse Mapping into Positive  '
ReDim RL(RLsiz)
For i = 0 To RLsiz
    (RL(i) = rRL(i
Next i
For i = 0 To RLsiz
    If Not (rRL(i) Mod 2 = 0) Then
        RL(i) = ((rRL(i) + 1) / 2) * -1
    Else
        RL(i) = rRL(i) / 2
    End If
Next i
' Start RL inv  '
Dim k As Integer
Dim temp As Integer
temp = 0
For ii = 0 To RLsiz - 1
        If RL(ii) = 0 Then temp = temp + RL(ii + 1) – 1
Next ii
For i = 0 To 3
    InvRL(i) = RL(i )
Next i
Set j = 4
Set i = 4
Dim count0 As Integer
count0 = 0
While j < Nm And i < Nm
        If RL(i) = 0 Then
            count0 = RL(i + 1)
            i = i + 1
        End If
    While RL(i - 1) = 0 And count0 > 0
                count0 = count0 - 1
                InvRL(j) = 0
                j = j + 1
      Wend
       If RL(i - 1) = 0 Then i = i + 1
      While RL(i) > 0 Or RL(i) < 0
            InvRL(j) = RL(i)
            i = i + 1
            j = j + 1
      Wend
Wend
U = 0
For i = 0 To Nm \ L - L
```

```
      For j = 0 To L - 1
          RL(U) = q(i, j )
       U = U + 1
       Next j
Next i
'Start De-Quantization'
ReDim DeQ(Nm \ L, L)
ReDim Qst(L)
Qstep = 1
Alpha = 0.1
For U = 0 To L - 1
      Qst(U) = Qstep * (1 + Alpha * U )
Next U
For U = 0 To Nm - 1 - L Step L
       For x = 0 To L – 1
             DeQ(U / L, x) = q(U / L, x) * Qst(x )
Next x, U

'INVERSE DCT '
For U = 0 To (Nm - 1 - L) Step L
SF = U \ L
For j = 0 To L - 1
temp = 0
For k = 0 To L - 1
      If k = 0 Then T2 = (1 / 2 ^ 0.5)
           Else T2 = 1
temp = temp + DeQ(SF, k) * Cos(((((2 * j) + 1) * k * 3.14) / (2 * L))) * T2

Next k
wavT(U + j) = ((2 / L) ^ 0.5) * temp
Next j
Next U
temp = 0
Save wavT at File De-Share
End Procedure
```

Chapter Two

Principles of Secret sharing Schemes and Compression

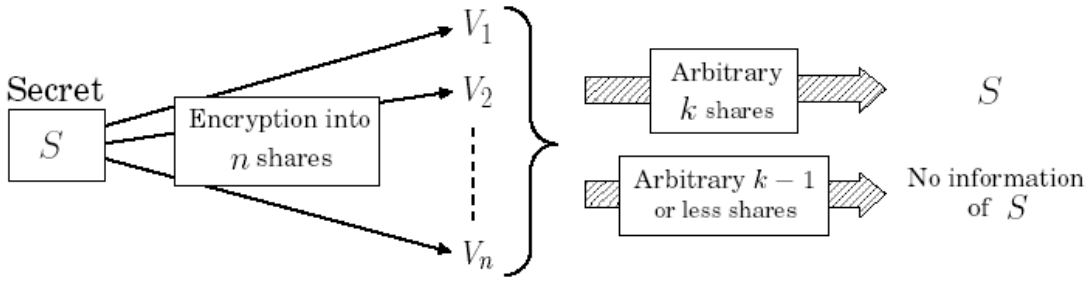
2.1 Introduction

This chapter is dedicated to explore the theoretical background need to establish system of audio coding to be transmitted through networks. In the sense of thesis context, audio coding means audio cryptography and compression. This chapter is divided into two main parts: The first one explained threshold secret sharing cryptography methodology while the principles of data compression based on Discrete Cosine Transform will be discussed in part two.

2.2 Concepts of threshold Secret Sharing Schemes

Threshold secret sharing refers to method for distributing a *secret* amongst a group of participants, each of whom is allocated a *share* of the secret. The secret can be reconstructed only when a sufficient number of shares are combined together; individual shares are of no use on their own. More formally, in a secret sharing scheme there are one *dealer* and *n players*. The dealer gives a secret to the players, but only when specific conditions are fulfilled. The dealer accomplishes this by giving each player a share in such a way that any group of t (for *threshold*) or more players can together reconstruct the secret but no group of fewer than t players can. Such a system is called a (t, n)-threshold scheme (sometimes it is written as an (n, t)-threshold scheme). [Sha12]

For example, consider a (k, n) threshold Secret Sharing scheme illustrated in figure 2.1. A secret S is encrypted into n pieces called shares V , each of which has no information of secret S, but S can be decrypted by collecting k shares whereas (k-1) or fewer cannot be decrypted the secret S.[Iwa03]



(Figure 2.1): A (k, n)–threshold Secret Sharing Scheme

In general, threshold SSS consists of multiple phases: [O1i13]

1. *Initialization*. It defines the environment of the scheme: the parameters, the possible space of secrets and shares and any other pre-requisites;
2. *Sharing*. It describes the splitting algorithm of the secret into multiple shares. Usually, a Trusted Third Party (TTP) called *dealer* performs the sharing; however, dealer-free SSS also exist;
3. *Distribution*. It indicates the share(s) that are sending to each participant via secure channels. Usually, during this phase, each participant receives one share. However, a single party may also own multiple shares (and therefore obtain more power within the group) or even an authorized set of shares (and hence becomes able to reconstruct the secret by him).
4. *Reconstruction*. It explicit the key recover formulas or algorithms performed to determine the secret from an authorized set of shares.

2.3 Constructions of Threshold Secret Sharing Scheme

The current section presents the common Threshold Secret Sharing Schemes explained in literature.

2.3.1. Trivial secret sharing

There are several *(t, n)* secret sharing schemes for $t = n$, when all shares are necessary to recover the secret. When space efficiency is not a concern, these schemes can be used to reveal a secret to any desired subsets of the players simply by applying the scheme for each subset. For

example, to reveal a secret *s* to any two of the three players Alice, Bob and Carol, create three different (2,2) secret shares for *s*, giving the three sets of two shares to Alice and Bob, Alice and Carol, and Bob and Carol. This approach quickly becomes impractical as the number of subsets increases, for example when revealing a secret to any 50 of 100 players, whereas the schemes described below allow secrets to efficiently be shared with a threshold of players. [Sha12]

The difficulty lies in creating schemes that are still secure, but do not require all *n* shares. For example, imagine that the Board of Directors of a company would like to protect their secret formula. The president of the company should be able to access the formula when needed, but in an emergency any 3 of the 12 board members would be able to unlock the secret formula together. This can be accomplished by a secret sharing scheme with $t = 3$ and $n = 15$, where 3 shares are given to the president, and 1 is given to each board member.

When space efficiency is not a concern, trivial $t = n$ schemes can be used to reveal a secret to any desired subsets of the players simply by applying the scheme for each subset. For example, to reveal a secret *s* to any two of the three players Alice, Bob and Carol, create three different (2,2) secret shares for *s*, giving the three sets of two shares to Alice and Bob, Alice and Carol, and Bob and Carol [Sti95].

2.3.2. Shamir's scheme

This secret sharing scheme is based on (*k*, *n*)-threshold based secret sharing technique ($k \leq n$) see figure (2.2). The technique allows any *k* out of *n* shares to construct a given secret, a (*k*-1) degree polynomial is necessary. This polynomial function of order (*k*-1) is constructed as,

$$F(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \pmod{p} \quad (2.1)$$

Now we can easily generate n number of shares by using above equation. Where a_0 is the secret, *p* is a prime number and all other coefficients are

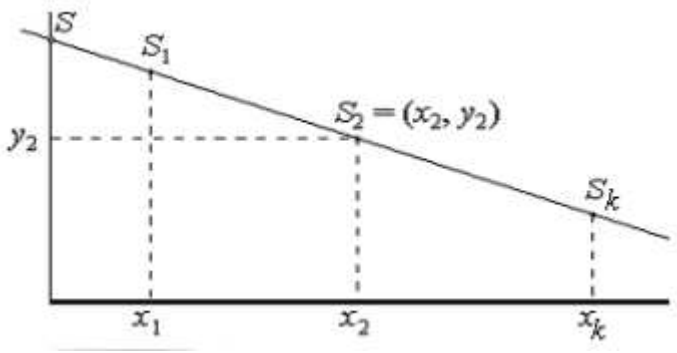
random elements from the secret. Each of the *n* shares is a pair (x_i, y_i) of numbers satisfying $f(x_i) = y_i$ and $x_i>0$, $1\leq i \leq n$ and $0< x_1< x_2\ldots< x\leq p-1$. Given any *k* shares, the polynomial is uniquely determined and hence the secret a_0 can be computed via Lagrange interpolation. However, given *k*-1or fewer shares, the secret can be any element in the field.

The polynomial function f(x) is destroyed after each shareholder possesses a pair of values (x_i, y_i) so that no single shareholder knows the secret value a_0. In fact, no groups of *k*-1 or fewer shares can discover the secret a_0. On the other hand, when *k* or more secret shares are available, then we may set at least *k* linear equations $y_i= f(x_i)$ for the unknown a_i's.

The unique solution to these equations shows that the secret value a can be easily obtained by using Lagrange interpolation [Sha79].

Shamir's SSS is regarded as a Perfect SS scheme because knowing even (k-1) linear equations doesn't expose any information about the secret.

Figure (2.2): Shamir's secret sharing scheme where k=2



Some of the useful properties of Shamir's (k, n)-threshold scheme are [Cha13]:

1. **Secure**: Information theoretic security.
2. **Minimal**: The size of each piece does not exceed the size of the original data.

3. **Extensible**: When *n* is kept fixed, D_i pieces can be dynamically added or deleted without affecting the other pieces.
4. **Dynamic**: Security can be easily enhanced without changing the secret, but by changing the polynomial occasionally (keeping the same free term) and constructing new shares to the participants.
5. **Flexible**: In organizations where hierarchy is important, we can supply each participant different number of pieces according to his importance inside the organization. For instance, the president can unlock the safe alone, whereas 3 secretaries are required together to unlock it.

- **Example**

The following example illustrates the basic idea of Shamir's scheme. Note, however, that calculations in the example are done using integer arithmetic rather than using finite field arithmetic. Therefore the example below does not provide perfect secrecy and is not a true example of Shamir's scheme. So we'll explain this problem and show the right way to implement it (using finite field arithmetic).

Suppose that our secret is 1234 $(S = 1234)$.

We wish to divide the secret into 6 parts $(n = 6)$, where any subset of 3 parts $(k = 3)$ is sufficient to reconstruct the secret. At random we obtain two $(k - 1)$ numbers: 166 and 94.

$(a_1 = 166; a_2 = 94)$

Our polynomial to produce secret shares (points) is therefore:

$$f(x) = 1234 + 166x + 94x^2$$

We construct 6 points $D_{x-1} = (x, f(x))$ from the polynomial:

$$D_0 = (1, 1494); D_1 = (2, 1942); D_2 = (3, 2578); D_3 = (4, 3402);$$
$$D_4 = (5, 4414); D_5 = (6, 5614)$$

We give each participant a different single point (both x and $f(x)$). Because we use D_{x-1} instead of D_x the points start from $(1, f(1))$ and not $(0, f(0))$. This is necessary because if one would have $(0, f(0))$ he would also know the secret $S = f(0)$. In order to reconstruct the secret any 3 points will be enough.

Let $(x_0, y_0) = (2, 1942); (x_1, y_1) = (4, 3402); (x_2, y_2) = (5, 4414)$

We will compute Lagrange basis polynomials:

$$\ell_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x-4}{2-4} \cdot \frac{x-5}{2-5} = \frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}$$

$$\ell_1 = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x-2}{4-2} \cdot \frac{x-5}{4-5} = -\frac{1}{2}x^2 + \frac{7}{2}x - 5$$

$$\ell_2 = \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x-2}{5-2} \cdot \frac{x-4}{5-4} = \frac{1}{3}x^2 - 2x + \frac{8}{3}$$

Therefore

$$f(x) = \sum_{j=0}^{2} y_j \cdot \ell_j(x)$$

$$= 1234 + 166x + 94x^2$$

Recall that the secret is the free coefficient, which means that $S = 1234$, and we are done.

2.3.3 Blakley's secret sharing scheme

Blakley's secret sharing scheme is geometric in nature. The secret is a point in an *m* dimensional space. *n* shares are constructed with each share defining an affine hyperplane in this space; an affine hyperplane is the set of solutions $x=(x_1, \ldots, x_m)$ to an equation of the form $a_1x_1 + \ldots + a_m x_m = b$. By finding the intersection of any *m* of these planes, the secret (that is, the point of intersection) can be obtained. This scheme is not perfect, as the person with a share of the secret knows the secret is a point on his or her hyperplane. Nevertheless, this scheme can be modified to achieve perfect security [Bla79]. Figure 2.3 presents this scheme.

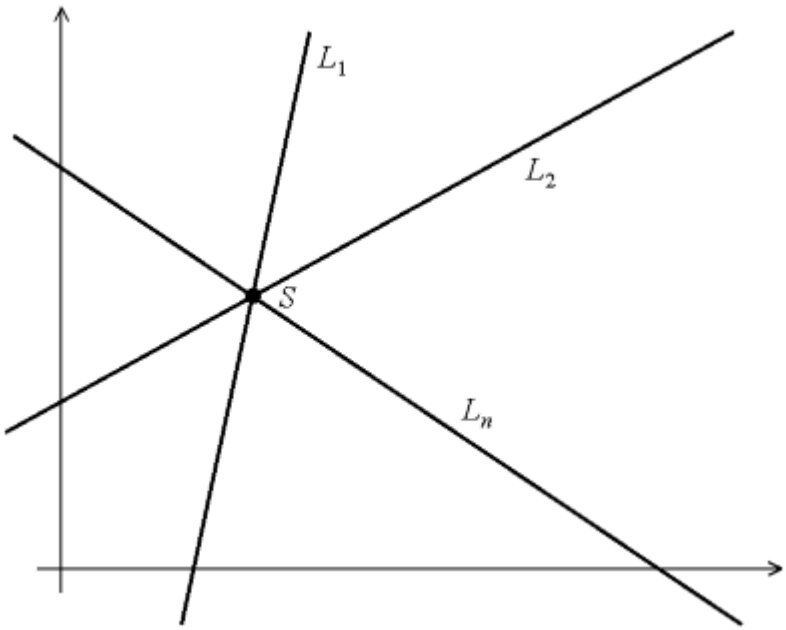


Figure (2.3) Blakley's Scheme

2.3.4 Chinese Remainder Theorem

The Chinese Remainder Theorem can also be used in secret sharing, for it provides us with a method to uniquely determine a number *S* modulo *k* many relatively prime integers [Sha12],

$$\boldsymbol{S < \prod_{i=1}^{k} m_i \quad m_1, m_2, \ldots m_k} \qquad (2.2)$$

There are two secret sharing schemes that make use of the Chinese Remainder Theorem; [Mig83] and [Asm83]. They are threshold secret sharing schemes, in which the shares are generated by reduction modulo the integers, and the secret is recovered by essentially solving the system of congruence using the Chinese Remainder Theorem.

2.3.5 Proactive secret sharing

Proactive secret sharing is a method to update distributed keys (shares) in a secret sharing scheme periodically such that an attacker has less time to compromise shares. This contrasts to a non-proactive scheme where if threshold number of shares is compromised during the lifetime of the secret, then secret is compromised.

If the players (holders of the shared secret) store their shares on insecure computer servers, an attacker could crack in and steal the shares. Since it is not often practical to change the secret, the uncompromised (Shamir-style) shares should be updated in a way that they generate the same secret, yet the old shares are invalidated.

In order to update the shares, the dealer (i.e., the person who gives out the shares) generates a new random polynomial with constant term zero and calculates for each remaining player a new ordered pair, where the x-coordinates of the old and new pairs are the same. Each player then adds the old and new y-coordinates to each other and keeps the result as the new y-coordinate of the secret [Her95].

- The dealer constructs a random polynomial over a field of degree k where k is the threshold
- Each player gets the share $x_i^0 = f^0(i)$ where $i \in \{1, ..., n\}$, n is the number of players, and x_i^0 is the share for player i at time interval 0
- The secret can be reconstructed via interpolation of k shares
- To update the shares, all parties need to construct a random polynomial of the form $\delta_i(z) = \delta_{i,1}z^1 + \delta_{i,2}z^2 + ... + \delta_{i,k}z^k$
- Each player i sends all other players $u_{i,j} = \delta_i(j)$
- Each player updates their share by $x_i^{t+1} = x_i^t + u_{1,i}^t + ... + u_{n,i}^t$ where t is the time interval in which the shares are valid

All of the non-updated shares the attacker accumulated become useless. An attacker can only recover the secret if he can find enough other non-updated shares to reach the threshold. This situation should not happen because the players deleted their old shares. Additionally, an attacker cannot recover any information about the original secret from the update process because it only contains random information.

The dealer can change the threshold number while distributing updates, but must always remain vigilant of players keeping expired shares.

2.4 Principals of Audio Compression

The next subsections will be devoted to presents all necessary methodologies for audio compression work.

2.4.1 Digital Audio

Sound can be digitized and broken up into numbers. When sound is played into a microphone, it is converted into a voltage that varies continuously with time. Figure 2.4 shows a typical example of sound that starts at zero and oscillates several times. Such voltage is the analog representation of the sound. Digitizing sound is done by measuring the voltage at many points in time, translating each measurement into a number, and writing the numbers on a file. This process is called sampling. The sound wave is sampled, and the samples become the digitized sound. The device used for sampling is called an analog-to-digital converter (ADC). The method of representing each sample with an independent code word is called pulse code modulation (PCM). Figure 2.4 shows the digital audio process. [digital audio compression]

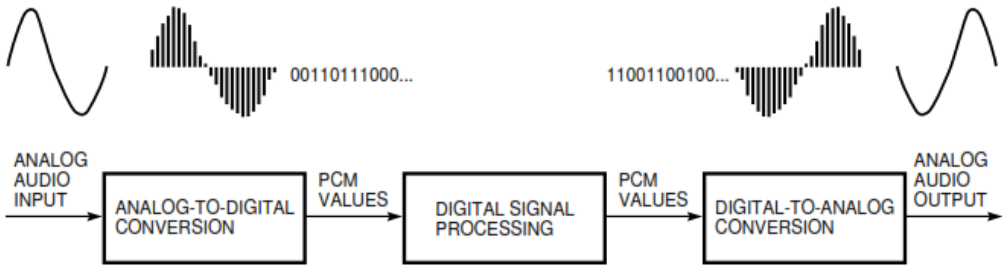


Figure (2.4) Digital Audio Process

Since the sound samples are numbers, they are easy to edit. However, the main use of a sound file is to play it back. This is done by converting the numeric samples back into voltages that are continuously fed into a

speaker. The device that does that is called a digital-to-analog converter (DAC). Intuitively, it is clear that a high sampling rate would result in better sound reproduction, but also in many more samples and therefore bigger files. Thus, the main problem in sound sampling is how often to sample a given sound [Sal04].

2.4.2 Concepts of Audio Compression

Huge amount of data transmission is very difficult both in terms of transmission and storage. Compression is basically to remove redundancy between neighboring samples and between adjacent cycles. Major objective of audio compression is to represent signal with lesser number of bits. The reduction of data should be done in such a way that there is acceptable loss of quality. [Pat13]

In general, compression methods have been classified as either **lossless** or **lossy**. *Lossless* techniques allow the exact original data to be reconstructed from the exact original data to be reconstructed from the compressed data. It is mainly used in cases where it is important that the original signal and the decompressed signal are almost same or identical. Examples of lossless compression are Run Length and Huffmann coding.

Lossy techinuqes are a data encoding method that compresses data by removing some of them. The aim of this technique is to minimize the amount of data that has to be transmitted. They are mostly used for multimedia data compression. Examples of lossy compression are MPEG and JPEC All real world measurements inherently contain a certain amount of *noise*. If the changes made to these signals resemble a small amount of additional noise, no harm is done. This distinction is important because lossy techniques are much more effective at compression than lossless methods. The higher the compression ratio (Compression ratio = size of the output file / size of the input file) the more noise added to the data. [Smi06]

2.4.3 Techniques for Audio compression

Audio compression is classified into three techniques [Pat13]; these are:

A. **Waveform Coding**

The signal that is transmitted as input is tried to be reproduced at the output which would be very similar to the original signal. Waveform-based codecs are intended to remove waveform correlation between audio samples to achieve audio compression. It aims to minimize the error between the reconstructed and the original speech waveforms. Typical ones are Pulse Code Modulation (PCM) and Adaptive Differential PCM (ADPCM) [Sun13].

B. Parametric coding

In this type of coding the signals are represented in the form of small parameters which describes the signals very accurately. In parametric extraction method a preprocessor is used to extract some features that can be later used to extract the original signal [Bri06].

C. Transform Coding

In this method the signal is transformed into frequency domain and then only dominant feature of signal is maintained. In transform method audio can be represented in terms of coefficients. Transform techniques do not compress the signal, they provide information about the signal and using various encoding techniques compressions of signal is done. Audio compression is done by neglecting small and lesser important coefficients and data and discarding them and then using quantization and encoding techniques [Pan93].

This technique of compression is performed in the following steps: 1- *Transform technique; 2-Quantization; and 3- Encoding*.

2.5 Discrete Cosine Transform

The discrete cosine transformation (DCT) is a technique for converting a signal into elementary frequency components. In

particular, The cosine transform translates a set of data points from the spatial domain to the frequency domain using Cosine basis functions[Sal07].

The DCT in one dimension is given by [Sal04] as equation 2.3

$$Gf = \left(\frac{2}{n}\right)^{\frac{1}{2}} Cf \sum_{t=0}^{n-1} pt \cos\left[\frac{(2t+1)f\pi}{2n}\right] \quad \dots 2.3$$

$$Cf = \begin{cases} \sqrt{2}, & for\ f = 0, \\ 1, & f > 0, \end{cases} \quad for\ f = 0,1,\dots,n-1$$

The input is a set of *n* data values p_t (pixels, audio samples, or other data) and the output is a set of n DCT transform coefficients (or weights) G_f. The first coefficient G_0 is called the DC coefficient and the rest are referred to as the AC coefficients. Notice that the coefficients are real numbers even if the input data consists of integers. Similarly, the coefficients may be positive or negative even if the input data consists of nonnegative numbers only. The decoder inputs the DCT coefficients in sets of *n* and uses the inverse DCT (IDCT) to reconstruct the original data values (also in groups of *n*), The IDCT in one dimension is given as equation 2.4.

$$pt = \left(\frac{2}{n}\right)^{\frac{1}{2}} \sum_{t=0}^{n-1} CjGj \cos\left[\frac{(2t+1)j\pi}{2n}\right] \quad \dots 2.4$$

The important feature of the DCT (i.e. the feature that makes it so useful in data compression) is that it takes correlated input data and concentrates its energy in just the first few transform coefficients. If the input data consists of correlated quantities, then most of the *n* transform

coefficients produced by the DCT are zeroes or small numbers, and only a few are large (normally the first ones). The early coefficients contain the important (low-frequency) image information and the later coefficients contain the less-important (high-frequency) image information. Compressing data with the DCT is therefore done by quantizing the coefficients. The small ones are quantized coarsely (possibly all the way to zero) and the large ones can be quantized finely to the nearest integer. After quantization, the coefficients (or variable-size codes assigned to the coefficients) are written on the compressed stream. Decompression is done by performing the inverse DCT on the quantized coefficients. This results in data items that are not identical to the original ones but are not much different.

In practical applications, the data to be compressed is partitioned into sets of *n* items each and each set is DCT-transformed and quantized individually. The value of *n* is critical, small values of *n* such as 3, 4, or 6 results in many small sets of data items. Such a small set is transformed to a small set of coefficients where the energy of the original data is concentrated in a few coefficients, but there are only few coefficients in such a set. Thus, there are not enough small coefficients to quantize. Large values of *n* result in a few large sets of data. The problem in such a case is that the individual data items of a large set are normally not correlated and therefore result in a set of transform coefficients where all the coefficients are large. Experience indicates that n = 8 is a good value and most data compression methods that employ the DCT use this value of n.

2.6 Scalar Quantization

The dictionary definition of the term “quantization” is “to restrict a variable quantity to discrete values rather than to a continuous set of

values." In the field of data compression, quantization is used in two ways [Sim06]:

1. If the data to be compressed are in the form of large numbers, quantization is used to convert them to small numbers. Small numbers take less space than large ones, so quantization generates compression. On the other hand, small numbers generally contain less information than large ones, so quantization results in lossy compression.
2. If the data to be compressed are analog (i.e., a voltage that changes with time) quantization is used to digitize it into small numbers. Smaller the numbers give better compression, but also the greater the loss of information. This aspect of quantization is used by several speech compression methods.

2.7 Run-Length Encoding

Digitized *signals* can have runs of the same value, indicating that the signal is not changing. For instance, digitized music might have a long run of zeros between songs. Run-length encoding is a simple method of compressing these types of files.

Figure 2.5 illustrates run-length encoding for a data sequence having frequent runs of *zeros*. Each time a zero is encountered in the input data, *two* values are written to the output file. The first of these values is a zero, a flag to indicate that run-length compression is beginning. The second value is the number of zeros in the run. If the average run-length is longer than two, compression will take place. On the other hand, many single zero in the data can make the encoded file larger than the original.

Many different run-length schemes have been developed. For example, the input data can be treated as individual bytes, or groups of bytes that represent something more elaborate, such as floating point numbers. Run-length encoding can be used on only *one* of the characters (as with the *zero* above), *several* of the characters, or *all* of the characters.

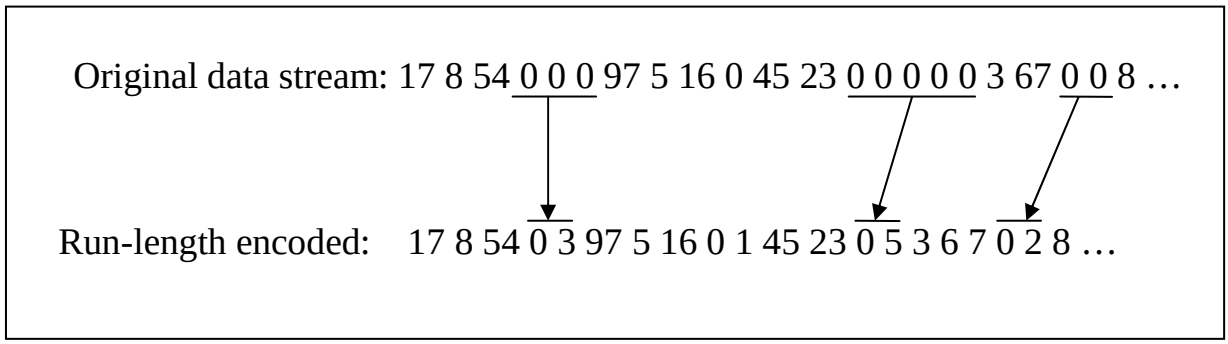


Figure (2.5) Example of Run-Length Encoding.

2.8 Shift Coding

The idea of this method is to encode the sequence of numbers by code words whose bit length is less than the bit length required to represent the maximum value of the sequence of numbers to be coded.

The numbers whose values are large may split into a sequence of code words, by using the following formula 2.5 [Gon02]

$$X = nW_m + W_r \qquad (2.5)$$

Where:

X is the number to be coded, n is the number of code words that used to encode the number X, W_m is the lowest value which cannot be coded by using a single code word, W_r is the value of the last code word used to encode X.

The values of W_m, W_r, and n are determined by using equations 2.6, 2.7, and 2.8 consequently;

$$W_m = 2b - 1 \qquad (2.6)$$

$$W_r = \text{X mod } W_m \qquad (2.7)$$

$$n = \text{X div } W_m \qquad (2.8)$$

Where

b is the number of bits used to represent each single shift code word.

The performance of Huffman coding and shift coding are better when the sequence of numbers has a histogram whose shape is highly peaked. The performance of shift coding is better than Huffman and arithmetic coding when the histograms have long tails [Mah07].

Dedication

To My father, mother, brothers

My Husband

And My Lovely Sons

With my love

List of Abbreviations

ACS	Audio Cryptography Scheme
ADC	analog-to-digital converter
ADPCM	Adaptive Differential PCM
ASS	Audio Secret Sharing
ASSS	Audio Secret Sharing Scheme
DAC	digital-to-analog converter
DB	Decibel
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
KB	Kilo Byte
MSE	Mean Square Error
PCM	pulse code modulation
PCM	Pulse Code Modulation
PSNR	Peak Signal to Noise Ratio
RLE	Run Length Encoding
SSS	Secret Sharing Scheme
TTP	Trusted Third Party
VSS	Visual Secret Sharing
VSSS	Visual Secret Sharing Scheme
WAV	Windows Audio Visual
JPEG	Joint Photographic Experts Group
MPEG	Moving Picture Experts Group

List of Algorithms

List of Figures

List of Tables

List of Contents

References

[Asm83] C. A. Asmuth and J. Bloom, "A modular approach to key safeguarding", IEEE Transactions on Information Theory, IT-29(2) pages:208–210, 1983.

[Bei96] Amos Beimel, "Secure Schemes for Secret Sharing and Key Distribution", Ph.D thesis, Senate of the Technion, Israel Institute of Technology, Haifa, 1996.

[Bei97] A. Beimel, A. Gal, and M. Paterson, "Lower bounds for monotone span programs", Computational Complexity, Conference version: FOCS '95., 6(1), pages: 29-45, 1997.

[Bla79] G.R. Blakley, "Safeguarding cryptographic keys", AFIPS Conference Proceedings, vol.48, pp.313–317, 1979.

[Bri06] Manuel Briand, David Virette, and Nadine Martin, "Parametric Coding of Stereo Audio Based on Principal Component Analysis", Proc. of the 9th Int. Conference on Digital Audio Effects (DAFX'06), Montreal, Canada, September 18-20, 2006.

[Cha06] Teng-Kuei Chang, "An Audio Feature Extraction Scheme Based on Secret Sharing and Wavelet Transform", M.Sc. thesis, Department of Computer Science and Engineering, Tatung University, 2006.

[Cha12] Ka Fai -Peter Chan, "Secret Sharing in Audio Steganography", Article, Provider: citeseer, 2012.

[Cha13] Ayan Chaudhuri, "Design of a secured secret sharing technique and its application on mobile handsets", M.Sc. thesis, School of Mobile Computing and Communication, Jadavpur University, Kolkata, India, 2013.

[Che12] Kai-Yuen Cheong, "A secret sharing scheme of prime numbers based on hardness of factorization, Japan advanced institute of science and technology, 2012.

[Csi96] L. Csirmaz, "The dealer's random bits in perfect secret sharing schemes", Studi Sci. Math. Hungar., 32(3-4), pages 429-437, 1996.

[Des93] Y. Desmedt,"Threshold cryptosystems", Advances in cryptology-Auscrypt'92, volum 718 of Lecture Notes in computer science, pages 5-14. Sepringer-Verlag, 1993.

[Des98] Y. Desmedt, S. Hou, and J. Quisquater, “Audio and optical cryptography,” Advances in Cryptology - Asiacrypt'98, Lecture Notes in Computer Science, LNCS 1514, pp.392–404, Springer Verlag, 1998.

[Ehd08] Mohammad Ehdaie, Taraneh Eghlidos, and Mohammad Reza Aref, "A Novel Secret Sharing Scheme from Audio Perspective", Internatioal Symposium on Telecommunications, pp.13-18, 2008.

[Fuj06] Norihiro Fujita, Ryouichi Nishimura and Yoˆiti Suzuki, “Audio secret sharing for 1-bit audio”, Acoust Sci. &Tech. 27, 3 (2006).

[Gon02] Gonzalez, R., and Woods, R.; "Digital Image Processing", Pearson Education International, Prentice Hall, Inc., 2nd Edition, New Jersey, 2002.

[Guo13] Cheng Guo , and Chin-Chen Chang," A Construction for Secret sharing Scheme with General Access Structure", Journal of Information Hiding and Multimedia Signal Processing, Volume 4, Number 1, January 2013.

[Her95] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage", In D. Coppersmith, editor, Advances in Cryptology - CRYPTO ’95, volume 963 of Lecture Notes in Computer Science, pages 339–352. Springer-Verlag, 1995.

[Hof08] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman, "An Introduction to Mathematical Cryptography", Springer, USA, 2008.

[Iwa03] Mitsugu Iwamoto, "General Construction Methods of Secret Sharing Schemes and Visual Secret Sharing Schemes", M.Sc. thesis, University of Tokyo ,Tokyo, Japan, 2003.

[Kur08] Jun Kurihara, Shinsaku Kiyomoto, Kazuhide Fukushima and Toshiaki Tanaka, 13." On a Fast (k,n)-Threshold Secret Sharing Scheme", Journal IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences archive, Volume E91-A Issue 9, Pages 2365-2378, Oxford University Press Oxford, UK September 2008.

[Lin03] Chen-chi Lin, Chi-sung Laih and Ching-Nung Yang, "New Audio Secret Sharing Schemes With Time Division Technique" - Journal Of Information Science And Engineering 19, pp.605-614, 2003.

[Mah07] Mhamood, R. F., "Improved Once-Time-Search Method Based on Inter-Block Correlation", M.Sc. Thesis, Al-Nahrain University,College of Science, 2007.

[Men96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. "Handbook of Applied Cryptography", CRC Press Inc., 1996.

[Mig83] M. Mignotte, "How to share a secret", in T. Beth, editor, Cryptography Proceedings of the Workshop on Cryptography, Burg Feuerstein, 1982, volume 149 of Lecture Notes in Computer Science, pages 371–375, Springer-Verlag, 1983.

[Nao94] Moni Naor and Adi Shamir. "Visual cryptography", In Proceedings of the 13th annual international conference on

Theory and application of cryptographic techniques, EUROCRYPT'94, pages 1-12. Springer, 1994.

[Nas11] Prabir Kr. Naskar, Hari Narayan Khan, Ujjal Roy, Ayan Chaudhuri , Atal Chaudhur, "Shared Cryptography with Embedded Session Key for Secret Audio", International Journal of Computer Applications (0975 – 8887), Volume 26– No.8, page5-9, July 2011.

[Nik10] Amresh Nikam, Poonam Kapade, and Sonali Patil,"Audio Cryptography: "A (2,2) Secret Sharing for Wave File", International Journal of Computer Science and Application Issue, ISSN 0974-0767, pp.96-99, 2010.

[O1i13] Ruxandra Florentina Olimid, "Secret Sharing-based Group Key Establishment", Ph.D. thesis, Department of mathematics and computer science, university of Bucharest, 2013.

[Pan93] Davis Yen Pan, "Digital Audio Compression", Digital Technical Journal Vol. 5 No. 2, Spring 1993.

[Pat13] M. V. Patil, Apoorva Gupta, Ankita Varma, and , Shikhar Salil, "Audio and Speech Compression Using DCT and DWT Techniques", International Journal of Innovative Research in Science, Engineering and Technology, *Vol. 2, Issue 5, May* 2013.

[Sal04] David Salomon, "Data Compression the Complete Reference", Third Edition, Springer –Verlag, New York, Inc., 2004.

[Sha79] Adi Shamir, "How to share a secret?", Commun. ACM, 22(11):612-613, pages 13, 21, 25, 26, November 1979.

[Sha12] Jyotsna Shaw, "A modern approach for secured transmission of audio files using fractal based chaos", Ms.c thesis, department

of computer science & Engineering, Jadavpur University, Kolkata, India, 2012.

[Smi06] Steven W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing, copyright", California Technical Publishing, 2006.

[Soc05] Socek, D., and Magliveras, S.S, "General access structures in audio cryptography", IEEE International Conference on Electro Information Technology ISBN: 0780392329, Pages: 1-6, 2005.

[Sti95] Douglas Stinson, "Cryptography: Theory and Practice", CRC Press, ISBN: 0849385210, 1995.

[Sun13] L. Sun Mkwawa, I.-H., Jammeh, E., Ifeachor, E., "*Guide to Voice and Video over IP*, Computer ommunications and Networks", DOI 10.1007/978-1-4471-4905-7_2, Springer-Verlag, London, 2013.

[Yan02] Ching-Nung Yang, "Improvements on Audio and Optical Cryptography", JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 18, 381-391 (2002).

[ShSu00] Shi, Y. Q., and Sun, H.; "Image and Video Compression for Multimedia Engineering"; CRC Press LLC; United States of America; 2000.

[Umba98] Umbaugh, S. E.; "Computer Vision and Image Processing: A Practical Approach using CVIP Tools";Prentice Hall, Inc.; 1998.

[Salo07] Salomon, D.; "Data Compression"; Springer; United Kingdom; London; 4th Edition; 2007.

[Sal07] D. Salomon." Variable- Length Codes for Data Compression", Springer-Verlag, London Limited.2007

[Grg01] S. Grgic, M. Grgic," Performance Analysis of Image Compression Using Waelet" Member,IEEE, and Branka Zovoko-Chilar, Member, IEEE.Vol.48, NO.3, JUNE 2001

Summary

An Effective and secure protection of sensitive information is the primary concern in communication systems or network storage systems, it is important for any information process to ensure data is not being tampered. To achieve confidentiality and integrity of multimedia information, various Secret Sharing Schemes (SSS) have been developed.

This thesis presents a (2,*n*) threshold Secret Sharing Scheme, which is one of the types of SSS and a special kind of (*k, n)* threshold scheme. This scheme share a secret file among *n* participants, and any *two p*articipants could cooperate to reconstruct the secret file, otherwise participants could get nothing. The proposed scheme is designed to solve the problem of increasing share size in most threshold secret sharing schemes besides guaranteeing security with low complexity..

The proposed scheme prototype consists of two modules: *Encoder* and *Decoder*, the input to the former is plain audio data (.WAV) while the output is *n* shares. *Encoder* is passes through two stages: *compression* and *encryption*. *Compression* is achieved by performing Discrete Cosine Transformation DCT, Quantizing, Run-Length Encoding, and Shifting Coding stages sequentially. While *encryption* is acquired by diffusion, implementing Random Coefficients generating and Share Functions generating stages consecutively.

The performance of the proposed scheme was tested, using some fidelity measures such as Mean Square Error (MSE) and Peak Signal Noise Ratio (PSNR) to measure the rate of error while Compression Ratio (CR) is used to measure the compression efficiency. The test result shows encourage results.

Republic of Iraq
Ministry of Higher Education
and scientific research
Al-Nahrain University
College of Science
Department of Computer Science

Share Audio Cryptography Using DCT Transform

A Thesis

Submitted to the College of Science/Al-Nahrain University as a partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

By

Sarah Saad Ali Albaghdady
B.Sc. 2003 Computer Science / College Science / Al-Nahrain University

Supervised by
Dr. Abeer M. Yousif
(Assistant Professor)

April 2014 **Second Jamadi 1435**