SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TNC TRANSACTIONS ON NETWORKS AND COMMUNICATIONS

VOLUME 4, ISSUE 6
ISSN: 2054 -7420

Secure File Sharing in Multi-clouds using Shamir's Secret Sharing Scheme

[1]Ibrahim Abdullah Althamary and [2]Talal Mousa Alkharobi
[1,2] Computer Engineering Department, King Fahd University of Petroleum & Minerals, KSA;
i.ibrahim42009@gmail.com; talalkh@kfupm.edu.sa

ABSTRACT

Cloud computing is a significant model for permitting on-demand network access to shared data, software's, infrastructure, and platform resources. However, cloud storage needs a certain level of availability, confidentiality, and integrity. Information sensitivity and value encourage users to select a highly secure protocol. This work proposes a new mechanism to increase the user trust in cloud computing using the secret sharing technique. The proposed algorithm is using Base64 encoding to convert any file type to ASCII strings. Base64 strings do not need any extra process to be compressed and this can speed up the share building process. Each string is used to produce N shares (using Shamir Secret Sharing Scheme) where each share is stored in a separate location in the cloud.

Keywords: Base64; Advance Encryption Standard; Multi-Cloud; Secure File Sharing (SFS); Secret Sharing.

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 server. Having many copies of data may increase the reliability. However, the existence of data in several different locations may reduce the confidentiality as it gives more chances to the attackers. Therefore, the need for a technique to enhance both the availability and the confidentiality motivates the use of secret sharing method.

Base64 is a mechanism to encode and decode data as ASCII string which is commonly used in e-mail to make the content encrypted. In addition, base64 is one of the best and most popular encoding/decoding schemes on the internet. Trillions of data 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 the secret sharing scheme. The proposed technique will take any file type as input and convert it using base64 to ASCII strings. Then, for each ASCII string, a set of *n* shares will be created then distributed one per cloud/location. The shares should be created such that the string can be regenerated by choosing any *t*

DOI: 10.14738/tnc.46.2560
Publication Date: 30th December 2016
URL: http://dx.doi.org/10.14738/tnc.46.2560

shares out of *n* shares (where *t* <= *n*). All outputs are in ASCII format which makes the process of storage and distribution easier.

The rest of this paper is structured as follows. We will introduce secret sharing in section 2 then base64 algorithms in section 3. Then section 4 describes the proposed scheme in details. Finally, the experimental results and discussion are presented in section 5.

2 Secret sharing

Secret sharing scheme is a decent tool for cryptography 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 Figure 1. We will focus only on one category of secret sharing schemes called threshold schemes.

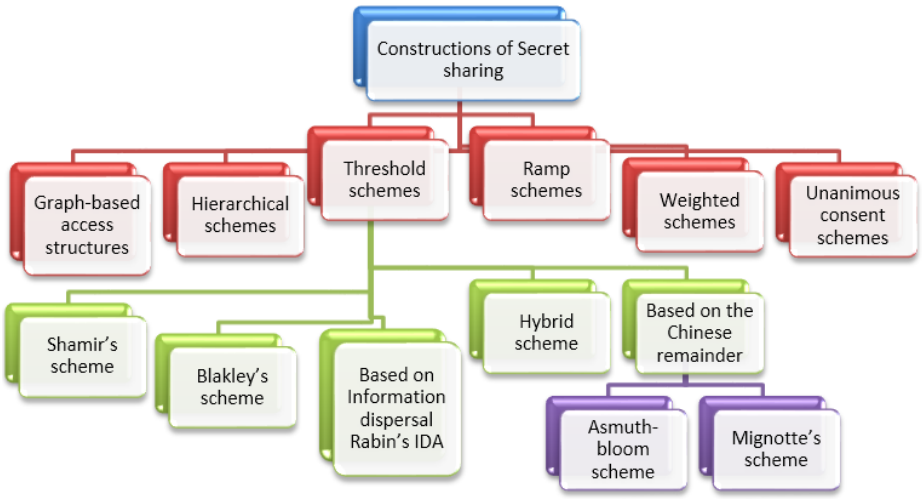

Figure 1 Schemes of Secret Sharing

2.1 Threshold secret sharing Techniques

In (*t*, *n*) threshold SSS, the secret *s* splits into *n* shares in such a way that *t* participants or more than *t* can reconstruct or obtain *s* but participants less than *t* cannot obtain any information about *s* [1].

The idea of threshold secret sharing was proposed independently by Shamir [2] and Blakley [3]. The threshold schemes contain Shamir's scheme, Blakley's scheme, information dispersal Rabin's IDA, the Chinese remainder, and Hybrid scheme [4].

Blakley[3] introduced a secret sharing method known as a threshold scheme which aims at the hyperplane. By using linear geometry, this method solves the secret sharing problem and it has been indeed used in secret image sharing technology. Moreover, as an example of Chinese remainder scheme, Mignott's[5] secret sharing scheme uses a special sequence of integers with CRT.

Table 1 Shows a Comparison Between the Most Important Threshold Secret Sharing Techniques.

method	Year	Techniques Used	Proactive	Advantage	Drawback
Shamir [2]	1979	Polynomial based	No	Perfect Ideal	It is not secure against cheaters. Not provide any extended features.
Blakley[1]	1979	Geometry based	No	Ideal	It is not perfect & Ideal SSS as Shamir‟s SSS. It is less space efficient than Shamir‟s scheme
Mignott's[5]	1982	CRT based	No	Ideal	It is not perfect secret sharing scheme

2.2 Shamir Secret sharing scheme

Shamir [2] 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 quorum of participant to reconstruct the secret. Generating the secret share use the following equation.

$$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \; mod \; p \tag{1}$$

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 the reconstruction, m participants, where t < m < n, are required to recollect their shares to the dealer and the dealer can perform the calculation using the Lagrange interpolation equation.

$$f(x) = \sum_{j=1}^{n} y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} \; (mod \; p) \tag{2}$$

To reconstruct the original polynomial, equation 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 original message

$$f(0) = s$$

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. Let us consider polynomial f(x) = 15x2 + 13x + 19 over the field Z_{23} where p is the prime number =23.

We get following shares of the five secrets.

$$Secret \; s = f(0) = 19$$

$$s1 = f(1) = 15 \times 12 + 13 \times 1 + 19 \; mod \; 23 = 1$$

$$s2 = f(2) = 15 \times 22 + 13 \times 2 + 19 \; mod \; 23 = 13$$

$$s3 = f(3) = 15 \times 32 + 13 \times 3 + 19 \; mod \; 23 = 9$$

$$s4 = f(4) = 15 \times 42 + 13 \times 4 + 19 \; mod \; 23 = 12$$

$$s5 = f(5) = 15 \times 52 + 13 \times 5 + 19 \; mod \; 23 = 22$$

By chosen shares s1, s2 and s3 the secret can be reconstructed as:

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

$$= -27 mod \; 23 = 19$$

$$f(0) = 19$$

2.3 Using secret sharing mechanism to secure images

In this section, we will review different types of research work conducted on secret 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 a number of shared images and distribute these shared images to participants.

Lin and Thien [6]. 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. [7] 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. The perfect reconstruction property recovers the original color image by logically decrypting the decomposed bit vector-arrays of the color shares and this process is called the decryption process.

Lou et al. [8] 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 shares happen 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. [9] 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 show how it is very good because the secret can be recognized by a human.

Alex et al. [11] 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 image it is not suffered by cross interference of share images.

Yang et al.[12] 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.[13] 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 after applying aspect ratio constraints. Over the partition and disguising processes, two shares where not meaningful images individually. They did not expose any data of the secret images. To reconstruct the secret, each share was flapped and 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.[14] introduced the formulation of VSS encryption for multiple images. The limitation of the EVCS Scheme 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 disallowing sets must be similar. Therefore, generalized VSS scheme for encrypting multiple secret images was introduced.

He, et al. [15] proposed novel (t, n) image that is gradually enhanced by using LOCO-I compression. Additionally, by embedding the hash-based message the three types of cheating will probably be detected Moreover, they improve the security by using a random strategy with dynamic embedding. This scheme and [16] divided the shadow or image to a group.

Askari et al. [17] developed the VSS scheme which is given by proposed (2, 2) VSS scheme without image size expansion. This scheme is based on encrypting a secret block with four pixels into two shares depending on the distribution of B&W 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. [18] developed a new color VCS that depends on the improved VC. In this scheme, the secret color image is shared using different random images and one noise 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 loss of quality. Due to this, the suggested scheme can deal with black & white pixels or color images. Table 2 shows a comparison of various secret image sharing techniques.

Table 2: Comparison of Different Secret Image Sharing Mechanism

Schemes	year	Techniques Used	Meaningful shares	Type of image	Pixel Expansion
Lin and Thien [6]	2002	Polynomial based	No	black & white	No
Lukac, et.at [7]	2004	Decomposed bit-levels	No	Color	No
Lou et al [8].	2011	Cover Image	Yes	Color	No
Tsai et al. [9]	2009	combination	No	Color	Yes
Chen et al[10]	2009	Random Grids	No	black & white	No
Alex et al. [11]	2011	Error Diffusion	No	black & white	Yes
Yang et al.[12]	2004	Probabilistic	No	black & white	No
Lin, et al.[13]	2010	Multiple Secrets	No	black & white	No
Sasaki et al.[14]	2014	Multiple Secrets	No	black & white	No
Askari et al. [17]	2012	XOR operation	No	black & white	No
Liu et al. [18]	2013	NVSS	Yes	Color	No

2.4 Using secret sharing mechanism to secure files

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

Kallahalla et al.[19] proposed 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 the RSA. The cryptography for the file is done on the client side, not on the server side which increases the trust.

Dong et al.[20] 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. [21] 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 sympatric encryption and distributed them in multi-cloud.

Alsolami and Boult [22] proposed CloudstaSh that applied Shamir secret sharing scheme [2]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 will increase. Moreover, they just created eight shares with a threshold of two which is not enough to show how good their work is.

3 Base64

Base64[23-28] 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 as shown in the following examples.

Example: Input data, 3 bytes, "ABC". Encoded output, 4 characters, "QUJD"

```
Input Data           A         B         C
Input Bits    01000001 01000010 01000011
                     \       \       \
Bit Groups    010000 010100 001001 000011
Mapping             Q       U       J       D
```

Example Decoding:

YW55IGNhcm5hbCBwbGVhcw==	two '='s	one	any carnal pleas

Base64 decoding process is the reverse of encoding process [27] 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[25, 29].

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 and decoded each day. 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[24, 28]. 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 [24]. This is to make sure that the data remain as such without modification during transmission[30]. 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. [27].

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. [31].Moreover, Base64 and AES algorithm are utilized to enhance the security of data[28] 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 [29] .

4 Proposed Scheme

The proposed technique is mainly used to protect data and increase the level of confidentiality and availability. Our work is based on base64 and Shamir[2] secret sharing scheme which is a perfect and ideal threshold scheme that boosts the security of data and gives the client more trust in cloud computing. This method will take any data file as input (JPG, GIF, PNG, ... txt, doc…etc.) and converts the file using base 64 then the file is compressed. Then Shamir[2] secret sharing mechanism is applied to generate *n* shares and distribute them to n different servers. Moreover, the original secret can be regenerated by *t*

of them (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 parts: Save and Load. **Error! Reference source not found.** A and B show flowcharts of saving a file to the cloud and downloading a file from the cloud.

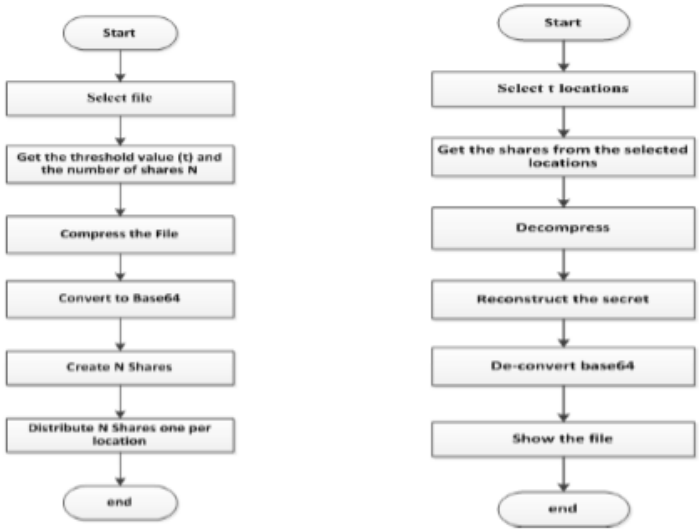

Figure 2: (A) Saving a File to the Cloud (B) Loading a File from the Cloud

4.1 Preparing the File

During this phase, the selected file will be converted using base 64 to. Afterwards, the file is compressed before and after Encoding/Decoding process using the base64 algorithm to convert the file to ASCII string to be ready for the next phase.

4.2 Shares building phase

In this phase, *n* shares of the file are generated ,We need the following information [32]:

- The secret: the ASCII string that is generated in the first phase.
- 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.
- The threshold value of secret: A qualified subset is a subset of the shareholders that should be able to rebuild the secret.

4.3 Share distribution phase

During this phase, only the file owner who first uploads the file is required to select the number of shares and is required to distribute such secret shares.

4.4 Secret reconstruction phase.

During this phase, users must access a threshold of storage servers through authentication and obtain the secret shares to reconstruct the file. In other words, the authorized users who own the file can reconstruct the file and get access to the share easily.

5 Experiment results

Our proposed scheme is efficient and does not cost a lot of time. To show the amount of time saved because of our scheme during the process of creating the shares and reconstructing them, we present a comparison between our scheme and sympatric encryption used in [21, 22, 33-35] . We use 256 bit AES algorithm with CFB mode and for hashing we use SHA512. Moreover, we apply secret sharing on the key to divide it into many shares and then distribute them to 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 72.2 Mbps. 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 [36].

5.1 Create The Secret.

We start 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 around 200KB and then we apply the same process for the small file in both methods. The following graphs illustrate the execution time in second and file size in MB. We have created shared files with different types and sizes. In general, we have chosen 59 files with 26 different types and different sizes and run the AES and secret file sharing(SFS) at the same time.

As we can see, In Figure 3 the line graph shows a comparison between SFS and AES algorithm when n=5 and t=3. The line graph illustrates that both schemes almost require the same execution time for low file sizes. However, for file sizes with 10 MB or more the SFS consumes less time than AES. More importantly, the difference in execution time increases with the file size.

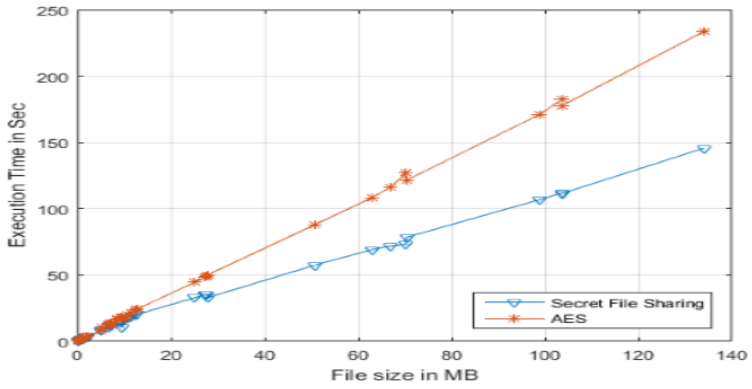

Figure 3 Performance Comparison of the Execution Time of Creating a Secret when the n=5 and the t=3.

In Figure 4 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. However, in this case the difference in the required time is almost constant and does not depend on the file size.

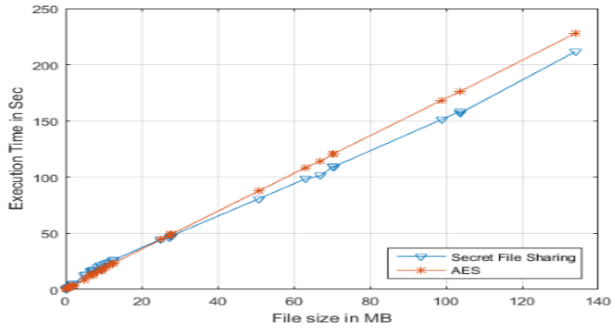

Figure 4 Performance Comparison of the Execution Time of Creating a Secret when the n=8 and the t=2.

In Figure 5 the line graph shows a comparison between SFS and AES algorithm when *n=8* and *t=3*. As we can notice when the threshold increases, the speed of creating the secret files using SFS falls down. Here, the both algorithms are almost the same when the file size is small but when the file size is large our algorithm runs slightly faster than AES algorithm.

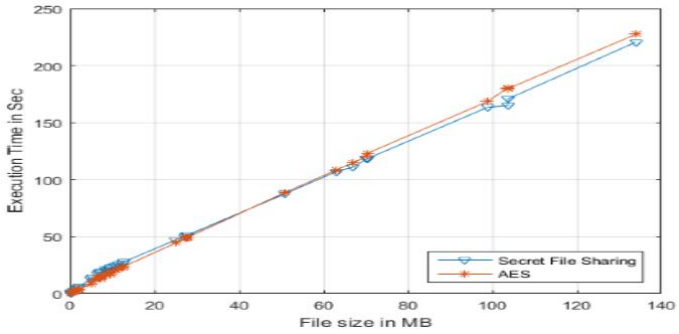

Figure 5 Performance Comparison of the Execution Time of Creating a Secret when the n=8 and the t=3.

Figure 6 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.

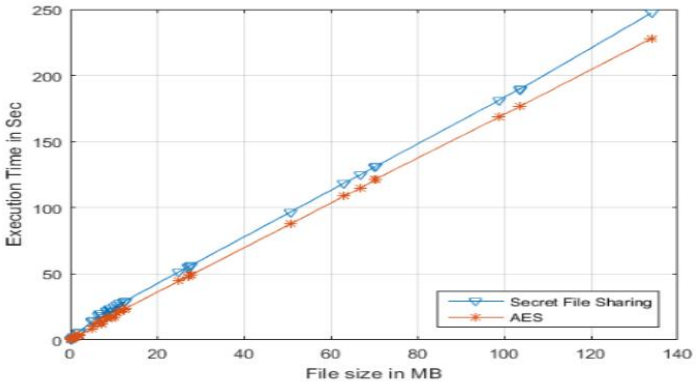

Figure 6 Performance Comparison of the Execution Time of Creating a Secret when the n=8 and the t=6

5.2 Reconstruct the secret

Error! Reference source not found.7 shows the execution time during the process of reconstructing the shares for our scheme and decryption using AES algorithm with different type and size when *n*=5 and

t=3. The line graph illustrates that the speed of reconstructing the secret file using SFS is fast compared to the decryption using AES algorithm regardless of the file size. Moreover, as the file size increases the difference in time excitation increases as well.

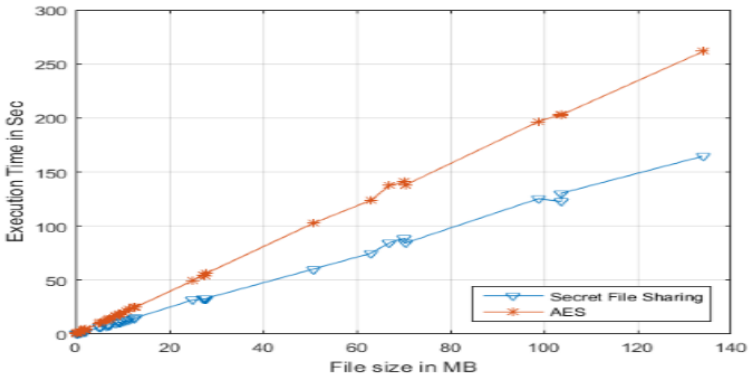

Figure 7 Performance Comparison of the Execution Time of Reconstructing the Secret when the n=5 and the t=3.

In Figure 8 the line graph shows a comparison between SFS and AES algorithm when. 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.

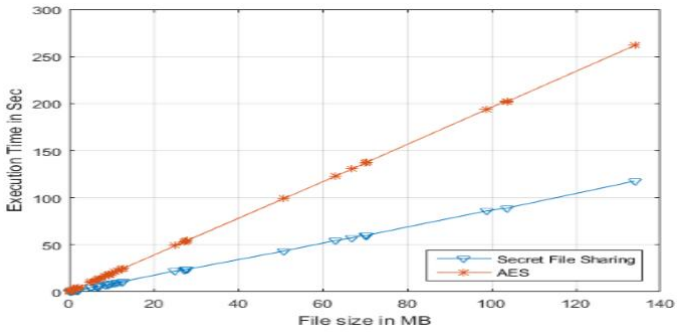

Figure 8 Performance Comparison of the Execution Time of Reconstructing the Secret when the n=8 and the t=2.

In figure 9 the line graph shows a comparison between SFS and AES algorithm when *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 affect the result of the reconstructing process while the threshold plays the critical role.

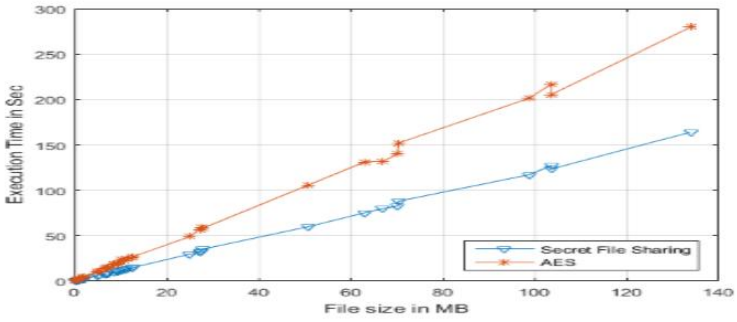

Figure 9 Performance Comparison of the Execution Time of Reconstructing the Secret when the n=8 and the t=3.

Figure 9 shows the result when the *n=8* and the *t*=6 .As the threshold increases ,the SFS execution time increases as well. Here, the decryption using AES algorithm runs faster than SFS but the differences are within the acceptable range.

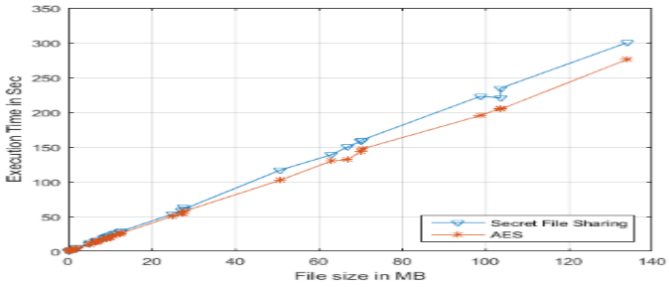

Figure 10 Performance Comparison of the Execution Time of Reconstructing the Secret when the n=8 and the t=6.

Overall, we can conclude that the results are all within the acceptable range. Also in using SFS to build the secret we noticed that the outputs change significantly according to the number of shares and the threshold. Increasing both parameters shows acceptable results level of security is definitely enhanced. Moreover, it is the threshold that plays the critical role in the reconstruction process not the number of shares.

5.3 Upload Process

Table 3 and the Figure 11 illustrate experimenting the uploading time for different files with different sizes in parallel to cloud storage. Overall, regarding uploading time Dropbox is the worst while best is SMEstorage which hosted in Amazon S3. Although, the time in AES_DropBox include the uploading time for the encryption file and the avenge time of the uploading all shares of the key.

Table 3 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

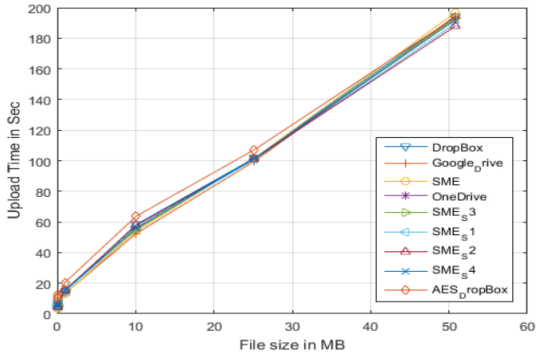

Figure 11 Performance Comparison of the Uploading Time for Different Files with Different Sizes in Parallel to Cloud Storage

5.4 Download Process

Table 4 and the Figure 12 show the results of the experiment on time of download for different files with different sizes in parallel from cloud storage. concerning download time Dropbox is the worst and 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 include the download time for the encryption file and the avenge time of the download all shares of the key.

Table 4 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

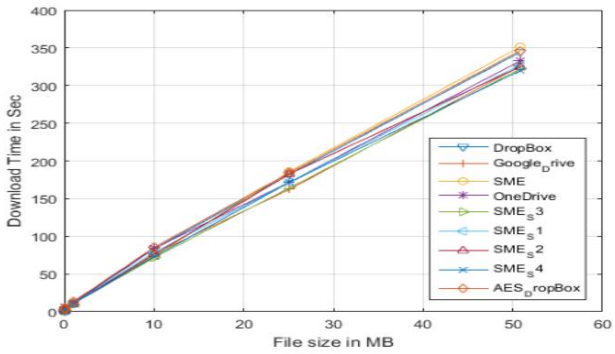

Figure 12 Performance Comparison of the Downloading Time for Different Files with Different Size in Parallel from Cloud Storage

6 Conclusion

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 DEFLATE 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 for them to have access to the data. 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 sympatric algorithm in both creating the or reconstructing the secret for any type of file.

ACKNOWLEDGEMENT

The investigators appreciate and acknowledge for King Fahd University of Petroleum and Minerals for the full support.

REFERENCES

[1] 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, pp. 3592-3601, 2015.

[2] A. Shamir, "How to share a secret," *Communications of the ACM,* vol. 22, pp. 612-613, 1979.

[3] G. R. Blakley, "Safeguarding cryptographic keys," in *afips*, 1979 p. 313.

[4] S. Iftene, "Secret Sharing Schemes with Applications in Security Protocols," *Sci. Ann. Cuza Univ.,* vol. 16, pp. 63-96, 2006.

[5] M. Mignotte, "How to share a secret," in *Cryptography*, ed: Springer, 1982, pp. 371-375.

[6] C.-C. Thien and J.-C. Lin, "Secret image sharing," *Computers & Graphics,* vol. 26, pp. 765-770, 2002.

[7] R. Lukac and K. N. Plataniotis, "Colour image secret sharing," *Electronics Letters,* vol. 40, p. 529, 2004.

[8] 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, pp. 118-134, 2011.

[9] 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, pp. 3247-3254, 2009.

[10] T.-H. Chen and K.-H. Tsao, "Visual secret sharing by random grids revisited," *Pattern Recognition,* vol. 42, pp. 2203-2217, 2009.

[11] 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*, 2011, pp. 393-397.

[12] C.-N. Yang, "New visual secret sharing schemes using probabilistic method," *Pattern Recognition Letters,* vol. 25, pp. 481-494, 2004.

[13] T.-L. Lin, S.-J. Horng, K.-H. Lee, P.-L. Chiu, T.-W. Kao, Y.-H. Chen*, et al.*, "A novel visual secret sharing scheme for multiple secrets without pixel expansion," *Expert systems with applications,* vol. 37, pp. 7858-7869, 2010.

[14] 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*, 2014, pp. 7391-7395.

[15] 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.

[16] 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.

[17] 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*, 2012, pp. 1-4.

[18] 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*, 2013, pp. 449-454.

[19] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage."

[20] J. Y. Xin Dong, Yuan Luo, Yingying Chen, Guangtao Xue, and Minglu Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *Computers & security,* pp. 151–164, 2014.

[21] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: dependable and secure storage in a cloud-of-clouds," *ACM Transactions on Storage (TOS),* vol. 9, p. 12, 2013.

[22] 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*, 2014, pp. 315-320.

[23] V. S. Agme and A. C. Lomte, "Cloud Data Storage Security Enhancement Using Identity Based Encryption," *Identity,* vol. 3, 2014.

[24] D. Esbensen, "Apparatus and method for fast data encoding and decoding," ed: Google Patents, 2012.

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

[26] 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*, 2014, pp. 2220-2223.

[27] M. Shirali-Shahreza and S. Shirali-Shahreza, "Sending pictures by SMS," in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, 2009, pp. 222-223.

[28] 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*, 2013, pp. 232-237.

[29] Wikipedia. (March 22). *Base64*. Available: https://en.wikipedia.org/wiki/Base64

[30] 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, 2015.

[31] R. Prajapati. (2014, March 22). *Base64 Images Advantages & Disadvantages - CodeRiddles*. Available: http://www.coderiddles.com/base64-images/

[32] T. Alkharobi, "Secure Repayable Storage System," in *Global E-Security*, ed: Springer, 2008, pp. 102-109.

[33] S. Gupta and S. Lamba, "An enhanced python based approach of secret sharing scheme with encryption," *Issues,* vol. 1, pp. 173-180.

[34] 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*, 2013, pp. 404-412.

[35] M. Kadam, S. Chaudhary, and B. Carvalho, "Security Approach for Multi-Cloud Data Storage," *International Journal of Computer Applications,* vol. 126, 2015.

[36] 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*, 2013, pp. 83-93.

