SECURE CLOUD STORAGE USING SECRET SHARING SCHEME

BY

IBRAHIM ABDULLAH AL-THAMARY

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER NETWORKS

MAY 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **IBRAHIM ABDULLAH AL-THAMARY** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKS** .

Thesis Committee

Dr. Talal Mousa Al-Kharobi
(Adviser)

Dr. (Co-adviser)

Dr. Tarek R. Sheltami
(Member)

Dr. Marwan H. Abu-Amara
(Member)

Dr. (Member)

Dr. Ahmad S. Almulhem
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

4/6/12
Date

Dedication to my parents for their guidance,patience, and support.
To my dear lovely wife and to my son for their extra patience.
To whom I know.

ACKNOWLEDGMENTS

All praise is due to Allah, Subhanahu-wa-Taala, for his limitless blessing and guidance. May Allah bestow peace on his prophet, Muhammad (Peace and blessing of Allah be upon him) and his family. All my appreciation and thanks to my thesis advisor, Dr. Talal Mousa Al-Kharobi , for his guidance and help all the way till the achievement of this thesis. I would like also to thank my thesis committee members, Prof. Tarek Sheltami and Dr. Marwan Abu-Amara for their cooperation and constructive comments. All my thanks to the Computer Engineering Department. My deepest appreciation,thanks ,and acknowledgment to King Fahd University of Petroleum and Minerals (KFUPM) for the full support. My thanks also to all my colleagues and friends, who encouraged me a lot in my way to the achievement of this work. Last but not least, my ultimate thank and love for my parents, brothers, sisters, son, and my wife for their endless support and love.

TABLE OF CONTENTS

ACKNOWLEDGEMENT **v**

LIST OF TABLES **viii**

LIST OF FIGURES **x**

ABSTRACT (ENGLISH) **xv**

ABSTRACT (ARABIC) **xvi**

CHAPTER 1 INTRODUCTION **1**

1.1 Cloud Computing 2

1.2 Problem Definition 4

1.3 Thesis Organization 5

CHAPTER 2 LITERATURE REVIEW **6**

2.1 Secret Sharing Scheme 6

2.1.1 Threshold Secret Sharing Schemes 6

2.1.2 Shamir Secret Sharing Scheme 8

2.1.3 Using Secret Sharing Mechanism to Secure Images 10

2.1.4 Securing Files in the cloud 14

2.2 Base64 Encoding 16

CHAPTER 3 PROPOSED SCHEME **19**

3.1 Saving Files 24

3.1.1 Preparing the File 24
3.1.2 Shares Building 24
3.1.3 Shares Distribution 25
3.2 Loading File 26
3.2.1 Secret Reconstruction 26

CHAPTER 4 EXPERIMENT RESULTS 27
4.1 Create Shares/Encryption 28
4.2 Reconstruct The Secret/Decryption 31
4.3 Enhance The Result 35
4.3.1 Create the Shares/Encryption 35
4.3.2 Reconstruct The Secret/Decryption 45
4.3.3 Compression and Decompression 54
4.4 Upload Process. 67
4.5 Download Process. 68
4.6 Conclusion 69

CHAPTER 5 CONCLUSION AND FUTURE DIRECTIONS 71

REFERENCES 73

APPENDICES 80

APPENDIX A CREATE THE SECRET 80

APPENDIX B RECONSTRUCT THE SECRET 102

VITAE 124

LIST OF TABLES

2.1 Comparison Between the most Important Threshold Secret Sharing Techniques. 8

2.2 Comparison of Different Secret Image Sharing Mechanism. 15

4.1 Sequential implementation of Creating the Shares and Encryption using AES of most Popular File Formats. 36

4.2 Parallel implementation of Creating the Shares and Encryption using AES of most Popular File Formats. 37

4.3 Sequential implementation of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes. 41

4.4 Parallel implementation of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes. 43

4.5 Sequential implementation of Reconstruct the Secret and Decryption using AES of most Popular File Formats. 45

4.6 Parallel implementation of Reconstruct the Secret and Decryption using AES of most Popular File Formats. 47

4.7 Sequential implementation of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes. 50

4.8 Parallel implementation of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes. 52

4.9 Uploading Time in Second for Different Files with Different Sizes in Parallel to Cloud Storage. 67

4.10 Download Time in Second for Different Files with Different Size in Parallel from Cloud Storage. 69

LIST OF FIGURES

1.1 An Example of Cloud Computing [1]. 3

2.1 Constructions of Secret Sharing. 7

3.1 Saving a File to the Cloud 20

3.2 SFS Model for Uploading a File to the Cloud 21

3.3 Loading a File from the Cloud 22

3.4 SFS Model for Download a File from the Cloud 23

3.5 Shares Building [2] 25

3.6 Shares Distribution [2] 25

3.7 Secret Reconstruction [2] 26

4.1 Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 5$ and $t = 3$ Versus Encryption the Same File using AES. 29

4.2 Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 8$ and $t = 2$ Versus Encryption the Same File using AES. 30

4.3 Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 8$ and $t = 3$ Versus Encryption the Same File using AES. 30

4.4 Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 8$ and $t = 6$ Versus Encryption the Same File using AES. 31

4.5 Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 5$ and $t = 3$ Versus Decryption the Same File using AES. 32
4.6 Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 8$ and $t = 2$ Versus Decryption the Same File using AES. 32
4.7 Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 8$ and $t = 3$ Versus Decryption the Same File using AES. 33
4.8 Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 8$ and $t = 6$ Versus Decryption the Same File using AES. 34
4.9 Performance Comparison of the Sequential Execution Timevof Creating the Shares and Encryption using AES of most Popular File Formats. 36
4.10 Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of most Popular File Formats. 37
4.11 Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of most Popular File Formats. 38
4.12 Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of most Popular File Formats. 39
4.13 Speed Up of Creating the Shares and Encryption using AES of most Popular File Formats 39
4.14 Speed Up of Creating the Shares and Encryption using AES of most Popular File Formats using pc with better features 40
4.15 Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes. 42

4.16 Throughput of the Sequential Execution Time of of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes. 42
4.17 Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes. 44
4.18 Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes. . 44
4.19 Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats. 46
4.20 Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats. 46
4.21 Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats. 47
4.22 Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats. . . 48
4.23 Speed Up of Reconstruct the Secret and Decryption using AES of most Popular File Formats. 49
4.24 Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes. 51
4.25 Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes. 51
4.26 Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes. 53

4.27 Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes. . . . 53
4.28 Performance of Compression PDF File type. . . . 54
4.29 Performance of Decompression PDF File type. . . . 55
4.30 File Share Size before Compression and after Compression of PDF File type. . . . 55
4.31 Performance of Compression Audio File type. . . . 56
4.32 Performance of Decompression Audio File type. . . . 56
4.33 File Share Size before Compression and after Compression of Audio File type. . . . 57
4.34 Performance of Compression Binary File Type. . . . 57
4.35 Performance of Decompression Binary File Type. . . . 58
4.36 File Share Size before Compression and after Compression of Binary File Type. . . . 58
4.37 Performance of Compression Document File Type. . . . 59
4.38 Performance of Decompression Document File Type. . . . 59
4.39 File Share Size before Compression and after Compression of Document File Type. . . . 59
4.40 Performance of Compression Executable File Type. . . . 60
4.41 Performance of Decompression Executable File Type. . . . 60
4.42 File Share Size before Compression and after Compression of Executable File Type. . . . 61
4.43 Performance of Compression Image File Type. . . . 61
4.44 Performance of Decompression Image File Type. . . . 62
4.45 File Share Size before Compression and after Compression of Image File Type. . . . 62
4.46 Performance of Compression Text File Type. . . . 63
4.47 Performance of Decompression Text File Type. . . . 63

4.48 File Share Size before Compression and after Compression of Text File Type. 63
4.49 Performance of Compression Video File Type. 64
4.50 Performance of Decompression Video File Type. 64
4.51 File Share Size before Compression and after Compression of Video File Type. 64
4.52 Performance of Compression Archive File Type. 65
4.53 Performance of Decompression Archive File Type. 65
4.54 File Share Size before Compression and after Compression of Archive File Type. 66
4.55 Performance Comparison of the Uploading Time for Different Files with Different Sizes in Parallel to Cloud Storage. 68
4.56 Performance Comparison of the Downloading Time for Different Files with Different Size in Parallel from Cloud Storage. 69

THESIS ABSTRACT

NAME: Ibrahim Abdullah Al-thamary

TITLE OF STUDY: Secure Cloud Storage Using Secret Sharing Scheme

MAJOR FIELD: Computer Networks

DATE OF DEGREE: May,2017

Cloud computing is a significant model for permitting on-demand network access to shared data, softwares, infrastructure, and platform resources. However, cloud storage needs a certain level of availability, confidentiality, and integrity. Information sensitivity and value require the use of a highly secure and reliable protocol. This work proposes a new mechanism to increase the user trust in cloud storage using secret sharing technique. The proposed algorithm uses Base64 encoding to convert files of any type to ASCII strings which will then be used to create the secret. The file does not need any extra process to be converted to Base64 string and this can speed up the share building process. To increase the trust on the cloud service provider and to store the data securely each string will be divided to N shares (using Shamir Secret Sharing Scheme) where each share is stored in different clouds. Then the secret should be recontract from the k shares.

خلاصة الرسالة

الاسم: **ابراهيم عبدالله مصلح الذماري**
عنوان الرسالة: **تأمين تخزين البيانات في للحوسبة السحابيه باستخدام تقنيه تقاسم السر(secret sharing)**
التخصص: **شبكات الحاسب الآلي**
تاريخ التخرج: **شعبان 1438 هـ**

تعتبر آلحوسبة السحابية من اهم النماذج التي تسمح بالوصول إلى المصادر آلمتوفرة في الشبكة عند الطلب حيث يتم الوصول إلى البيانات المشتركة والبرمجيات والبنية التحتية والمنصات البرمجية . بالرغم أن الحوسبة السحابية لأزالت بحاجةٍ ماسةٍ إلى السرية و المصداقية والتوافر، فلذلك المعلومات المهمة والحساسة بحاجة إلى بروتوكول مرن وامن لحمايتها. هذا العمل يقدم تقنية جديدة تعمل على زيادة الثقة أتناء تخزين البياناتات في الحوسبة السحابية باستخدام تقنية تقاسم السر (secret sharing) . نستخدم في هذه آلتقنية ترميز آلآساس 64 لتحويل البيانات من آي نوع إلى نص "أسكي" وسيتم استخدام هذا النص بعد ذلك للإنشاء أجزاء السر. إن عمليه تحويل الملف إلى آلآساس 64 لا تتطلب معالجه أضافية مما يعمل على تسريع عمليه أنشاء أجزاء السر. للعمل على زيادة آلوثوقية في مزود خدمه آلحوسبة آلسحابية ولضمان خزن البيانات بشكل امن ، سيتم تقسيم كل نص إلى عدد من آلأجزاء باستخدام (Shamir Secret Sharing Scheme) حيث سيخزن كل جزء (share) في حوسبة سحابية مختلفه. كما ان السر لأيمكن استرجاعه أذا لم يتوفر الحد آلأدنى من آلأجزاء (shares).

درجة الماجستير في شبكات الحاسب الآلي
جامعة الملك فهد للبترول والمعادن
الظهران- اللملكة العربية السعودية
شعبان 1438 هـ

CHAPTER 1

INTRODUCTION

Due to the rapid advancement in the e-world and the fast growth of using the internet, information security systems should be developed to protect the privacy of users. This can be accomplished using cryptography, steganography, or/and secret sharing. Securing data becomes a big concern in certain environments like local network, wireless network, the internet or/and cloud computing. Having a single copy of the data will increase the possibility of losing the data as it is impossible to retrieve the data if this copy is destroyed. In other words, the possibility of losing data is high when there is only a single copy of the information on a single location. Having many copies of data may increase the reliability. However, the existence of data in more than one locations may reduce the confidentiality as it gives more chances to the attackers. Therefore, there is a need for a technique to enhance both the availability and the confidentiality of the data which motivates the use of secret sharing method. Base64 encoding is a mechanism to convert data to ASCII string which is commonly used in e-mail to make the content unreadable. In addition, base64 is one of the best and most popular encoding/decoding schemes on the internet. Trillions of bytes are encoded and decoded each day using base64. In this work, we propose a new approach to increase the user trust in the cloud using secret sharing. The proposed technique will take any file type as input and convert it using base64 to ASCII strings. Then, each ASCII string, is a set to generate n shares. Then , the n shares are distributed one per cloud/location. The shares should be created such that the string can be regenerated by any t shares out of the n shares (where $t <= n$). The reconstruct process will use the ASCII format which makes the ability for storage and distributed easily.

1.1 Cloud Computing

According to National Institute of Standards and Technology(NIST), ”the cloud computing is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models” [3] . Cloud computing is an expression that indicates to resources and computer systems that are available on demand through the network, which can provide a number of combined computing facilities as shown in Fig. 1.1 without following the local resources in order to make it easier for the user, and include those resources space for data storage, backup and selfsynchronization, also include processing abilities of software and arrangement of tasks and push e-mail and remote printing, and the user can control when it is connected to the network in these resources using a simple software interface simplifies and ignores many details and internal operation [4, 5].

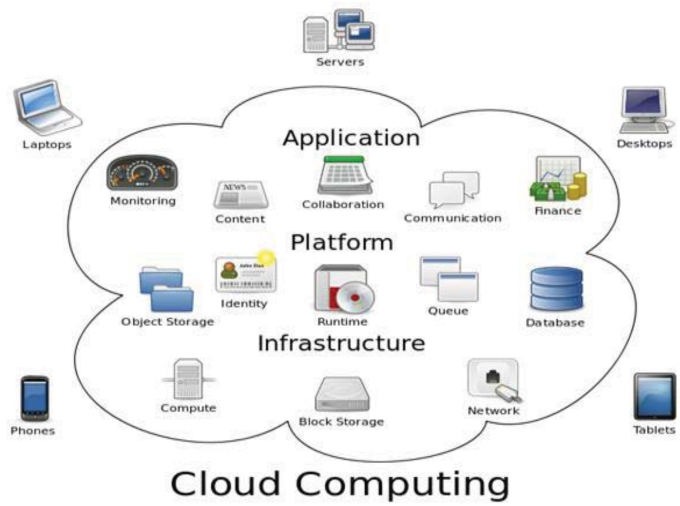

Figure 1.1: An Example of Cloud Computing [1].

There are many types of cloud depend on deployment public cloud, private cloud, hybrid cloud, and community cloud. In addition, the services provided by the cloud are divided into four main categories: IaaS (Infrastructure as a Service), SaaS (Software as a Service), PaaS (Platform as a Service), and (XaaS/AaaS) Anything as a Service. Moreover, the cloud consists of a number of storage servers and node manager/ a front-end server which manages the storage servers within the cloud [4, 6, 7].

1.2 Problem Definition

In the last few years, there is a remarkable development in the cloud. This development makes societies and companies start to use the cloud to store information. These societies and companies require certain security guarantees to be made before using these services. Thus, those security concern needs to be addressed because the attackers will attack any valuable data. Hence the focus is to find good techniques that will offer more than confidentiality. Pervious studies have introduced many techniques, one of this is a secret sharing. Group of those studies adopt using secret sharing. However, non of them showed the support of different file types such as (images, sound, video, executable, document file, etc). those technics have their own weakness and strength points. To support multiple file types, we will use Base64. As Base64 encoding will significantly increase the file size approximately 20-25% more than the size of the original file [8, 9] . Moreover, Base64 Encoding/Decoding process consumes resources [10] but it gives the ability to compress the file and reduce the size. Our proposed scheme increases the trust and security because each file is converted to the base64 string before applied the secret sharing mechanism, and then the string is compressed using GZIP compression [11] before/after creating the secret using the secret sharing scheme. We send the file in compressed form and the receiver decompress the file and gets the original file. In addition, The proposed scheme increases the confidentiality, and availability, and it gives the user more privacy because the data will be separated in different clouds.

1.3 Thesis Organization

The organization of this thesis is as follows. In Chapter 2, a literature review of several compression and compaction of the existing and most recent techniques is proposed for Secret Sharing Scheme and Base64. In Chapter 3, our proposed scheme is illustrated in details. Experimental results are demonstrated in Chapter 4. Finally, in Chapter 5, we conclude the thesis and suggest directions for the future work.

CHAPTER 2

LITERATURE REVIEW

This chapter is structured as follows section 1 a secret sharing and section 2 base64 encoding.

2.1 Secret Sharing Scheme

Secret sharing is a cryptographic tool that allows secret information to be shared among a group of people/machines such that predefined set(s) of them can together reveal the secret. There are different schemes of secret sharing as shown in Fig. 2.1. We will focus only on one category of secret sharing schemes called threshold schemes.

2.1.1 Threshold Secret Sharing Schemes

The idea of threshold secret sharing was proposed independently by Shamir [12] and Blakley [13]. In (t, n) threshold SSS, the secret s is split into n shares in such a way that any t participants or more can reconstruct or obtain s but participants

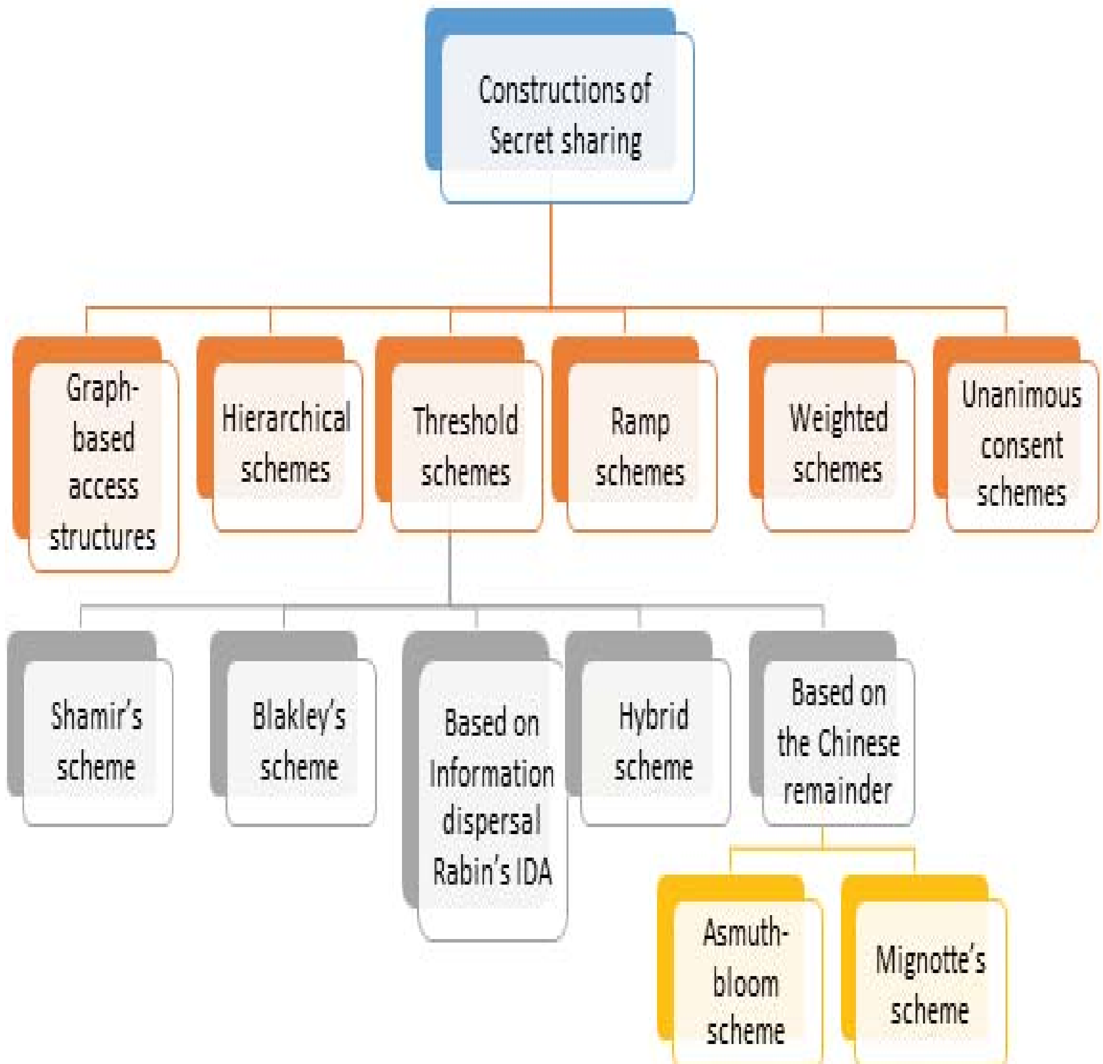

Figure 2.1: Constructions of Secret Sharing.

less than t cannot obtain any information about s [14] . The threshold schemes contains Shamirs scheme, Blakleys scheme, Information Dispersal Rabins IDA, the

Chinese remainder, and Hybrid scheme [15]. Blakley [13] introduced a threshold secret sharing method using linear geometry. His method solves the secret sharing problem and it has been used in secret image sharing technology. Moreover, as an example of Chinese remainder scheme, Mignott's [16] secret sharing scheme uses a special sequence of integers with CRT. Table 2.1 shows a comparison between the most important threshold secret sharing techniques.

Table 2.1: Comparison Between the most Important Threshold Secret Sharing Techniques.

method	**Year**	**Techniques Used**	**Advantage**	**Drawback**
Shamir [12]	1979	Polynomial based	Perfect ,Ideal	Not secure against cheaters.
Blakley [14]	1979	Geometry based	Ideal	Not perfect . It is less space efficient than Shamir"s scheme
Mignott's [16]	1982	CRT based	Ideal	Not perfect .

2.1.2 Shamir Secret Sharing Scheme

Shamir [12] introduced a threshold secret sharing approach in 1979 where particular secret messages are shared over n servers, the dealer D generates the polynomial $y = f(x)$ with degree $t - 1$, where t is the threshold. The polynomial will be as the following.

$$f(x) = s + a_1x + a_2x^2 + \cdots + a_{t-1}x^{t-1} \ mod \ p \tag{2.1}$$

Where p is a prime number, the coefficient $a_{\ i} \in Z_p$, $i = [2..., n]$ and x is the participant's ID. The dealer determines the shares and distributes them to n participants. For reconstruction, m participants, where $t \leq m \leq n$, are required to recollect their shares to the dealer and the dealer can perform the calculation using the lagrange interpolation equation (2.2) .

$$f\ (x) = \sum_{j=1}^{n} y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}\ (mod\ p) \tag{2.2}$$

To reconstruct the original polynomial, equation (2.2) is used for this purpose , where x_j, are the participant p_j's ID and y_i are the participant's share. Finally, the dealer adds the value at x = 0 to the $f\ (x)$ which gives the secret

$$f\ (0) = s$$

For better understanding an example of the secret of Shamir's is in order.

Let $n = 5$ and $t = 3$ and the secret is 19. A possible polynomial is $f(x) = 15x^2 + 13x + 19$ over the field Z_{23} where p is 23.

We generate the following five shares of secrets.

$$s1 = f(1) = (15 \times 1^2 + 13 \times 1 + 19)\ mod\ 23 = 1$$

$$s2 = f(2) = (15 \times 2^2 + 13 \times 2 + 19)\ mod\ 23 = 13$$

$$s3 = f(3) = (15 \times 3^2 + 13 \times 3 + 19)\ mod\ 23 = 9$$

$$s4 = f(4) = (15 \times 4^2 + 13 \times 4 + 19)\ mod\ 23 = 12$$

$$s5 = f(5) = (15 \times 5^2 + 13 \times 5 + 19)\ mod\ 23 = 22$$

To reconstruct the secret we choose shares $s1(1,1)$, $s2(2,13)$ and $s3(3,9)$. We use lagrange interpolation as follows :

$$1 \times \frac{2}{2-1} \times \frac{3}{3-1} + 13 \times \frac{1}{1-2} * \frac{3}{3-2} + 9 \times \frac{1}{1-3} \times \frac{2}{2-3} \quad mod \quad 23$$

$$= (1 * 3 + 13 * 20 + 9 * 1) \quad mod \quad 23$$

$$= 272 \quad mod \quad 23 = 19$$

$$f\ (0) = 19$$

2.1.3 Using Secret Sharing Mechanism to Secure Images

In this section, we will review pervious of research work conducted on image sharing techniques. Although many of them are quite good, there are still many challenges in this field. The main idea of secret image sharing schemes is to hide the secret image into number of images and distribute these images to different participants. Table 2.2 shows a comparison of various secret image sharing techniques.

Lin and Thien [17]. proposed secret image sharing scheme with the ability of share data reduction. A secret image is first distributed into blocks of size less than 250 pixels, and by decreasing the size of the shared images, it is easy to deal with each part in the image individually.

Lukac, et.at. [18] proposed colour image secret sharing that works in the decomposed bit-levels (binary pixels of binary share) of the input color vectors to change both spectral correlation characteristics and spatial position of the share results and generate random, color- noise-like images for protecting communication and secure access.In the decryption process ,the perfect reconstruction property recovers the original color image by logically decrypting the decomposed bit vector-arrays of the color shares.

Lou et al. [19] proposed color visual secret sharing scheme which uses non-expanded meaningful shares. They are used to hide a secret image into two meaningful cover images. The build of shares occurs without using pixel expansion. At the same time, this scheme makes the sharing of a color image more secure and adds extra confidentiality. The secret image can be revealed by overlapping both of them without complexity. Moreover, the validity of the secret image can be checked at the receiver side.

Tsai et al. [20] introduced a secret color image sharing method with the size constraint that uses neural networks combined with visual secret sharing. Adding neural networks improved the memory usage, increased performance of bandwidth, and saved power and time. Furthermore, this method supports

24-bit color and the results show the good quality of the reconstructed image but the variance between cover images and camouflage images are not visually distinguishable.

Chen et al[10]. proposed (2, n) and (n, n) scheme for secret image sharing based on random grids. During the process of image encrypting and decryption, there is no pixel expansion which gives this scheme an advantage. In this method, codebook is used in the encryption process. At the receiver end the decryption shows up by superimposing not less than 2 shares in (2, n) scheme and all n shares in (n, n) scheme without requiring any computation. The results of the secret reconstruction can be recognized by a human.

Alex et al. [21] suggested various methods for error diffusion in order to increase the quality of the image in the halftone shares. The halftone visual cryptographic is used to fit snugly the pixels of secret information into previously encoded halftone shares. Visual cryptographic combined with halftone in which the continuous-tone image is transformed into a binary image then apply visual secret sharing to it. By using the error diffusion, the complexity is decreased and the quality of the image is increased. The secret image reconstructs occurs when the stacking shares combine together and the reconstructed secret image does not suffer from cross interference of share images.

Yang et al. [22] introduced visual secret sharing scheme using (2, 2), (2, n), and (k, n) which is based on a probabilistic method with non-expandable shares size of pixels. The contrast level of this scheme is similar to the traditional visual secret sharing scheme. Moreover, they showed by using transfer function how to convert from the traditional VSS scheme to probabilistic VSS scheme. The rate of the white pixels is used for displaying the color contrast of the reconstructed secret image.

Lin, et al. [23] introduced a framework for multiple secret sharing scheme without pixel expansion. In this framework, encoding the secret images does not require codebook. It was found that the pixel expansion was four times less compared to earlier schemes in their literature review after applying aspect ratio constraints. Over the separation and camouflaging processes, two share images turn into meaningless images which did not leak any information about the secret images. To reconstruct the secret, each share was flapped and human visual system (HVS) was capable of identifying the reconstructed image. This scheme has very good quality in reconstructing the secret and resolve the pixel expansion problem.

Sasaki et al. [24] introduced the formulation of VSS encryption for multiple images. The limitation of the extended visual cryptography schemes (EVCS) is that each share had the further secret image linked with it. The limitation of VSS-q-PI is the multiple secret images associated with the matching shares in capable sets but the shares in forbidden sets must be similar. Therefore, generalized VSS scheme for encrypting multiple secret images was introduced.

He, et al. [25] proposed a novel (t, n) image that is gradually enhanced by using lossless compression for Images (LOCO-I) compression. Additionally, by embedding the hash-based message the three types of cheating will probably be detected. Moreover, they improved the security by using a random strategy with dynamic embedding. This scheme and the proposed scheme in [26] divided the shadow or image into groups.

Askari et al. [27] developed the VSS scheme which is given by proposed (2, 2) VSS scheme without image size expansion. His scheme is based on encrypting a secret block with four pixels into two shares depending on the distribution of BW pixels. This can lead to reconstruct the secret image by using XOR operation. This scheme can apply to binary or halftone images.

Liu et al. [28] developed a new color VCS that depends on the improved VC. In this scheme, the secret color image is shared over n-1 arbitrary natural images and one noise-like share image. Instead of modification natural image properties, the encryption takes the features from all the natural images. This proposed scheme can efficiently reduce the transmission risk and solve the share management problems. This method succeeds in dealing with the problem of expansion of pixel and makes it easy to reconstruct the secret images without any change in the image quality. Due to this, the suggested scheme can deal with greyscale pixels or color images. Table 2.2 shows a comparison of various secret image sharing techniques.

2.1.4 Securing Files in the cloud .

As all information is basically converted to digital format, the need for secure manipulation is dramatically increasing. Attacking data storage is a target for

Table 2.2: Comparison of Different Secret Image Sharing Mechanism.

Schemes	**year**	**Techniques Used**	**Meaningful shares**	**Type of image**	**Pixel Expansion**
Lin and Thien [17]	2002	Polynomial based	No	Grayscale	No
Lukac, et.at [18]	2004	Decomposed bit-levels	No	Color	No
Lou et al [[19].	2011	Cover Image	Yes	Color	No
Tsai et al. [20]	2009	combination	No	Color	Yes
Chen et al [29]	2009	Random Grids	No	Grayscale	No
Alex et al. [21]	2011	Error Diffusion	No	Grayscale	Yes
Yang et al. [22]	2004	Probabilistic	No	Grayscale	No
Lin, et al. [23]	2010	Multiple Secrets	No	Grayscale	No
Sasaki et al. [24]	2014	Multiple Secrets	No	Grayscale	No
Askari et al. [27]	2012	XOR operation	No	Grayscale	No
Liu et al. [28]	2013	NVSS	Yes	Color	No

the attackers in order to access to unauthorized information. In order to keep this information secure and to allow only legitimate access, many researchers have proposed different methods for securing the process of storing files.

Kallahalla et al. [30] proposed a scalable secure file sharing on untrusted storage called PLUTUS. The main goal of this method is to provide information owners with direct control access to their files as well as the key management. This scheme is based on RSA. The encrypt/decrypt of the file is done on the client side, not on the server side which increases the trust.

Dong et al. [31] proposed a high level of scalability, user privacy, and effective data sharing in the cloud by merging the CP-ABE (Cipher text-Policy -Attribute

Based Encryption scheme) with IBE (Identity Based Encryption Scheme). This proposal gives data owners the ability to assign different access privileges to users as well as to give or deny any access privileges to them. At the same time, the cloud is not allowed to read or access files shared by data owners.

Bessani, et al. [32] proposed DEPSKY-CA protocol dependable and secure storage in a cloud-of-clouds to improve the confidentiality and availability by using secret sharing combined with symmetric encryption and distributed them in multi-cloud. Alsolami and Boult [33] proposed CloudstaSh that applied Shamir secret sharing scheme [12] directly on the file and distribute the shares to multi-cloud. According to this work , the CloudStach is not statically significant for large file .By applying Shamir secret sharing scheme on the text file with different sizes (1KB, 10KB, 100KB, 1MB, and 10MB) , the confidentiality and availability were increased. Moreover, they just created eight shares with a threshold of two which is not enough to show how good their work .

2.2 Base64 Encoding

Base64 [34–39] is an encoding scheme that scans a stream of bytes and converts every 3 bytes (24 bits) into 4 blocks of 6 bits. Then the algorithm uses its dictionary to convert each resulting block (decimal 0 - 63) into US-ASCII character (encoded with 8 bits) by converting the binary data to "ASCII string" and then sending the data. On the receiver side, the "ASCII string" is converted back into the original binary data. Base64 encoding adds a padding character when the number of bytes is less than three or not a multiple of 3. If the total number of bits in the text are 3n+1, the encoder adds one ”=” at the end of encoded text while if the total number of bits in the text are 3n+2, it adds two ”==” at the end of encoded text.

Base64 decoding process is the reverse of encoding process when decoding Base64 text, four characters are returned back to be three bytes. In addition, the padding character ’==’ shows that the four characters will be decoded to only a single byte while ’=’ shows that the four characters will be decoded to only two bytes [36,40]. Base64 algorithm is mainly used when there is a necessity to encode binary data as ASCII text that needs to be stored or transferred. Trillions of data bytes are base64 encoded/decoded each day [35]. Base64 is generally used for sending e-mail via MIME (Multipurpose Internet Mail Extensions) .However, the idea of base64 is not to send secure email but rather to convert the e-mail to make it difficult to understand its content directly [39]. Moreover, it is specifically used with email attachments, including files of many different types, such as images, sound, video, executable, document file, etc. In addition, base64 is one of the most popular encoding styles to transfer 8-byte code on the internet and Base64 is used widely through which the data is usually put in URL [35]. This is to make sure that the data remains as such without modification during transmission [8]. Furthermore, Base64 has various applications more than sending e-mail such as sending the image as SMS, using Base64 to obscure passwords and sending secret messages without using cryptography and using keys to encrypt and decrypt the message. It can be used for inserting binary data in an XML file and it can be used against web filters because Base64 changes the input file hence the keyword filtering cannot be used in the encoded file [38]. Additionally, Base64 is used to minimize the number of requests to the server by adding image data in HTML code and image encoded data can be saved inside the database and can generate the image file [9]. Moreover, Base64 and AES algorithm are utilized to enhance the security of data [39] and to represent a hash block size such as 128bit or 256bit (SHA/MD5). Converting the output into Base64 makes it much easier to display the hash [40] .

CHAPTER 3

PROPOSED SCHEME

The proposed technique called Secure File Sharing (SFS) which is mainly used to protect data and increase the level of availability because the data will be available in multi cloud and also increase the level of confidentiality because the attackers need to compromise more than one cloud (equal the threshold) to get access to the data. Our work is based on base64 and Shamir Secret Sharing Scheme [12] which is a perfect and ideal threshold scheme that boosts the security of data and gives the client more trust in cloud computing. Our method can take any data file as input (image, document, system file, audio, and video... etc.) and compress the file, then the file is converted to base64. Then Shamir Secret Sharing mechanism [12] is applied to generate n shares and distribute them to n different cloud providers. The secret should be regenerated by any t of the n shares (where $t <= n$). The threshold value, t, can be selected according to the security requirements for particular situations. All outputs are in ASCII printable text which makes them easy to store and distribute. The design of our scheme can be divided into two main procedure: Save and Load. Fig. 3.1 shows the flowcharts of uploading a file to the cloud and Fig. 3.2 illustrates Secure File Sharing (SFS) model for uploading a file to the cloud. Fig. 3.3 shows the flowcharts of downloading a file from the cloud and and Fig. 3.4 demonstrates SFS model for download a file from the cloud.

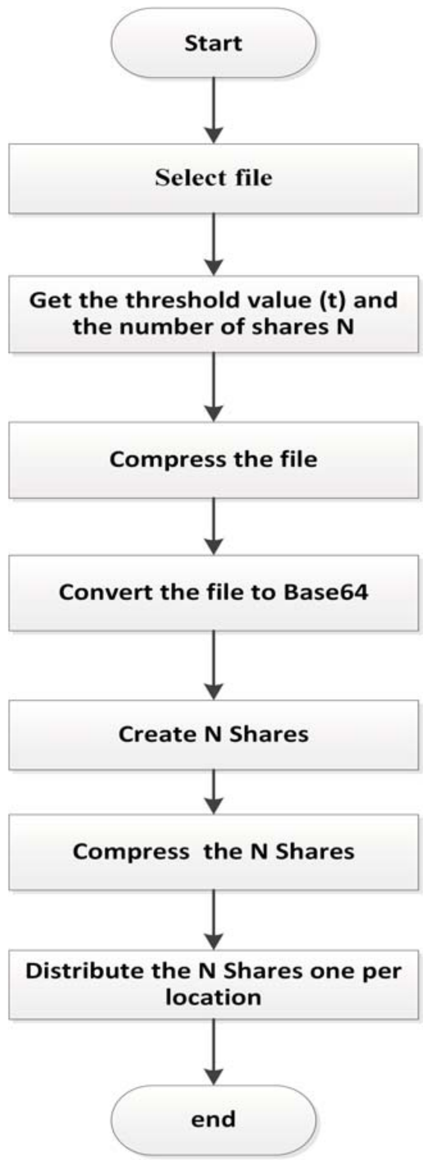

Figure 3.1: Saving a File to the Cloud

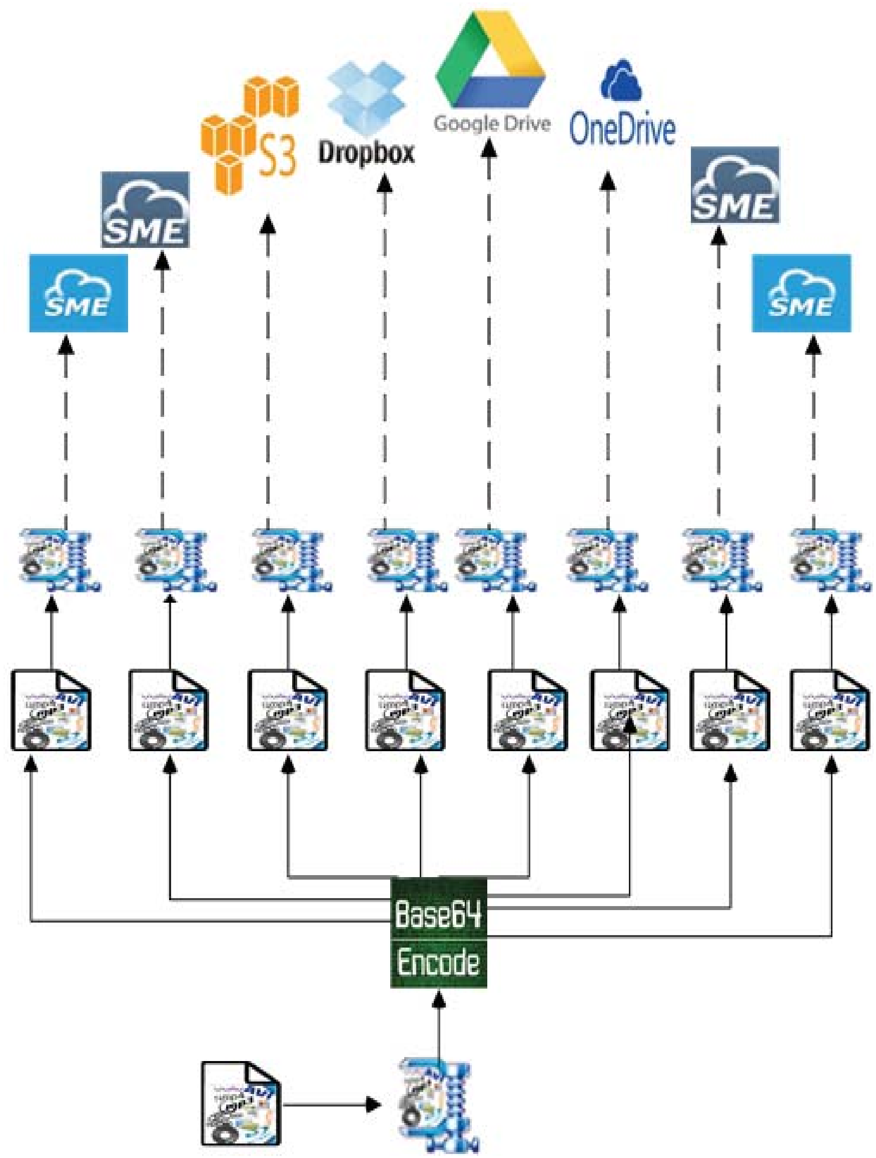

Figure 3.2: SFS Model for Uploading a File to the Cloud

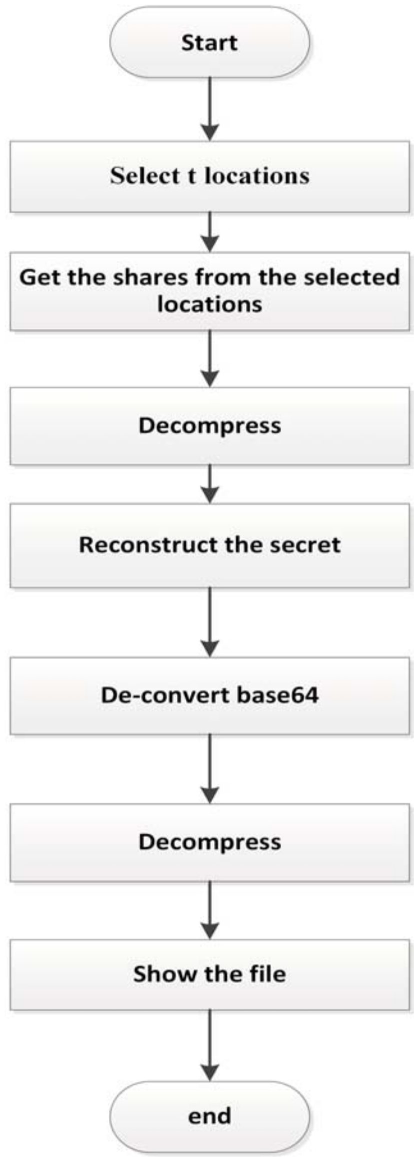

Figure 3.3: Loading a File from the Cloud

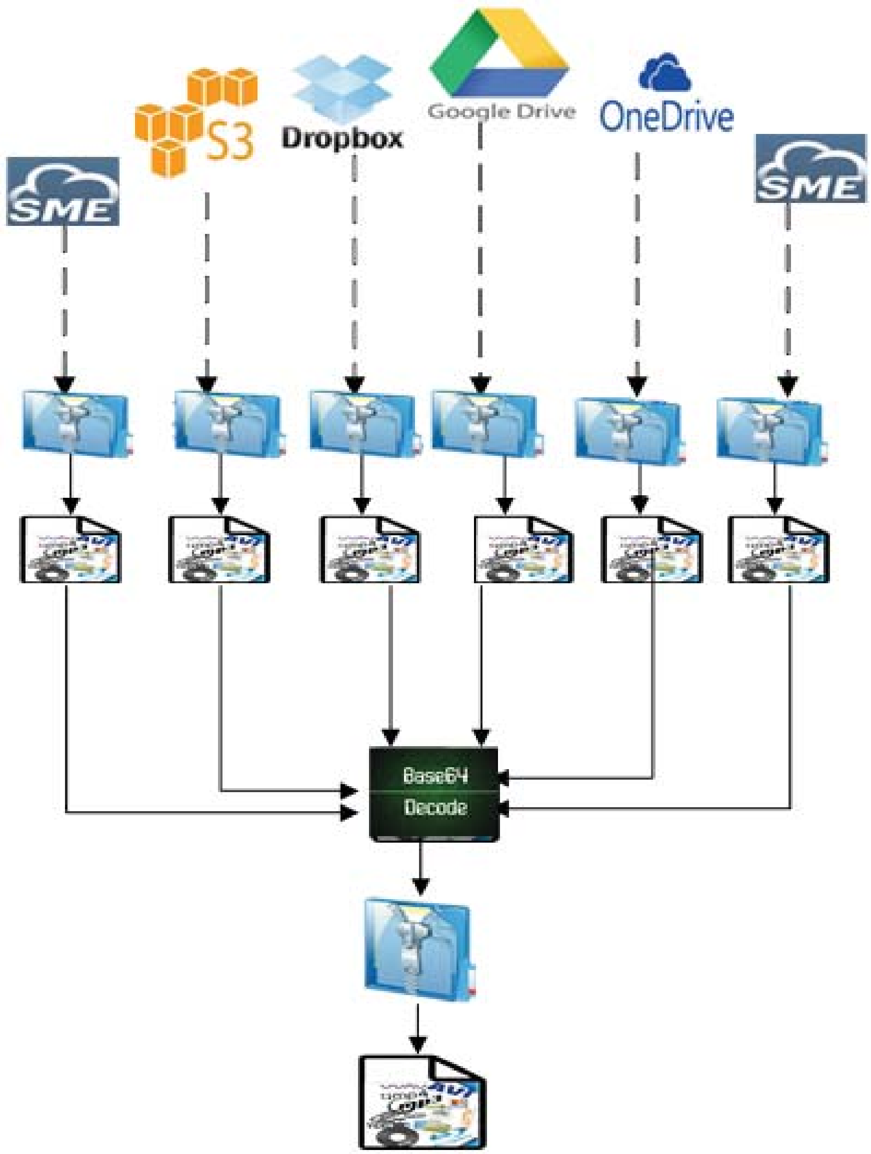

Figure 3.4: SFS Model for Download a File from the Cloud

3.1 Saving Files

3.1.1 Preparing the File

After selecting a file, the file will be compressed and then it will be converted using base 64 to be ready for the next step.

3.1.2 Shares Building

In this step , we divide the file to chunks and generate n shares as illustrate in Fig. 3.5 ,we need the following information [2] :

- The secret: one of the divided chunks.
- The trusted participants (shareholders): the people/machines that can keep the generated shares of the secret. These shares will be distributed by allocating one share for each participant. In our case, the number of cloud provider that we will use to store the data.
- The threshold value of secret: A qualified subset is a subset of the shareholders that should be able to rebuild the secret. In our case, it is the minimum number of location that we require to reconstruct the secret.

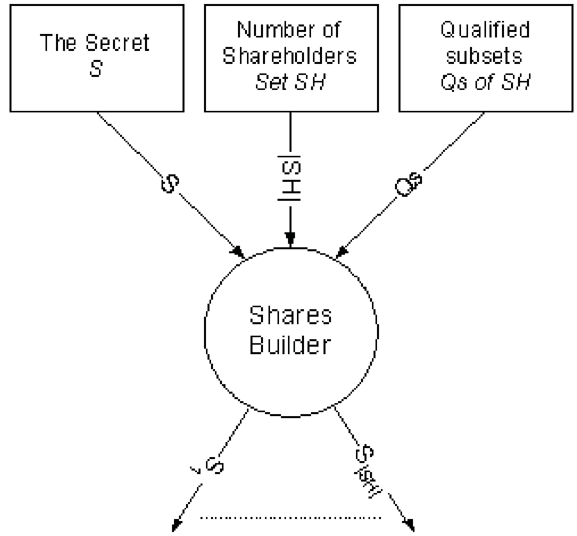

Figure 3.5: Shares Building [2]

3.1.3 Shares Distribution

After generating n shares , the file is compressed . Then we will have n files ,each of which will be uploaded to a separated cloud (shareholder) as demonstrate in Fig. 3.6.

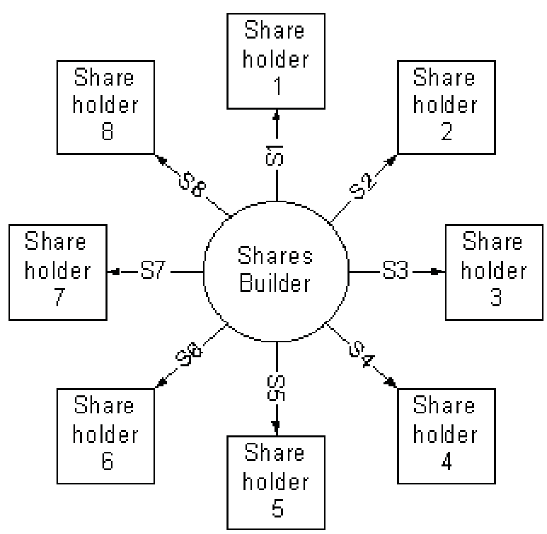

Figure 3.6: Shares Distribution [2]

3.2 Loading File

3.2.1 Secret Reconstruction

To reconstruct the secret, the users must select number of clouds provider that are equal to the threshold that is selected during the shares building. The authorized users who own the file can reconstruct the file and get access to the share easily as shown in Fig. 3.7.

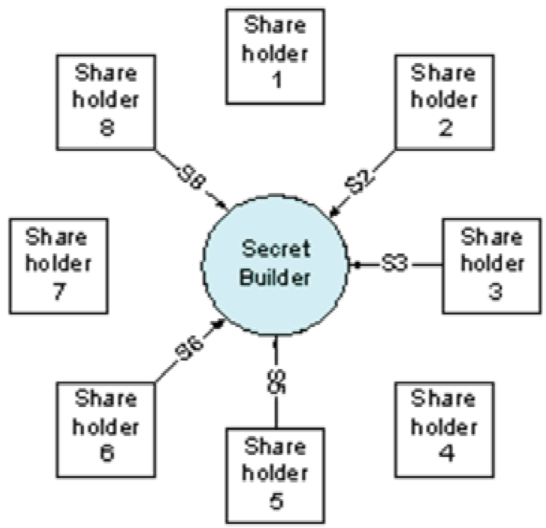

Figure 3.7: Secret Reconstruction [2]

CHAPTER 4

EXPERIMENT RESULTS

We conducted several experiments to evaluate our proposed scheme and compare it with existing solution. We conducted the average time needed by our scheme during the process of creating the shares and reconstructing them with confidence interval 95%, we compare between time needed by our scheme and symmetric encryption Advanced Encryption Standard (AES). We select AES to compare with because it is the encryption algorithm that is announced by NIST to replace the DES and 3DES and it was selected as the best encryption standard [41, 42] . Moreover, it has been used in [32, 33, 43–45]. We use AES 256 bit with cipher feedback mode (CFB) mode and for hashing we use SHA512 [33] . We use System.Security.Cryptography in C#.NET to implement them. Moreover, we apply secret sharing on the key to divide it into many shares and then distribute them and storage them in n-cloud. Our work is implemented using C#.NET with different cloud API and on a machine with this features "Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz (4 CPUs), 2.4GHz ,6GB RAM and 64-bit Windows operating system". The speed of the Internet is 2.2 Mbps on average. We used four different clouds (OneDrive, Google Drive, Dropbox, and SMEStorage hosted on Amazon S3 in five different places) and the performance assessment of these clouds storage can be found in [46].

4.1 Create Shares/Encryption

We run the experiment by applying different number of shares and different number of thresholds for the same dataset that contains different file sizes and different file types. To deal with a large file, we divide the file into chunks where every chunk is at most 200KB and if the file less than 200KB we take as its . Then we apply the same process for the small file in both methods. We run the experiment for 59 files with 26 different types of varying sizes. We compare our scheme Secret File Sharing(SFS) with the time needed to encrypt the same file using AES and Secret File Sharing(SFS) .

Fig. 4.1 shows the first test set using n=5 and t=3. The line graph illustrates that both schemes almost require the same execution time for small files. However, for file sizes of 10 MB or more , SFS consumes less time compered to AES. More importantly, the difference in execution time increases as the file size increase.

In Fig. 4.2 the line graph shows a comparison between SFS and AES algorithm when n=8 and t=2. For this case, it is obvious that both schemes consume comparable execution time. The line graph indicates that AES is slightly better than SFS for small file size while SFS is better than AES for the file sizes of

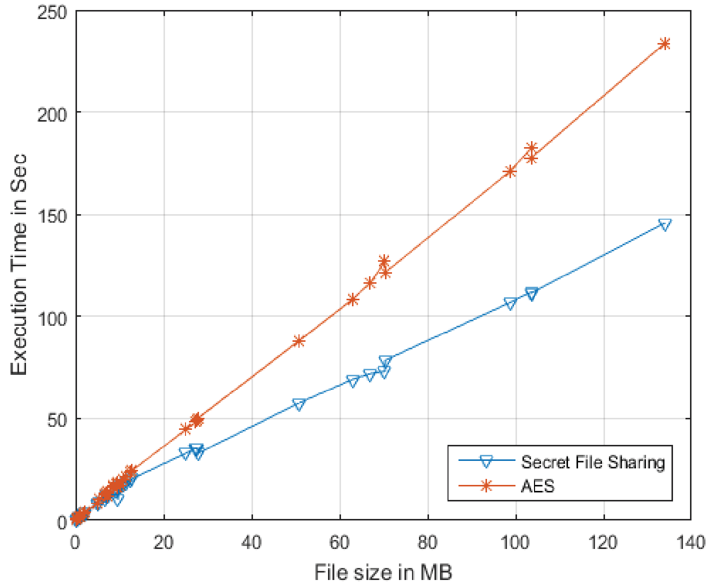

Figure 4.1: Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 5$ and $t = 3$ Versus Encryption the Same File using AES.

25MB and more . However, in this case the difference in the required time is almost constant and does not depend on the file size.

In Fig. 4.3 the line graph shows a comparison between SFS and AES algorithm when n=8 and t=3. As we can see when the threshold increases, the time needed of create the shares is almost equivalent to the AES encryption time. Fig. 4.4 shows the results when n=8 and t=6. As the threshold increases, the SFS execution time increases too. Here, the AES algorithm runs faster than SFS but the results are within the acceptable range.

Fig. 4.4 shows the results when n=8 and t=6. As the threshold increases, the SFS execution time increases too. Here, the AES algorithm runs faster than SFS but the results are within the acceptable range.

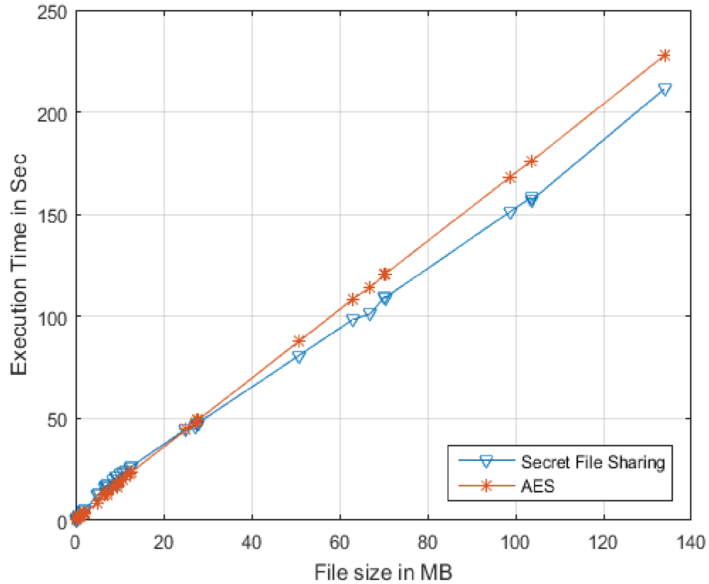

Figure 4.2: Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 8$ and $t = 2$ Versus Encryption the Same File using AES.

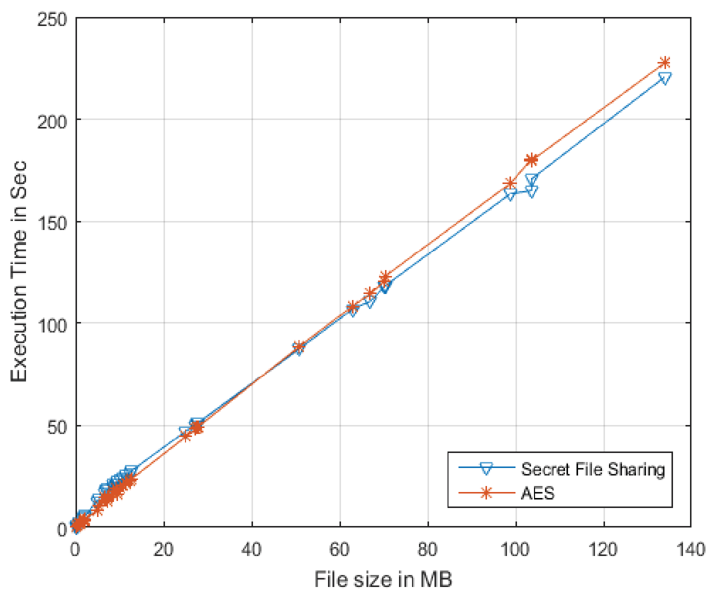

Figure 4.3: Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 8$ and $t = 3$ Versus Encryption the Same File using AES.

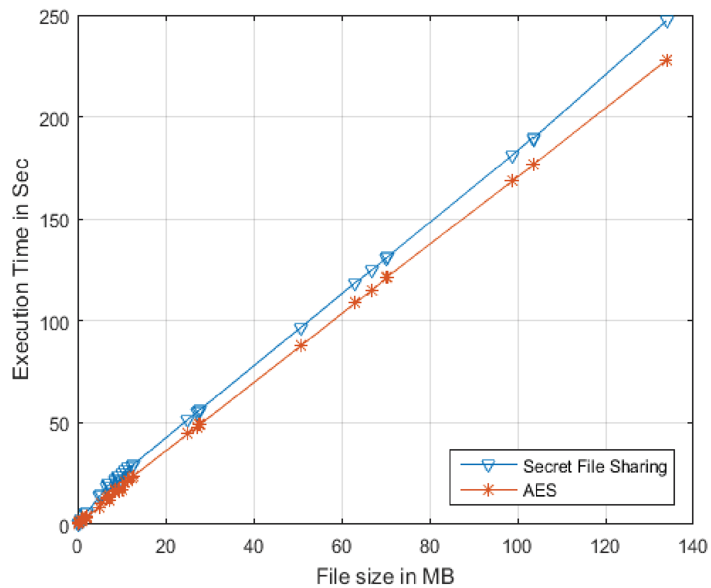

Figure 4.4: Performance Comparison of the Execution Time of Creating Shared of Files of Different Size when $n = 8$ and $t = 6$ Versus Encryption the Same File using AES.

4.2 Reconstruct The Secret/Decryption

Fig. 4.5 shows the execution time during the process of reconstructing the secret for our scheme versus decryption of the same file using AES algorithm with different types and sizes when n=5 and t=3. The line graph illustrates that the time of reconstructing the secret file using SFS is shorter than the decryption time using AES algorithm regardless of the file size. Moreover, as the file size increases the difference in excitation time increases as well. In Fig. 4.6 the line graph shows a comparison between SFS and AES algorithm when n=8 and t=2. The line graph clearly demonstrates the superiority of our scheme where the difference in execution time can reach more than two minutes for files of size 120MB. In Fig. 4.7 the line graph shows a comparison between SFS and AES algorithm when

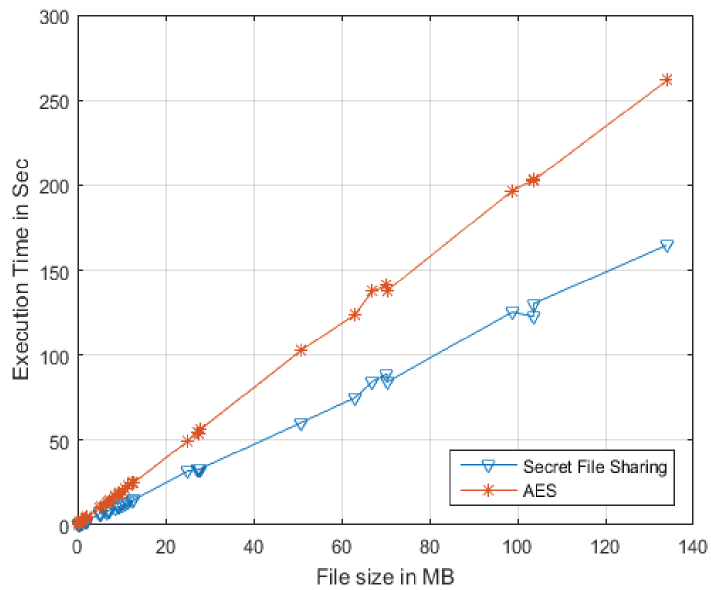

Figure 4.5: Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 5$ and $t = 3$ Versus Decryption the Same File using AES.

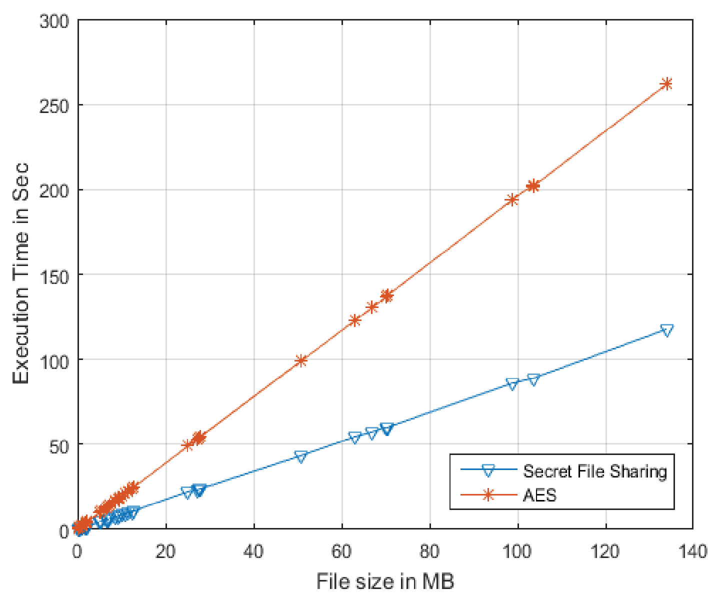

Figure 4.6: Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 8$ and $t = 2$ Versus Decryption the Same File using AES.

n=8 and t=3. The line graph shows that our SFS method is faster than AES in reconstructing the secret file. Moreover, the number of shares does not effect the result of the reconstructing process while the threshold plays the critical role. Fig. 4.8 shows the result when n=8 and t=6. As the threshold increases ,the SFS

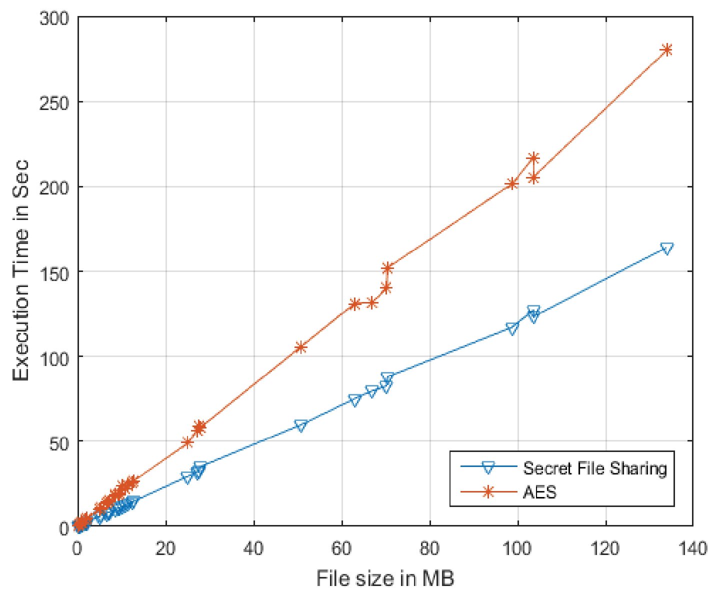

Figure 4.7: Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 8$ and $t = 3$ Versus Decryption the Same File using AES.

execution time increases as well. Here, the decryption using AES algorithm runs faster than SFS but the differences are within the acceptable range.

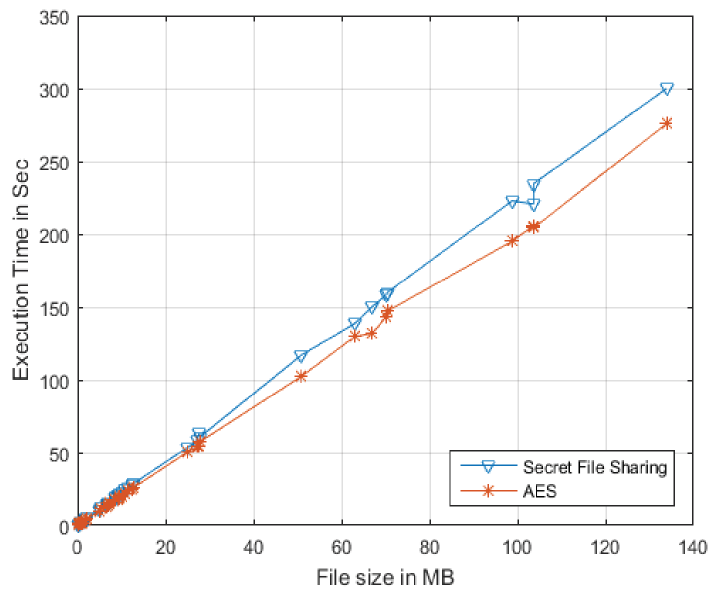

Figure 4.8: Performance Comparison of the Execution Time of Reconstruct the the File of Different Size when $n = 8$ and $t = 6$ Versus Decryption the Same File using AES.

4.3 Enhance The Result

We run our experiment with some modification to enhance the results . First, instead of applying the secret sharing on the base64 ,we use the index of the character in the Index Array. Index Array is an array that contains all the characters in Base64 encoding. Using Index Array will solve the cheating problem in Shamir Secret Sharing Scheme. During creating the shares , the files will be compressed using GZIP compression because the files becomes large when it converts to Base64. We run the experiment using most popular file types . Moreover, we apply the parallelization on both algorithms AES and our SFS scheme.

4.3.1 Create the Shares/Encryption

The most popular file formats

Table 4.1 and Fig. 4.9 show a performance comparison of the sequential execution time of create a secret and encrypt a different file types. The line graph illustrates that SFS schemes when (n=8, t=6), (n=8, t=3), and (n=8, t=2) almost require the same execution time for all file sizes . Moreover, both SFS schemes when (n=5, t=3) and (n=5, t=2) almost require the same execution time for all file sizes, while SFS scheme when (n=3, t=2) is better than other SFS schemes. Generally, it is noticed that SFS scheme consumes less time than AES. In addition, increasing the number of the shares effects on the execution time more than the number of thresholds.

We can see that SFS scheme when (n=3, t=2) is faster in performance than

Table 4.1: Sequential implementation of Creating the Shares and Encryption using AES of most Popular File Formats.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000806	0.272519	0.001215	0.003352	0.01846	0.002557	0.012505	0.008658
.tgz	0.011756	0.286453	0.014629	0.015214	0.015883	0.022333	0.021726	0.021126
.png	0.129359	0.597376	0.117879	0.166015	0.169303	0.231597	0.233924	0.257071
.exe	1.223342	3.74412	1.086153	1.574501	1.614991	2.215864	2.277921	2.351582
.pdf	13.15998	35.36956	11.22547	16.11115	15.59375	22.38146	22.29741	23.05456
.doc	20.19169	53.38245	16.8928	23.88537	23.55916	33.52248	34.19957	35.21744
.mp3	63.33237	170.8936	58.76135	79.57114	77.82943	114.5714	116.1646	119.5822
.jpg	114.8889	309.3546	106.4165	145.1851	138.5865	205.9828	205.8134	213.6779
Sum=	212.9382	573.9007	194.516	266.5119	257.3874	378.9305	381.0211	394.1706
Throughput (MB / Sec)		0.371037	1.094708	0.798982	0.827306	0.561945	0.558862	0.540218

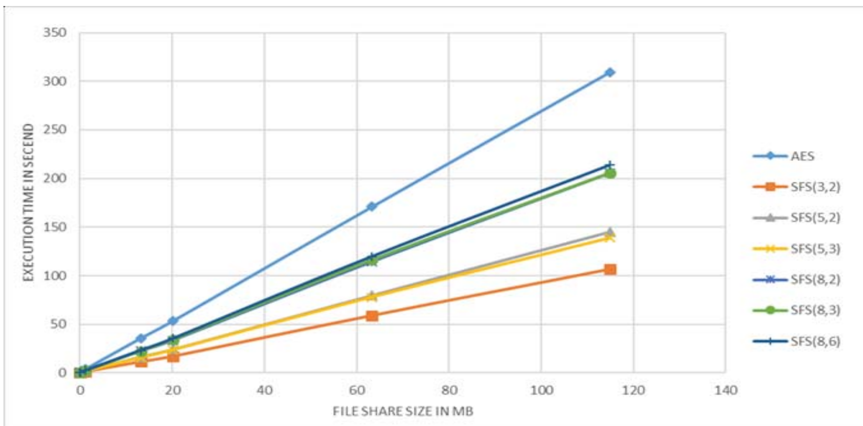

Figure 4.9: Performance Comparison of the Sequential Execution Timevof Creating the Shares and Encryption using AES of most Popular File Formats.

other schemes. Another point can be noticed here is that the difference of the throughput among the SFS schemes is relatively small in general. Summary of execution time throughput of SFS and AES schemes is shown in Fig. 4.10. Considering the throughput of all files, we can see that AES has a lower throughput than SFS schemes. Therefore, consumes less power than the other schemes. Moreover, we observed that SFS schemes with (n=8, t=6), (n=8, t=3), (n=8, t=2), (n=5, t=3) and (n=5, t=2) have quite the same throughput. However, SFS with (n=5, t=3) and (n=5, t=2) are slightly faster.

Table 4.2 and Fig. 4.11 show a parallel implementation of create a secret and

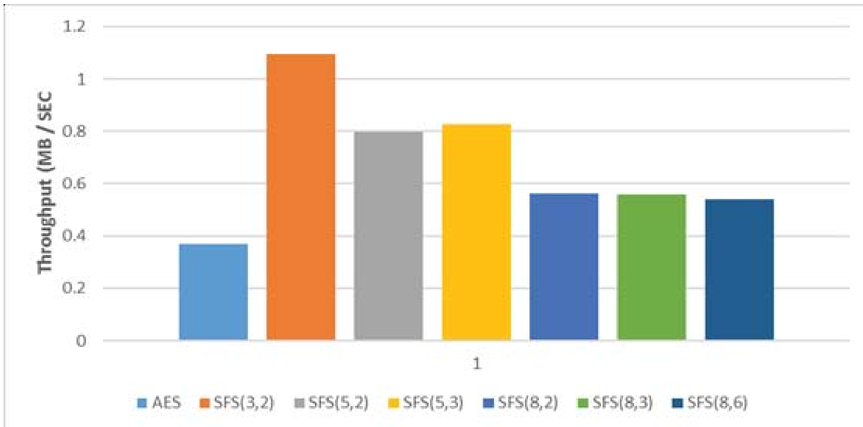

Figure 4.10: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of most Popular File Formats.

encrypt a different file types. The line graph illustrates that SFS schemes when (n=8, t=6), (n=8, t=3), and (n=8, t=2) almost have quite the same execution time. However, (n=8, t=3) is slightly slower than (n=8, t=2) whereas (n=8, t=6) is slightly faster than others . Moreover, both SFS schemes when (n=5, t=3) and (n=5, t=2) almost require the same execution time for all file sizes, while SFS scheme when (n=3, t=2) is better than other SFS schemes. Generally, it is noticed that SFS scheme consumes less time than AES. In addition, increasing the number of the shares effects on the execution time more than the number of thresholds.

Table 4.2: Parallel implementation of Creating the Shares and Encryption using AES of most Popular File Formats.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000806	0.264419	0.007069	0.005957	0.006572	0.006545	0.011622	0.006735
.tgz	0.011756	0.308018	0.02259	0.024005	0.025113	0.034285	0.037243	0.039994
.png	0.129359	0.614951	0.145908	0.222404	0.232572	0.334189	0.355175	0.392909
.exe	1.223342	2.812096	0.787751	1.240126	1.242515	1.976161	2.236696	2.374491
.pdf	13.15998	21.14621	5.530591	8.309103	9.098822	15.95286	15.28679	16.45471
.doc	20.19169	33.84927	6.556674	10.20914	10.93939	16.46419	18.14756	21.22733
.mp3	63.33237	104.7168	28.78238	49.01058	50.38144	75.11581	77.08215	83.43849
.jpg	114.8889	188.3953	56.02712	88.0518	89.62887	127.8719	137.6849	144.2431
sum=	212.9382	352.1071	97.86008	157.0731	161.5553	237.756	250.8422	268.1777
Throughput (MB / Sec)		0.604754	2.175946	1.355663	1.318052	0.895617	0.848893	0.794019

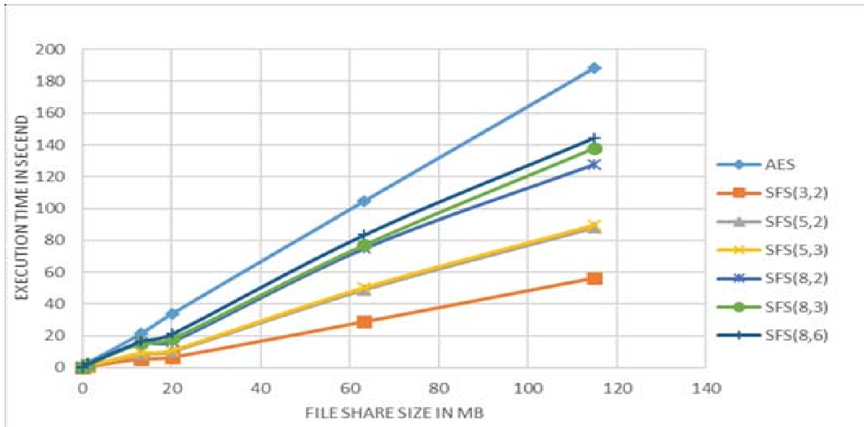

Figure 4.11: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of most Popular File Formats.

We can see that SFS scheme when (n=3, t=2) is faster in performance than other schemes. Another point can be noticed here is that the difference of the throughput among the SFS schemes is relatively small in general. Summary of execution time throughput of SFS and AES schemes is shown in Fig. 4.12. Considering the throughput of different files type, we can see that AES has a lower throughput than SFS schemes. therefore, consumes less power than the other schemes. Moreover, we observed that SFS schemes with (n=8, t=6), (n=8, t=3), (n=8, t=2), (n=5, t=3) and (n=5, t=2) have quite the same throughput. However, SFS with (n=5, t=3) and (n=5, t=2) are slightly faster.

Fig. 4.13 illustrates the difference in execution time for sequential and parallel implementation for create the secret and encryption different file type. We can note that the performance is improved in the parallel implementation. Also, it can be seen that the performance is not fixed or constant for all schemes. Here, we can observe that the performance of parallel implementation for small size file is less and it is increased as the file size is increased. But it will increase till a

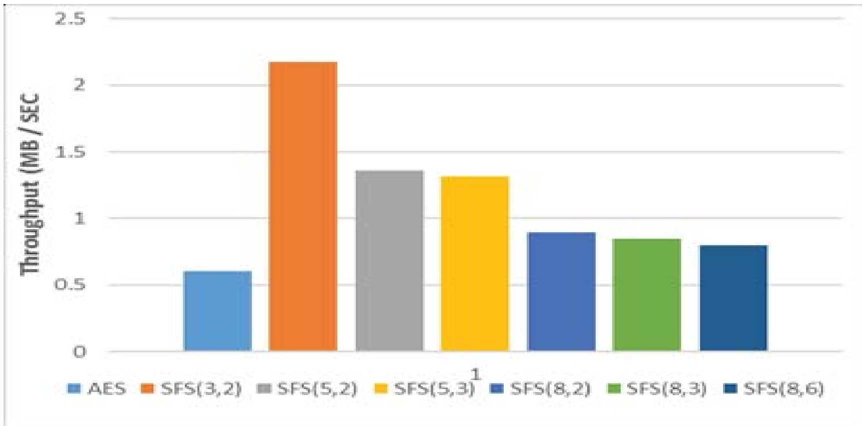

Figure 4.12: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of most Popular File Formats.

particular value and after that it will be a constant value. Generally , we observed more improvement for AES in parallel than in sequential. However, SFS schemes is still better. Moreover, we can see that there is decreasing in the speed up due to the devices features . Therefore, we run our experiments again in a different device with the following features: " Intel Core(TM) i7 -6700HQ cpu @ 2.59GHz 16GB Memory - 1TB Hard Drive + 128GB "and we get better results as shown in Fig. 4.14.

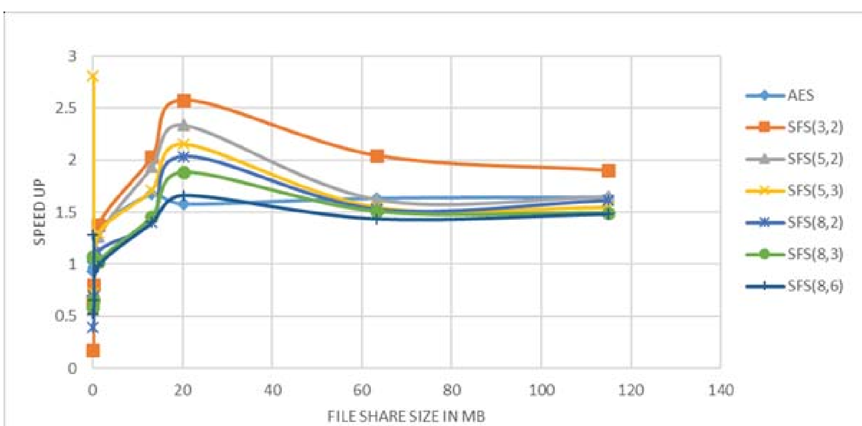

Figure 4.13: Speed Up of Creating the Shares and Encryption using AES of most Popular File Formats .

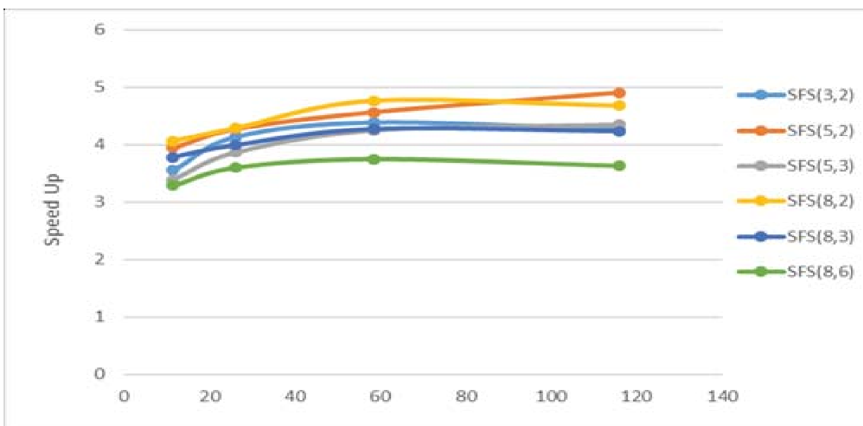

Figure 4.14: Speed Up of Creating the Shares and Encryption using AES of most Popular File Formats using pc with better features .

PDF file formats

In this section, we explain PDF format in details as an example for create the shares and encrypt a different file types. The result of creating the shares and encryption for the other files is given in the Appendix A

The Table 4.3 and Fig. 4.15 show sequential implementation of creating the shares and encryption using AES of PDF file type of different sizes. The line graph illustrates that SFS schemes when (n=8, t=6), (n=8, t=3), and (n=8, t=2) almost have quite the same execution time. However, (n=8, t=6) is slightly slower than others. Moreover, both SFS schemes when (n=5, t=3) and (n=5, t=2) almost require the same execution time for all file sizes, while SFS scheme when (n=3, t=2) is better than other SFS schemes. Generally, it is noticed that SFS scheme consumes less time than AES. In addition, increasing the number of the shares effects the execution time more than the number of thresholds.

Table 4.3: Sequential implementation of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.pdf	0.000834	0.305094	0.001005	0.001188	0.001429	0.002764	0.001963	0.150591
.pdf	0.006684	0.266376	0.005391	0.006383	0.008004	0.010362	0.01138	0.028576
.pdf	0.106799	0.579613	0.089085	0.115132	0.136405	0.187575	0.172644	0.216746
.PDF	0.830359	2.272317	0.546168	0.897309	0.905497	1.219139	1.288421	1.416338
.pdf	1.36139	3.562602	0.920778	1.422522	1.449138	2.113922	2.139843	2.179916
.pdf	13.16027	32.53742	8.792271	13.60339	14.19589	20.48834	20.18966	21.4545
.PDF	26.9594	68.96663	17.97656	28.57989	29.51048	43.40014	42.77134	44.89893
.pdf	60.9044	156.2485	42.7172	67.51218	68.36503	98.6137	98.96433	105.1793
.pdf	128.7644	330.8267	97.68768	152.1946	149.8191	219.9563	218.184	234.9162
sum=	232.0945	595.5653	168.7361	264.3326	264.3909	385.9922	383.7236	410.4411
Throughput (MB / Sec)		0.389705	1.375488	0.87804	0.877846	0.601293	0.604848	0.565476

We can see that SFS scheme when (n=3, t=2) is faster in performance than other schemes. Another point can be noticed here is that the difference of the throughput among the SFS schemes is relatively small in general. Summary of

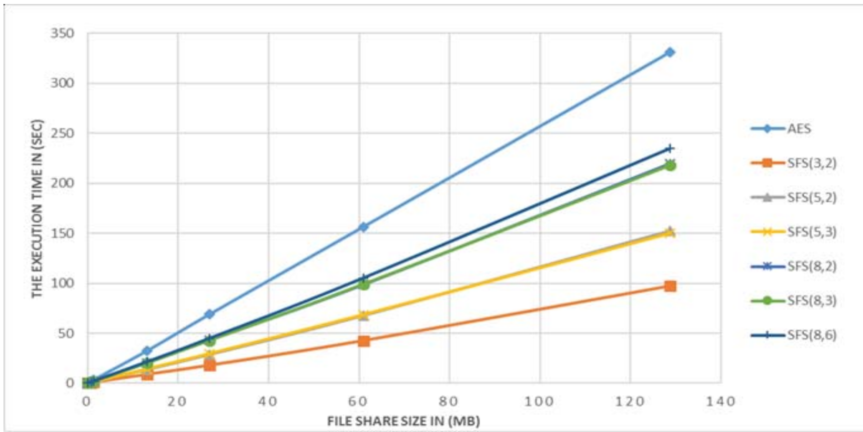

Figure 4.15: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes.

execution time throughput of SFS and AES schemes is shown in Fig. 4.16. Considering the throughput of all files, we can see that AES has a lower throughput than SFS schemes. therefore, consumes less power than the other schemes. Moreover, we observed that SFS schemes with (n=8, t=6), (n=8, t=3), (n=8, t=2), (n=5, t=3) and (n=5, t=2) have quite the same throughput. However, SFS with (n=5, t=3) and (n=5, t=2) are slightly faster.

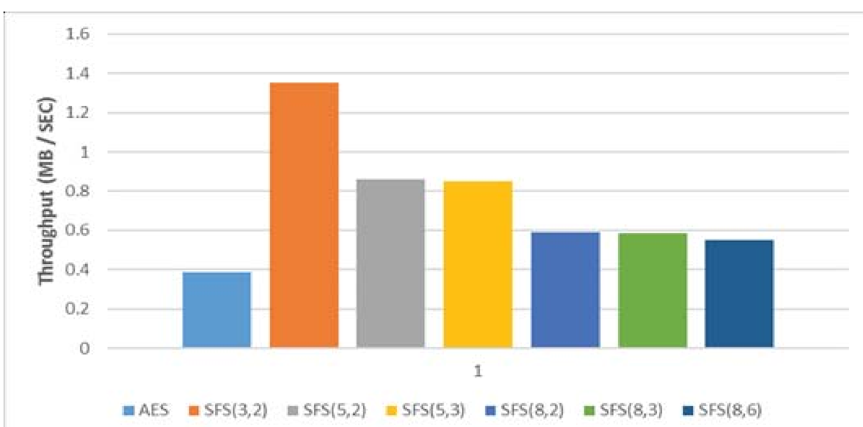

Figure 4.16: Throughput of the Sequential Execution Time of of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes.

The Table 4.4 and Fig. 4.17 show a performance comparison of the parallel execution time of create a secret and encrypt a PDF file type. The line graph illustrates that SFS schemes when (n=8, t=6), (n=8, t=3), and (n=8, t=2) almost have quite the same execution time. However, (n=8, t=6) is slightly slower than others. Moreover, SFS schemes when (n=5, t=3) is slightly slower than (n=5, t=2), while SFS scheme when (n=3, t=2) is better than other SFS schemes. Generally, it is noticed that SFS scheme consumes less time than AES. We also observed that SFS schemes are slightly better in Parallel than in sequential. In addition, increasing the number of the shares effects on the execution time more than the number of thresholds.

Table 4.4: Parallel implementation of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.pdf	0.000834	0.208378	0.00635	0.004335	0.003938	0.004655	0.00655	0.005321
.pdf	0.006684	0.215344	0.017837	0.013165	0.016623	0.019169	0.023733	0.022129
.pdf	0.106799	0.463873	0.10624	0.188977	0.251047	0.345154	0.296366	0.319448
.PDF	0.830359	1.222278	0.468225	0.795654	0.886653	1.22869	1.136174	1.26054
.pdf	1.36139	2.005185	0.741972	1.219852	1.356754	1.943219	1.802287	2.000625
.pdf	13.16027	18.59345	6.617767	11.27738	11.74478	17.90399	16.68294	18.8062
.PDF	26.9594	32.4183	13.50459	21.97078	22.15551	34.82851	33.95971	37.86716
.pdf	60.9044	80.24629	26.31003	46.18155	47.84104	66.146	76.59416	86.01726
.pdf	128.7644	166.119	53.01418	90.15962	99.86032	136.7923	169.8715	186.7328
sum=	232.0945	301.4921	100.7872	171.8113	184.1167	259.2117	300.3735	333.0315
Throughput (MB / SEC)		0.76982	2.302818	1.350869	1.260584	0.895386	0.772686	0.696915

We can see that SFS scheme when (n=3, t=2) is faster in performance than other schemes. Another point can be noticed here is that the difference of the throughput among the SFS schemes is relatively small in general. Summary of execution time throughput of SFS and AES schemes is shown in Fig. 4.18. Considering the throughput of all files, we can see that AES has a lower throughput than SFS schemes. Therefore, consumes less power than the other schemes. Moreover, we observed that SFS schemes with (n=8, t=6), (n=8, t=3), (n=8, t=2)

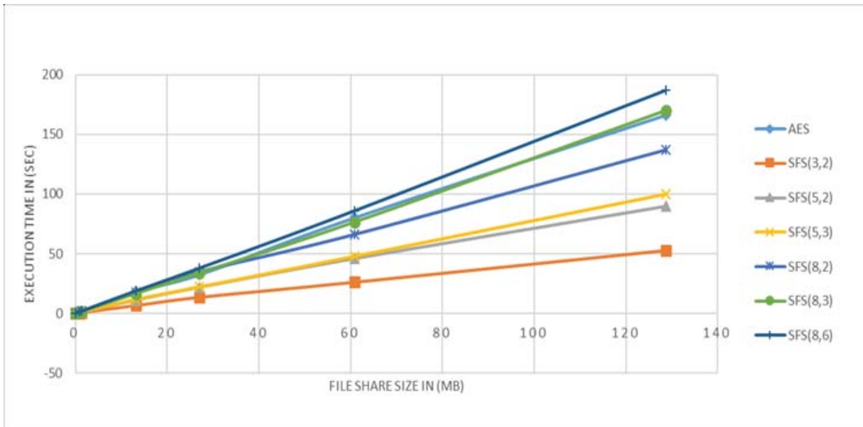

Figure 4.17: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes.

have quite the same throughput. However, SFS scheme with (n=8, t=6) is slightly slower. Whereas SFS schemes with (n=5, t=3) and (n=5, t=2) have also quite the same throughput. However, SFS with (n=5, t=2) is slightly faster.

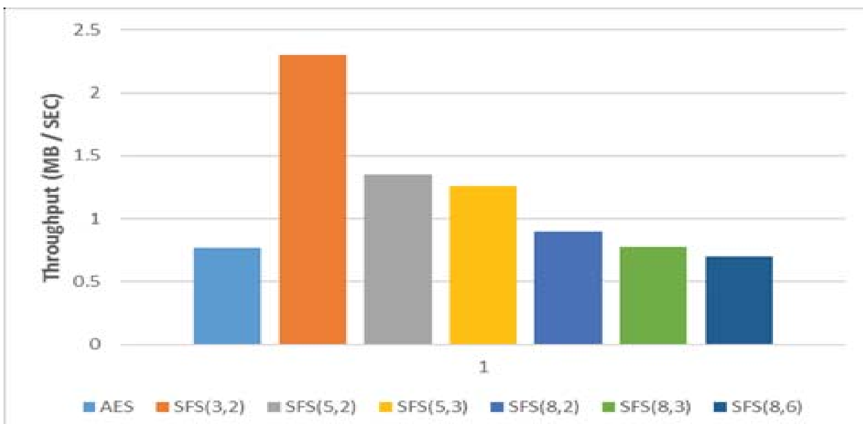

Figure 4.18: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of PDF File Type of Different Sizes.

4.3.2 Reconstruct The Secret/Decryption

The most popular file formats

The Table 4.5 and Fig. 4.19 show a performance comparison of the sequential execution time of reconstruct and decryption using AES of different file types. The line graph illustrates that the execution time of SFS scheme is improved dramatically and therefore consumes very less time than AES. Moreover, we observed that SFS schemes when t=2 almost require the same execution time for all file sizes, and both SFS schemes t=3 almost require the same execution time for all file sizes. However, they are slightly slower than SFS with t=2. While SFS scheme when t=6 is slower than other SFS schemes, but still better than AES. In addition, we can see that increasing the number of the thresholds effects on the execution time.

Table 4.5: Sequential implementation of Reconstruct the Secret and Decryption using AES of most Popular File Formats.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000806	0.259497	0.000392	0.000317	0.000924	0.00031	0.000357	0.000596
.tgz	0.011756	0.261842	0.004937	0.005026	0.005581	0.004869	0.006141	0.011531
.png	0.129359	0.523923	0.045061	0.043325	0.052838	0.044957	0.052975	0.080394
.exe	1.223342	3.445708	0.396014	0.390648	0.472805	0.409168	0.47547	0.745363
.pdf	13.15998	36.49325	4.352675	4.292135	5.181315	4.414166	5.092969	8.028322
.doc	20.19169	56.82038	6.922966	6.901265	8.0602	6.931192	8.060043	12.44748
.mp3	63.33237	175.4779	21.34528	21.43678	25.56629	21.45706	25.65903	37.29487
.jpg	114.8889	313.7745	37.30818	37.05509	44.98251	36.98753	44.34046	64.61524
sum=	212.9382	587.057	70.3755	70.12459	84.32247	70.24925	83.68745	123.2238
Throughput (KB / ms)		0.362722	3.025744	3.03657	2.525285	3.031182	2.544447	1.728061

We can see that SFS scheme when t=2 is faster in performance than other SFS schemes. However, it can be noticed that the difference of the throughput among different files reconstructed using the SFS schemes is relatively small. Summary of throughput of SFS and AES schemes is shown in Fig. 4.20. Considering the throughput of all files, we can see that AES has a lower throughput than SFS schemes. Therefore, consumes less power . Moreover, we observed that SFS schemes when t=2 almost have quite the same throughput, and both SFS schemes t=3 almost have also quite the same throughput. However, they are slightly slower. While SFS scheme when t=6 is slower than other SFS schemes, but still better than AES.

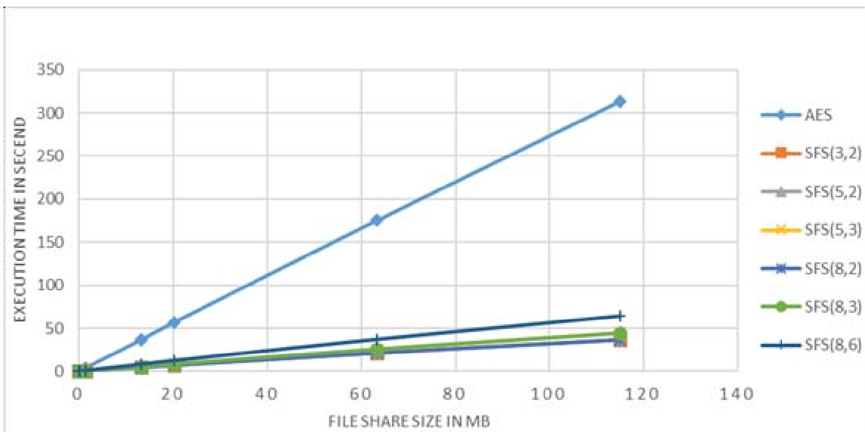

Figure 4.19: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats.

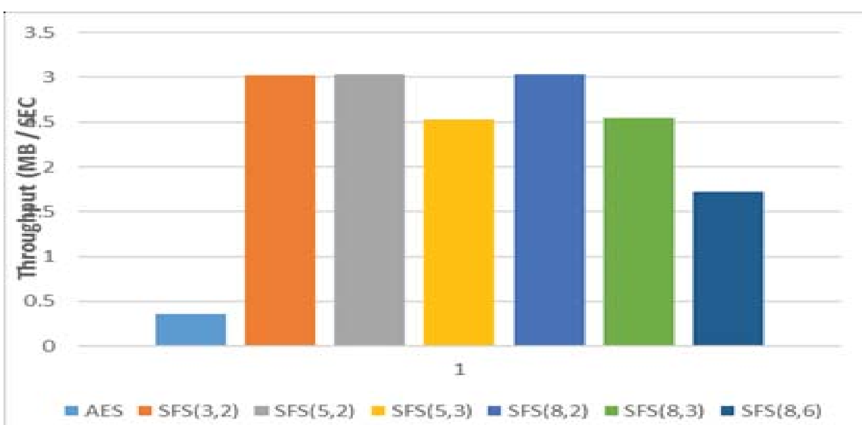

Figure 4.20: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats.

The Table 4.6 and Fig. 4.21 show a performance comparison of the parallel execution time of reconstruct the secret and decryption using AES of most popular file formats. The line graph illustrates that the execution time of SFS scheme is improved dramatically and therefore consumes very less time than AES. Moreover, we observed that SFS schemes when t=2 almost require the same execution time for all file sizes, and both SFS schemes t=3 almost require the same execution time for all file sizes. However, they are slightly slower. While SFS scheme when t=6 is slower than other SFS schemes, but still better than AES. In addition, we can see that increasing the number of the thresholds effects on the execution time.

Table 4.6: Parallel implementation of Reconstruct the Secret and Decryption using AES of most Popular File Formats.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000806	0.300373	0.002389	0.002899	0.002588	0.002789	0.004162	0.005223
.tgz	0.011758	0.308509	0.015617	0.012056	0.021017	0.018118	0.016687	0.042235
.png	0.129373	0.434473	0.06281	0.060916	0.099873	0.063399	0.101994	0.232432
.exe	1.223363	2.292842	0.364921	0.331134	0.610034	0.358143	0.528697	1.428648
.pdf	13.15995	22.87396	4.199058	3.860987	6.895496	4.017666	6.250292	16.63279
.doc	20.19173	35.06085	7.509392	8.692852	10.42986	8.561988	10.39432	22.67067
.mp3	63.33208	107.9735	27.31328	26.96642	33.29497	27.14255	32.41477	52.62848
.jpg	114.8893	190.5882	40.0472	40.96611	51.44264	39.9216	52.36443	86.90132
sum=	212.9384	359.8327	79.51467	80.89338	102.7965	80.08625	102.0754	180.5418
Throughput (KB / ms)		0.591771	2.677976	2.632334	2.071456	2.658863	2.08609	1.179441

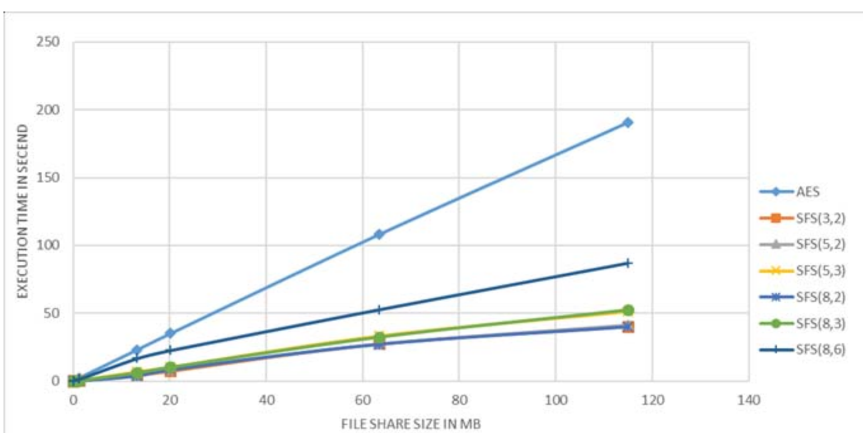

Figure 4.21: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats.

In Fig. 4.20 Summary of execution time throughput of SFS and AES schemes.

We notice that SFS scheme when t=2 is faster in performance than other SFS schemes. However, it can be noticed that the difference of the throughput among the SFS schemes is relatively small in general. Considering the throughput of all files, we can see that AES has a lower throughput than SFS schemes. Therefore, consumes less power . Moreover, we observed that SFS schemes when t=2 almost have quite the same throughput, and both SFS schemes t=3 almost have also quite the same throughput. However, they are slightly slower. While SFS scheme when t=6 is slower than other SFS schemes, but still better than AES.

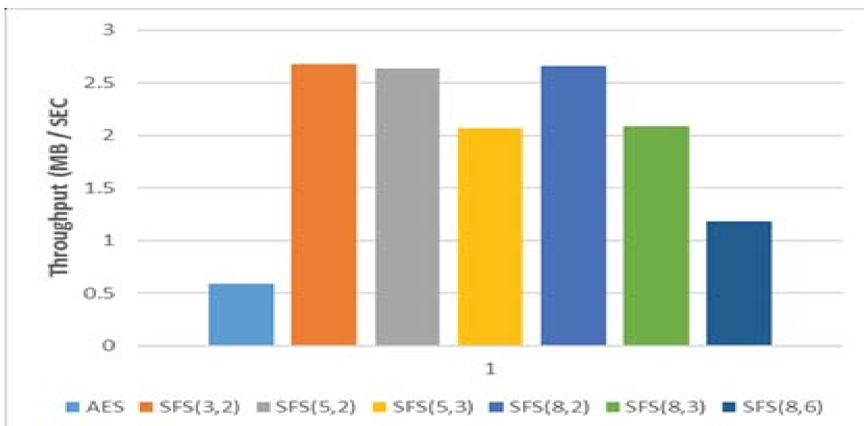

Figure 4.22: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of most Popular File Formats.

Fig.4.23 illustrates the difference in execution time for sequential and parallel implementation for reconstruct and decryption different file type. We can note that the performance is improved in the parallel implementation. In this also it can be seen that the performance is not fixed or constant for all schemes. Here we can see that the performance of parallel implementation for small size file is less and it will increase as the file size increases till a particular value after which it will be a constant value. Generally , we observed more improvement for AES in parallel than in sequential. However, SFS schemes is still better.

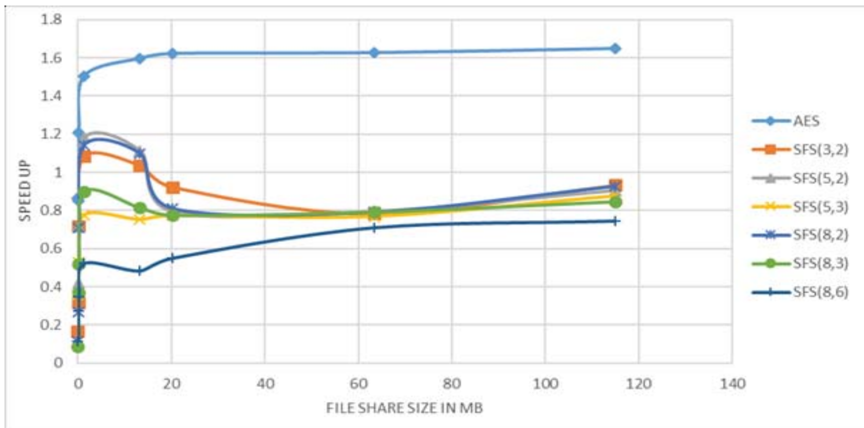

Figure 4.23: Speed Up of Reconstruct the Secret and Decryption using AES of most Popular File Formats.

PDF file formats

In this section, we explain PDF format in details as an example for reconstruct the secret and decryption using AES of PDF File type of different sizes. The result of reconstructing the secret for the other file types is shown in Appendix B.

The Table 4.7 and Fig. 4.24 show performance comparison of the sequential execution time of reconstructing the secret and decryption using AES of PDF file type of different sizes. The line graph illustrates that the execution time of SFS scheme is improved dramatically and therefore consumes very less time than AES. Moreover, we observed that SFS schemes when t=2 almost require almost the same execution time for all file sizes, and both SFS schemes t=3 almost require the same execution time for all file sizes. However, they are slightly slower. While SFS scheme when t=6 is slower than other SFS schemes, but still better than AES. In addition, we can see that increasing the value of the thresholds effects the execution time.

Table 4.7: Sequential implementation of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.pdf	0.000834	0.237528	0.000393	0.000274	0.000481	0.000384	0.000467	0.000703
.pdf	0.006684	0.260077	0.00368	0.002372	0.004506	0.00388	0.004204	0.006337
.pdf	0.106799	0.562031	0.047703	0.036748	0.066607	0.062995	0.065124	0.075652
.PDF	0.830359	2.424523	0.270143	0.280818	0.331581	0.274417	0.338507	0.515038
.pdf	1.36139	4.053242	0.52511	0.502577	0.678461	0.474573	0.613416	0.848786
.pdf	13.16027	37.10253	4.910213	4.567748	5.639248	4.600203	5.646171	8.219183
.PDF	26.9594	74.1524	9.488832	9.610638	11.00949	9.353508	11.63609	16.97265
.pdf	60.9044	158.8993	21.77991	21.32283	25.32741	21.77832	25.80141	38.63707
.pdf	128.7644	333.0299	42.18129	41.85679	49.24716	42.03524	49.97805	72.78362
sum=	232.0945	610.7215	79.20727	78.1808	92.30494	78.58352	94.08344	138.059
Throughput (MB / SEC		0.380033	2.930217	2.96869	2.514432	2.953475	2.466901	1.681125

We can notice that SFS scheme when t=2 is faster than other SFS schemes. However, it can be noticed that the difference of the throughput among different

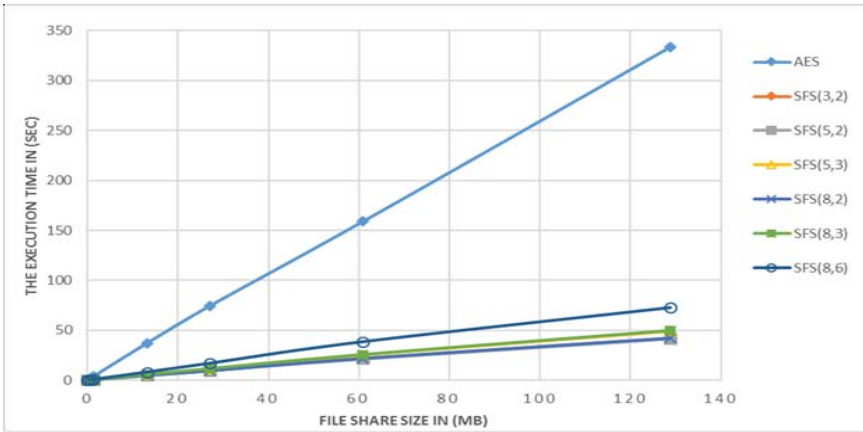

Figure 4.24: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes.

experiments is relatively small in general. Summary of execution time throughput of SFS and AES scheme is shown in Fig. 4.25. Considering the throughput of all files, we can see that AES has a lower throughput than SFS schemes. Therefore, SFS consumes less power . Moreover, we observed that SFS schemes when t=2 almost have quite the same throughput, and both SFS schemes t=3 almost have also quite the same throughput. However, they are slightly slower. While SFS scheme when t=6 is slower than other SFS schemes, but still better than AES.

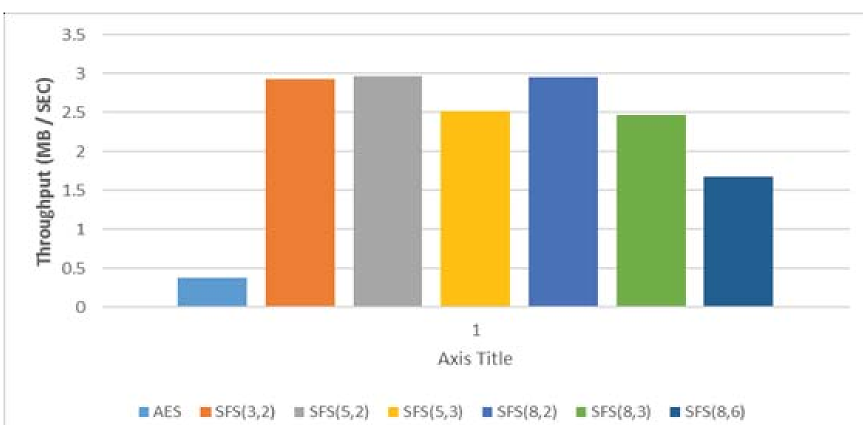

Figure 4.25: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes.

The Table 4.8 and Fig. 4.26 show a performance comparison of the parallel execution time of reconstruct and decryption PDF file type. The line graph illustrates that the execution time of SFS scheme is improved dramatically and therefore consumes very less time than AES. Moreover, we observed that SFS schemes when t=2 almost require the same execution time for all file sizes even if the number of shares is different , and both SFS schemes t=3 almost require the same execution time for all file sizes even if the number of shares is different. However, they are slightly slower. When SFS uses t=6 , it takes long time more than other threshold values , but still better than AES. In addition, we can see that increasing the number of the thresholds effects on the execution time.

Table 4.8: Parallel implementation of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.pdf	0.000834	0.208378	0.00635	0.004335	0.003938	0.004655	0.00655	0.005321
.pdf	0.006684	0.215344	0.017837	0.013165	0.016623	0.019169	0.023733	0.022129
.pdf	0.106799	0.463873	0.10624	0.188977	0.251047	0.345154	0.296366	0.319448
.PDF	0.830359	1.222278	0.468225	0.795654	0.886653	1.22869	1.136174	1.26054
.pdf	1.36139	2.005185	0.741972	1.219852	1.356754	1.943219	1.802287	2.000625
.pdf	13.16027	18.59345	6.617767	11.27738	11.74478	17.90399	16.68294	18.8062
.PDF	26.9594	32.4183	13.50459	21.97078	22.15551	34.82851	33.95971	37.86716
.pdf	60.9044	80.24629	26.31003	46.18155	47.84104	66.146	76.59416	86.01726
.pdf	128.7644	166.119	53.01418	90.15962	99.86032	136.7923	169.8715	186.7328
sum=	232.0945	301.4921	100.7872	171.8113	184.1167	259.2117	300.3735	333.0315
Throughput (MB / SEC)		0.76982	2.302818	1.350869	1.260584	0.895386	0.772686	0.696915

We notice that SFS scheme when t=2 is faster in performance than other SFS with different values of t. However, it can be noticed that the difference of the throughput among the SFS schemes is relatively small in general. Summary of execution time throughput of SFS and AES schemes is shown in Fig. 4.27. Considering the throughput of all files, we can see that AES has a lower throughput than SFS schemes. therefore, consumes less power than the other schemes.

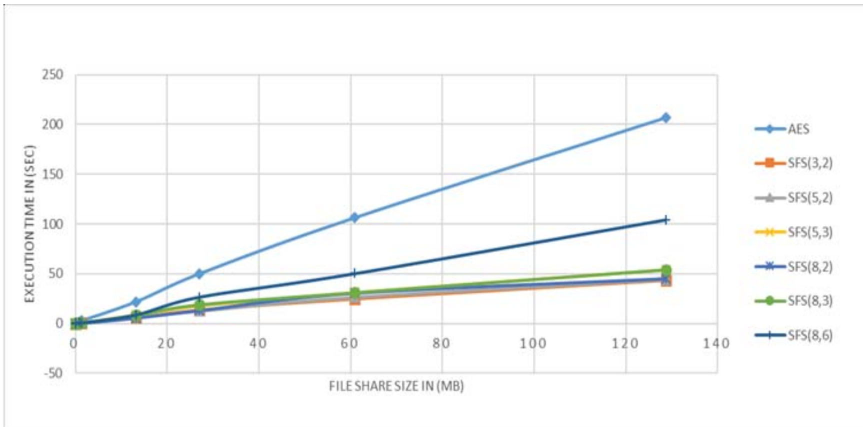

Figure 4.26: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes.

Moreover, we observed that SFS schemes when t=2 almost have quite the same throughput, and both SFS schemes t=3 almost have also quite the same throughput. However, they are slightly slower. While SFS scheme when t=6 is slower than other SFS with different values of t, but still better than AES.

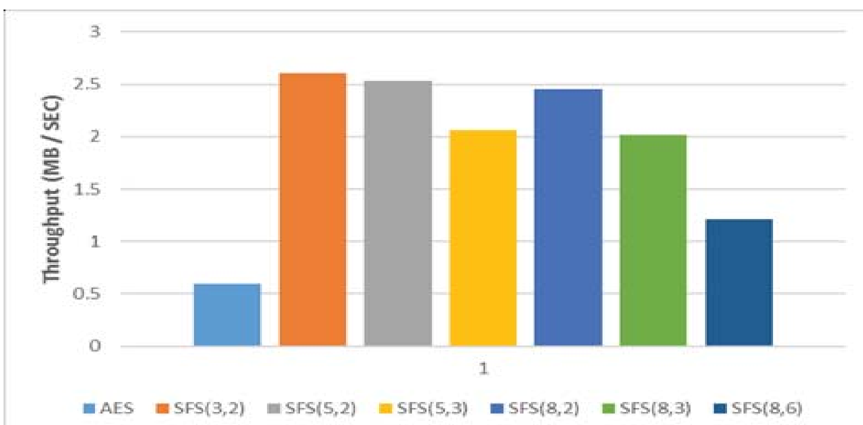

Figure 4.27: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of PDF File Type of Different Sizes.

4.3.3 Compression and Decompression

Converting file to Base64 increases the file size. We select the GZIP algorithm for the compression and decompression of the converted file. GZIP is very fast and has small memory footprint according to [11, 47, 48] . In this section, we show the execution time for compression and decompression of the different file type before/after create the share and compare the file size with the original file. Reducing the file size will reduce the time needed to create the shares and reconstruct the secret . Some fluctuations happened in some graph because of the file size and type.

Fig. 4.28, Fig. 4.29 , and Fig. 4.30 shown performance of compression, decompression , and file share size before compression and after compression respectively for PDF file.

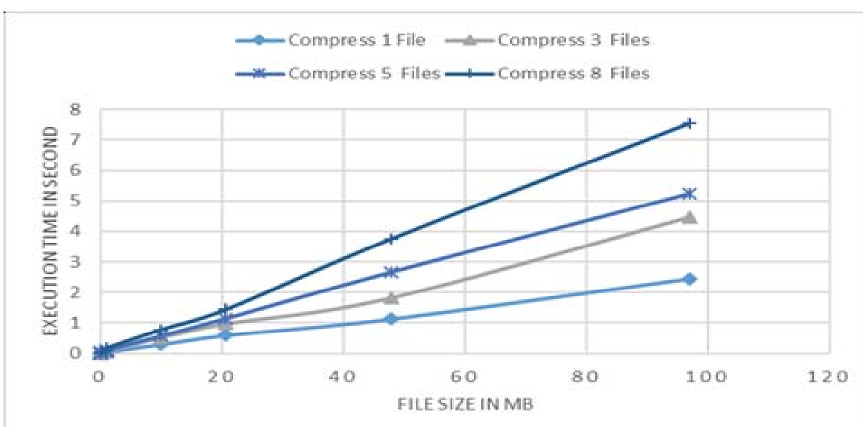

Figure 4.28: Performance of Compression PDF File type.

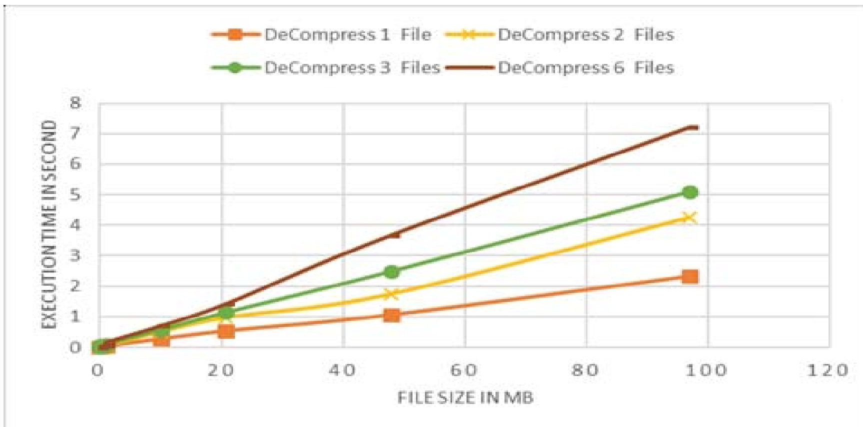

Figure 4.29: Performance of Decompression PDF File type.

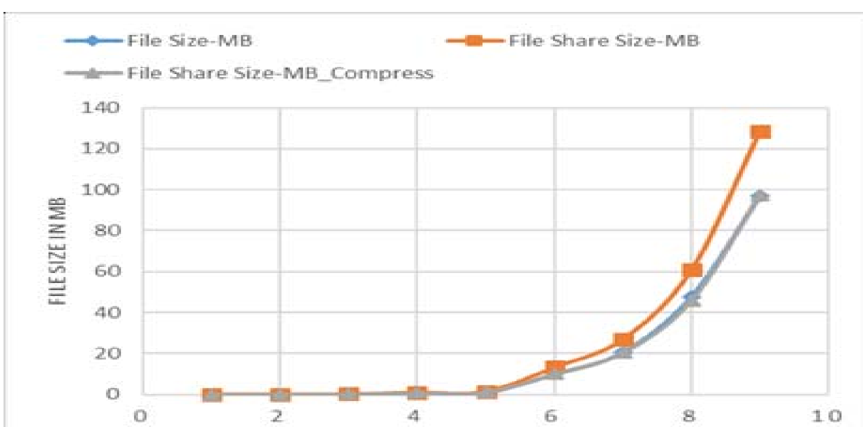

Figure 4.30: File Share Size before Compression and after Compression of PDF File type.

Fig. 4.31, Fig. 4.32 , and Fig. 4.33 shown performance of compression, decompression , and file share size before compression and after compression respectively for Audio file.

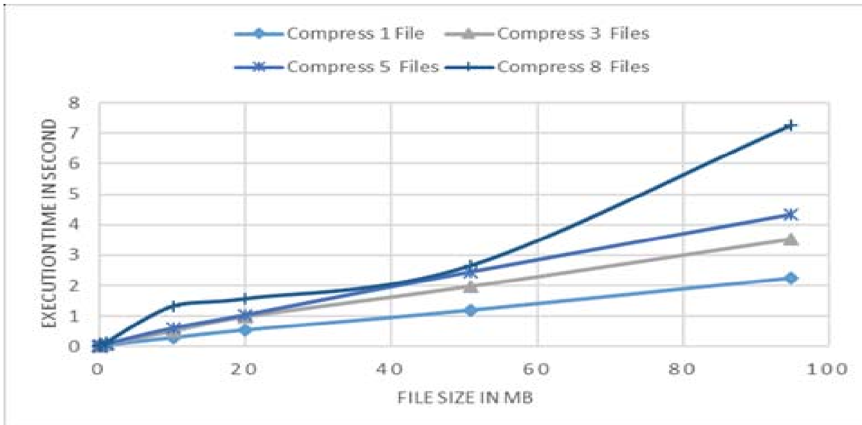

Figure 4.31: Performance of Compression Audio File type.

Figure 4.32: Performance of Decompression Audio File type.

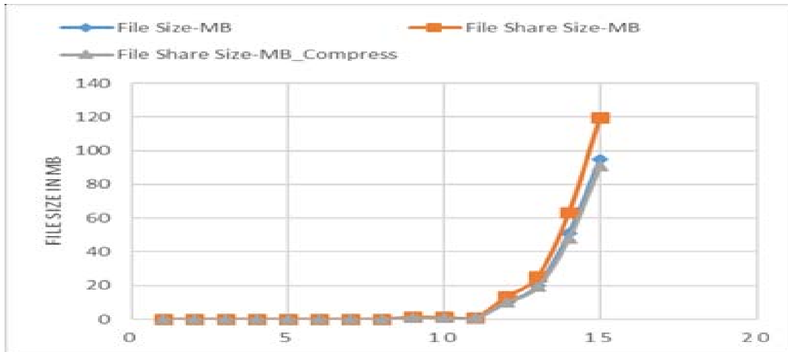

Figure 4.33: File Share Size before Compression and after Compression of Audio File type.

Fig. 4.34, Fig. 4.35 , and Fig. 4.36 shown performance of compression, decompression , and file share size before compression and after compression respectively for Binary file.

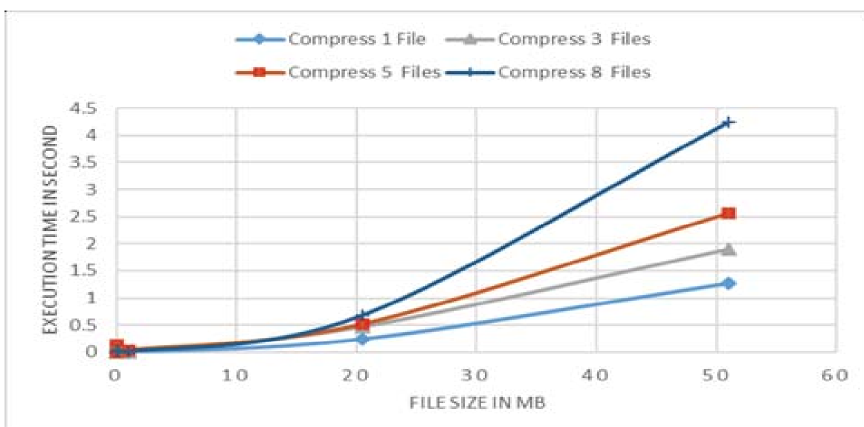

Figure 4.34: Performance of Compression Binary File Type.

Figure 4.35: Performance of Decompression Binary File Type.

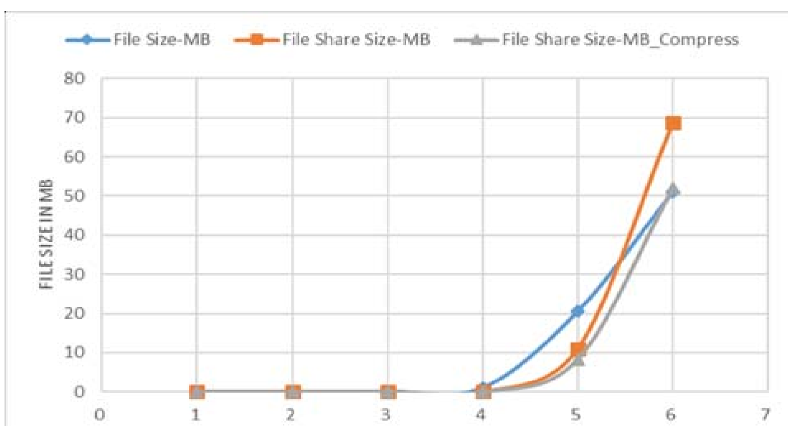

Figure 4.36: File Share Size before Compression and after Compression of Binary File Type.

Fig. 4.37, Fig. 4.38 , and Fig. 4.39 shown performance of compression, decompression , and file share size before compression and after compression respectively for Document file.

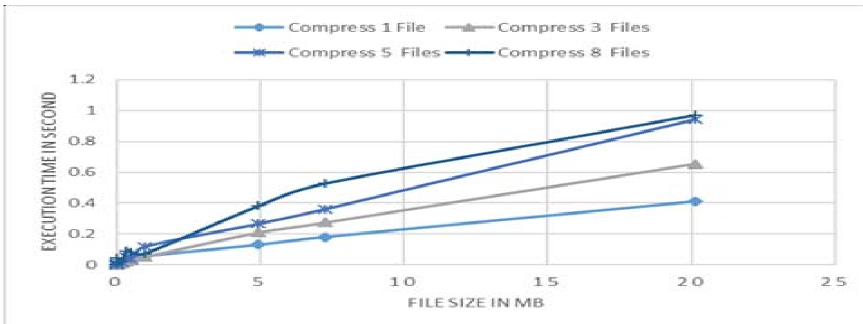

Figure 4.37: Performance of Compression Document File Type.

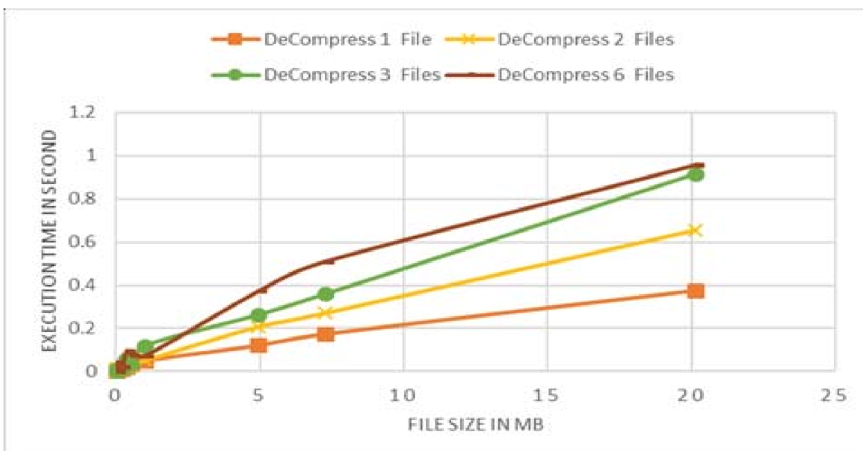

Figure 4.38: Performance of Decompression Document File Type.

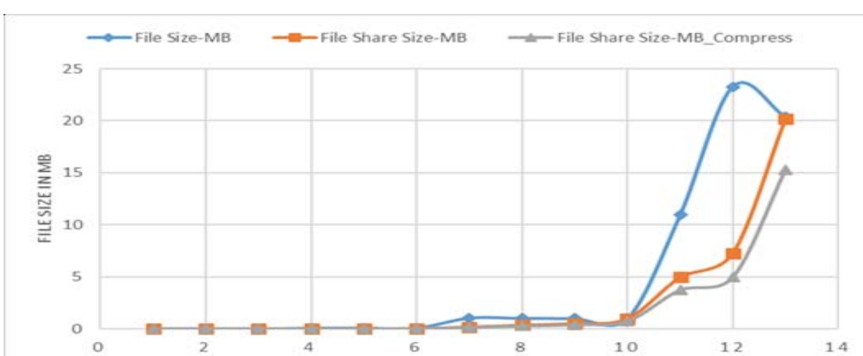

Figure 4.39: File Share Size before Compression and after Compression of Document File Type.

Fig. 4.40, Fig. 4.41 , and Fig. 4.42 shown performance of compression, decompression , and file share size before compression and after compression respectively for Executable file.

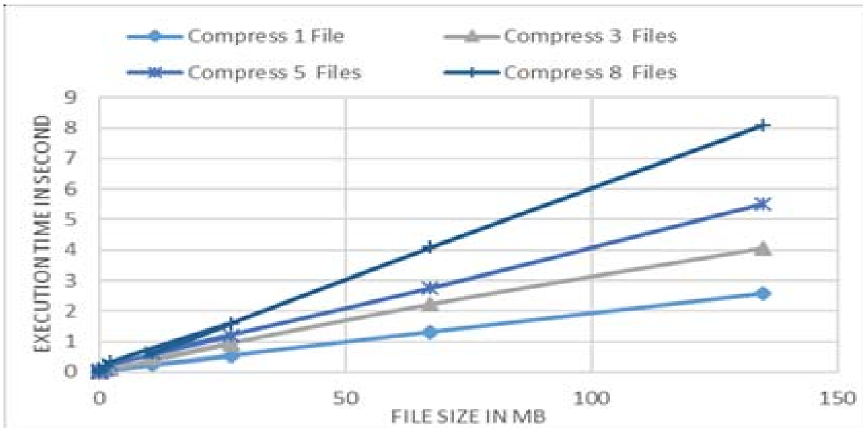

Figure 4.40: Performance of Compression Executable File Type.

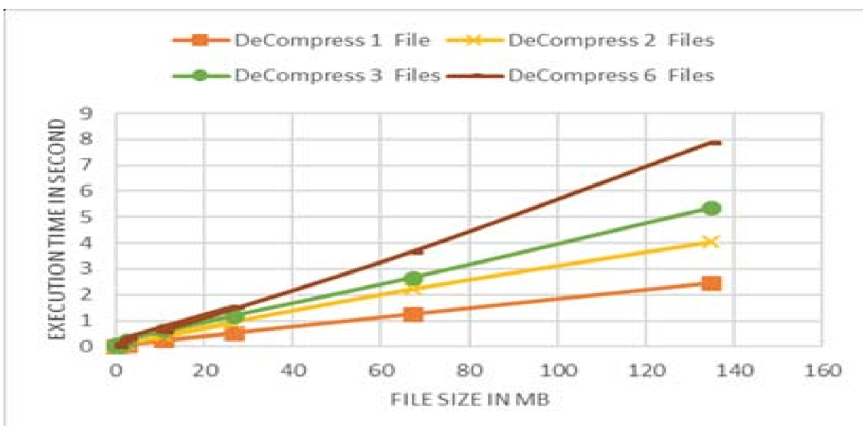

Figure 4.41: Performance of Decompression Executable File Type.

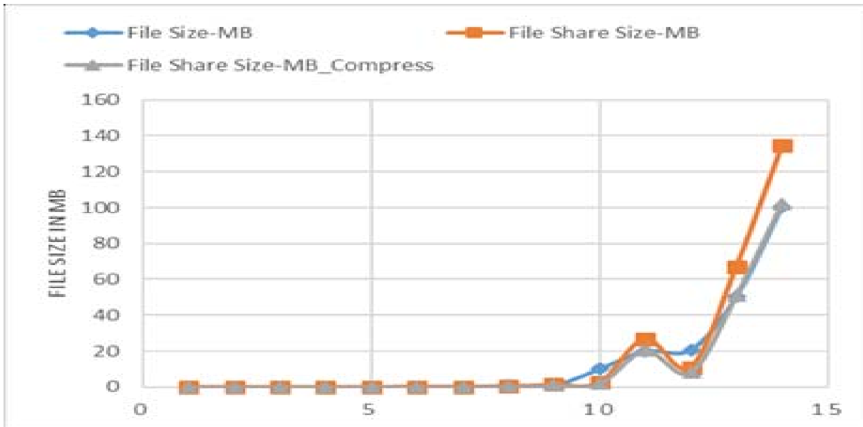

Figure 4.42: File Share Size before Compression and after Compression of Executable File Type.

Fig. 4.43, Fig. 4.44 , and Fig. 4.45 shown performance of compression, decompression , and file share size before compression and after compression respectively for Image file.

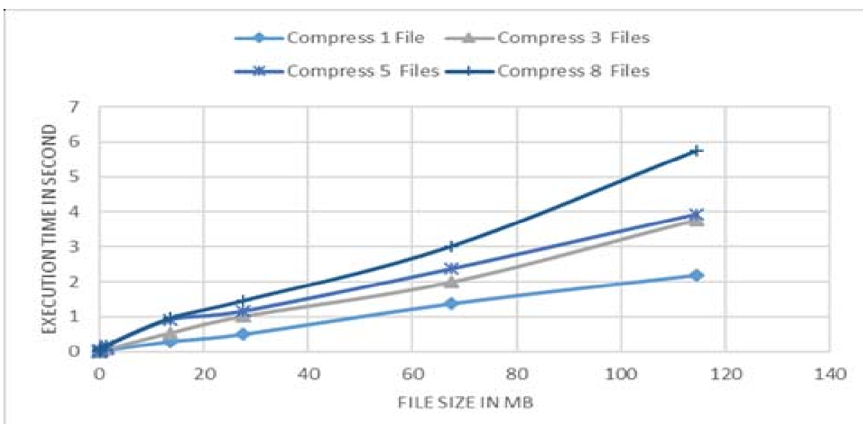

Figure 4.43: Performance of Compression Image File Type.

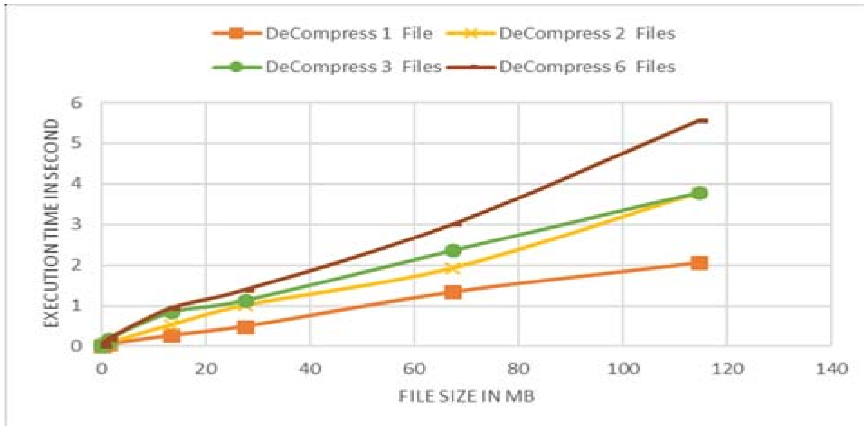

Figure 4.44: Performance of Decompression Image File Type.

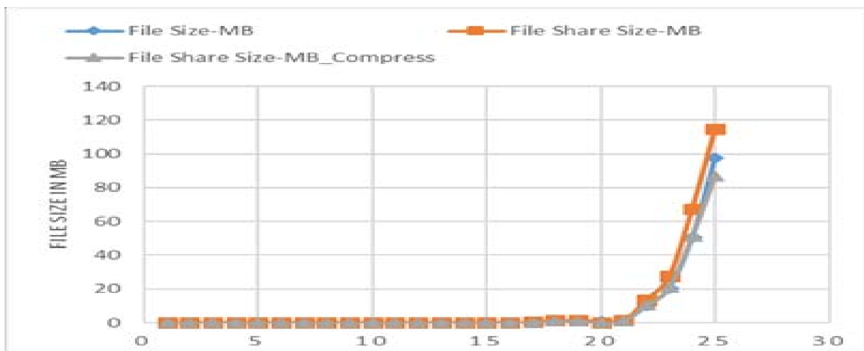

Figure 4.45: File Share Size before Compression and after Compression of Image File Type.

Fig. 4.46, Fig. 4.47 , and Fig. 4.48 shown performance of compression, decompression , and file share size before compression and after compression respectively for Text file.

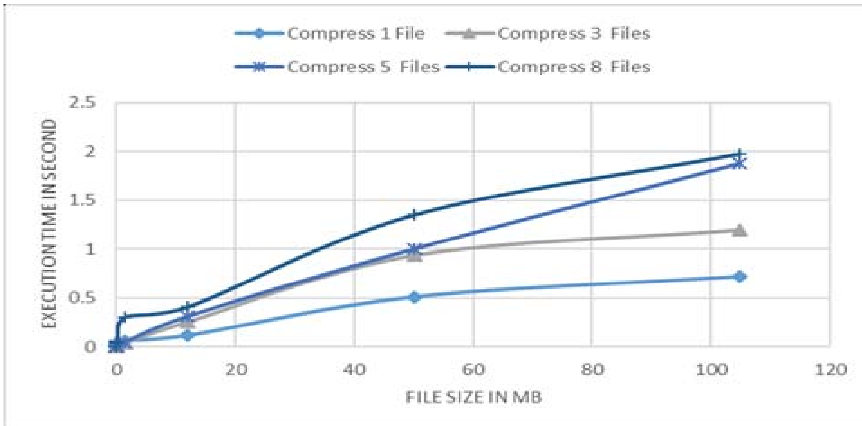

Figure 4.46: Performance of Compression Text File Type.

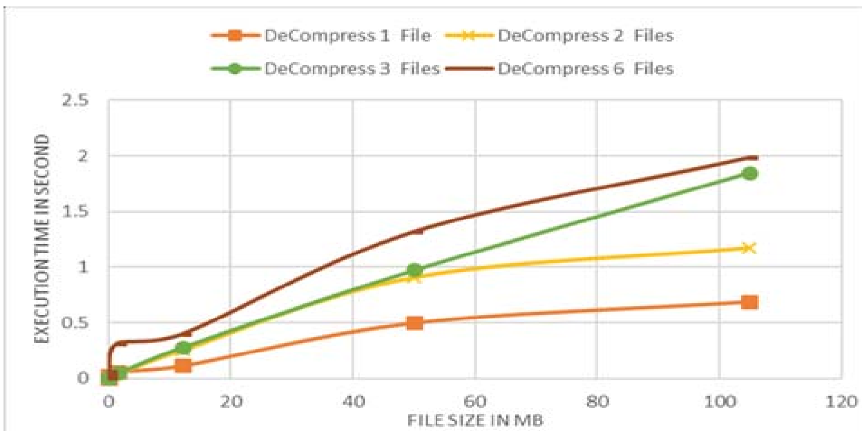

Figure 4.47: Performance of Decompression Text File Type.

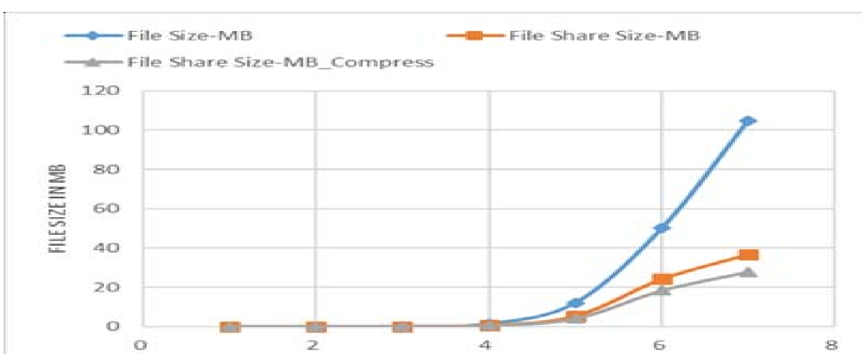

Figure 4.48: File Share Size before Compression and after Compression of Text File Type.

Fig. 4.49, Fig. 4.50 , and Fig. 4.51 shown performance of compression, decompression , and file share size before compression and after compression respectively for Video file.

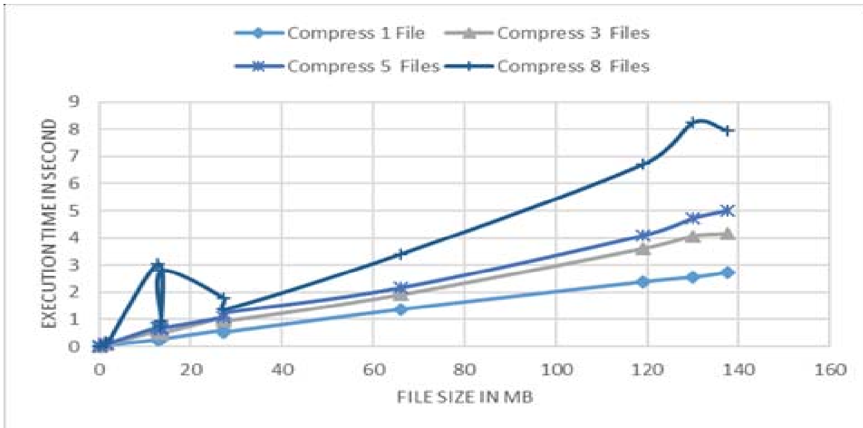

Figure 4.49: Performance of Compression Video File Type.

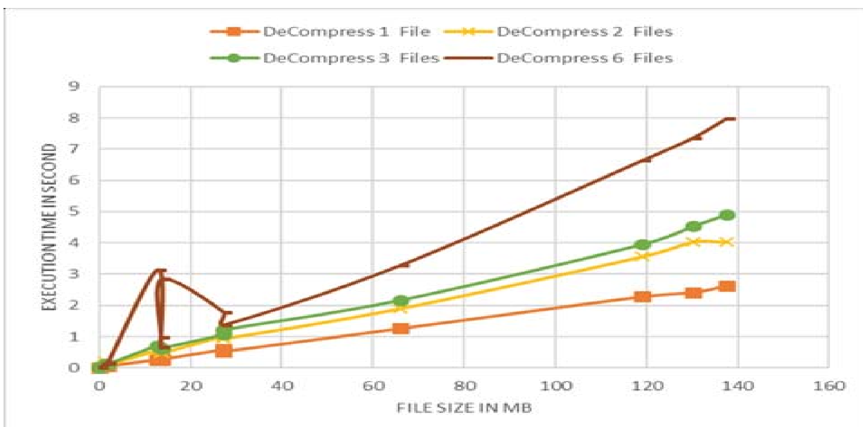

Figure 4.50: Performance of Decompression Video File Type.

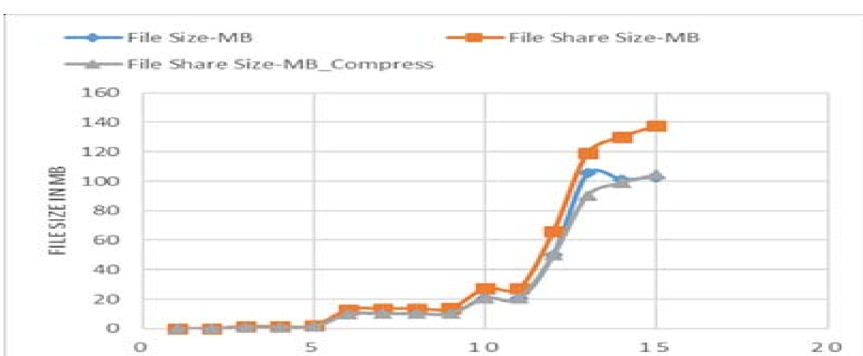

Figure 4.51: File Share Size before Compression and after Compression of Video File Type.

Fig. 4.52, Fig. 4.53 , and Fig. 4.54 shown performance of compression, decompression , and file share size before compression and after compression respectively for Archive file.

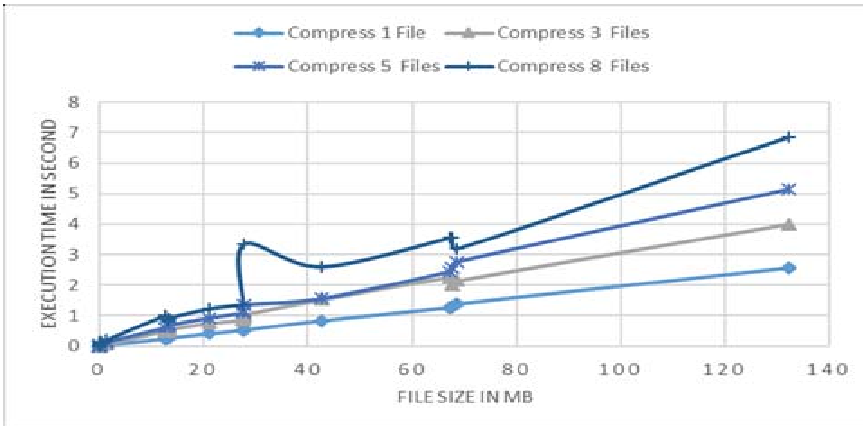

Figure 4.52: Performance of Compression Archive File Type.

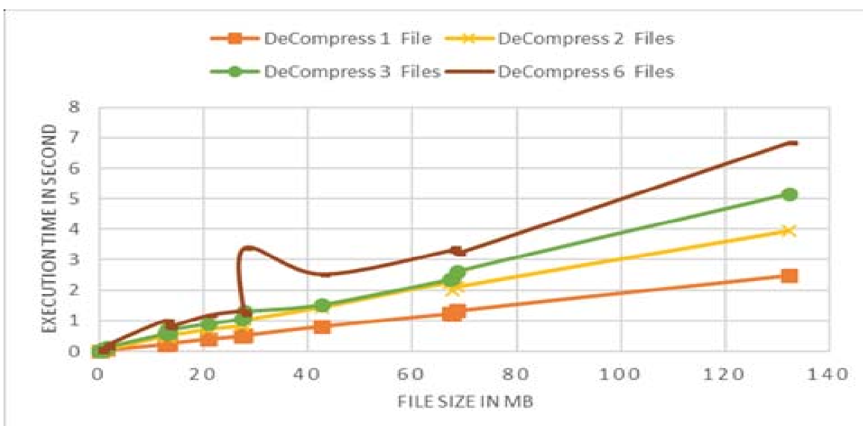

Figure 4.53: Performance of Decompression Archive File Type.

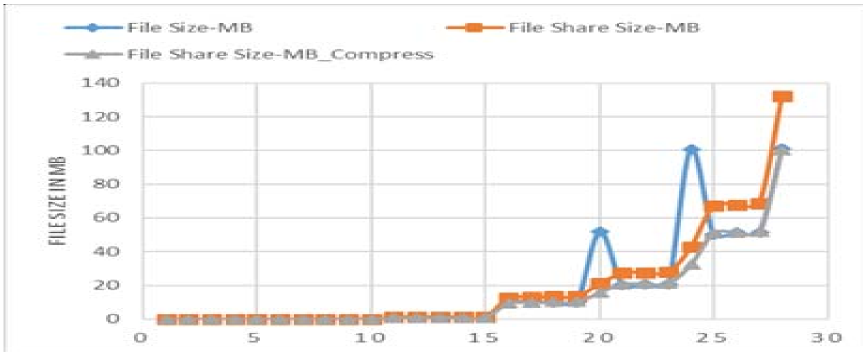

Figure 4.54: File Share Size before Compression and after Compression of Archive File Type.

4.4 Upload Process.

Table 4.9 and the Fig. 4.55 illustrate the experiment of the uploading time for different files with different sizes in parallel to cloud storage. Overall, regarding uploading time, Dropbox is the worst, while the best is SMEstorage which is hosted in Amazon S3. Although, the time in AES_DropBox includes the uploading time for the encryption file and the average time of the uploading all shares of the key.

Table 4.9: Uploading Time in Second for Different Files with Different Sizes in Parallel to Cloud Storage.

File Size-MB	**DropBox**	**Google Drive**	**SME**	**OneDrive**	**SME S3**	**SME S1**	**SME S2**	**SME S4**	**AES DropBox**
0.0007	5.0044	4.1324	3.6020	5.1743	4.6917	4.5937	4.5613	4.7232	10.4428
0.0127	5.1847	4.4554	3.4053	6.0728	5.5025	5.4818	4.5357	4.9421	10.6230
0.0479	5.4428	5.2285	3.8557	5.8736	5.4568	5.2115	4.6294	5.9285	10.8812
0.0990	7.3329	4.6575	4.9211	5.6339	5.3054	5.1047	4.8727	5.0761	12.7713
0.9775	14.9435	13.1944	13.0493	14.5562	15.3208	14.4413	15.5067	15.2434	20.3819
10.0778	58.3193	52.4350	54.1062	55.5180	55.1580	56.1183	57.7684	56.2539	63.7576
25.1418	101.7160	99.5973	101.6725	101.0624	101.1508	100.6766	101.1795	101.5205	107.1544
50.8237	194.5096	193.9219	196.8051	192.1513	193.1364	189.7932	188.0017	192.0535	193.9234

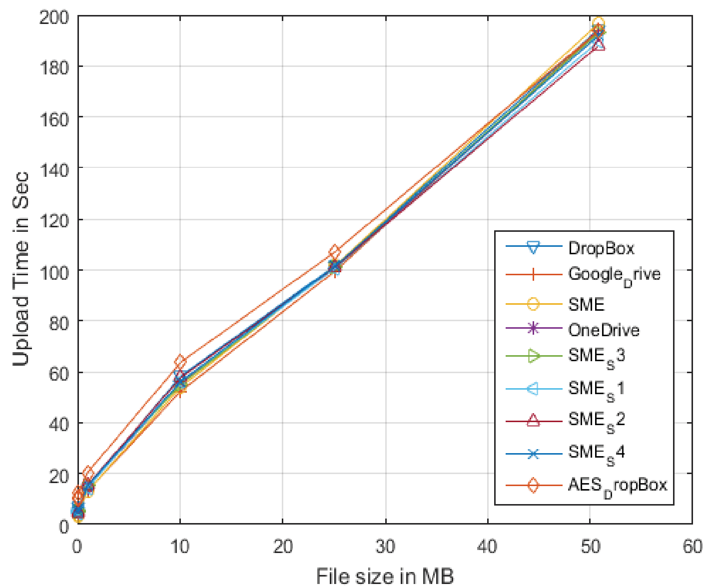

Figure 4.55: Performance Comparison of the Uploading Time for Different Files with Different Sizes in Parallel to Cloud Storage.

4.5 Download Process.

Table 4.10 and the Fig. 4.56 show experiment results of the time of download for different files with different sizes in parallel from cloud storage. concerning download time , Dropbox is the worst, while the best is SMEstorage which is hosted in Amazon S3. However, when the file size is 50MB, the SMEstorage shows the worst performance which indicates the dependence of the results on the network state and the download rate. Although, the time in AES_DropBox includes the download time for the encryption file and the average time of the download all shares of the key.

Table 4.10: Download Time in Second for Different Files with Different Size in Parallel from Cloud Storage.

File Size-MB	DropBox	Google Drive	SME	OneDrive	SME S3	SME S1	SME S2	SME S4	AES DropBox
0.0005	1.1174	1.4544	0.9619	2.2992	1.8707	1.1074	1.1376	1.0545	3.1280
0.0007	1.2886	1.2459	0.8784	1.7981	0.8525	1.0962	0.8456	0.9029	3.2992
0.0127	1.5160	1.5185	1.4284	2.7746	1.4394	1.4522	1.4587	1.5642	3.5266
0.0479	2.3980	1.9101	2.4369	2.9549	2.0131	2.0281	1.9668	3.0042	4.4086
0.0990	2.6569	2.9511	3.0598	5.8820	2.6076	2.8406	2.8947	2.9987	4.6675
0.9775	10.9620	10.6109	11.0750	12.1696	10.3473	10.5648	10.3478	10.1664	12.9726
10.0778	83.5843	76.6780	74.5356	84.0721	71.9610	78.9003	76.2712	73.3049	85.5949
25.1418	182.4045	162.5270	186.0100	172.0826	164.4602	172.1932	182.9029	171.8020	184.4151
50.8237	344.2008	325.6052	351.2803	332.7302	322.0095	327.5514	324.9523	319.7728	346.2114

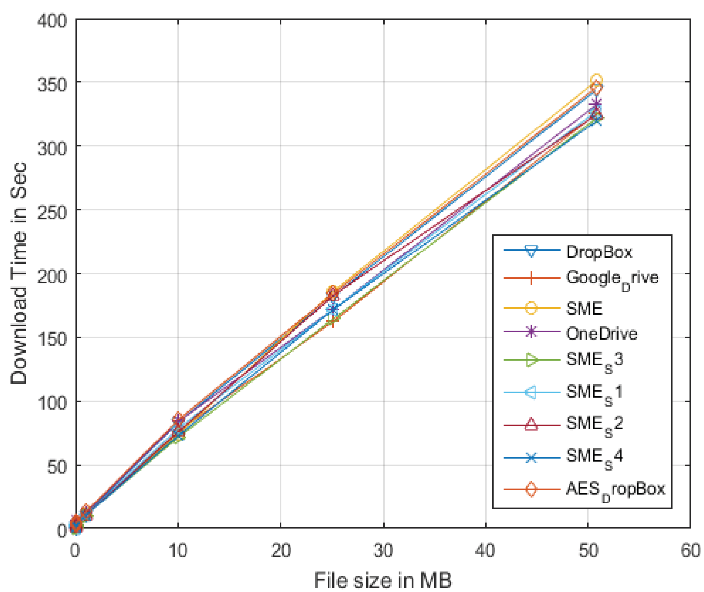

Figure 4.56: Performance Comparison of the Downloading Time for Different Files with Different Size in Parallel from Cloud Storage.

4.6 Conclusion

Overall, we can conclude that the results are all within the acceptable range and by using the index array. Parallel implementation of the scheme shows significantly improve in the results . Also using SFS to build the shares, we noticed that the outputs change significantly according to the number of shares and the threshold .Moreover, increasing both parameters shows acceptable results, and the level of security is definitely enhanced. It is clear that the threshold value plays critical role in the reconstruction process . Finally, compressing file before preparing the shares reduce the time needed significantly.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

In conclusion, securing files in the cloud is a vital issue because large amount of data has been moved to the cloud. Applying the secret file sharing (SFS) for all types of files such as (image, document, system file, audio, and video... etc.) increases the trust and achieves the security goal. Each file is converted to the base64 string before applying the secret sharing mechanism, and the string is compressed using GZIP compression before and after applying the secret sharing scheme. As a result, the file is sent in compressed form and the receiver should decompress the file and get the original shares. Using compression makes the size of file less than the original file even if we convert it to base64 unless the file is already compressed. Our scheme adds more security, extra confidentiality, and availability because the data will be available in multi cloud and the attackers will need to compromise number of clouds more than or equal to the threshold.

It should be noted that there is a trade-off between the execution time and the threshold which means that the outputs change significantly due to the number of shares and the threshold. Increasing the threshold leads to increase the trust in the cloud. Finally, SFS shows significant results compared with symmetric algorithm in both creating and reconstructing the secret for any type of file.

As for future improvements, applying secret file sharing in different field such as the social media. In addition, doing more experiment for large file size .

REFERENCES

[1] A. Westerheim, "What is cloud computing?" accessed:13.02.2016. [Online]. Available: http://www.ekaru.com/blog/bid/92650/What-is-Cloud-Computing?$_sm_au_=iVVsP1FpvNQ6pnvN$

[2] T. Alkharobi, *Secure Repayable Storage System.* Springer, 2008, pp. 102–109.

[3] P. M. Mell and T. Grance, "Sp 800-145. the nist definition of cloud computing," 2011.

[4] "Searchcloudcomputing," accessed: 22.07.2016. [Online]. Available: http://searchcloudcomputing.techtarget.com/

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[6] M. Miller, *Cloud computing: Web-based applications that change the way you work and collaborate online.* Que publishing, 2008.

[7] S. Singh, Y.-S. Jeong, and J. H. Park, "A survey on cloud computing security: Issues, threats, and solutions," *Journal of Network and Computer Applications*, vol. 75, pp. 200–222, 2016.

[8] M. Gobi and M. R. Sridevi, "Performance analysis of biometric image encryption in transformed formats using public key cryptography," *International Journal of Scientific Engineering Research*, vol. 6, no. 2, 2015.

[9] R. Prajapati, "Base64 images advantages disadvantages - coderiddles," 2014. [Online]. Available: http://www.coderiddles.com/base64-images/

[10] A. D. Diary, "Advantages of base64 encoding," 2012. [Online]. Available: http://dev-faqs.blogspot.com/2012/12/advantages-of-base-64-encoding.html?_sm_au_=iVVsP1FpvNQ6pnvN

[11] L. P. Deutsch, "Gzip file format specification version 4.3," 1996.

[12] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[13] G. R. Blakley, "Safeguarding cryptographic keys," vol. 48, 1979, pp. 313–317.

[14] Z. Chen, S. Li, Y. Zhu, J. Yan, and X. Xu, "A cheater identifiable multi-secret sharing scheme based on the chinese remainder theorem," *Security and Communication Networks*, vol. 8, no. 18, pp. 3592–3601, 2015.

[15] S. Iftene, "Secret sharing schemes with applications in security protocols," *Sci. Ann. Cuza Univ.*, vol. 16, pp. 63–96, 2006.

[16] M. Mignotte, *How to share a secret.* Springer, 1982, pp. 371–375.

[17] C.-C. Thien and J.-C. Lin, "Secret image sharing," *Computers Graphics*, vol. 26, no. 5, pp. 765–770, 2002.

[18] R. Lukac and K. N. Plataniotis, "Colour image secret sharing," *Electronics Letters*, vol. 40, no. 9, p. 529, 2004.

[19] D.-C. Lou, H.-H. Chen, H.-C. Wu, and C.-S. Tsai, "A novel authenticatable color visual secret sharing scheme using non-expanded meaningful shares," *Displays*, vol. 32, no. 3, pp. 118–134, 2011.

[20] D.-S. Tsai, G. Horng, T.-H. Chen, and Y.-T. Huang, "A novel secret image sharing scheme for true-color images with size constraint," *Information Sciences*, vol. 179, no. 19, pp. 3247–3254, 2009.

[21] N. S. Alex and L. J. Anbarasi, "Enhanced image secret sharing via error diffusion in halftone visual cryptography," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, vol. 2. IEEE, Conference Proceedings, pp. 393–397.

[22] C.-N. Yang, "New visual secret sharing schemes using probabilistic method," *Pattern Recognition Letters*, vol. 25, no. 4, pp. 481–494, 2004.

[23] T.-L. Lin, S.-J. Horng, K.-H. Lee, P.-L. Chiu, T.-W. Kao, Y.-H. Chen, R.-S. Run, J.-L. Lai, and R.-J. Chen, "A novel visual secret sharing scheme for multiple secrets without pixel expansion," *Expert systems with applications*, vol. 37, no. 12, pp. 7858–7869, 2010.

[24] M. Sasaki and Y. Watanabe, "Formulation of visual secret sharing schemes encrypting multiple images," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on.* IEEE, Conference Proceedings, pp. 7391–7395.

[25] J. He, W. Lan, and S. Tang, "A secure image sharing scheme with high quality stego-images based on steganography," *Multimedia Tools and Applications*, 2016.

[26] P. Li, C.-N. Yang, and Z. Zhou, "Essential secret image sharing scheme with the same size of shadows," *Digital Signal Processing*, vol. 50, pp. 51–60, 2016.

[27] N. Askari, C. Moloney, and H. M. Heys, "A novel visual secret sharing scheme without image size expansion," in *Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on.* IEEE, Conference Proceedings, pp. 1–4.

[28] X.-Y. Liu, M.-S. Chen, and Y.-L. Zhang, "A new color visual cryptography scheme with perfect contrast," in *Communications and Networking in China (CHINACOM), 2013 8th International ICST Conference on.* IEEE, Conference Proceedings, pp. 449–454.

[29] T.-H. Chen and K.-H. Tsao, "Visual secret sharing by random grids revisited," *Pattern Recognition*, vol. 42, no. 9, pp. 2203–2217, 2009.

[30] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage." in *Fast*, vol. 3, 2003, pp. 29–42.

[31] Y. L. Y. C. G. X. Xin Dong, Jiadi Yu and M. Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *Computers security*, no. 42, p. 151164, 2014.

[32] A. Bessani, M. Correia, B. Quaresma, F. Andr, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," *ACM Transactions on Storage (TOS)*, vol. 9, no. 4, p. 12, 2013.

[33] F. Alsolami and T. Boult, "Cloudstash: using secret-sharing scheme to secure data, not keys, in multi-clouds," in *Information Technology: New Generations (ITNG), 2014 11th International Conference on.* IEEE, Conference Proceedings, pp. 315–320.

[34] V. S. Agme and A. C. Lomte, "Cloud data storage security enhancement using identity based encryption," *Identity*, vol. 3, no. 4, 2014.

[35] D. Esbensen, "Apparatus and method for fast data encoding and decoding," 2012.

[36] S. Josefsson, "The base16, base32, and base64 data encodings," 2006.

[37] W. L. Li, R. X. Zhu, J. Kang, L. Tao, and G. H. Cai, "A design of improved base64 encoding algorithm based on fpga," in *Applied Mechanics and Materials*, vol. 513. Trans Tech Publ, 2014, pp. 2220–2223.

[38] M. Shirali-Shahreza and S. Shirali-Shahreza, "Sending pictures by sms," in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, vol. 1. IEEE, Conference Proceedings, pp. 222–223.

[39] G. Singh, "Modified vigenere encryption algorithm and its hybrid implementation with base64 and aes," in *Advanced Computing, Networking and Security (ADCONS), 2013 2nd International Conference on*. IEEE, Conference Proceedings, pp. 232–237.

[40] Wikipedia, "Base64," 2013. [Online]. Available: https://en.wikipedia.org/wiki/Base64

[41] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.

[42] N. F. Pub, "197: Advanced encryption standard (aes)," *Federal Information Processing Standards Publication*, vol. 197, no. 441, p. 0311, 2001.

[43] S. Gupta and S. Lamba, "An enhanced python based approach of secret sharing scheme with encryption," *Issues*, vol. 1, no. 1, pp. 173–180, 2014.

[44] T. Guo, F. Liu, C. Wu, C. Yang, W. Wang, and Y. Ren, "Threshold secret image sharing," in *International Conference on Information and Communications Security*. Springer, 2013, pp. 404–412.

[45] M. Kadam, S. Chaudhary, and B. Carvalho, "Security approach for multi-cloud data storage," *International Journal of Computer Applications*, vol. 126, no. 4, 2015.

[46] M. Villari, A. Celesti, F. Tusa, and A. Puliafito, "Data reliability in multi-provider cloud storage service with rrns," in *European Conference on Service-Oriented and Cloud Computing.* Springer, 2013, pp. 83–93.

[47] L. Collin, "A quick benchmark: Gzip vs. bzip2 vs. lzma," May 2005. [Online]. Available: https://tukaani.org/lzma/benchmarks.html

[48] N. T. Ltd, "Dotnetcompression," accessed: 22.05.2017. [Online]. Available: https://www.noemax.com/dotnetcompression/compression-analysis-tool

APPENDIX A

CREATE THE SHARES

A.1 Audio file formats

Table A.1: Sequential implementation of Creating the Shares and Encryption using AES of Audio File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.wav	0.000881	0.258789	0.001101	0.001506	0.001522	0.001806	0.001921	0.036963
.mp3	0.001864	0.259201	0.001985	0.002791	0.003727	0.003432	0.003353	0.330767
.wav	0.007831	0.294619	0.005677	0.008783	0.010861	0.052654	0.01421	0.012654
.mp3	0.010812	0.273951	0.007457	0.011918	0.012524	0.046865	0.146118	0.017968
.ogg	0.012504	0.271327	0.008783	0.016023	0.015435	0.026	0.351585	0.037681
.aac	0.012911	0.298225	0.022086	0.01627	0.013338	0.052738	0.048324	0.04841
.WAV	0.121142	0.551301	0.105401	0.15469	0.156741	0.32877	0.224957	0.22039
.MP3	0.130061	0.548204	0.122379	0.150613	0.152555	0.23105	0.228292	0.257148
.WAV	1.113782	3.265649	0.815514	1.299425	1.264672	1.927354	1.933144	2.08116
.ogg	1.198285	3.582327	0.906589	1.335281	1.394397	2.088357	2.076078	2.204324
.MP3	1.340312	3.954375	0.986417	1.494072	1.599407	2.226711	2.518357	2.546987
.mp3	13.39954	35.3693	9.358547	15.65688	15.54519	21.38748	21.29459	23.6293
.mp3	25.38321	69.01725	18.81481	30.15872	30.39643	44.30236	45.30303	51.80042
.mp3	63.33239	160.4711	47.91844	74.07479	74.86859	108.1481	109.4823	114.2877
.mp3	120.1339	304.848	87.90203	138.0069	141.0048	202.002	203.049	212.0839
Sum=	226.1994	583.2636	166.9772	262.3887	266.4402	382.8257	386.6753	409.5957
Throughput (MB/ Sec)		0.387817	1.354672	0.862078	0.848969	0.590868	0.584986	0.55225

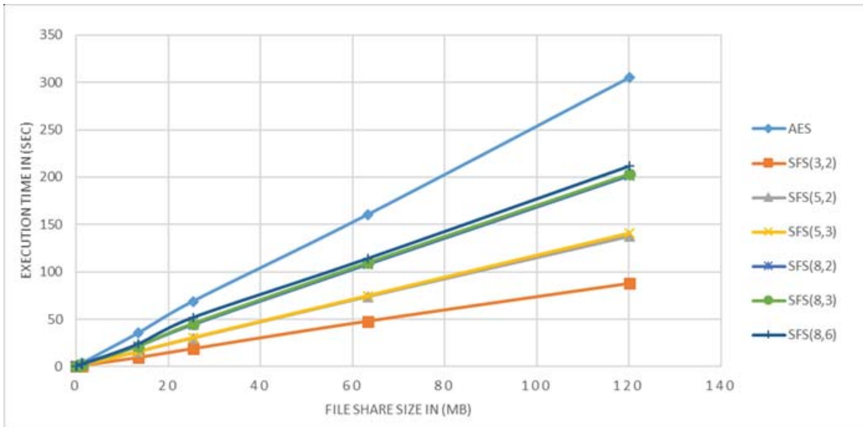

Figure A.1: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Audio File Type of Different Sizes.

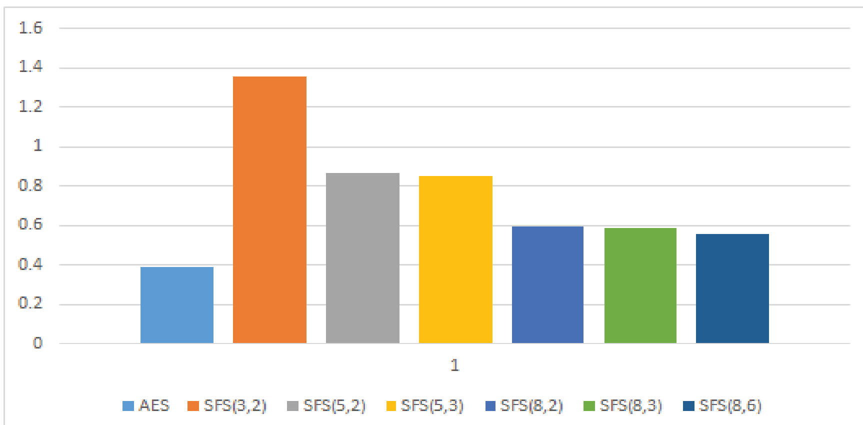

Figure A.2: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Audio File Type of Different Sizes.

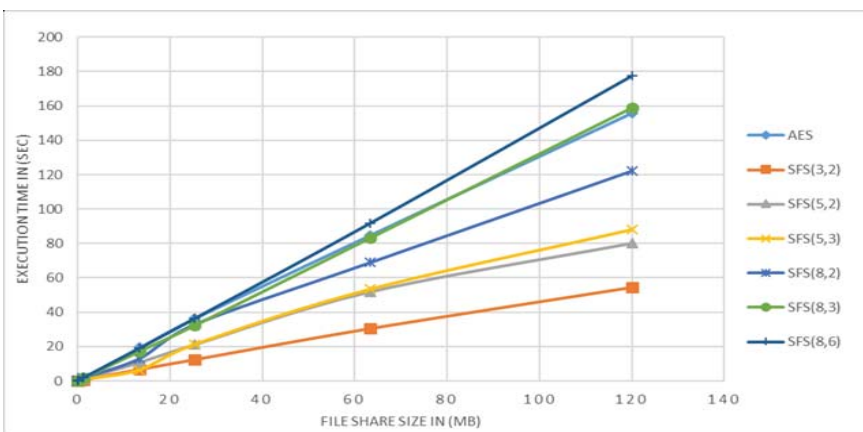

Figure A.3: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Audio File Type of Different Sizes.

Table A.2: Parallel implementation of Creating the Shares and Encryption using AES of Audio File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.wav	0.000881	0.212544	0.003598	0.005382	0.006298	0.007093	0.006465	0.005412
.mp3	0.001864	0.212733	0.004581	0.008444	0.008387	0.007231	0.007532	0.008529
.wav	0.007831	0.216555	0.010176	0.021127	0.021199	0.022377	0.023183	0.037674
.mp3	0.010812	0.221017	0.012955	0.020222	0.021914	0.029407	0.030511	0.065329
.ogg	0.012504	0.214901	0.014742	0.026708	0.027311	0.047649	0.035634	0.078343
.aac	0.012911	0.219033	0.016109	0.023323	0.029781	0.033199	0.036728	0.048151
.WAV	0.121142	0.425984	0.137705	0.202679	0.213944	0.304942	0.349024	0.35694
.MP3	0.130061	0.428282	0.132465	0.204694	0.266967	0.368537	0.342154	0.452074
.WAV	1.113782	1.691637	0.614169	1.098647	1.140247	1.624402	1.472002	1.74404
.ogg	1.198285	1.743921	0.609442	1.051533	1.223562	1.69918	1.561549	1.730436
.MP3	1.340312	1.856065	0.669877	1.225101	1.13647	1.946015	1.718602	2.043991
.mp3	13.39954	19.27124	6.715657	10.7606	6.412136	12.95901	17.18541	19.03588
.mp3	25.38321	36.33403	12.43381	21.13461	21.31677	32.9218	32.30493	36.29371
.mp3	63.33239	84.57047	30.49876	51.82066	53.35351	68.85662	83.14607	91.55714
.mp3	120.1339	155.721	54.54024	80.16999	88.11212	122.2988	159.0777	177.3213
sum=	226.1994	303.3394	106.4143	167.7737	173.2906	243.1262	297.2975	330.7789
Throughput (MB / SEC)		0.745698	2.125649	1.348241	1.305319	0.930379	0.760852	0.683839

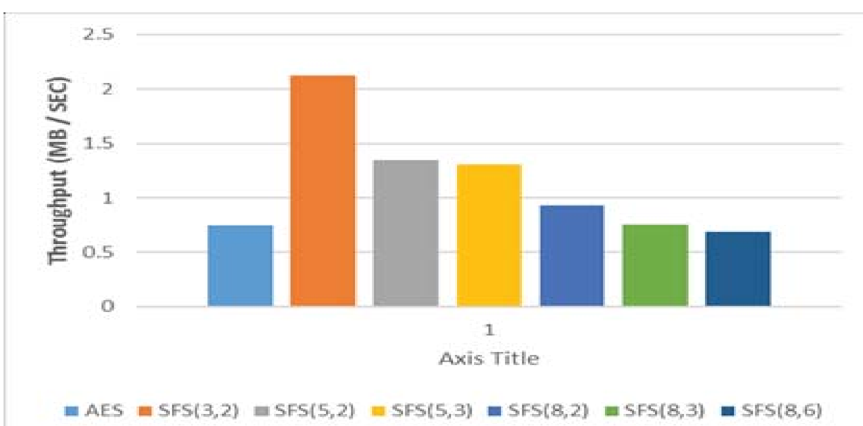

Figure A.4: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Audio File Type of Different Sizes.

A.2 Binary file formats

Table A.3: Sequential implementation of Creating the Shares and Encryption using AES of Binary File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bin	0.000469	0.260092	0.001124	0.001519	0.001056	0.001593	0.001285	0.001665
.bin	0.001416	0.272884	0.001551	0.001632	0.001797	0.003898	0.002612	0.002727
.bin	0.006151	0.263719	0.005139	0.007253	0.009235	0.015373	0.012344	0.019479
.bin	0.013671	0.260876	0.012379	0.015352	0.016438	0.022141	0.022835	0.054751
.BIN	0.099341	0.5704	0.075895	0.107014	0.107434	0.163852	0.176037	0.183152
.bin	0.136086	0.539318	0.100944	0.171087	0.145995	0.231123	0.265223	0.218306
.BIN	11.06026	29.72963	8.497002	12.66103	12.90051	18.31017	20.1091	20.03366
.bin	68.75715	176.3291	55.19748	80.41183	80.55144	111.095	122.2123	123.6259
Sum=	80.07454	208.226	63.89152	93.37672	93.73391	129.8431	142.8017	144.1397
Throughput (MB/ Sec)		0.384556	1.253289	0.857543	0.854275	0.616702	0.560739	0.555534

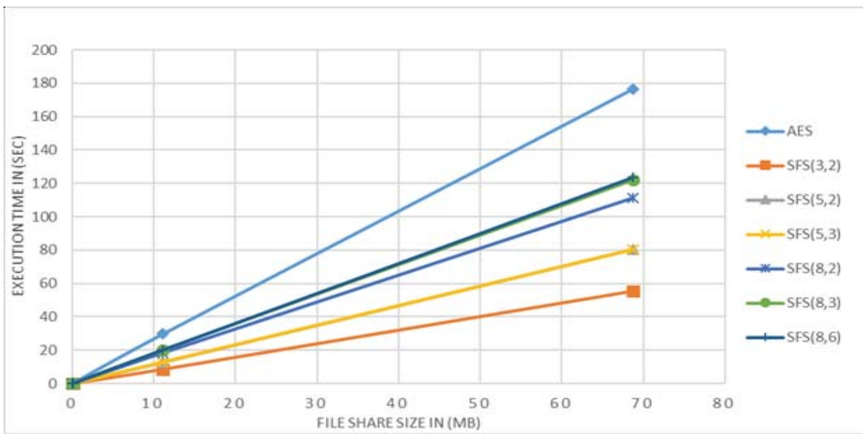

Figure A.5: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Binary File Type of Different Sizes.

Table A.4: Parallel implementation of Creating the Shares and Encryption using AES of Binary File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bin	0.000469	0.210581	0.021321	0.005805	0.006655	0.010671	0.007664	0.015982
.bin	0.001416	0.212847	0.118866	0.010359	0.011488	0.00953	0.011657	0.010061
.bin	0.006151	0.220952	0.012162	0.016129	0.017953	0.023464	0.02309	0.023072
.bin	0.013671	0.215129	0.020866	0.027378	0.028539	0.391576	0.05124	0.078118
.BIN	0.099341	0.427601	0.119191	0.165678	0.195634	0.259694	0.278518	0.309176
.bin	0.136086	0.441484	0.139459	0.244893	0.249088	0.436146	0.428882	0.412394
.BIN	11.06026	15.06531	5.601418	9.278003	9.897554	13.57165	15.24423	15.84823
.bin	68.75715	93.41486	35.86503	53.97019	57.10886	79.08547	77.76194	86.11326
sum=	80.07454	110.2088	41.89831	63.71844	67.51577	93.78821	93.80722	102.8103
Throughput (MB / SEC)		0.726571	1.911164	1.256693	1.186012	0.85378	0.853607	0.778857

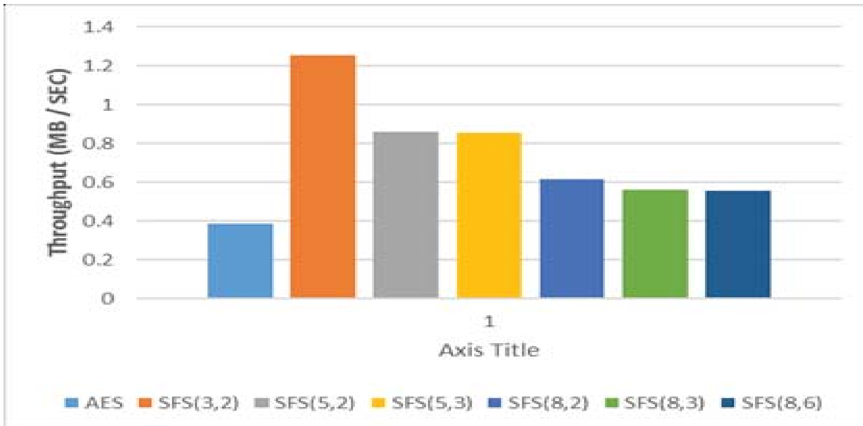

Figure A.6: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Binary File Type of Different Sizes.

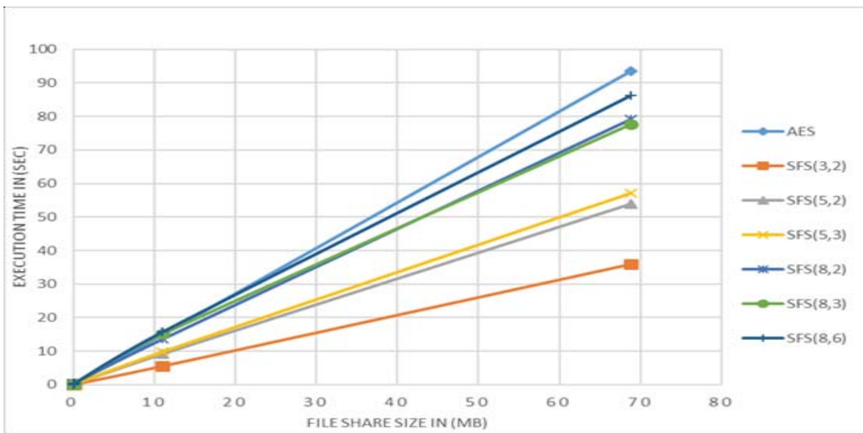

Figure A.7: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Binary File Type of Different Sizes.

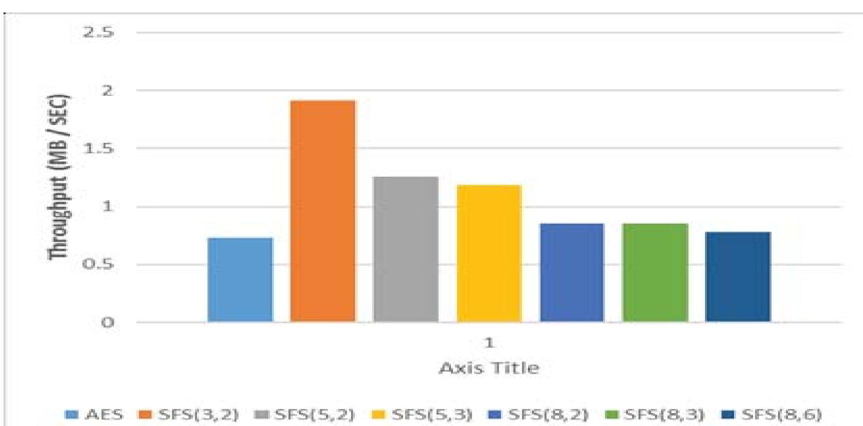

Figure A.8: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Binary File Type of Different Sizes.

A.3 Document file formats

Table A.5: Sequential implementation of Creating the Shares and Encryption using AES of Document File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.rtf	0.000682	0.25657	0.00087	0.001034	0.00115	0.001674	0.002312	0.001945
.doc	0.00391	0.282181	0.004034	0.004856	0.005231	0.008066	0.007063	0.007883
.rtf	0.00412	0.272458	0.003338	0.004924	0.004977	0.015225	0.008683	0.010046
.xls	0.015959	0.3028	0.009538	0.018354	0.018126	0.029702	0.032278	0.095342
.doc	0.024937	0.281358	0.017203	0.060271	0.032458	0.064207	0.039818	0.060842
.ppt	0.029634	0.270333	0.025412	0.029638	0.036121	0.054576	0.059647	0.062144
.rtf	0.176815	0.539536	0.137009	0.199765	0.219962	0.343091	0.304243	0.329191
.xls	0.351154	1.072159	0.307644	0.500562	0.389111	0.6178	0.59127	0.672995
.doc	0.514131	1.589471	0.38096	0.592942	0.583752	0.907477	0.94272	0.938984
.ppt	0.977554	2.914775	0.764354	1.273339	1.120937	1.853127	1.945434	2.057141
.rtf	2.758348	7.745389	2.140532	3.15942	3.103664	5.001559	4.989784	5.426379
.doc	4.955577	13.82073	3.795599	5.653618	5.616455	8.310213	8.638379	9.0385
.xls	7.267596	19.87585	5.556785	8.434565	8.229632	10.66159	11.53966	13.23916
.doc	20.19167	53.35038	15.61758	23.37844	23.45928	35.16928	36.81468	38.29706
Sum=	37.27208	102.574	28.76086	43.31173	42.82086	63.03759	65.91597	70.23761
Throughput (MB/ Sec)		0.363368	1.295931	0.860554	0.870419	0.591268	0.565448	0.530657

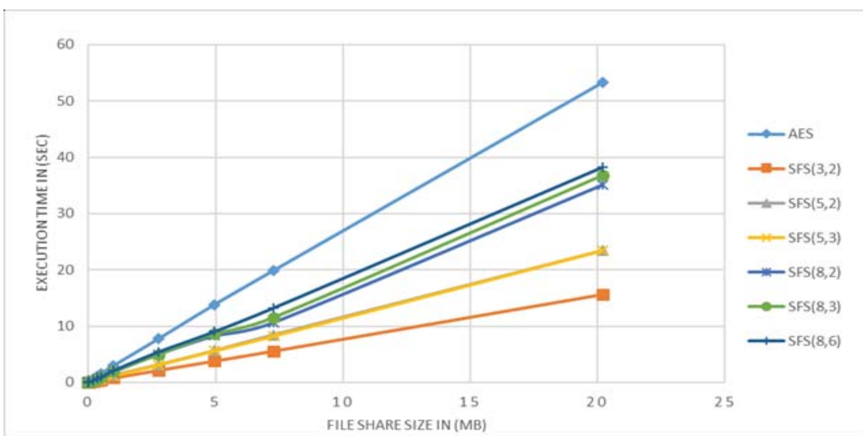

Figure A.9: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Document File Type of Different Sizes.

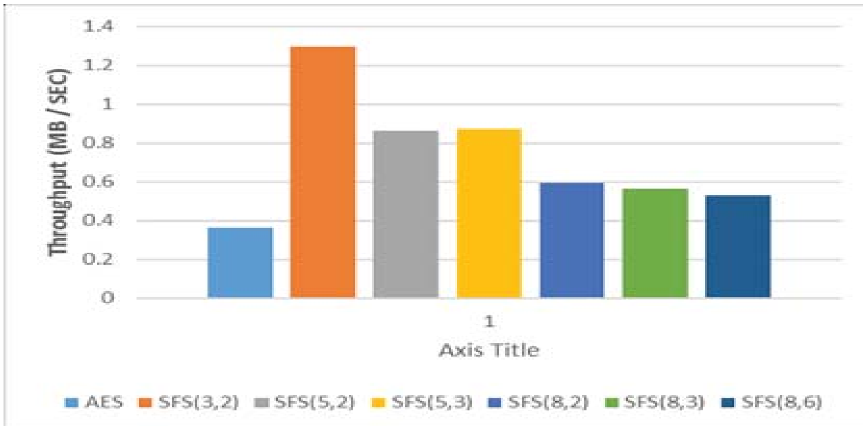

Figure A.10: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Document File Type of Different Sizes.

Table A.6: Parallel implementation of Creating the Shares and Encryption using AES of Document File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.rtf	0.000682	0.22177	0.017078	0.007246	0.006886	0.008299	0.022362	0.022258
.doc	0.00391	0.216435	0.014056	0.012681	0.019312	0.066418	0.024594	0.053482
.rtf	0.00412	0.211404	0.010761	0.018826	0.412218	0.017453	0.024363	0.06178
.xls	0.015959	0.21169	0.022814	0.041189	0.032128	0.056442	0.055528	0.09306
.doc	0.024937	0.219155	0.031692	0.05265	0.068825	0.079153	0.117189	0.106038
.ppt	0.029634	0.214551	0.051261	0.053423	0.0951	0.082261	0.085379	0.106089
.rtf	0.176815	0.42811	0.192223	0.301592	0.626554	0.489394	0.48686	0.514473
.xls	0.351154	0.882336	0.363713	0.668518	0.615499	0.999196	1.097724	1.153876
.doc	0.514131	0.96838	0.564774	0.883676	0.952018	1.718909	1.585221	1.494305
.ppt	0.977554	1.530761	0.62087	1.272644	1.377048	2.014145	2.087577	2.151535
.rtf	2.758348	3.683754	1.619502	2.628539	2.755792	4.117527	4.279383	4.31965
.doc	4.955577	6.791374	2.799002	4.622872	4.646973	6.486644	6.880884	7.427726
.xls	7.267596	10.33937	4.087246	6.353829	6.703177	9.499622	10.14086	10.54591
.doc	20.19167	29.22544	10.56182	17.06578	17.50842	25.83571	27.12501	29.36498
Sum=	37.27208	55.14453	20.95682	33.98346	35.81995	51.47117	54.01293	57.41516
Throughput (MB / SEC)		0.675898	1.778519	1.096771	1.04054	0.724135	0.690059	0.649168

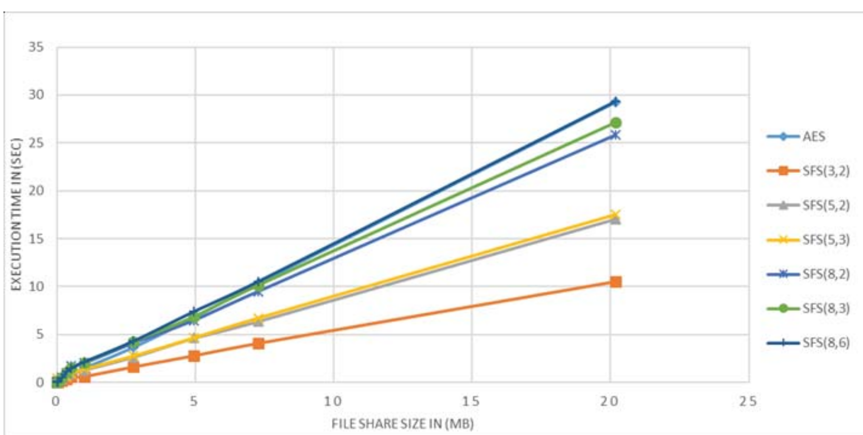

Figure A.11: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Document File Type of Different Sizes.

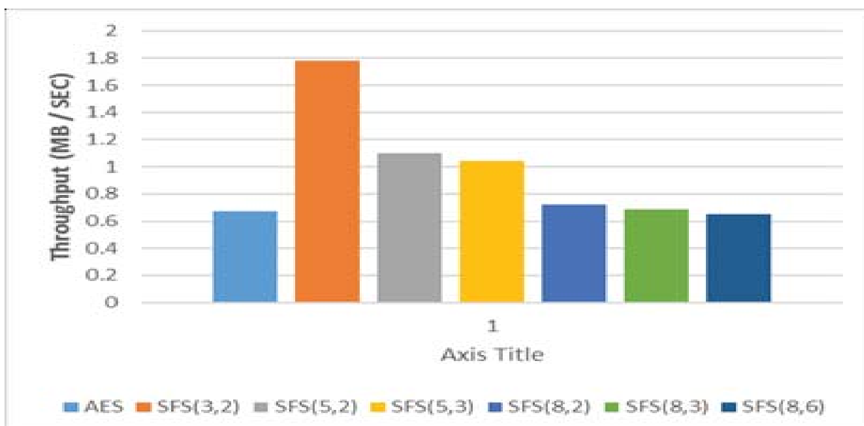

Figure A.12: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Document File Type of Different Sizes.

A.4 Executable file formats

Table A.7: Sequential implementation of Creating the Shares and Encryption using AES of Executable File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bat	0.000713	0.244369	0.000845	0.001197	0.001043	0.001493	0.001945	0.002084
.dll	0.001415	0.274988	0.00151	0.001782	0.001882	0.002211	0.009267	0.002838
.exe	0.001495	0.284219	0.001475	0.001992	0.002045	0.003332	0.004632	0.003099
.dll	0.005783	0.283335	0.004423	0.006806	0.006662	0.008781	0.009836	0.011689
.exe	0.006152	0.2303	0.004985	0.006714	0.008062	0.01182	0.011509	0.011364
.dll	0.04107	0.264275	0.031726	0.065644	0.047674	0.07236	0.096915	0.090005
.exe	0.066244	0.259295	0.055403	0.073512	0.077957	0.117566	0.123413	0.111054
.dll	0.248048	0.818193	0.191385	0.284103	0.304107	0.399444	0.436465	0.448274
.exe	1.223361	3.450952	0.920674	1.366027	1.425149	2.113238	2.083826	2.210319
.dll	2.333558	6.630939	1.730802	2.695176	3.373681	4.228074	4.24525	4.357876
.dll	10.86065	29.06437	8.233721	13.1182	13.0774	19.21185	18.82423	20.01682
.exe	26.83663	68.62444	20.30046	32.11373	32.54346	47.40163	47.0745	49.92586
.exe	67.34037	172.5429	50.85121	78.15311	78.34051	114.4208	114.9852	119.9217
.exe	135.1324	342.9647	100.2372	153.7538	155.466	228.755	228.2236	240.3368
Sum=	244.0979	625.9372	182.5658	281.6418	284.6756	416.7476	416.1305	437.4498
Throughput (MB/ Sec)		0.389972	1.337041	0.866696	0.85746	0.585721	0.58659	0.558002

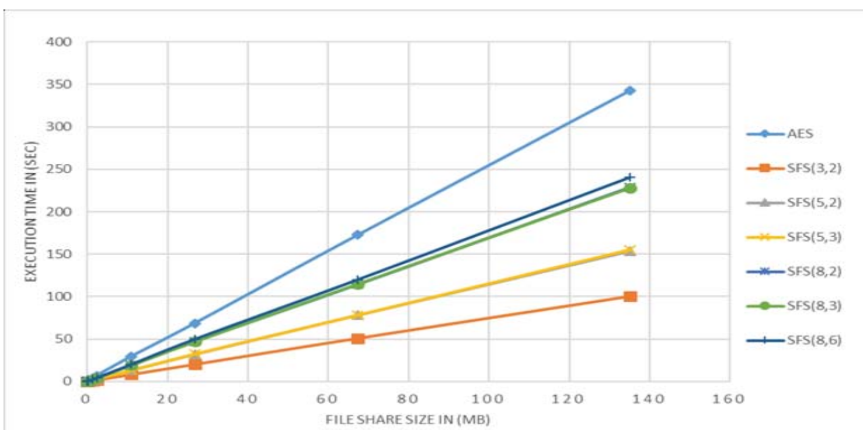

Figure A.13: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Executable File Type of Different Sizes.

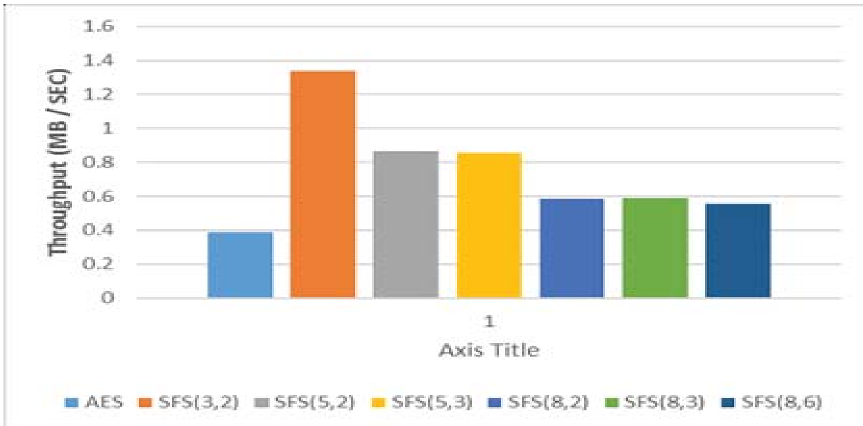

Figure A.14: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Executable File Type of Different Sizes.

Table A.8: Parallel implementation of Creating the Shares and Encryption using AES of Executable File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bat	0.000713	0.213739	0.003598	0.003121	0.005068	0.078625	0.005974	0.004703
.dll	0.001415	0.20849	0.004408	0.004694	0.004798	0.007484	0.007125	0.011655
.exe	0.001495	0.215081	0.004252	0.006489	0.007339	0.009656	0.009148	0.013773
.dll	0.005783	0.207518	0.009643	0.01181	0.012659	0.018706	0.026482	0.020472
.exe	0.006152	0.211504	0.012021	0.020524	0.016289	0.021564	0.018982	0.02145
.dll	0.04107	0.226455	0.042681	0.065367	0.114305	0.123692	0.121251	0.127043
.exe	0.066244	0.219964	0.070777	0.105348	0.11523	0.205284	0.169415	0.203267
.dll	0.248048	0.640215	0.247825	0.466582	0.425578	0.800424	0.644667	0.709498
.exe	1.223361	1.779023	0.631442	1.099701	1.237363	1.933157	1.621186	1.81041
.dll	2.333558	3.223319	1.211213	2.068216	1.423098	2.955583	3.199935	3.633691
.dll	10.86065	15.46298	5.453487	8.68426	9.285773	14.61379	13.59542	15.15177
.exe	26.83663	37.99645	13.87555	22.55861	23.30909	35.69784	33.9087	37.81834
.exe	67.34037	89.63716	33.79753	52.88443	55.95502	90.8836	86.52061	100.0499
.exe	135.1324	176.3424	61.02184	90.86358	92.96739	181.6937	176.8927	202.8125
Sum=	244.0979	326.5843	116.3863	178.8427	184.879	329.0431	316.7416	362.3885
Throughput (MB / SEC)		0.747427	2.097308	1.364874	1.320312	0.741842	0.770653	0.673581

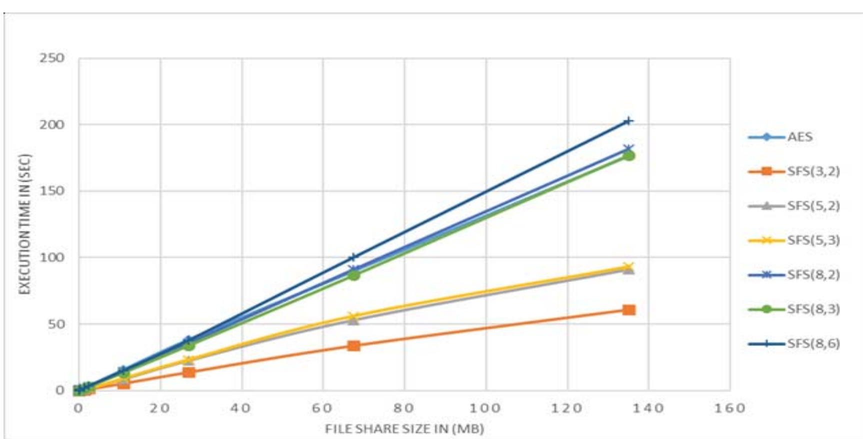

Figure A.15: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Executable File Type of Different Sizes.

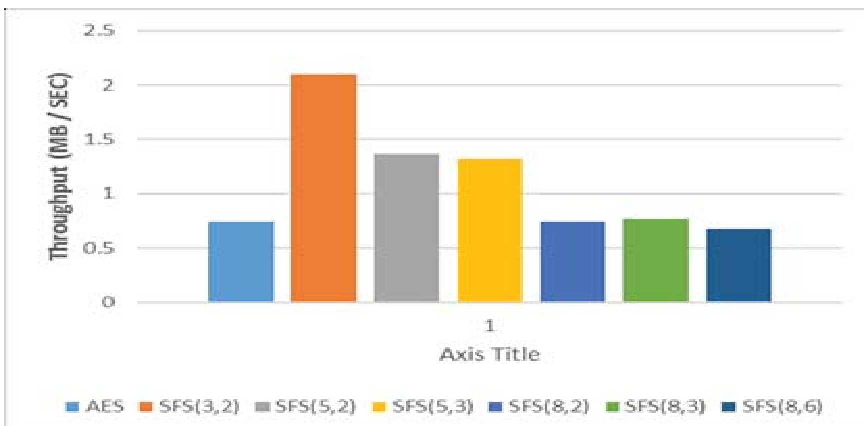

Figure A.16: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Executable File Type of Different Sizes.

A.5 Image file formats

Table A.9: Sequential implementation of Creating the Shares and Encryption using AES of Image File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ico	0.000775	0.26154	0.000905	0.001321	0.002072	0.001593	0.004676	0.002925
.bmp	0.000807	0.267638	0.001277	0.001177	0.001161	0.001723	0.001756	0.002494
.jpg	0.001109	0.263	0.001013	0.001641	0.001667	0.002318	0.002568	0.002366
.gif	0.001202	0.316324	0.003517	0.001806	0.00189	0.002231	0.002264	0.002411
.png	0.001417	0.26047	0.001752	0.00169	0.001741	0.004881	0.002607	0.003035
.bmp	0.003551	0.303355	0.003708	0.00369	0.004199	0.005792	0.005834	0.040669
.jpg	0.005877	0.298653	0.004796	0.006924	0.007445	0.009013	0.017379	0.010203
.jpg	0.006936	0.283171	0.006992	0.007034	0.00958	0.03364	0.043605	0.012393
.jpg	0.012669	0.280681	0.011543	0.017913	0.016	0.021337	0.046872	0.029333
.png	0.012883	0.258304	0.055577	0.013994	0.013029	0.020287	0.047906	0.021543
.gif	0.013248	0.299246	0.010646	0.014103	0.258932	0.021056	0.021449	0.024236
.png	0.013301	0.303843	0.010123	0.041552	0.015814	0.020709	0.04604	0.060099
.bmp	0.03936	0.271087	0.034906	0.044479	0.046854	0.07155	0.074185	0.071894
.jpg	0.114242	0.533996	0.10096	0.146746	0.166983	0.224292	0.228562	0.211573
.png	0.129358	0.531398	0.119743	0.157928	0.183688	0.234433	0.253805	0.23771
.jpg	0.129999	0.565075	0.103761	0.165103	0.167235	0.213376	0.224194	0.361685
.gif	0.130485	0.553679	0.105285	0.186552	0.16837	0.21483	0.307666	0.242841
.jpg	0.278825	0.789374	0.217536	0.316601	0.378456	0.500222	0.482507	0.565068
.gif	1.343994	3.759202	1.05984	1.571618	1.631586	2.170897	2.730744	2.602271
.png	1.365189	3.995258	1.02548	1.576976	1.587642	2.212854	2.503148	2.689167
.jpg	1.377728	3.992126	1.009566	1.608666	1.690152	2.009282	2.644322	2.821811
.jpg	13.55293	36.6108	10.79649	15.79177	16.02471	21.86269	24.35159	25.74785
.jpg	27.60827	70.17239	21.14492	31.71205	30.87441	41.15647	47.6556	50.65194
.jpg	67.6302	170.7265	53.15942	79.34566	80.19234	108.7535	122.1022	122.2144
.jpg	114.8891	290.0348	90.64383	133.962	134.8911	182.304	197.8783	205.6673
Sum=	228.6635	585.9319	179.6336	266.699	268.3371	362.073	401.6797	414.2972
Throughput (MB/ Sec)		0.390256	1.272944	0.857384	0.85215	0.63154	0.569268	0.551931

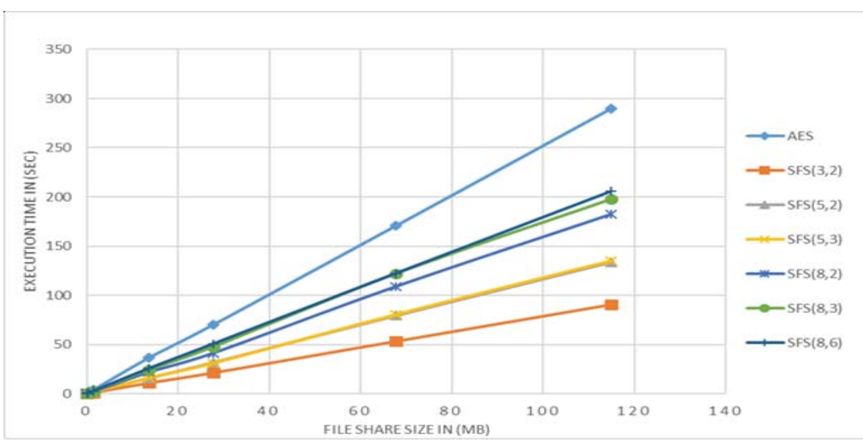

Figure A.17: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Image File Type of Different Sizes.

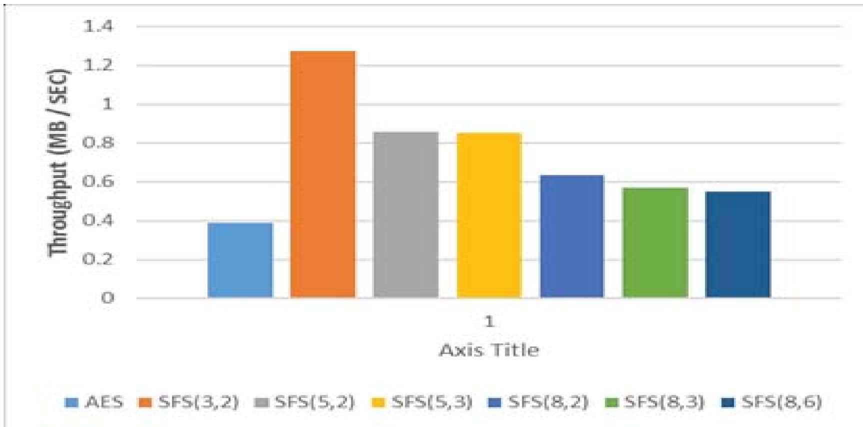

Figure A.18: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Image File Type of Different Sizes.

Table A.10: Parallel implementation of Creating the Shares and Encryption using AES of Image File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ico	0.000775	0.207467	0.008654	0.007275	0.007688	0.009064	0.01013	0.015162
.bmp	0.000807	0.209836	0.007422	0.02198	0.009449	0.081768	0.0107	0.015228
.jpg	0.001109	0.210709	0.009503	0.011064	0.007492	0.009034	0.009331	0.008268
.gif	0.001202	0.212189	0.023548	0.011473	0.013421	0.009382	0.028617	0.025267
.png	0.001417	0.206505	0.007472	0.008562	0.008097	0.008905	0.012053	0.043258
.bmp	0.003551	0.208624	0.010847	0.030567	0.014178	0.021826	0.026642	0.041532
.jpg	0.005877	0.212804	0.011722	0.035109	0.016442	0.022211	0.024464	0.023615
.jpg	0.006936	0.210046	0.020734	0.02312	0.018069	0.032686	0.034866	0.026817
.jpg	0.012669	0.209779	0.033818	0.046068	0.029402	0.045234	0.039285	0.085553
.png	0.012883	0.227681	0.022824	0.146297	0.027478	0.040673	0.099511	0.061354
.gif	0.013248	0.212815	0.085903	0.041822	0.048648	0.058377	0.063148	0.094535
.png	0.013301	0.21411	0.021279	0.027278	0.307864	0.057364	0.042239	0.077219
.bmp	0.03936	0.212838	0.050551	0.089628	0.08981	0.119262	0.126191	0.138153
.jpg	0.114242	0.430414	0.11745	0.237439	0.204744	0.399508	0.446457	0.519105
.png	0.129358	0.424773	0.162726	0.246521	0.275861	0.347335	0.374399	0.383393
.jpg	0.129999	0.421199	0.144001	0.230935	0.244129	0.383875	0.364692	0.376488
.gif	0.130485	0.443856	0.179598	0.213325	0.261551	0.340514	0.510153	0.412996
.jpg	0.278825	0.643629	0.401487	0.51326	0.513809	0.803113	0.784673	1.094782
.gif	1.343994	1.840062	0.914034	1.472087	1.642216	2.292468	2.38604	2.502315
.png	1.365189	1.935442	0.983557	1.515872	1.620416	2.644309	2.433163	2.525222
.jpg	1.377728	1.960808	1.223339	0.897234	0.925212	2.513832	2.480095	1.631172
.jpg	13.55293	19.25334	7.064779	11.01736	11.43894	17.55378	18.02951	19.2361
.jpg	27.60827	36.63105	12.92628	12.59721	15.37501	27.86897	30.84277	32.11911
.jpg	67.6302	89.56975	33.27388	47.52559	50.4043	72.29256	76.56782	81.29878
.jpg	114.8891	150.7191	49.12106	76.95854	84.14759	119.1356	130.8456	144.5832
sum=	228.6635	307.0288	106.8265	153.9256	167.6518	247.0917	266.5926	287.3386
Throughput (MB / SEC)		0.744762	2.140513	1.485545	1.363919	0.925419	0.857726	0.795798

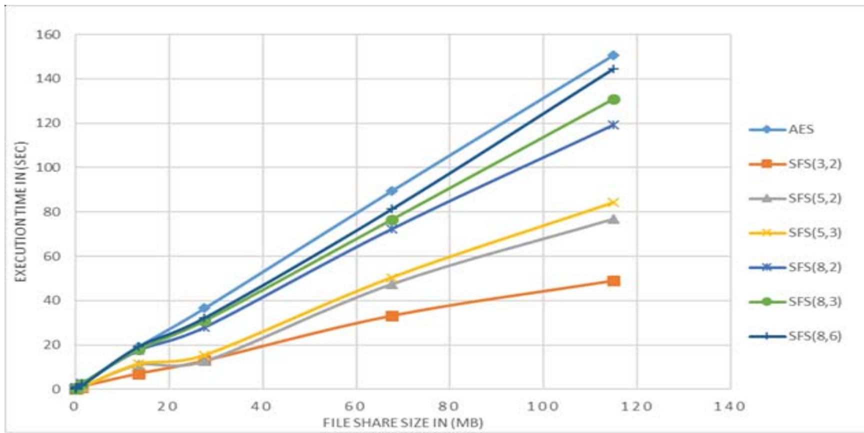

Figure A.19: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Image File Type of Different Sizes.

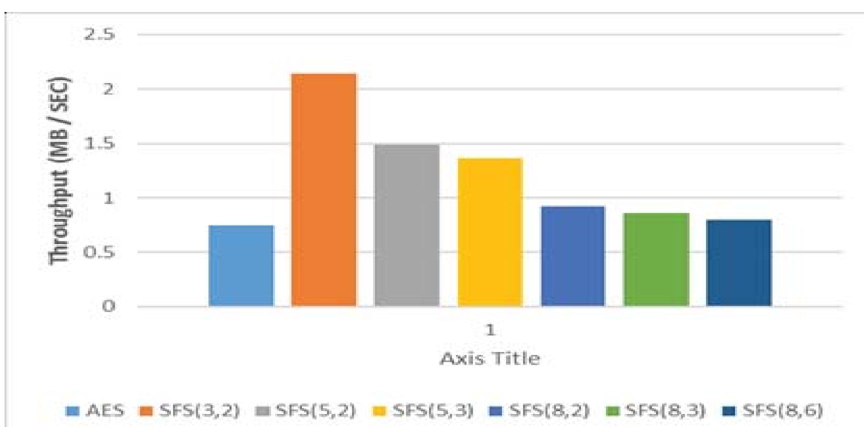

Figure A.20: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Image File Type of Different Sizes.

A.6 Text file formats

Table A.11: Sequential implementation of Creating the Shares and Encryption using AES of Text File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000807	0.3021	0.000907	0.001259	0.001076	0.001599	0.003731	0.005654
.txt	0.005672	0.240221	0.004316	0.005555	0.006091	0.011364	0.008621	0.00868
.txt	0.015554	0.266474	0.011712	0.019246	0.018859	0.026282	0.031968	0.058434
.txt	0.704043	1.886232	0.463386	0.695211	0.740363	1.076405	1.05143	1.130196
.txt	5.26441	14.73039	4.513525	6.122312	6.123311	8.885614	8.992157	9.443522
.txt	24.19914	61.58744	18.59172	27.063	27.74973	37.38928	40.03352	42.12983
.txt	36.85302	95.29174	28.9812	44.12774	44.14673	64.01463	65.61778	68.83184
Sum=	67.04264	174.3046	52.56676	78.03432	78.78616	111.4052	115.7392	121.6082
Throughput (MB/ Sec)		0.384629	1.275381	0.859143	0.850944	0.601791	0.579256	0.551301

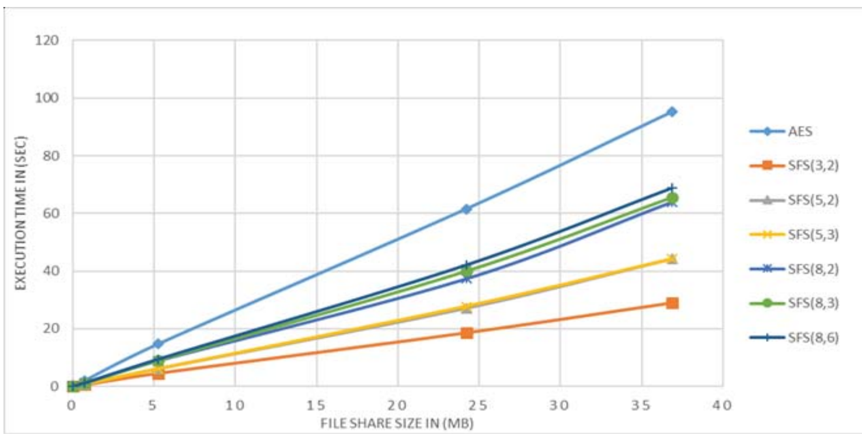

Figure A.21: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Text File Type of Different Sizes.

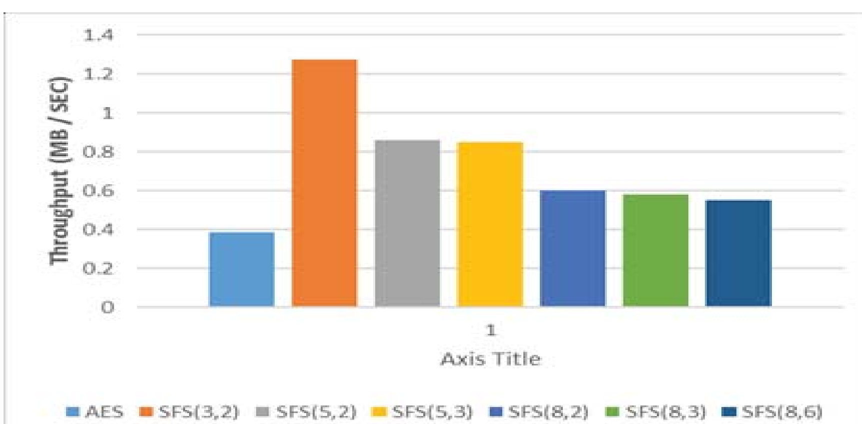

Figure A.22: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Text File Type of Different Sizes.

Table A.12: Parallel implementation of Creating the Shares and Encryption using AES of Text File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000807	0.211733	0.131485	0.007159	0.007605	0.018394	0.008555	0.010492
.txt	0.005672	0.215084	0.012916	0.02575	0.022799	0.024164	0.034651	0.034829
.txt	0.015554	0.240679	0.021144	0.039163	0.043415	0.044661	0.082896	0.079835
.txt	0.704043	1.28985	0.747303	1.21432	1.323482	1.951733	2.098843	2.312144
.txt	5.26441	7.475327	3.001531	4.612706	4.905367	7.213662	7.574712	8.237012
.txt	24.19914	34.19009	12.497	19.27192	19.96731	27.13081	31.796	33.85541
.txt	36.85302	52.1411	19.66458	30.03736	32.18727	40.80034	47.78314	49.20029
sum=	67.04264	95.76387	36.07596	55.20838	58.45725	77.18376	89.37879	93.73001
Throughput (MB / SEC)		0.700083	1.858374	1.214356	1.146866	0.868611	0.750096	0.715274

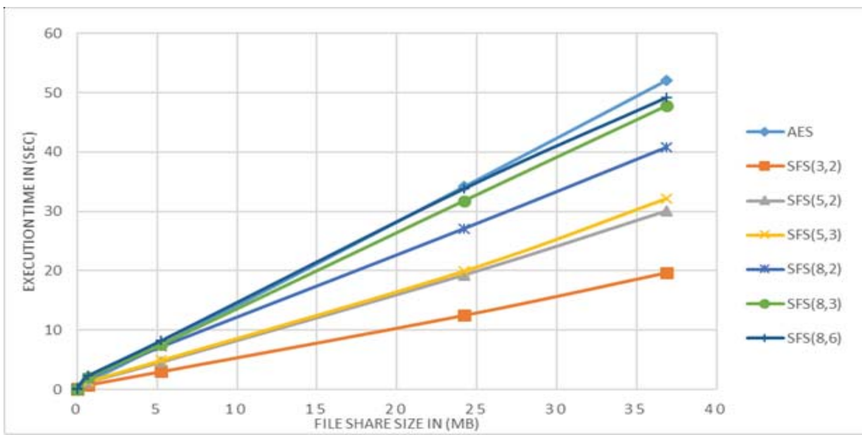

Figure A.23: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Text File Type of Different Sizes.

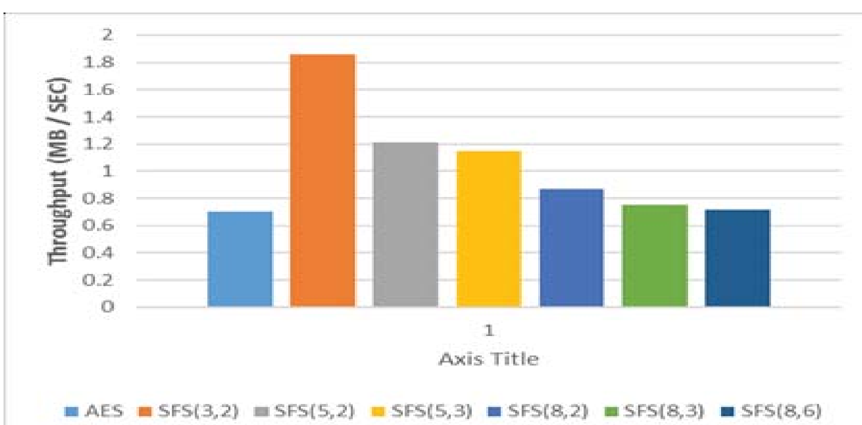

Figure A.24: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Text File Type of Different Sizes.

A.7 Video file formats

Table A.13: Sequential implementation of Creating the Shares and Encryption using AES of Video File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ts	0.000669	0.244085	0.001178	0.001503	0.001259	0.002193	0.002123	0.001474
.ts	0.002464	0.240815	0.003307	0.00302	0.003326	0.003607	0.004225	0.003958
.flv	1.387849	4.040784	1.046781	1.534642	1.623827	2.305611	2.38564	2.532559
.MKV	1.405831	3.968854	1.051176	1.584132	1.676247	2.10116	2.194908	2.515951
.MKV	1.755652	5.103292	1.35474	1.985873	2.108826	2.97388	3.065907	3.204205
.avi	12.81602	33.83223	9.34734	13.35568	13.94558	20.05421	22.12938	21.79297
.MKV	13.44737	35.68476	10.33202	15.34349	16.44393	20.44667	23.48308	24.17046
.flv	13.53714	37.96164	9.828212	15.37841	16.16786	23.3123	23.6713	24.78966
.MKV	13.70051	35.89625	10.36824	15.60317	18.22771	21.78381	22.36971	23.66909
.FLV	27.32828	72.26743	20.95478	33.17578	33.06738	47.63057	49.46713	51.30326
.MKV	27.38996	73.46028	20.85068	32.78063	32.94681	48.04299	50.07635	52.11477
.MKV	66.155	169.2735	50.25245	77.80298	80.26006	112.9988	116.1001	121.3827
.mov	119.4465	301.5958	88.24741	139.5351	139.4253	201.1956	204.3368	216.9755
.MKV	130.4061	330.2066	97.16172	150.426	150.0391	218.0424	224.9972	237.521
.avi	137.9018	346.0712	103.6173	159.1928	162.312	227.6705	229.6781	248.0756
Sum=	566.6811	1449.848	424.4173	657.7033	668.2493	948.5644	973.9619	1030.053
Throughput (MB/ Sec)		0.390856	1.335198	0.861606	0.848009	0.597409	0.581831	0.550147

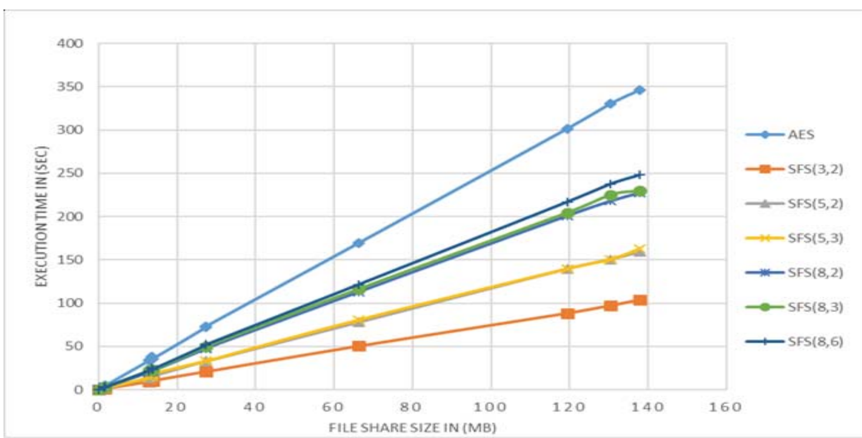

Figure A.25: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Video File Type of Different Sizes.

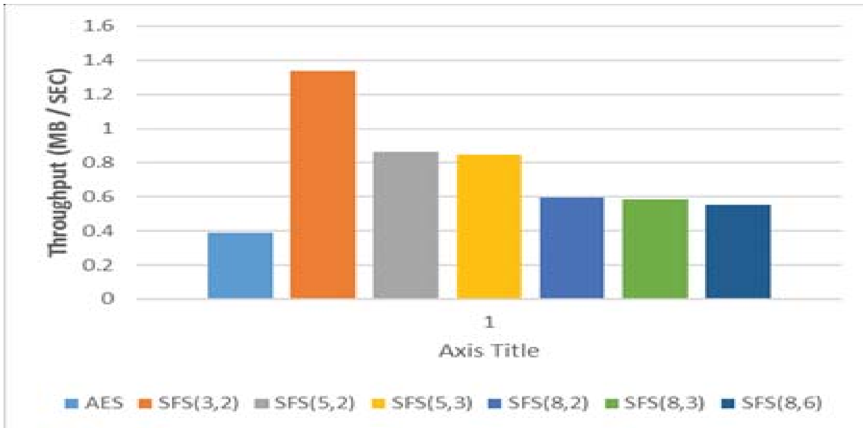

Figure A.26: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Video File Type of Different Sizes.

Table A.14: Parallel implementation of Creating the Shares and Encryption using AES of Video File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ts	0.000669	0.184383	0.006452	0.006969	0.014137	0.011379	0.048981	0.006647
.ts	0.002464	0.18373	0.007958	0.009599	0.00968	0.008578	0.009171	0.017234
.flv	1.387849	2.117553	0.858279	1.637039	1.704443	2.632385	2.559958	2.930011
.MKV	1.405831	1.964219	0.980868	1.700778	1.681489	2.695033	2.617101	2.906916
.MKV	1.755652	2.722497	1.175957	1.945766	2.017536	3.072792	3.293765	3.567436
.avi	12.81602	18.53886	6.651224	5.76569	8.737804	13.8445	13.42963	16.61818
.MKV	13.44737	19.22737	7.691684	11.43127	11.87164	18.46591	18.69308	21.59098
.flv	13.53714	19.61065	6.990308	11.50166	12.1788	17.57379	18.13515	20.37754
.MKV	13.70051	16.3108	7.073713	9.952842	6.650318	13.36864	15.54856	17.73247
.FLV	27.32828	37.92532	14.3619	22.37964	23.89967	34.99093	36.71413	41.22584
.MKV	27.38996	38.08249	15.24409	22.45874	23.51405	36.55736	35.55556	42.92936
.MKV	66.155	90.40687	32.59527	50.32345	54.32381	75.53735	84.12947	88.63307
.mov	119.4465	158.1147	48.30215	84.94016	89.12784	135.2178	137.5684	160.1241
.MKV	130.4061	172.9231	46.20615	92.90914	99.15387	142.8584	148.366	170.9934
.avi	137.9018	180.4383	58.21084	98.67343	104.9238	152.4514	152.885	180.7646
sum=	566.6811	758.7508	246.3568	415.6362	439.8089	649.2862	669.554	770.4177
Throughput (MB / SEC)		0.746861	2.300245	1.363407	1.288471	0.872775	0.846356	0.73555

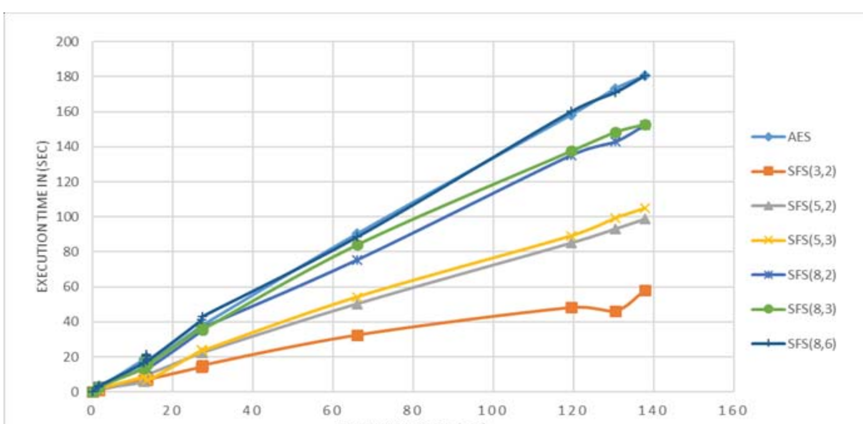

Figure A.27: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Video File Type of Different Sizes.

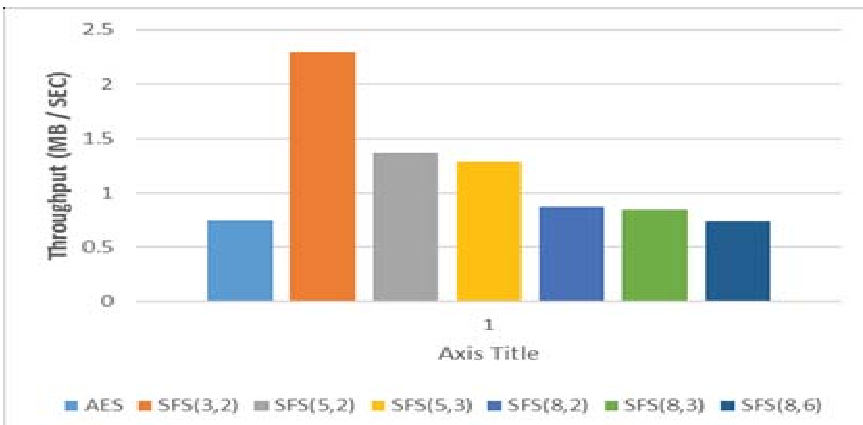

Figure A.28: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Video File Type of Different Sizes.

A.8 Archive file formats

Table A.15: Sequential implementation of Creating the Shares and Encryption using AES of Archive File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.tar	0.000533	0.26528	0.002938	0.001356	0.001222	0.00155	0.001737	0.001965
.jar	0.001053	0.267974	0.001208	0.001756	0.001858	0.001619	0.002203	0.002489
.zip	0.00113	0.27578	0.001148	0.001548	0.001528	0.002162	0.002129	0.002773
.tgz	0.001496	0.262906	0.001422	0.001934	0.002452	0.00279	0.003252	0.00285
.zip	0.007612	0.294287	0.006919	0.006833	0.008125	0.01071	0.01372	0.253649
.jar	0.011474	0.259132	0.00902	0.012432	0.01525	0.016147	0.019902	0.022005
.tgz	0.011759	0.288956	0.008129	0.144608	0.016327	0.016807	0.018508	0.021788
.jar	0.036816	0.246671	0.058294	0.037919	0.037177	0.06279	0.060714	0.059513
.tar	0.091844	0.264071	0.071464	0.096797	0.113511	0.159354	0.152423	0.19083
.bz2	0.131865	0.519244	0.100655	0.156497	0.188986	0.217895	0.226256	0.233533
.zip	1.318213	3.303679	1.229269	1.379277	1.381425	2.138486	2.058948	2.147091
.gz	1.334643	3.733418	0.981979	1.530097	1.498154	2.200661	2.054917	2.474532
.cab	1.344069	3.754602	1.069848	1.656337	1.598301	2.355648	2.457156	2.470547
.jar	1.35046	3.581442	1.044623	1.491795	1.603974	2.078314	2.229113	2.485373
.rar	1.353995	3.559951	0.971724	1.388622	1.45478	2.030451	2.327711	2.276004
.jar	12.62993	34.30476	9.904787	14.38281	15.01945	22.14223	22.61611	23.98629
.rar	13.02323	36.44472	10.54137	15.25329	15.28601	21.30421	23.16577	24.60699
.zip	13.36134	32.86407	10.40675	14.41442	15.32784	19.96134	23.66191	23.13263
.gz	13.39809	36.68574	10.54945	15.44803	15.6029	22.96284	24.11456	24.62229
.jar	20.97486	51.98819	16.01221	23.12955	22.91316	32.49338	35.17731	36.92568
.rar	27.44257	74.32415	21.85323	32.75942	32.74193	47.20672	48.4346	51.91759
.cab	27.68423	74.65109	22.01327	32.47095	32.878	47.44926	48.96709	51.68246
.zip	27.84394	70.46894	22.04262	30.0966	32.70848	41.89507	46.41074	47.93042
.jar	42.70904	109.7715	33.94955	49.91811	50.13963	70.26502	72.30565	78.90181
.cab	67.44812	172.3997	52.64786	78.25493	78.601	113.3462	116.799	121.426
.zip	67.87289	173.4575	53.85516	79.7104	79.21954	113.4751	118.0412	122.5157
.bz2	68.7001	177.0263	54.60911	79.58382	80.96236	110.7014	119.234	124.091
.rar	132.5344	335.4979	103.2131	153.1637	153.0636	212.3813	229.7249	236.8785
Sum=	542.6197	1400.762	427.1571	626.4939	632.3869	886.8794	940.2815	981.2624
Throughput (MB/ Sec)		0.387375	1.270305	0.866121	0.85805	0.61183	0.577082	0.552981

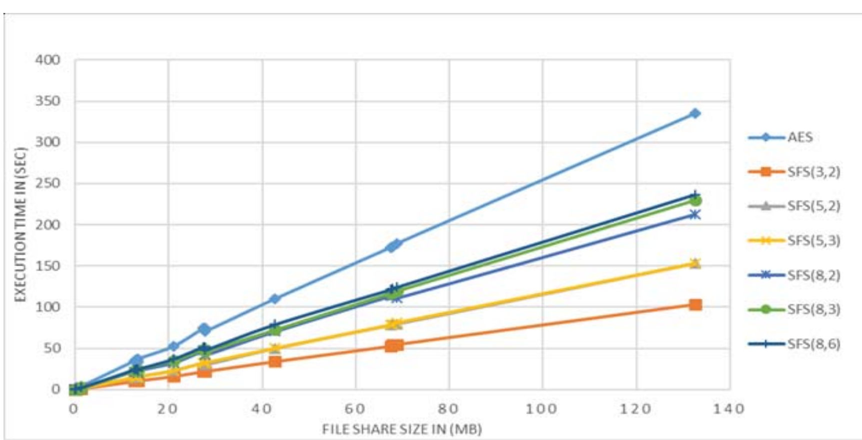

Figure A.29: Performance Comparison of the Sequential Execution Time of Creating the Shares and Encryption using AES of Archive File Type of Different Sizes.

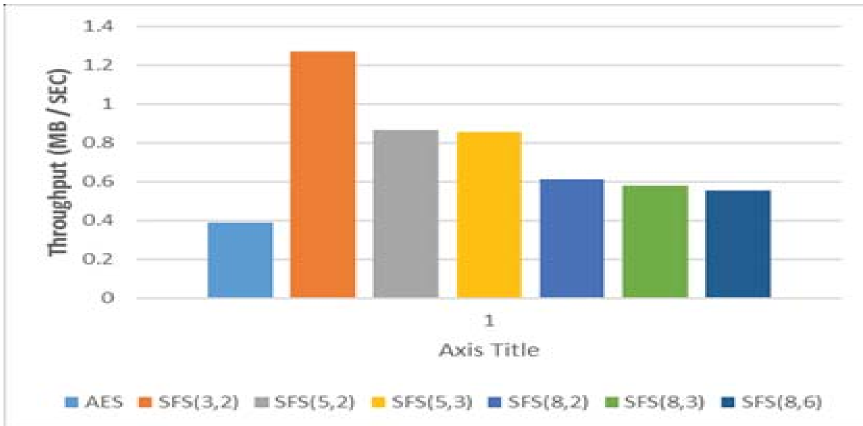

Figure A.30: Throughput of the Sequential Execution Time of Creating the Shares and Encryption using AES of Archive File Type of Different Sizes.

Table A.16: Parallel implementation of Creating the Shares and Encryption using AES of Archive File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.tar	0.000533	0.214641	0.007593	0.004151	0.003816	0.011969	0.007307	0.00558
.jar	0.001053	0.215522	0.005225	0.006241	0.004638	0.006157	0.028372	0.006388
.zip	0.00113	0.22302	0.003786	0.012842	0.007039	0.006367	0.014508	0.006079
.tgz	0.001496	0.221678	0.004651	0.006077	0.005275	0.00853	0.012755	0.027725
.zip	0.007612	0.210432	0.010132	0.015171	0.015669	0.022339	0.02575	0.030642
.jar	0.011474	0.181526	0.014006	0.035418	0.023381	0.032752	0.047877	0.045531
.tgz	0.011759	0.189121	0.014581	0.021087	0.022256	0.035389	0.053477	0.077518
.jar	0.036816	0.217075	0.039003	0.059119	0.061435	0.089891	0.117946	0.108884
.tar	0.091844	0.215591	0.089888	0.150298	0.18199	0.222595	0.338322	0.348689
.bz2	0.131865	0.426827	0.111829	0.226586	0.239959	0.321499	0.477082	0.401263
.zip	1.318213	1.929565	0.751393	1.211158	1.218222	1.651659	3.335875	1.934106
.gz	1.334643	1.817429	0.662164	1.201045	1.289681	1.681213	3.405802	2.073699
.cab	1.344069	1.827014	0.685123	1.242893	1.256804	1.689911	3.490201	1.901789
.jar	1.35046	2.00548	0.692658	1.294897	1.310784	1.665376	3.35232	1.932799
.rar	1.353995	1.747367	0.796415	1.222818	1.28243	1.690006	3.50681	1.970929
.jar	12.62993	18.03885	6.270601	10.11057	10.75763	15.47928	18.71159	17.4263
.rar	13.02323	18.62211	6.585283	10.67261	10.83543	15.8751	18.90561	17.93968
.zip	13.36134	18.50181	6.754231	8.36998	9.964911	16.24021	21.33709	18.41719
.gz	13.39809	19.24553	6.705059	11.28207	11.94678	16.52535	20.89908	18.89354
.jar	20.97486	27.60034	10.44766	15.14109	17.89576	25.37648	32.09092	29.32713
.rar	27.44257	38.59434	13.52581	23.25422	24.02039	34.46649	39.78756	38.25546
.cab	27.68423	39.14305	13.97467	23.08016	24.71458	33.99184	41.00914	39.08563
.zip	27.84394	36.26288	10.9331	19.32445	15.76131	33.5387	42.42107	39.27345
.jar	42.70904	58.51731	20.79835	33.12746	36.84034	52.40613	65.64318	60.59498
.cab	67.44812	93.78899	32.50352	52.56874	54.11802	82.6981	99.4618	96.85045
.zip	67.87289	90.89594	34.31373	53.63226	53.6111	84.62525	99.95392	96.91834
.bz2	68.7001	90.93116	35.36137	51.65609	54.87644	84.11406	102.6577	98.7098
.rar	132.5344	171.8773	55.05326	87.48828	95.14789	167.5214	195.2636	194.7235
sum=	542.6197	733.6619	257.1151	406.4178	427.414	671.994	816.3567	777.2871
Throughput (MB / SEC)		0.739605	2.110416	1.335128	1.269541	0.807477	0.664685	0.698094

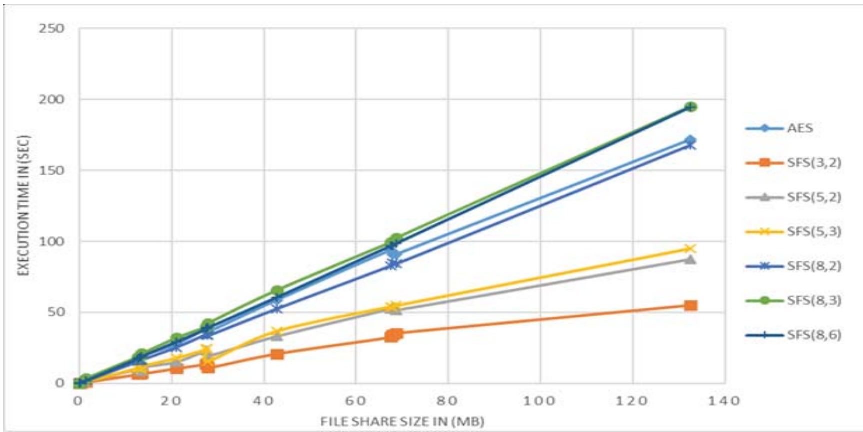

Figure A.31: Performance Comparison of the Parallel Execution Time of Creating the Shares and Encryption using AES of Archive File Type of Different Sizes.

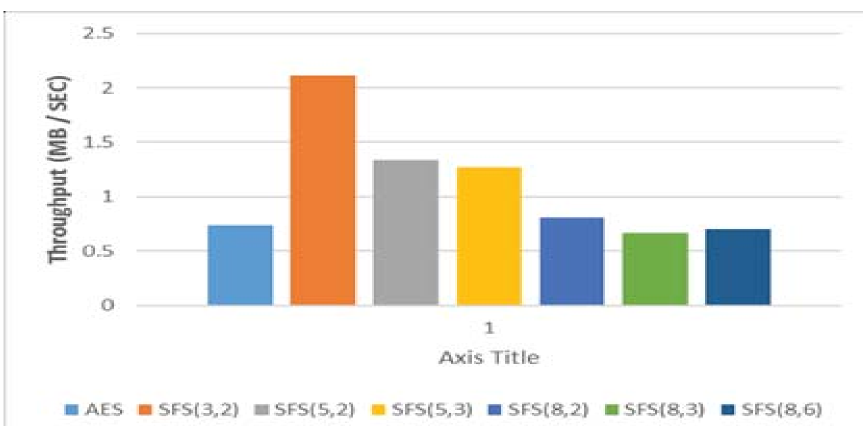

Figure A.32: Throughput of the Parallel Execution Time of Creating the Shares and Encryption using AES of Archive File Type of Different Sizes.

APPENDIX B

RECONSTRUCT THE SECRET

B.1 Audio file formats

Table B.1: Sequential implementation of Reconstruct the Secret and Decryption using AES of of Audio File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.wav	0.000881	0.266704	0.000443	0.000417	0.0005	0.000313	0.000519	0.000553
.mp3	0.001864	0.256475	0.001103	0.000907	0.001094	0.000643	0.001467	0.001275
.wav	0.007831	0.267861	0.004538	0.004608	0.003253	0.003006	0.005285	0.004204
.mp3	0.010812	0.257966	0.02061	0.004406	0.007316	0.003874	0.004677	0.006157
.ogg	0.012504	0.270494	0.011542	0.005148	0.006491	0.004426	0.009025	0.012107
.aac	0.012911	0.266263	0.00582	0.007003	0.008533	0.004425	0.007116	0.00758
.WAV	0.121142	0.526905	0.048723	0.058641	0.060934	0.052316	0.049689	0.081812
.MP3	0.130061	0.538363	0.042318	0.042656	0.063739	0.049198	0.052725	0.085387
.WAV	1.113782	2.961481	0.377997	0.369009	0.454035	0.392167	0.450319	0.832329
.ogg	1.198285	3.093872	0.440574	0.464943	0.533791	0.400703	0.586032	0.788632
.MP3	1.340312	3.758824	0.4492	0.448864	0.553225	0.531021	0.525743	0.817227
.mp3	13.39954	35.73699	4.792736	5.298058	4.908169	5.185837	4.962159	8.532472
.mp3	25.38321	68.24257	8.989292	8.864306	10.60341	8.861075	10.64335	16.60936
.mp3	63.33239	160.7493	21.82624	22.29331	26.41494	21.80935	27.12788	38.33652
.mp3	120.1339	305.4833	38.9719	38.8627	48.92713	39.26023	48.52871	69.03022
Sum=	226.1994	582.6774	75.98304	76.72498	92.54655	76.55858	92.9547	135.1458
Throughput (MB / Sec)		0.388207	2.976973	2.948185	2.444169	2.954593	2.433437	1.673743

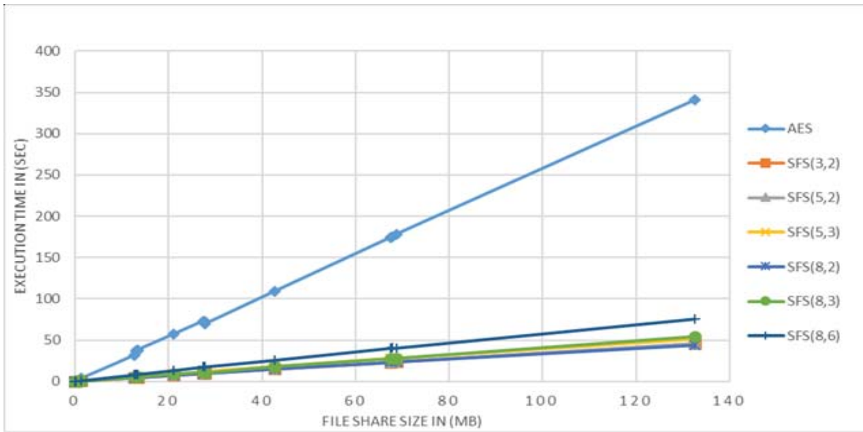

Figure B.1: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Audio File Type of Different Sizes.

Figure B.2: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Audio File Type of Different Sizes.

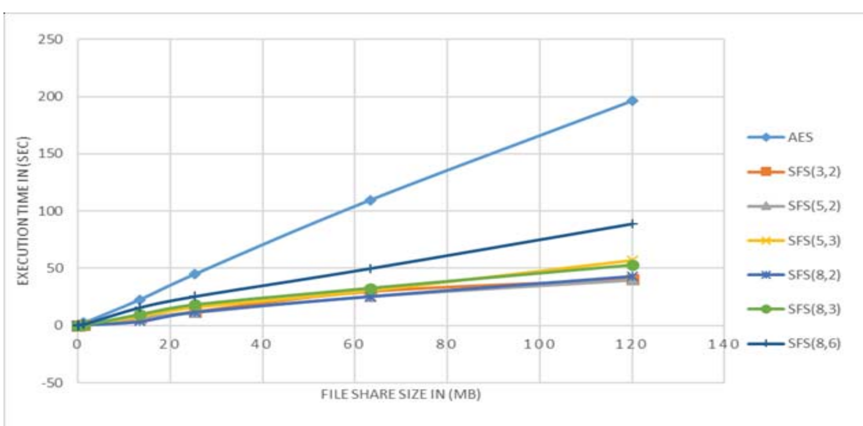

Figure B.3: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Audio File Type of Different Sizes.

Table B.2: Parallel implementation of Reconstruct the Secret and Decryption using AES of of Audio File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.wav	0.000881	0.212544	0.003598	0.005382	0.006298	0.007093	0.006465	0.005412
.mp3	0.001864	0.212733	0.004581	0.008444	0.008387	0.007231	0.007532	0.008529
.wav	0.007831	0.216555	0.010176	0.021127	0.021199	0.022377	0.023183	0.037674
.mp3	0.010812	0.221017	0.012955	0.020222	0.021914	0.029407	0.030511	0.065329
.ogg	0.012504	0.214901	0.014742	0.026708	0.027311	0.047649	0.035634	0.078343
.aac	0.012911	0.219033	0.016109	0.023323	0.029781	0.033199	0.036728	0.048151
.WAV	0.121142	0.425984	0.137705	0.202679	0.213944	0.304942	0.349024	0.35694
.MP3	0.130061	0.428282	0.132465	0.204694	0.266967	0.368537	0.342154	0.452074
.WAV	1.113782	1.691637	0.614169	1.098647	1.140247	1.624402	1.472002	1.74404
.ogg	1.198285	1.743921	0.609442	1.051533	1.223562	1.69918	1.561549	1.730436
.MP3	1.340312	1.856065	0.669877	1.225101	1.13647	1.946015	1.718602	2.043991
.mp3	13.39954	19.27124	6.715657	10.7606	6.412136	12.95901	17.18541	19.03588
.mp3	25.38321	36.33403	12.43381	21.13461	21.31677	32.9218	32.30493	36.29371
.mp3	63.33239	84.57047	30.49876	51.82066	53.35351	68.85662	83.14607	91.55714
.mp3	120.1339	155.721	54.54024	80.16999	88.11212	122.2988	159.0777	177.3213
sum=	226.1994	303.3394	106.4143	167.7737	173.2906	243.1262	297.2975	330.7789
Throughput (MB / SEC)		0.745698	2.125649	1.348241	1.305319	0.930379	0.760852	0.683839

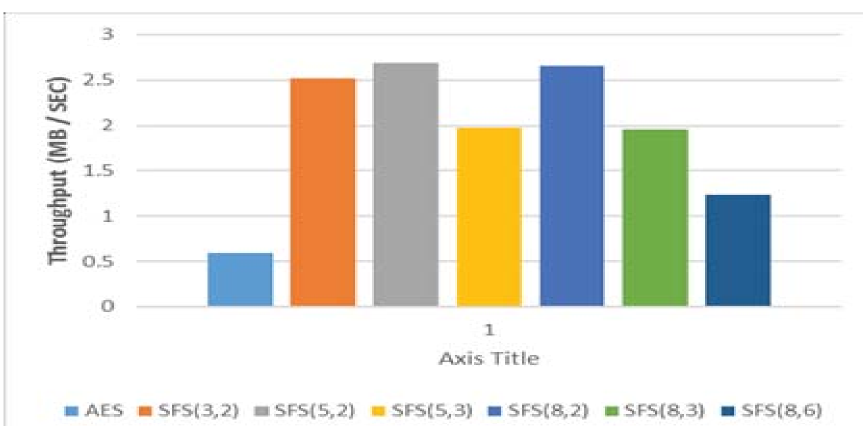

Figure B.4: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Audio File Type of Different Sizes.

B.2 Binary file formats

Table B.3: Sequential implementation of Reconstruct the Secret and Decryption using AES of of Binary File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bin	0.000469	0.231307	0.000193	0.000135	0.000538	0.000195	0.000174	0.000214
.bin	0.001416	0.23592	0.000753	0.000846	0.000447	0.001058	0.000656	0.001633
.bin	0.006151	0.258013	0.002014	0.002196	0.00348	0.004073	0.003453	0.005384
.bin	0.013671	0.254833	0.004164	0.004553	0.005551	0.007807	0.00676	0.013598
.BIN	0.099341	0.558249	0.037402	0.082877	0.047695	0.043811	0.055286	0.05561
.bin	0.136086	0.472206	0.057019	0.04548	0.046658	0.048788	0.048984	0.082861
.BIN	11.06026	29.70218	3.88771	3.968977	4.104478	4.04534	4.67626	7.071209
.bin	68.75715	177.2776	23.58591	24.01403	28.0626	23.82384	27.83008	40.60998
Sum=	80.07454	208.9903	27.57517	28.11909	32.27145	27.97491	32.62165	47.84049
Throughput (MB / Sec)		0.38315	2.903864	2.847693	2.481281	2.86237	2.454644	1.673782

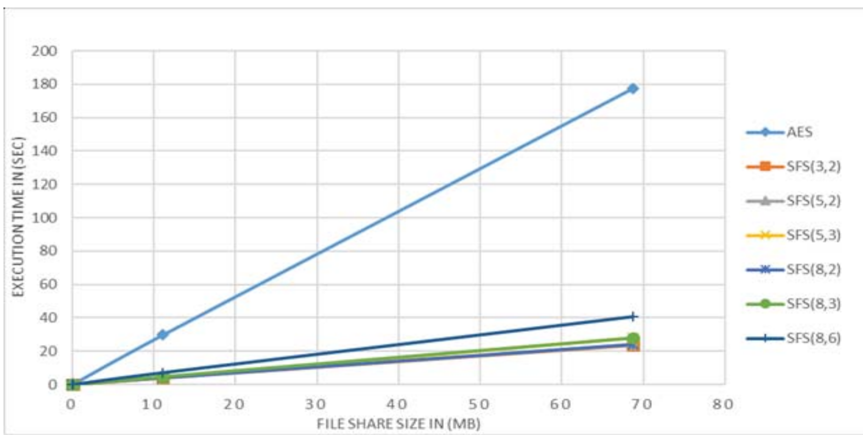

Figure B.5: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Binary File Type of Different Sizes.

Table B.4: Parallel implementation of Reconstruct the Secret and Decryption using AES of of Binary File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bin	0.000469	0.210581	0.021321	0.005805	0.006655	0.010671	0.007664	0.015982
.bin	0.001416	0.212847	0.118866	0.010359	0.011488	0.00953	0.011657	0.010061
.bin	0.006151	0.220952	0.012162	0.016129	0.017953	0.023464	0.02309	0.023072
.bin	0.013671	0.215129	0.020866	0.027378	0.028539	0.391576	0.05124	0.078118
.BIN	0.099341	0.427601	0.119191	0.165678	0.195634	0.259694	0.278518	0.309176
.bin	0.136086	0.441484	0.139459	0.244893	0.249088	0.436146	0.428882	0.412394
.BIN	11.06026	15.06531	5.601418	9.278003	9.897554	13.57165	15.24423	15.84823
.bin	68.75715	93.41486	35.86503	53.97019	57.10886	79.08547	77.76194	86.11326
sum=	80.07454	110.2088	41.89831	63.71844	67.51577	93.78821	93.80722	102.8103
Throughput (MB / SEC)		0.726571	1.911164	1.256693	1.186012	0.85378	0.853607	0.778857

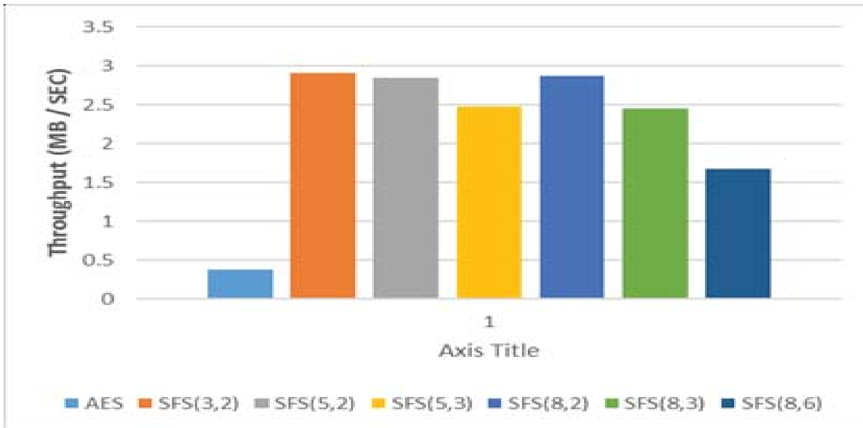

Figure B.6: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Binary File Type of Different Sizes.

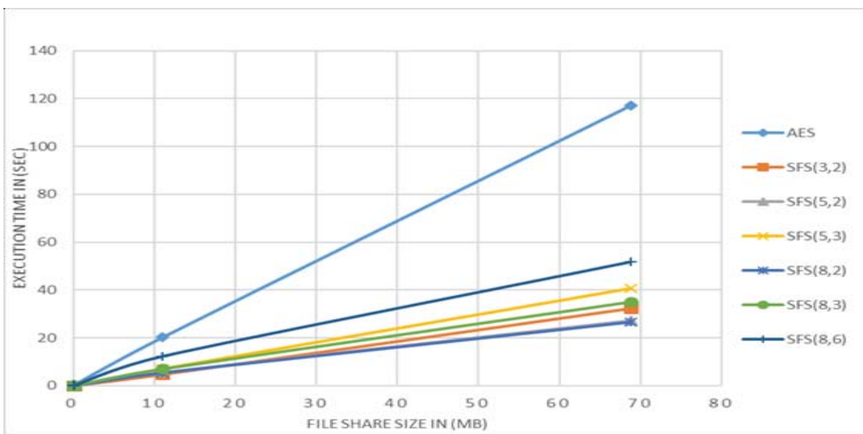

Figure B.7: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Binary File Type of Different Sizes.

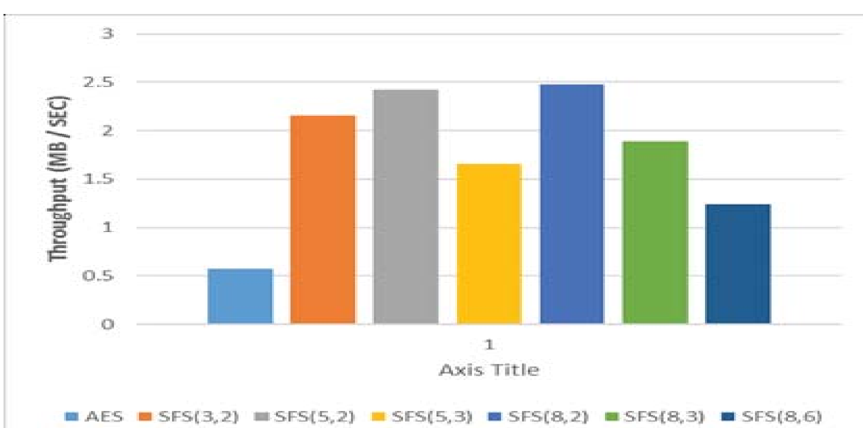

Figure B.8: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Binary File Type of Different Sizes.

B.3 Document file formats

Table B.5: Sequential implementation of Reconstruct the Secret and Decryption using AES of of Document File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.rtf	0.000682	0.225538	0.001212	0.000346	0.000341	0.000426	0.000215	0.000336
.doc	0.00391	0.275867	0.004319	0.001516	0.001868	0.004457	0.003267	0.002119
.rtf	0.00412	0.239225	0.001971	0.002423	0.001517	0.001594	0.001573	0.002432
.xls	0.015959	0.247336	0.007446	0.005097	0.019087	0.01208	0.006242	0.007694
.doc	0.024937	0.254311	0.012096	0.023319	0.025143	0.008627	0.01199	0.014774
.ppt	0.029634	0.277894	0.010971	0.011687	0.010984	0.012558	0.019906	0.039508
.rtf	0.176815	0.491175	0.076853	0.071089	0.067288	0.055648	0.058276	0.101099
.xls	0.351154	0.963935	0.140702	0.131818	0.120932	0.148874	0.122447	0.203602
.doc	0.514131	1.417535	0.144952	0.183496	0.202214	0.178158	0.179373	0.263059
.ppt	0.977554	2.600946	0.277173	0.309414	0.3296	0.319692	0.324887	0.600683
.rtf	2.758348	6.918239	1.004094	0.930535	1.009524	0.987415	0.982597	1.482225
.doc	4.955577	14.27966	1.744648	1.775981	1.850111	1.696236	2.116852	3.502684
.xls	7.267596	20.4323	2.781647	2.590862	3.078215	2.510083	3.026076	4.610353
.doc	20.19167	56.3689	7.099734	7.02722	8.546578	7.22949	8.406233	12.91145
Sum=	37.27208	104.9929	13.30782	13.0648	15.2634	13.16534	15.25993	23.74202
Throughput (MB / Sec)		0.354996	2.800766	2.852863	2.441925	2.831077	2.44248	1.569879

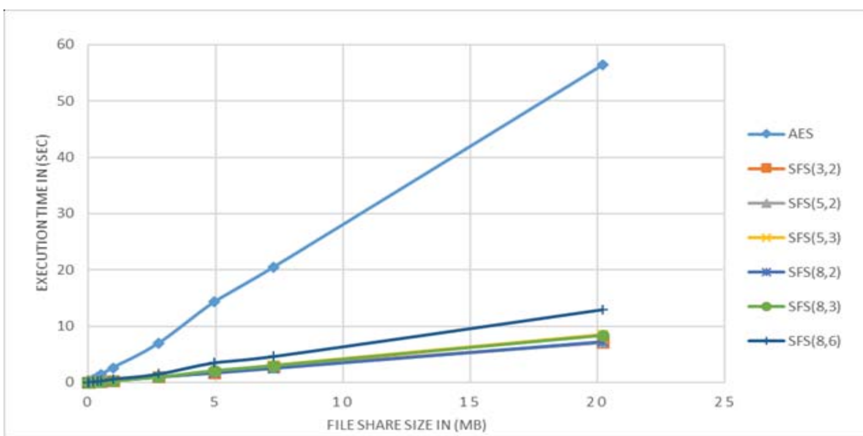

Figure B.9: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Document File Type of Different Sizes.

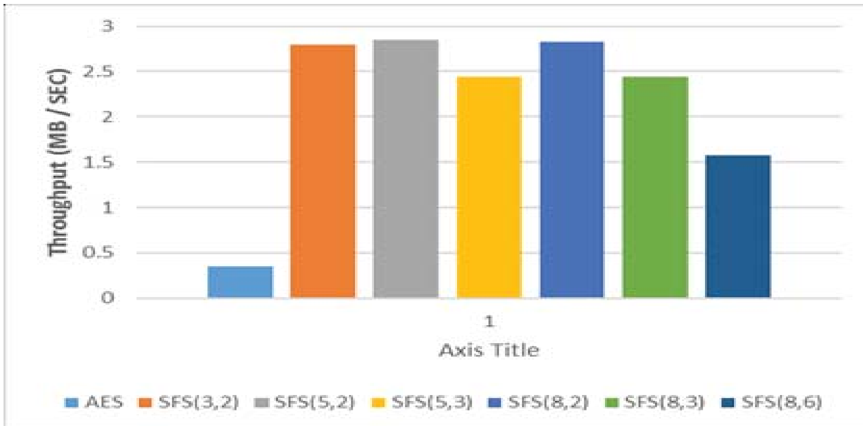

Figure B.10: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Document File Type of Different Sizes.

Table B.6: Parallel implementation of Reconstruct the Secret and Decryption using AES of of Document File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.rtf	0.000682	0.22177	0.017078	0.007246	0.006886	0.008299	0.022362	0.022258
.doc	0.00391	0.216435	0.014056	0.012681	0.019312	0.066418	0.024594	0.053482
.rtf	0.00412	0.211404	0.010761	0.018826	0.412218	0.017453	0.024363	0.06178
.xls	0.015959	0.21169	0.022814	0.041189	0.032128	0.056442	0.055528	0.09306
.doc	0.024937	0.219155	0.031692	0.05265	0.068825	0.079153	0.117189	0.106038
.ppt	0.029634	0.214551	0.051261	0.053423	0.0951	0.082261	0.085379	0.106089
.rtf	0.176815	0.42811	0.192223	0.301592	0.626554	0.489394	0.48686	0.514473
.xls	0.351154	0.882336	0.363713	0.668518	0.615499	0.999196	1.097724	1.153876
.doc	0.514131	0.96838	0.564774	0.883676	0.952018	1.718909	1.585221	1.494305
.ppt	0.977554	1.530761	0.62087	1.272644	1.377048	2.014145	2.087577	2.151535
.rtf	2.758348	3.683754	1.619502	2.628539	2.755792	4.117527	4.279383	4.31965
.doc	4.955577	6.791374	2.799002	4.622872	4.646973	6.486644	6.880884	7.427726
.xls	7.267596	10.33937	4.087246	6.353829	6.703177	9.499622	10.14086	10.54591
.doc	20.19167	29.22544	10.56182	17.06578	17.50842	25.83571	27.12501	29.36498
Sum=	37.27208	55.14453	20.95682	33.98346	35.81995	51.47117	54.01293	57.41516
Throughput (MB / SEC)		0.675898	1.778519	1.096771	1.04054	0.724135	0.690059	0.649168

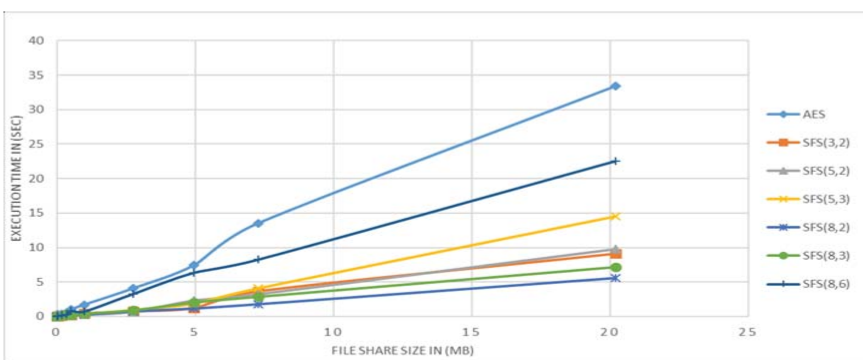

Figure B.11: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Document File Type of Different Sizes.

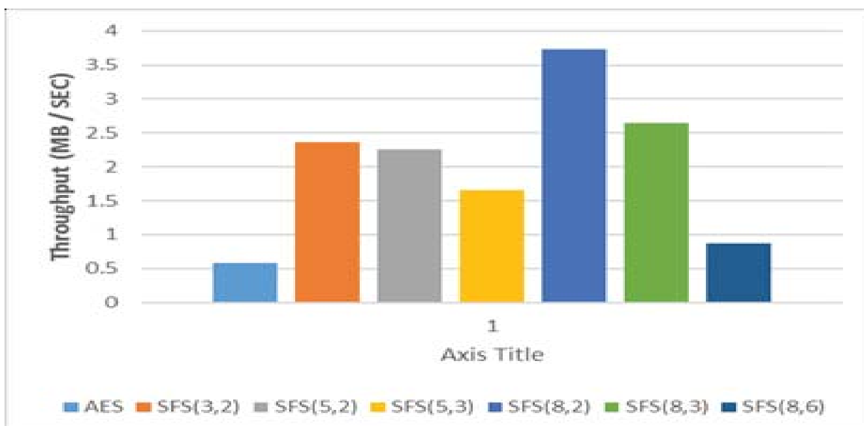

Figure B.12: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Document File Type of Different Sizes.

B.4 Executable file formats

Table B.7: Sequential implementation of Reconstruct the Secret and Decryption using AES of of Executable File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bat	0.000713	0.267205	0.00019	0.000734	0.000609	0.000342	0.000678	0.00091
.dll	0.001415	0.294345	0.000541	0.002589	0.00189	0.000939	0.00133	0.001699
.exe	0.001495	0.25951	0.000701	0.001147	0.001179	0.001019	0.0013	0.001602
.dll	0.005783	0.230387	0.004567	0.002015	0.00836	0.003027	0.002239	0.003445
.exe	0.006152	0.235662	0.002984	0.007224	0.003097	0.002882	0.002354	0.010557
.dll	0.04107	0.236151	0.031733	0.019597	0.027034	0.013004	0.021453	0.030865
.exe	0.066244	0.247727	0.024079	0.022347	0.025516	0.037111	0.047553	0.059116
.dll	0.248048	0.735027	0.085684	0.102072	0.374102	0.078061	0.093783	0.169503
.exe	1.223361	3.527363	0.425338	0.43445	0.531791	0.472638	0.484404	0.880089
.dll	2.333558	6.660625	0.815485	0.670252	0.88138	0.744838	0.882904	1.271991
.dll	10.86065	29.92286	4.126446	3.888614	4.616005	4.197897	4.653631	6.714568
.exe	26.83663	70.83018	9.434205	9.388177	11.34905	9.38265	11.49654	16.97195
.exe	67.34037	174.5782	23.24232	23.43106	27.73457	23.2306	27.72915	39.44042
.exe	135.1324	346.2312	44.4568	47.18127	54.64361	43.81838	56.00021	76.17412
Sum=	244.0979	634.2564	82.65107	85.15155	100.1982	81.98339	101.4175	141.7308
Throughput (MB / Sec)		0.384857	2.953354	2.866629	2.436151	2.977407	2.406861	1.722264

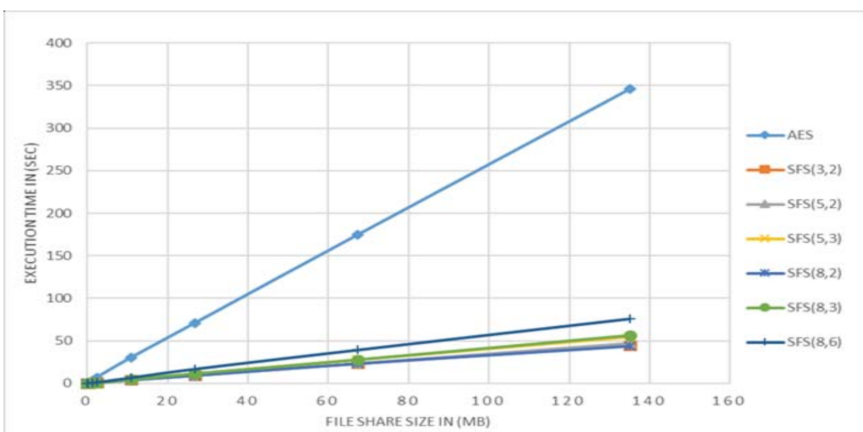

Figure B.13: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Executable File Type of Different Sizes.

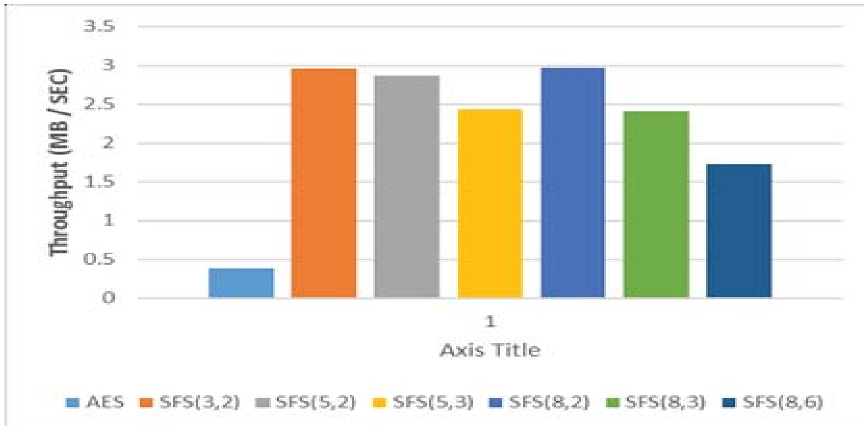

Figure B.14: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Executable File Type of Different Sizes.

Table B.8: Parallel implementation of Reconstruct the Secret and Decryption using AES of of Executable File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.bat	0.000713	0.213739	0.003598	0.003121	0.005068	0.078625	0.005974	0.004703
.dll	0.001415	0.20849	0.004408	0.004694	0.004798	0.007484	0.007125	0.011655
.exe	0.001495	0.215081	0.004252	0.006489	0.007339	0.009656	0.009148	0.013773
.dll	0.005783	0.207518	0.009643	0.01181	0.012659	0.018706	0.026482	0.020472
.exe	0.006152	0.211504	0.012021	0.020524	0.016289	0.021564	0.018982	0.02145
.dll	0.04107	0.226455	0.042681	0.065367	0.114305	0.123692	0.121251	0.127043
.exe	0.066244	0.219964	0.070777	0.105348	0.11523	0.205284	0.169415	0.203267
.dll	0.248048	0.640215	0.247825	0.466582	0.425578	0.800424	0.644667	0.709498
.exe	1.223361	1.779023	0.631442	1.099701	1.237363	1.933157	1.621186	1.81041
.dll	2.333558	3.223319	1.211213	2.068216	1.423098	2.955583	3.199935	3.633691
.dll	10.86065	15.46298	5.453487	8.68426	9.285773	14.61379	13.59542	15.15177
.exe	26.83663	37.99645	13.87555	22.55861	23.30909	35.69784	33.9087	37.81834
.exe	67.34037	89.63716	33.79753	52.88443	55.95502	90.8836	86.52061	100.0499
.exe	135.1324	176.3424	61.02184	90.86358	92.96739	181.6937	176.8927	202.8125
Sum=	244.0979	326.5843	116.3863	178.8427	184.879	329.0431	316.7416	362.3885
Throughput (MB / SEC)		0.747427	2.097308	1.364874	1.320312	0.741842	0.770653	0.673581

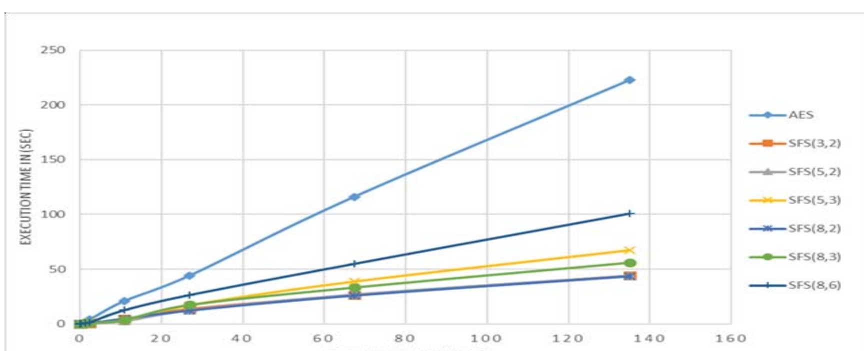

Figure B.15: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Executable File Type of Different Sizes.

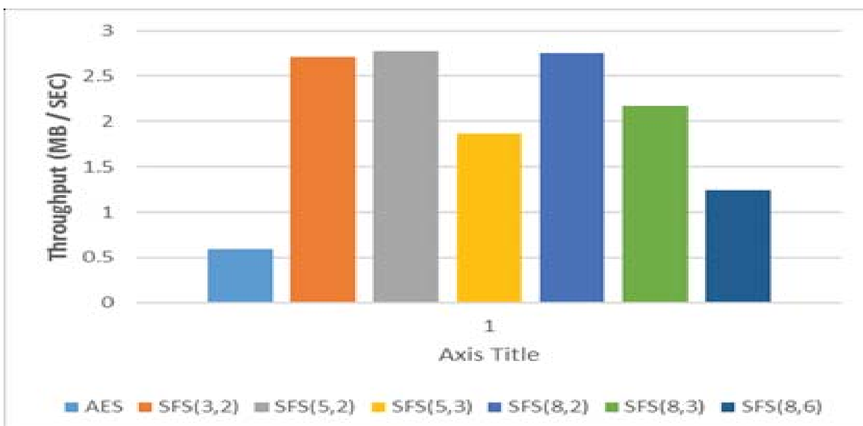

Figure B.16: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Executable File Type of Different Sizes.

B.5 Image file formats

Table B.9: Sequential implementation of Reconstruct the Secret and Decryption using AES of of Image File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ico	0.000775	0.283962	0.000265	0.000201	0.000544	0.000268	0.000361	0.000711
.bmp	0.000807	0.228693	0.000615	0.000245	0.000887	0.00027	0.000355	0.000685
.jpg	0.001109	0.29788	0.000632	0.000449	0.000362	0.000925	0.000475	0.001681
.gif	0.001202	0.231668	0.000434	0.000503	0.000573	0.000902	0.002278	0.000974
.png	0.001417	0.230551	0.001091	0.001113	0.000729	0.000496	0.00045	0.000751
.bmp	0.003551	0.27284	0.001194	0.001208	0.002184	0.001442	0.001429	0.003804
.jpg	0.005877	0.22911	0.002541	0.001711	0.003166	0.001994	0.002114	0.005624
.jpg	0.006936	0.24059	0.002495	0.002148	0.008187	0.004039	0.002526	0.005217
.jpg	0.012669	0.230585	0.006608	0.003806	0.005124	0.004377	0.007174	0.008402
.png	0.012883	0.229307	0.00437	0.011451	0.004325	0.004254	0.008136	0.012967
.gif	0.013248	0.27627	0.004409	0.006358	0.007451	0.00716	0.00539	0.008693
.png	0.013301	0.228764	0.006409	0.003997	0.017911	0.005525	0.00459	0.012909
.bmp	0.03936	0.229886	0.013184	0.010818	0.017069	0.012573	0.014728	0.03494
.jpg	0.114242	0.463345	0.041667	0.04991	0.038088	0.048161	0.039898	0.083772
.png	0.129358	0.514058	0.040394	0.040218	0.044032	0.041608	0.050842	0.098939
.jpg	0.129999	0.501337	0.051583	0.042926	0.062498	0.052854	0.043001	0.072227
.gif	0.130485	0.470558	0.18517	0.036797	0.041958	0.051863	0.053309	0.100515
.jpg	0.278825	0.713863	0.092266	0.075158	0.123232	0.1018	0.098379	0.170692
.gif	1.343994	3.319946	0.488396	0.378363	0.479867	0.433836	0.498907	0.847751
.png	1.365189	3.572244	0.508978	0.412667	0.472038	0.472825	0.509048	0.925527
.jpg	1.377728	4.013582	0.400682	0.43411	0.539797	0.453008	0.498119	0.85353
.jpg	13.55293	36.18946	4.860618	4.827061	5.673581	4.793114	6.023042	8.663898
.jpg	27.60827	70.86314	9.601606	10.15581	12.16054	10.27412	11.96856	17.80969
.jpg	67.6302	174.9266	23.54142	23.51935	28.36164	23.52656	27.39076	41.11191
.jpg	114.8891	293.3882	40.56186	39.08564	45.50476	37.82236	45.92455	66.62804
Sum=	228.6635	592.1464	80.41888	79.10201	93.57054	78.11633	93.14843	137.4638
Throughput (MB / Sec)		0.38616	2.843405	2.890741	2.443755	2.927217	2.454829	1.663444

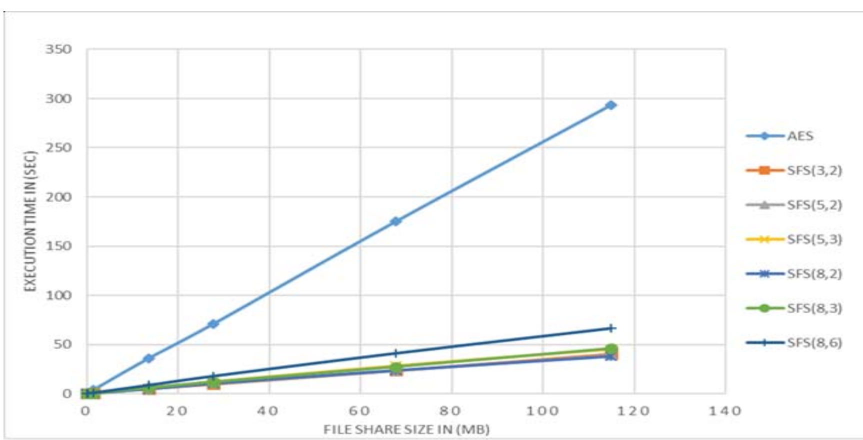

Figure B.17: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Image File Type of Different Sizes.

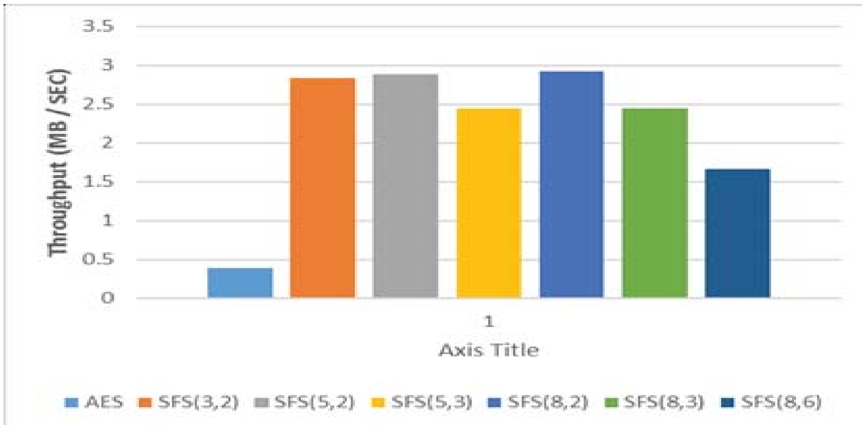

Figure B.18: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Image File Type of Different Sizes.

Table B.10: Parallel implementation of Reconstruct the Secret and Decryption using AES of of Image File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ico	0.000775	0.207467	0.008654	0.007275	0.007688	0.009064	0.01013	0.015162
.bmp	0.000807	0.209836	0.007422	0.02198	0.009449	0.081768	0.0107	0.015228
.jpg	0.001109	0.210709	0.009503	0.011064	0.007492	0.009034	0.009331	0.008268
.gif	0.001202	0.212189	0.023548	0.011473	0.013421	0.009382	0.028617	0.025267
.png	0.001417	0.206505	0.007472	0.008562	0.008097	0.008905	0.012053	0.043258
.bmp	0.003551	0.208624	0.010847	0.030567	0.014178	0.021826	0.026642	0.041532
.jpg	0.005877	0.212804	0.011722	0.035109	0.016442	0.022211	0.024464	0.023615
.jpg	0.006936	0.210046	0.020734	0.02312	0.018069	0.032686	0.034866	0.026817
.jpg	0.012669	0.209779	0.033818	0.046068	0.029402	0.045234	0.039285	0.085553
.png	0.012883	0.227681	0.022824	0.146297	0.027478	0.040673	0.099511	0.061354
.gif	0.013248	0.212815	0.085903	0.041822	0.048648	0.058377	0.063148	0.094535
.png	0.013301	0.21411	0.021279	0.027278	0.307864	0.057364	0.042239	0.077219
.bmp	0.03936	0.212838	0.050551	0.089628	0.08981	0.119262	0.126191	0.138153
.jpg	0.114242	0.430414	0.11745	0.237439	0.204744	0.399508	0.446457	0.519105
.png	0.129358	0.424773	0.162726	0.246521	0.275861	0.347335	0.374399	0.383393
.jpg	0.129999	0.421199	0.144001	0.230935	0.244129	0.383875	0.364692	0.376488
.gif	0.130485	0.443856	0.179598	0.213325	0.261551	0.340514	0.510153	0.412996
.jpg	0.278825	0.643629	0.401487	0.51326	0.513809	0.803113	0.784673	1.094782
.gif	1.343994	1.840062	0.914034	1.472087	1.642216	2.292468	2.38604	2.502315
.png	1.365189	1.935442	0.983557	1.515872	1.620416	2.644309	2.433163	2.525222
.jpg	1.377728	1.960808	1.223339	0.897234	0.925212	2.513832	2.480095	1.631172
.jpg	13.55293	19.25334	7.064779	11.01736	11.43894	17.55378	18.02951	19.2361
.jpg	27.60827	36.63105	12.92628	12.59721	15.37501	27.86897	30.84277	32.11911
.jpg	67.6302	89.56975	33.27388	47.52559	50.4043	72.29256	76.56782	81.29878
.jpg	114.8891	150.7191	49.12106	76.95854	84.14759	119.1356	130.8456	144.5832
sum=	228.6635	307.0288	106.8265	153.9256	167.6518	247.0917	266.5926	287.3386
Throughput (MB / SEC)		0.744762	2.140513	1.485545	1.363919	0.925419	0.857726	0.795798

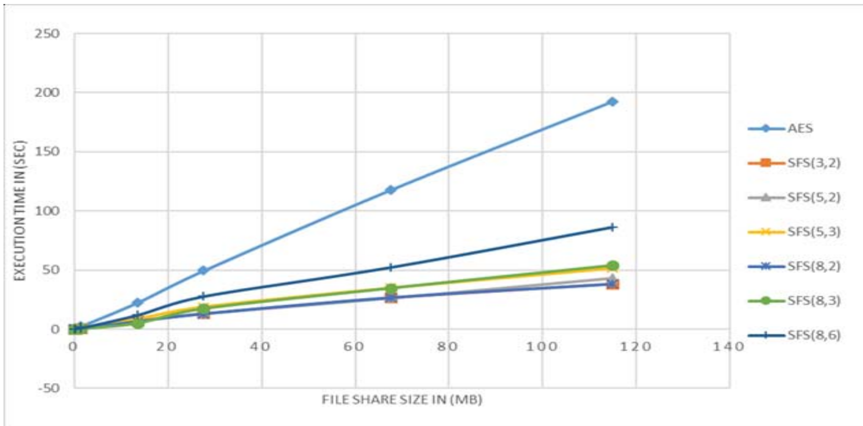

Figure B.19: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Image File Type of Different Sizes.

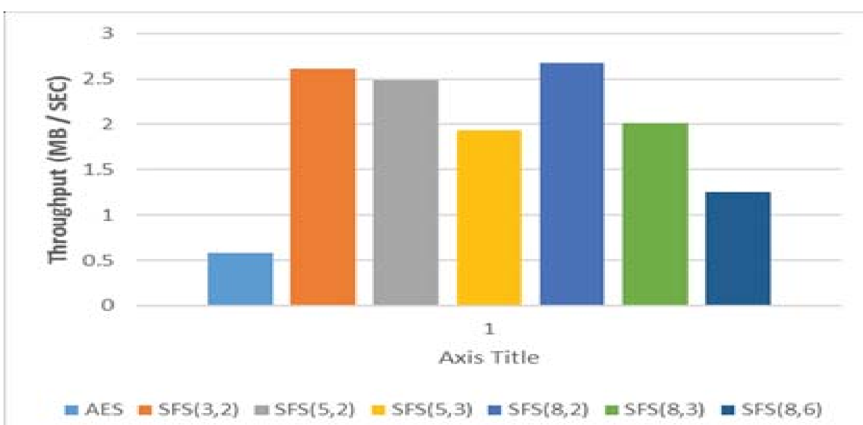

Figure B.20: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Image File Type of Different Sizes.

B.6 Text file formats

Table B.11: Sequential implementation of Reconstruct the Secret and Decryption using AES of of Text File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000807	0.319752	0.000237	0.000377	0.000279	0.000213	0.000809	0.000408
.txt	0.005672	0.308658	0.002759	0.002635	0.003712	0.007858	0.003228	0.006105
.txt	0.015554	0.262268	0.009543	0.006333	0.006414	0.005294	0.010244	0.008673
.txt	0.704043	2.09437	0.27176	0.238526	0.285303	0.24012	0.322035	0.452985
.txt	5.26441	14.86831	1.858954	1.894026	2.348347	1.870894	2.312914	3.323454
.txt	24.19914	65.71618	8.514996	8.491031	10.19678	8.754758	10.1514	15.47788
.txt	36.85302	94.2097	13.21783	12.94392	15.66598	13.10733	15.50164	22.95322
Sum=	67.04264	177.7792	23.87608	23.57685	28.50681	23.98647	28.30228	42.22272
Throughput (MB / Sec)		0.377112	2.807941	2.84358	2.351811	2.79502	2.368807	1.587833

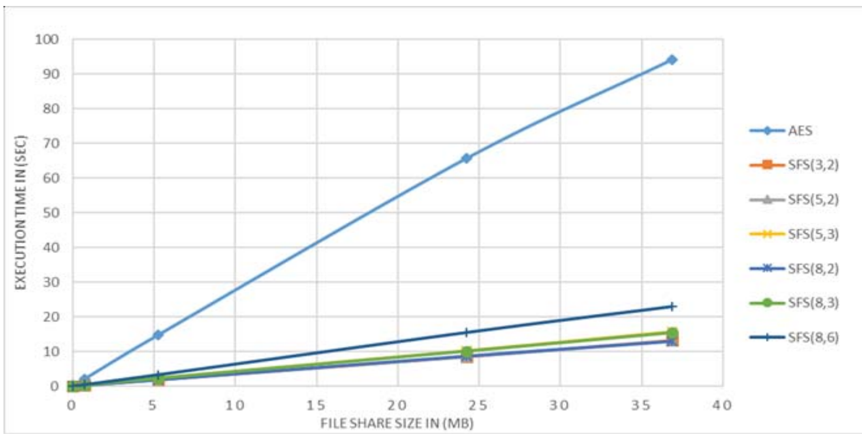

Figure B.21: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Text File Type of Different Sizes.

Table B.12: Parallel implementation of Reconstruct the Secret and Decryption using AES of of Text File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.txt	0.000807	0.211733	0.131485	0.007159	0.007605	0.018394	0.008555	0.010492
.txt	0.005672	0.215084	0.012916	0.02575	0.022799	0.024164	0.034651	0.034829
.txt	0.015554	0.240679	0.021144	0.039163	0.043415	0.044661	0.082896	0.079835
.txt	0.704043	1.28985	0.747303	1.21432	1.323482	1.951733	2.098843	2.312144
.txt	5.26441	7.475327	3.001531	4.612706	4.905367	7.213662	7.574712	8.237012
.txt	24.19914	34.19009	12.497	19.27192	19.96731	27.13081	31.796	33.85541
.txt	36.85302	52.1411	19.66458	30.03736	32.18727	40.80034	47.78314	49.20029
sum=	67.04264	95.76387	36.07596	55.20838	58.45725	77.18376	89.37879	93.73001
Throughput (MB / SEC)		0.700083	1.858374	1.214356	1.146866	0.868611	0.750096	0.715274

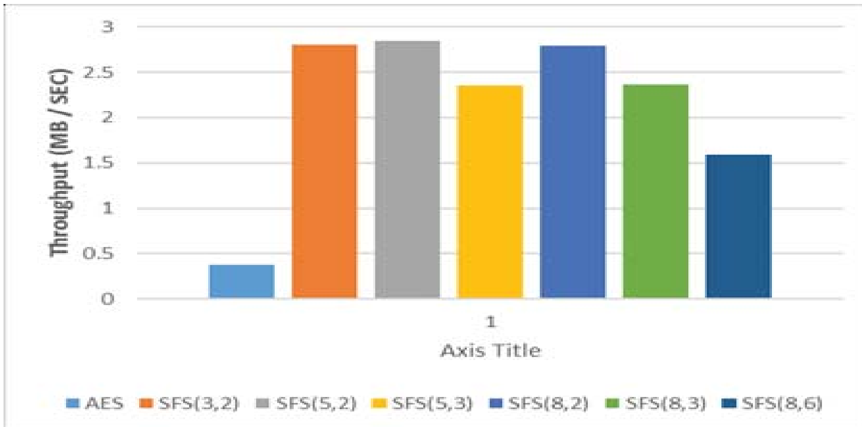

Figure B.22: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Text File Type of Different Sizes.

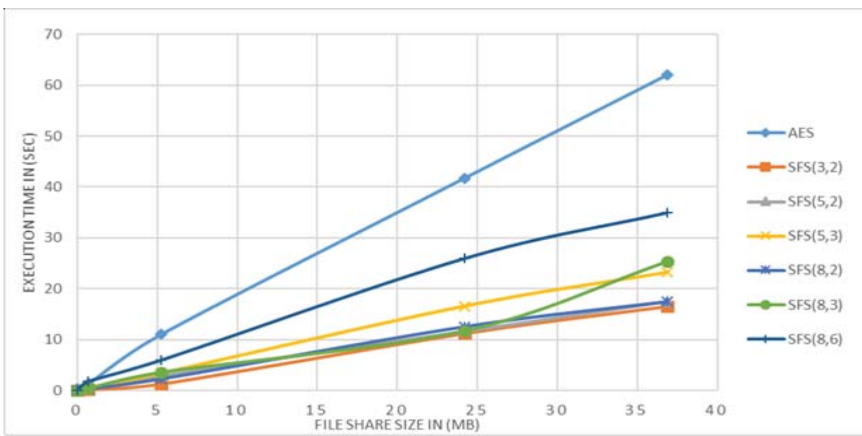

Figure B.23: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Text File Type of Different Sizes.

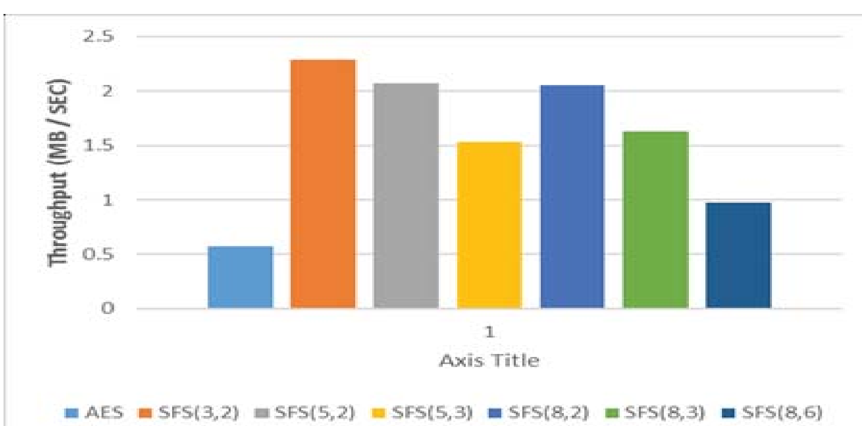

Figure B.24: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Text File Type of Different Sizes.

B.7 Video file formats

Table B.13: Sequential implementation of Reconstruct the Secret and Decryption using AES of of Video File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ts	0.000669	0.256799	0.000212	0.000626	0.00032	0.000312	0.000332	0.000485
.ts	0.002464	0.293664	0.001637	0.005602	0.003467	0.001838	0.001198	0.002566
.flv	1.387849	4.03047	0.461222	0.428631	0.714023	0.540466	0.607379	0.877016
.MKV	1.405831	3.803419	0.480534	0.503221	0.643316	0.520095	0.64709	0.8997
.MKV	1.755652	4.719978	0.632293	0.498171	0.963268	0.601641	0.802551	1.093034
.avi	12.81602	35.2293	3.934831	3.912593	4.776419	3.980353	6.196741	8.332461
.MKV	13.44737	33.35583	4.724364	5.084271	5.870088	4.83259	5.615623	8.551873
.flv	13.53714	35.31849	5.56572	4.162075	5.044161	5.208081	5.72568	8.578368
.MKV	13.70051	37.00518	5.231454	5.153261	5.101218	4.869663	5.934921	8.870448
.FLV	27.32828	74.47737	9.540214	9.706642	11.65917	9.561162	11.65794	17.95487
.MKV	27.38996	73.62939	9.629694	9.868815	11.51664	9.580877	11.54118	17.45302
.MKV	66.155	171.4361	22.85618	22.77387	28.1996	23.13909	27.15777	39.61638
.mov	119.4465	304.2607	39.0111	38.79849	47.08623	38.94526	46.20445	68.4274
.MKV	130.4061	334.0517	45.83825	42.86129	51.44863	42.953	51.16815	74.67544
.avi	137.9018	352.8027	47.47269	46.022	55.27507	47.72214	53.09463	79.25238
Sum=	566.6811	1464.671	195.3804	189.7796	228.3016	192.4566	226.3556	334.5854
Throughput (MB / Sec)		0.3869	2.900399	2.985997	2.48216	2.944462	2.503499	1.693681

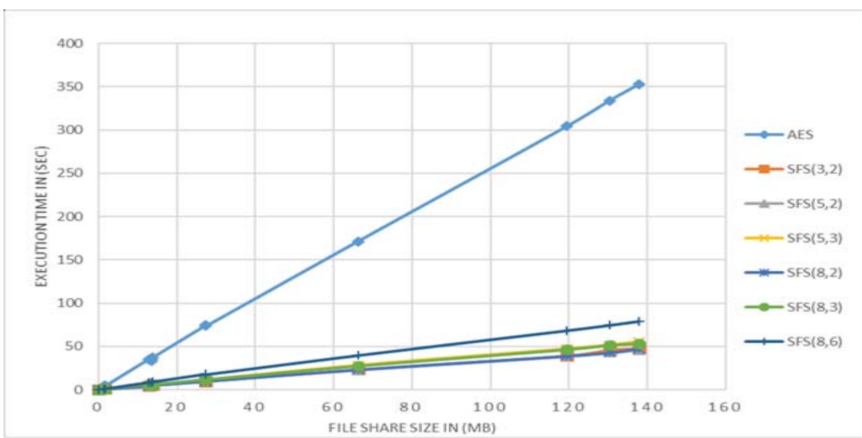

Figure B.25: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Video File Type of Different Sizes.

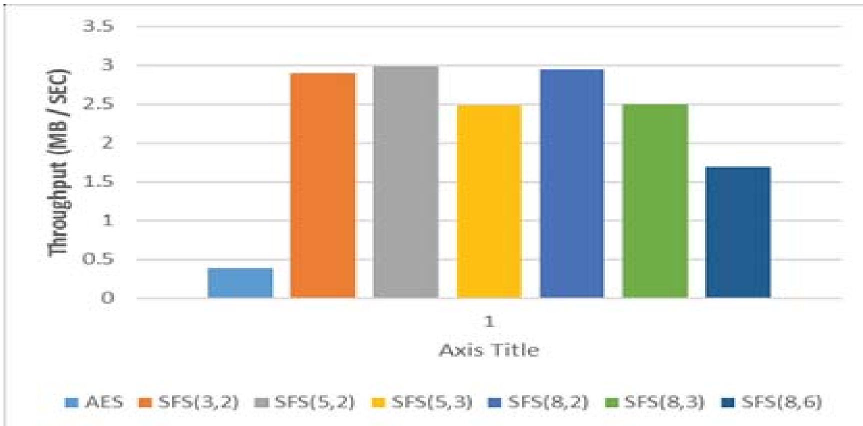

Figure B.26: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of of Video File Type of Different Sizes.

Table B.14: Parallel implementation of Reconstruct the Secret and Decryption using AES of of Video File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.ts	0.000669	0.184383	0.006452	0.006969	0.014137	0.011379	0.048981	0.006647
.ts	0.002464	0.18373	0.007958	0.009599	0.00968	0.008578	0.009171	0.017234
.flv	1.387849	2.117553	0.858279	1.637039	1.704443	2.632385	2.559958	2.930011
.MKV	1.405831	1.964219	0.980868	1.700778	1.681489	2.695033	2.617101	2.906916
.MKV	1.755652	2.722497	1.175957	1.945766	2.017536	3.072792	3.293765	3.567436
.avi	12.81602	18.53886	6.651224	5.76569	8.737804	13.8445	13.42963	16.61818
.MKV	13.44737	19.22737	7.691684	11.43127	11.87164	18.46591	18.69308	21.59098
.flv	13.53714	19.61065	6.990308	11.50166	12.1788	17.57379	18.13515	20.37754
.MKV	13.70051	16.3108	7.073713	9.952842	6.650318	13.36864	15.54856	17.73247
.FLV	27.32828	37.92532	14.3619	22.37964	23.89967	34.99093	36.71413	41.22584
.MKV	27.38996	38.08249	15.24409	22.45874	23.51405	36.55736	35.55556	42.92936
.MKV	66.155	90.40687	32.59527	50.32345	54.32381	75.53735	84.12947	88.63307
.mov	119.4465	158.1147	48.30215	84.94016	89.12784	135.2178	137.5684	160.1241
.MKV	130.4061	172.9231	46.20615	92.90914	99.15387	142.8584	148.366	170.9934
.avi	137.9018	180.4383	58.21084	98.67343	104.9238	152.4514	152.885	180.7646
sum=	566.6811	758.7508	246.3568	415.6362	439.8089	649.2862	669.554	770.4177
Throughput (MB / SEC)		0.746861	2.300245	1.363407	1.288471	0.872775	0.846356	0.73555

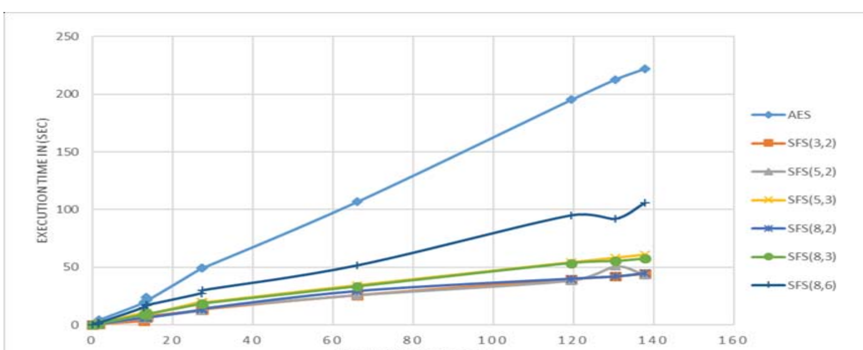

Figure B.27: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Video File Type of Different Sizes.

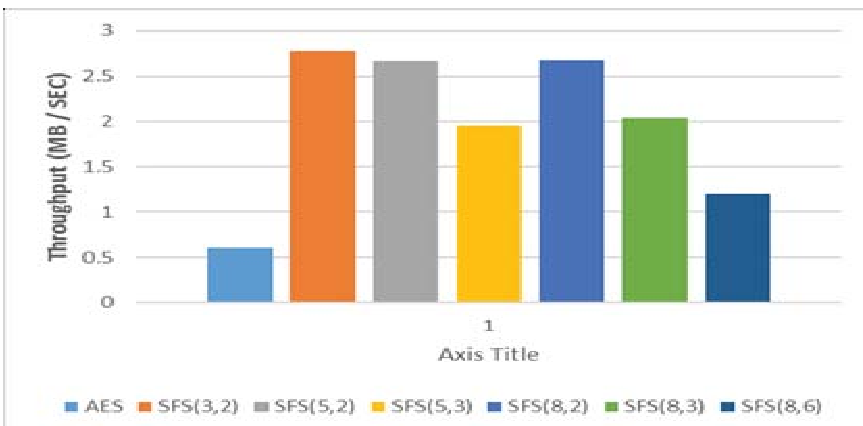

Figure B.28: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of of Video File Type of Different Sizes.

B.8 Archive file formats

Table B.15: Sequential implementation of Reconstruct the Secret and Decryption using AES of Archive File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.tar	0.000533	0.242376	0.000247	0.00024	0.00018	0.000239	0.000291	0.000263
.jar	0.001053	0.234573	0.000294	0.000364	0.000572	0.000544	0.000875	0.001277
.zip	0.00113	0.299076	0.000333	0.000451	0.000393	0.000463	0.000644	0.000653
.tgz	0.001496	0.278332	0.000742	0.001779	0.000529	0.000752	0.000885	0.001756
.zip	0.007612	0.255407	0.004522	0.002927	0.005253	0.005586	0.011559	0.008053
.jar	0.011474	0.292837	0.005575	0.006713	0.004775	0.013113	0.015511	0.0072
.tgz	0.011759	0.26441	0.004514	0.013109	0.003994	0.006407	0.00871	0.006437
.jar	0.036816	0.257485	0.012898	0.015473	0.015178	0.029047	0.028208	0.022504
.tar	0.091844	0.278502	0.030515	0.028311	0.036073	0.0319	0.03547	0.054143
.bz2	0.131865	0.485364	0.063491	0.07223	0.064509	0.06189	0.059715	0.095637
.zip	1.318213	3.753446	0.50226	0.508672	0.549625	0.428201	0.513374	0.838018
.gz	1.334643	3.795429	0.435248	0.37148	0.458152	0.390704	0.541494	0.841931
.cab	1.344069	3.753309	0.460387	0.441517	0.548179	0.461132	0.643197	0.837472
.jar	1.35046	3.998891	0.482514	0.458766	0.560475	0.513973	0.576656	0.855937
.rar	1.353995	3.574848	0.446467	0.516675	0.615695	0.511206	0.526367	1.023215
.jar	12.62993	31.64608	4.695976	4.596131	4.625538	4.80168	5.285697	7.907785
.rar	13.02323	36.62286	4.622633	4.65696	5.491159	4.67042	5.373151	8.133996
.zip	13.36134	37.55304	4.786923	4.050668	4.858523	4.010406	5.423294	8.483888
.gz	13.39809	38.11117	4.713219	5.087152	5.588199	4.728284	5.620838	8.568021
.jar	20.97486	56.81981	7.433035	7.769295	8.845101	7.568561	8.808253	13.19771
.rar	27.44257	72.91977	9.478216	9.604407	11.56578	9.641595	11.54791	17.50508
.cab	27.68423	72.1966	9.609181	9.754971	11.71301	9.914219	11.63645	17.63824
.zip	27.84394	69.81085	9.889767	9.848058	12.16307	9.772322	10.36968	17.6818
.jar	42.70904	109.0025	15.02562	15.07055	17.96355	15.18873	18.0401	25.70592
.cab	67.44812	174.1442	22.96422	23.08314	27.80905	23.31829	27.49221	40.20562
.zip	67.87289	175.0772	23.83713	23.43831	27.73775	24.70881	27.77851	40.47685
.bz2	68.7001	178.1055	24.07647	23.8861	27.88458	23.8102	27.9228	40.37151
.rar	132.5344	340.6436	46.12009	45.80777	50.9975	43.97627	54.15067	75.50048
Sum=	542.6197	1414.417	189.7025	189.0922	220.1064	188.5649	222.4125	325.9714
Throughput (MB / Sec)		0.383635	2.860372	2.869604	2.465261	2.877628	2.439699	1.664624

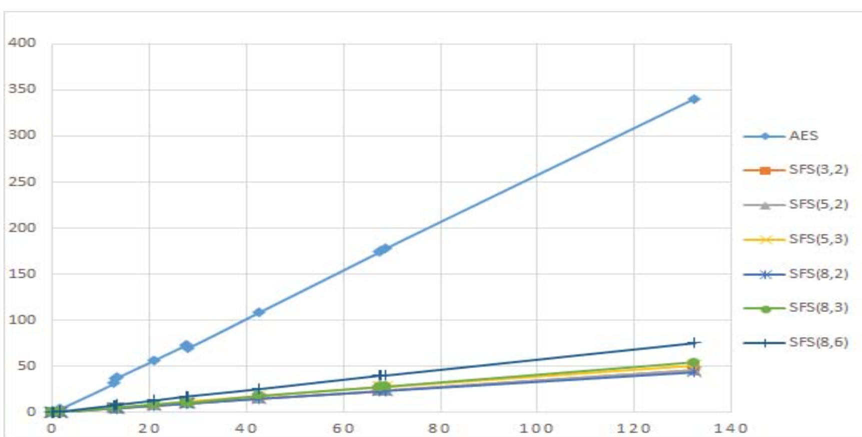

Figure B.29: Performance Comparison of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of Archive File Type of Different Sizes.

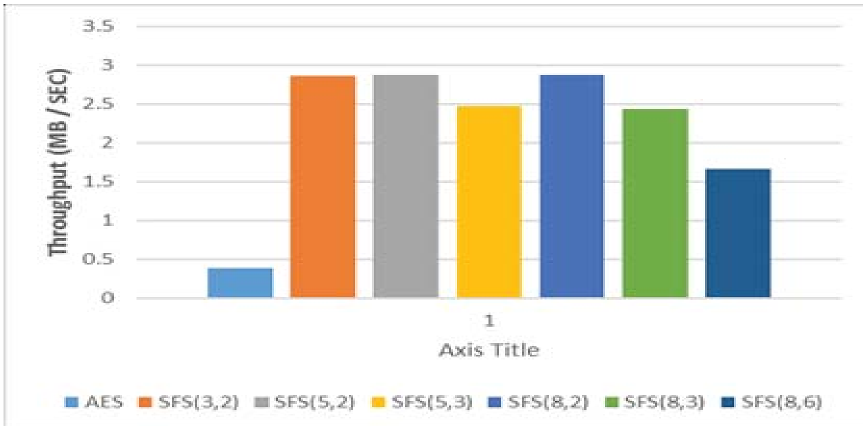

Figure B.30: Throughput of the Sequential Execution Time of Reconstruct the Secret and Decryption using AES of Archive File Type of Different Sizes.

Table B.16: Parallel implementation of Reconstruct the Secret and Decryption using AES of Archive File Type of Different Sizes.

		Cpu Time(sec)						
File Type	**File Share Size(MB)**	**AES**	**SFS(3,2)**	**SFS(5,2)**	**SFS(5,3)**	**SFS(8,2)**	**SFS(8,3)**	**SFS(8,6)**
.tar	0.000533	0.214641	0.007593	0.004151	0.003816	0.011969	0.007307	0.00558
.jar	0.001053	0.215522	0.005225	0.006241	0.004638	0.006157	0.028372	0.006388
.zip	0.00113	0.22302	0.003786	0.012842	0.007039	0.006367	0.014508	0.006079
.tgz	0.001496	0.221678	0.004651	0.006077	0.005275	0.00853	0.012755	0.027725
.zip	0.007612	0.210432	0.010132	0.015171	0.015669	0.022339	0.02575	0.030642
.jar	0.011474	0.181526	0.014006	0.035418	0.023381	0.032752	0.047877	0.045531
.tgz	0.011759	0.189121	0.014581	0.021087	0.022256	0.035389	0.053477	0.077518
.jar	0.036816	0.217075	0.039003	0.059119	0.061435	0.089891	0.117946	0.108884
.tar	0.091844	0.215591	0.089888	0.150298	0.18199	0.222595	0.338322	0.348689
.bz2	0.131865	0.426827	0.111829	0.226586	0.239959	0.321499	0.477082	0.401263
.zip	1.318213	1.929565	0.751393	1.211158	1.218222	1.651659	3.335875	1.934106
.gz	1.334643	1.817429	0.662164	1.201045	1.289681	1.681213	3.405802	2.073699
.cab	1.344069	1.827014	0.685123	1.242893	1.256804	1.689911	3.490201	1.901789
.jar	1.35046	2.00548	0.692658	1.294897	1.310784	1.665376	3.35232	1.932799
.rar	1.353995	1.747367	0.796415	1.222818	1.28243	1.690006	3.50681	1.970929
.jar	12.62993	18.03885	6.270601	10.11057	10.75763	15.47928	18.71159	17.4263
.rar	13.02323	18.62211	6.585283	10.67261	10.83543	15.8751	18.90561	17.93968
.zip	13.36134	18.50181	6.754231	8.36998	9.964911	16.24021	21.33709	18.41719
.gz	13.39809	19.24553	6.705059	11.28207	11.94678	16.52535	20.89908	18.89354
.jar	20.97486	27.60034	10.44766	15.14109	17.89576	25.37648	32.09092	29.32713
.rar	27.44257	38.59434	13.52581	23.25422	24.02039	34.46649	39.78756	38.25546
.cab	27.68423	39.14305	13.97467	23.08016	24.71458	33.99184	41.00914	39.08563
.zip	27.84394	36.26288	10.9331	19.32445	15.76131	33.5387	42.42107	39.27345
.jar	42.70904	58.51731	20.79835	33.12746	36.84034	52.40613	65.64318	60.59498
.cab	67.44812	93.78899	32.50352	52.56874	54.11802	82.6981	99.4618	96.85045
.zip	67.87289	90.89594	34.31373	53.63226	53.6111	84.62525	99.95392	96.91834
.bz2	68.7001	90.93116	35.36137	51.65609	54.87644	84.11406	102.6577	98.7098
.rar	132.5344	171.8773	55.05326	87.48828	95.14789	167.5214	195.2636	194.7235
sum=	542.6197	733.6619	257.1151	406.4178	427.414	671.994	816.3567	777.2871
Throughput (MB / SEC)		0.739605	2.110416	1.335128	1.269541	0.807477	0.664685	0.698094

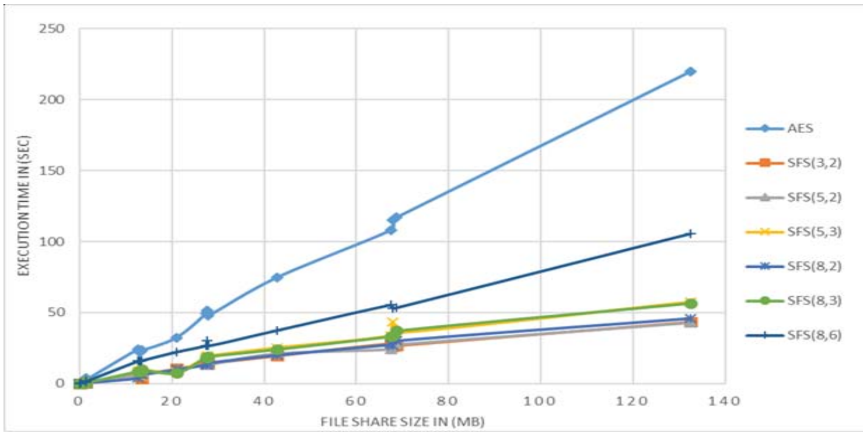

Figure B.31: Performance Comparison of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of Archive File Type of Different Sizes.

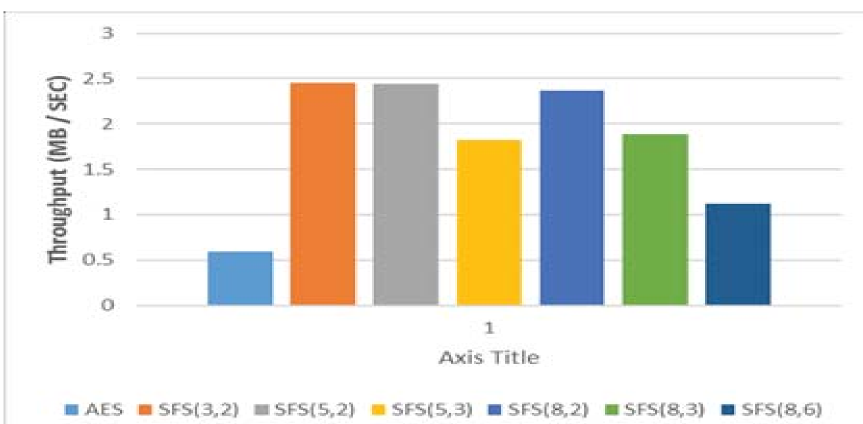

Figure B.32: Throughput of the Parallel Execution Time of Reconstruct the Secret and Decryption using AES of Archive File Type of Different Sizes.

Vitae

- Name: Ibrahim Abdullah Musleh Althamary
- Nationality: Yemen
- Date of Birth: January 1, 1985
- Email: *i.ibrahim42009@gmail.com*
- Permenant Address: Thamar, Yemen
- EDUCATION:

  2014-2017 Graduated from KFUPM with a MS in Computer Engineering Department, Major: Computer Network.

  2005-2009 Graduated from Thamar University-Computer Science Information System with a BS in Computer Science, Thamar Yemen.
- PUBLICATIONS :

  -Althamary, Ibrahim Abdullah, and Talal Mousa Alkharobi. "Secure File Sharing in Multi-clouds using Shamirs Secret Sharing Scheme." Transactions on Networks and Communications 4.6 (2017): 43.

  - Althamary, Ibrahim Abdullah, and El-Sayed M. El-Alfy. "A More Secure Scheme for CAPTCHA-Based Authentication in Cloud Environment." Int. Conf. Information Technologies 2017.