Performance Evaluation of a Fragmented Secret Share System

Elochukwu Ukwandu, Prof William J Buchanan
The Cyber Academy, Edinburgh Napier University, Edinburgh. UK
e.ukwandu@napier.ac.uk, w.buchanan@napier.ac.uk

Dr Gordon Russell
The Cyber Academy, Edinburgh Napier University, Edinburgh. UK
g.russell@napier.ac.uk

Abstract: There are many risks in moving data into public storage environments, along with an increasing threat around large-scale data leakage. Secret sharing scheme has been proposed as a keyless and resilient mechanism to mitigate this, but scaling through large scale data infrastructure has remained the bane of using secret sharing scheme in big data storage and retrievals. This work applies secret sharing methods as used in cryptography to create robust and secure data storage and retrievals in conjunction with data fragmentation. It outlines two different methods of distributing data equally to storage locations as well as recovering them in such a manner that ensures consistent data availability irrespective of file size and type. Our experiments consist of two different methods – data and key shares. Using our experimental results, we were able to validate previous works on the effects of threshold on file recovery. Results obtained also revealed the varying effects of share writing to and retrieval from storage locations other than computer memory. The implication is that increase in fragment size at varying file and threshold sizes rather than add overheads to file recovery, do so on creation instead, underscoring the importance of choosing a varying fragment size as file size increases.

Keywords—data, key, secret shares, disaster contention, thresholds scheme.

I. INTRODUCTION

With the introduction of cloud services for disaster management on a scalable rate, there appears to be the needed succour by small business owners to get a cheaper and scalable disaster recovery mechanism so as to provide business continuity and remain competitive with other large businesses. But that is not to be so, as cloud outages became a nightmare. Recent statistics by Bill [1] on Cost of Data Centre Outages, shows an increasing rate of 38% from $505,502 in 2010 to $740,357 as at January 2016. Using activity-based costing, they were able to capture direct and indirect cost to: Damage to mission-critical data; Impact of downtime on organisational productivity; Damages to equipment and other assets and so on, and was derived from 63 data centres based in the USA.

These events may have encouraged the adoption of multi-cloud services so as to divert customers traffic in the event of cloud outage. Some fine-grained proposed solutions on these are focused on redundancy and backup such as: local backup [2]; geographical redundancy and backup [3]; inter-private cloud storage [4]; resource management for data recovery in storage clouds [5], and so on. But in all these, cloud service providers see disaster recovery as a way of getting the system back online and making data available after a service disruption, and not on contending disaster by providing robustness that is capable of mitigating shocks and losses resulting from these disasters.

The current practice of using public key infrastructure in protecting data being moved to the public cloud has some inherent challenges of possible loss, leakage or theft of encryption keys. The case of the carbanak cybergang attacks on banks resulting to loss of over $1bn from 100 financial institutions around the world [6] shows the weakness of protecting data using public key infrastructure. Using secret sharing scheme to provide a resilient and keyless mechanism has been proposed but using such scheme for large scale data infrastructure has remained a daunting task as the scheme is based on finite field arithmetic and so limited in scope.

In the face of these current realities, this paper outlines a Secured Threshold Storage system using Fragmented Secret Sharing system. It applies secret sharing methods as used in cryptography to create robust and secure Cloud-based data storage. Our experiments consist of two different methods – data and key shares. Data share implies using secret sharing scheme to break data into shares and using a certain number of the share (threshold) recreate the data and any number less than the threshold cannot.

While key share involves breaking data into chunks using a pre-defined chunk size, encrypting the chunk each with an AES-256-bit key and then create shares out of the encryption key based on a share policy. The shares, as well as the chunks, are stored in different storage locations and when the file is required, the key shares are recovered using the same key share policy. The recovered key is therefore used to decrypt each chunk and with the chunks, the original file is recombined.

The key contribution of the paper is in the evaluation of the performance overhead in processing and recovering files using secret shares. It outlines two different methods of distributing data equally to hosts as well as recovering them in such a manner that ensures consistent data availability irrespective of file size and type. It also shows that using fragmented secret share system is the most scalable in terms of big data infrastructure compare to using only threshold secret sharing scheme.

The rest of the paper is organized as follows: Section II is a review of related literature, while Section III took an overview of RESCUE with details of design and implementations. IV is

about the results and their evaluations, while we concluded in V as well as detailed our area of future works.

II. LITERATURE REVIEW

Loruenser et al [7] presented an architecture for secure cloud-based data sharing known as ARCHISTAR based on secret sharing scheme. The focus of the system is on providing adequate confidentiality to data; make it available against any active attacks as well as robust even in the face of failures.

Ermakova and Fabian [8] defined a secret sharing for health data in multi-provider clouds. Their work was based on the need to provide a scheme that will make data readily available, provide confidentiality and integrity to medical records stored in clouds. They used a secret sharing scheme to distribute data as fragments to several cloud in order to provide the needs as stated above. In all these, secret sharing was seen as limited in scope and therefore cannot scale large data infrastructure.

There are other research solutions based on different variants of secret sharing schemes and multi-cloud architecture that give credence to its resilience in the face of failures, data security in keyless manner, such as:

- Ukwandu *et al*, [9] - RESCUE: Resilient Secret Sharing Cloud-based Architecture.
- Alsolami & Boult, [10] - CloudStash: Using Secret-Sharing Scheme to Secure Data, Not Keys, in Multi-Clouds.
- Fabian *et al*, [11] - on collaborative and secure sharing of healthcare data in multi-clouds.

While RESCUE provides an architecture for a resilient cloud-based storage with keyless data security capabilities using secret sharing scheme for data splitting, storage and recovery, CloudStash also relied on the above strengths to prove security of data using secret sharing schemes in a multi-cloud environment and Fabian *et al* proved resilience and robust sharing in the use of secret sharing scheme in a multi-cloud environment for data sharing.

III. OVERVIEW OF RESCUE

RESCUE is a secured threshold Cloud-based storage infrastructure using the Fragmented Secret Sharing System design philosophy, and is based on multi-cloud architecture for data storage. Replication for backup and restore of data from a primary site to other sites separated geographically which according to [2, 3, 4, 5] shows little-known potential in eliminating system downtime because of the:

- **Effects of latency on performance:** the effect of latency on performance is a source of performance lag in using replication for backup and restore of data or virtualised infrastructure from a primary site to backup sites. Using synchronized replication in a multi-cloud storage system has an increasingly large overhead, and, on the other hand, asynchronous replication reduces the integrity of the replicated data.
- **Data integrity on recovery:** quality assurance of recovered data is an issue not readily discussed in data storage and retrieval, but a very strong necessity. So checking the integrity of data after recovery is necessary to eliminate possible data corruption.
- **Consistent data availability:** data availability is key to the knowledge economy and therefore needful to mitigate factors that add large overheads to systems and thus using a robust, and all-encompassing, system is a necessity.

RESCUE is designed to handle: the latency effect on performance by defining the usage of key share mechanism rather than data sharing, when the file size is large.

It also addresses the issue of data integrity on recovery, as it is highly minimal or non-existent depending on the combination used in terms of file size, fragment size and key share policy. With regards to key share method; files are broken into chunks and the chunks in turn encrypted with AES and safely decrypted using recovered key before decrypting the chunks and combining file. In terms of shared data, data is treated as a sequence of bytes so data encoding does not matter and recovered file are cross-checked with the original file using SHA-512 hash function for data checksum. Additionally, using secret sharing scheme to split data and recover it assures data security in a keyless manner devoid of corruption as appropriate measure is put in place in the algorithm to detect wrong shares during data recovery.

A. Architecture

The architecture of RESCUE involves data/key splitting, storage, and retrievals. The two different methods involve data splitting or key splitting with data encryption and decryption using recovered key. The method implemented starts by defining the policy, which is the number of shares to be generated from each file, (N) and the least required number of shares (M) needed to come together to recover the file. The Policy of M-out-of-N, here the policy is 2-out-of-5, 3-out-of-5 and 4-out-of-5 shares, implying that for example 2 shares out-of-5 generated shares from a file are needed to recreate the file. The unique identifier is similar a magic number, unique to each session that is appended to the share when created.

B. File share

Files are scanned as in Figure 1 from a designated folder and encoded to byte streams. Using a pre-defined share policy, the encoded data is broken into shares. The shares generated are stored in separate containers and from where they are read-in and files recovered during file recovery on request.

C. Key share

Files are scanned as in Figure 1 from a designated folder as above, then using a predefined chunk size say 1024 Bytes, files are broken into the defined chunk size, encrypted and the encryption key shared as above. When the files are required, the shares generated from the key are brought together and the key recovered from where encrypted chunks are decrypted and the chunks brought together and the files are recreated.

D. Share generation

Using the equation below we generate shares:

$$f(X, A) = \sum_{i=0}^{M-1} GF\_SUM\ A[i] X^i$$

To create N shares from a secret, with a threshold of M, we will take a look on how each octet of the secret is generated.

An array A of M octets is created at first in which the array element A[0] contains a portion of the secret, while A[1], A[2],..., A[M-1] are selected independently and uniformly at random. Each share is generated by computing the value of *f*(*X*,A), where X is the share index and the resulting octet is appended to the share. A, B, C,... are arrays of *M* octets and each zero element of the array contains a portion of the share. A[0], B[0], C[0], ... are equal to first, second and third octets of the secret and so on. The power of X is the coefficient and M-1 is the threshold. GF_SUM is Galois field summation, which takes place over GF(256), different from integer addition as each addition uses the exclusive-or operation.

E. Secret Recovery

Just as in Shamir [12] authorised participants following earlier stated rules are able to recover the secret using Lagrangian interpolation once the conditions:

1. All zero elements of the array of M octets are retrieved.
2. Number of retrieved elements greater or equal to the threshold.
3. All contributed shares from participants are certified as genuine and satisfies 2 above.

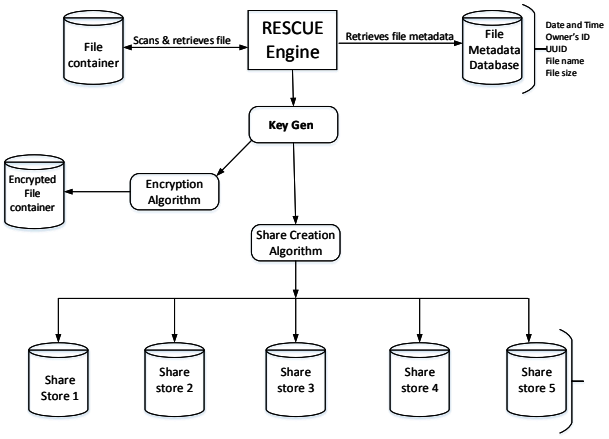

Figure 1: Key/File share creation

F. Recovery: Files

When files are to be recovered as in Figure 2, the user types in the destination folder for recovered files; the program picks up each filename, the associated values that identify the owner of the files – the UUID all in the metadata database and used the values gathered to confirm ownership and thereafter scans and retrieves all shares associated with the filename. With these file recovery is made using the Recovery algorithm.

G. Recovery: Key

Following the initial method use above in ownership identification, the system retrieves the encrypted file, recover key and use the key so recovered to decrypt the file. See Figure 3 for details.

The number of shares recovered can be less than N, but equals or greater than M (Threshold). The shares must be of equal length, else they are inconsistent. In file recovering, the output string is initialised to zero and the initial octet (share indexes are grouped in octets) of the share is stripped from each share and none of these octets are same else error will be reported, which halts the process. For each of these shares an array V of M octets is created, in which an array element V[i] contains the octet from ith share. These stripped octets are appended to the octets array U, formed by setting U[i] equals to the first octet of the *i*th share. The value of I(U, V) is computed, and appended to the output string, which is returned as the secret. This contains one fewer octet than the shares.

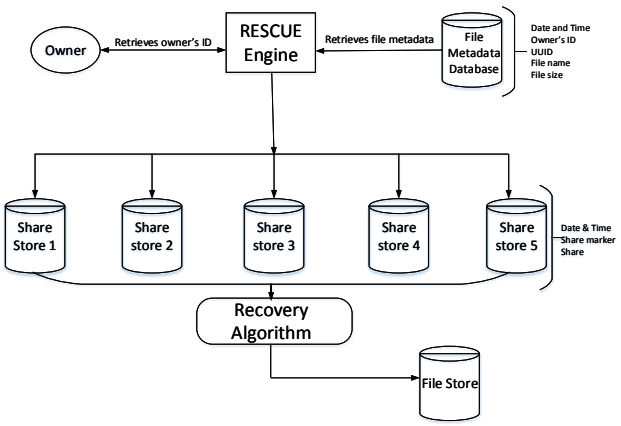

Figure 2: File recovery

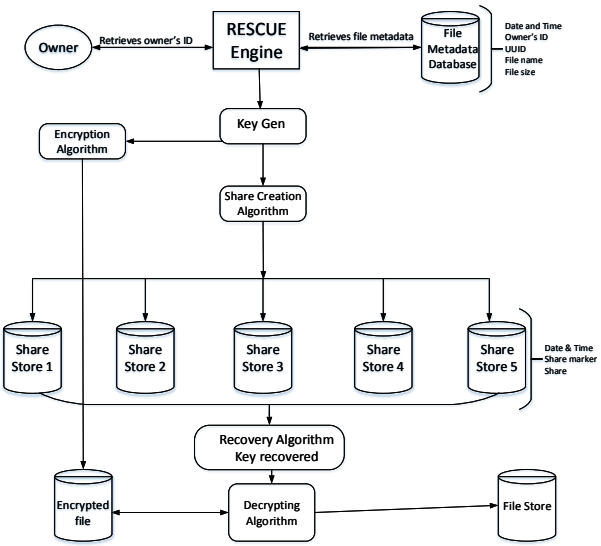

Figure 3: Key recovery and File decryption

IV. RESULTS AND EVALUATIONS

Two different sets of experiments were performed: file/data share; and key share methods. In file share, files of different sizes are created into share and stored in folders. When the files are needed, the several shares are recovered from the folders and the file recreated. Each file involved in the process is created into shares using M-out-of-N threshold secret sharing scheme and the shares stored in folders, while in key share, files of different sizes are broken into chunks; each chunk is encrypted using AES of 256-bits key length then stored in folder, the encryption key is thereafter shared, stored in folders as well.

When the files are needed, the shares are recovered from the folders for each key based on policy and the key recreated, using each key to decrypt a chunk as retrieved from the folder and the file recombined. The secret sharing scheme used is modified Social Secret sharing scheme. The issue of confidentiality and integrity in the use of secret sharing scheme has been validated by many works in secret sharing schemes such as Abdallah and Salleh [13], Buchanan et al. [14]. Since RESCUE is concentrated on Data Availability at Zero Downtime [15], the essence of the experiment is to understand all performance overheads that will impact negatively on the objective of the system so as to eliminate them or validating already known facts.

Evaluation of the results: Secret sharing schemes have been used successfully in data splitting and reconstruction, thereby providing data security in a keyless manner. This section outlines an experiment involving two main methods of secret sharing application – data sharing and key sharing. In Experiment One, figures 4 and 7 show normal curve with an increasing size of Threshold (M) and file size but figure 5 and 6 showed otherwise, a varying curve indicating the effects of Share Writing and Recovery from folders on systems performance. In Experiment Two, figures 8, 9 and 10 showed the validation of [13, 14] on the effect of increasing Threshold and file size on the system performance as in figure 8, the threshold is 2, so the overhead is with the Process not on Recovery as in figures 9 and 10. But a look at figures 11, 12 and 13 indicate entirely different results from the previous ones thus giving an understanding that there are resultant effects of file size, fragment size on share policy. The fragment size was varied in all as well as share policy using file sizes from 1KB to 1GB. In all the results shown, it is evident that using fragmented secret share system is the best option while dealing with big data infrastructure than using threshold secret sharing scheme alone, which has proved impossible to be used to scale large data infrastructure due to inherent characteristics of finite field arithmetic.

The evaluator, in this case, is the performance overhead at an increasing thresholds and data sizes. The experiments showed that Share Writing and Recovery adds more performance overhead in Experiments One, while in Experiment Two, the performance overheads of File and Fragment Sizes on Share Policy were obvious. These depict their strengths and weaknesses at different application scenarios.

The aim of the experiment is to discover all factors capable of adding performance overhead thereby derailing total system performance both in File and Key Sharing methods. Because we aim to apply the methods further in both network and cloud scenarios, we will work in eliminating the discovered factors that add to performance overhead to the system as this method has proved scalable with big data infrastructure. The test machine is a Duo Core Intel Pentium N3530 2.16GHz, 2.16 GHz, 64bit x64-based processor, Windows 8 operating system on 4GB of RAM.

Two primary sets of results are presented which use the parameters of *M=2, N=5; M=3, N=5 and M=4, N=5*. The variable *N* relates to the number of shares to create while the variable *M* relates to the number of shares required for recreation of the original arbitrary data (using each SSS algorithm). Results are presented in seconds for Time, while in KB, MB and GBs for variables file sizes. From the figures and tables presented, it can be clearly demonstrated that key share is the fastest method regardless of file sizes as well the method capable of scaling over large volumes of varying file sizes.

The key share experiment involves more stages than the previous and we therefore use the terms, **Process** and **Recover**. Process time involves time taken to split the file into chunks using a pre-defined chunk size, fragment encryption time, key share creation and writing times while Recover time involves time taken to recover key shares from folders, key recreation time, fragment decryption and file recombination times.

V. Conclusions

Experiments performed using secret sharing scheme has proved resilience in the face of failures as not all hosts are required to reconstruct data after splitting, but a major drawback remains the effect of latency on performance. This is worsened as data size increase as well as the distance between each of the hosts thus giving rise to our work. Lessons learnt are that using Key Share rather than Data Share method in combination with an appropriate fragment and share policy is the only way to scale large data infrastructure and with this lessons and validations we intend to eliminate all factors revealed as capable of adding large overhead to the system. This will provide a platform capable of achieving data availability at zero downtime.

Tables and Figures

Table 1: Share creation against policy

	Policy:	**2 from 5**	**3 from 5**	**4 from 5**
S/N	**File Size (KB)**	**Creation Time (Sec)**	**Creation Time (Sec)**	**Creation Time (Sec)**
1	1	0.106119	0.10933	0.143713
2	10	0.913352	1.075088	1.427096
3	100	1.833184	2.115918	2.461108

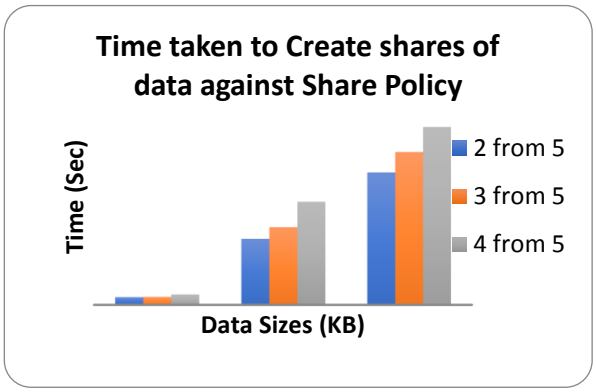

Fig. 4: **Time taken to Create share against Policy**

Table 2: Share Writing to folders against Policy

	Policy:	2 from 5	3 from 5	4 from 5
S/N	File Size (KB)	Writing Shares (Sec)	Writing Shares(Sec)	Writing Shares(Sec)
1	1	0.020532	0.03125	0.164257
2	10	0.066987	0.100468	0.03355
3	100	0.090945	0.085788	0.068099

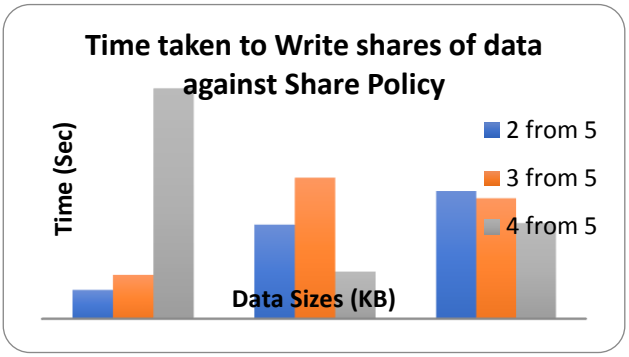

Fig. 5: Share Writing to folders against Policy

Table 3: Share Recovery against Policy

	Policy:	2 from 5	3 from 5	4 from 5
S/N	File Size (KB)	Share Recovery (Sec)	Share Recovery (Sec)	Share Recovery (Sec)
1	1	0.008113	0.004693	0.015012
2	10	0.083933	0.009362	0.005608
3	100	0.025136	0.010948	0.008912

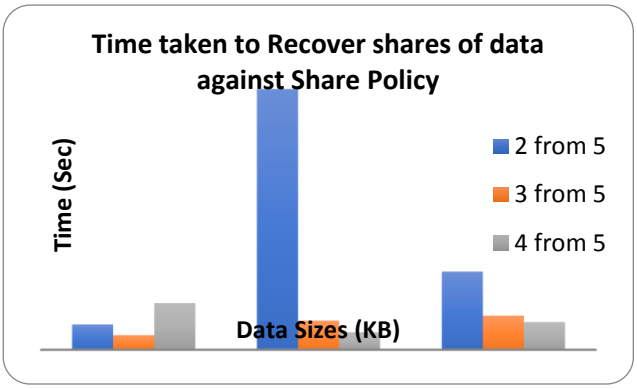

Fig. 6: Share Recovery from folders against Policy

Table 4: File Recreation against Policy

	Policy:	**2 from 5**	**3 from 5**	**4 from 5**
S/N	**File Size (KB)**	**File Recreation**	**File Recreation**	**File Recreation**
1	1	0.03405	0.054628	0.10265
2	10	0.434176	0.558682	0.92636
3	100	0.674842	1.091936	1.704002

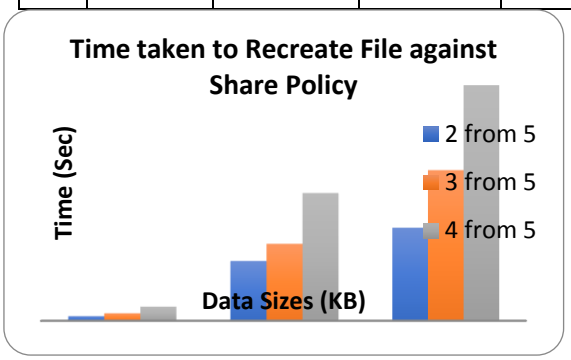

Fig. 7: File Recreation against Policy

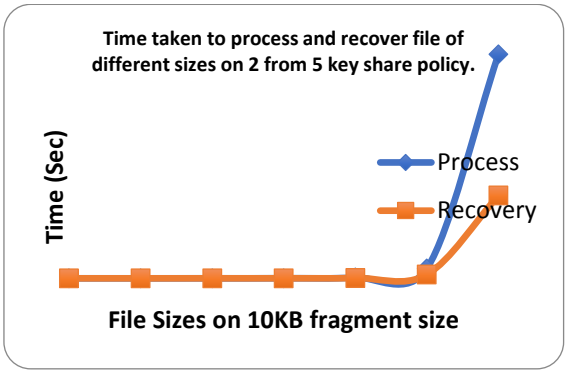

Fig. 8: Process and Recover of file using 10KB fragment size on 2 from 5 Policy.

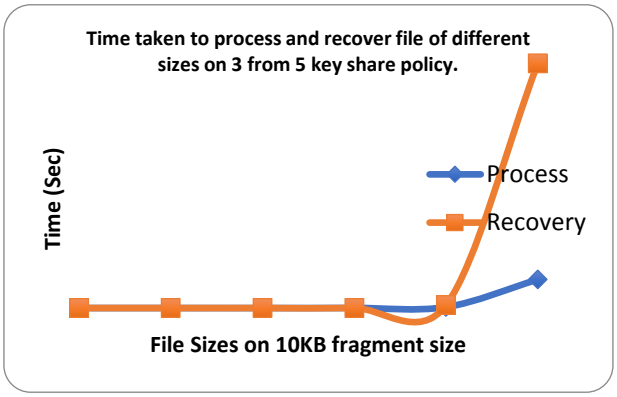

Fig. 9: Process and Recover of file using 10KB fragment size on 3 from 5 Policy.

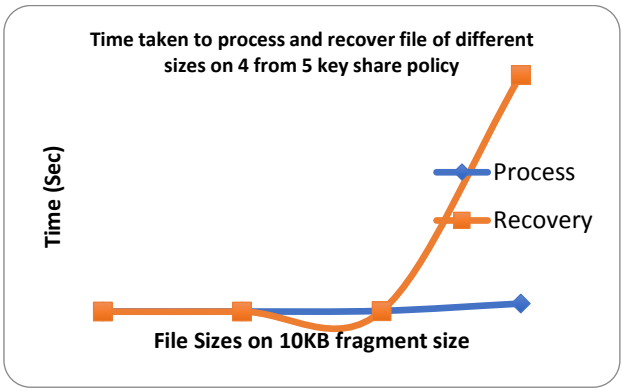

Fig. 10: Process and Recover of file using 10KB fragment size on 4 from 5 Policy.

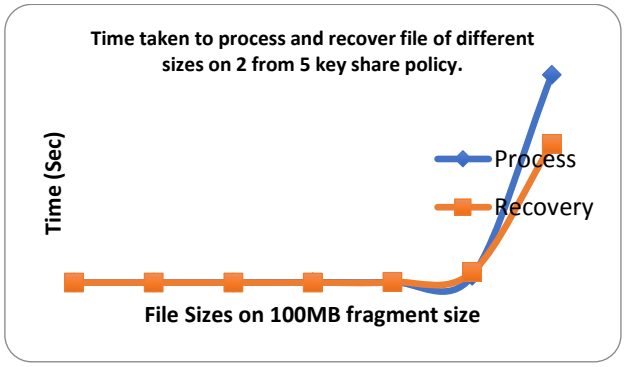

Fig. 11: Process and Recover of file using 100MB fragment size on 2 from 5 Policy.

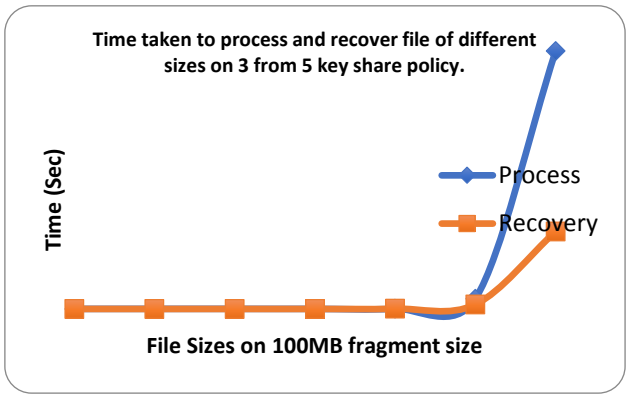

Fig. 12: Process and Recover of a file using 100MB fragment size on 3 from 5 Policy.

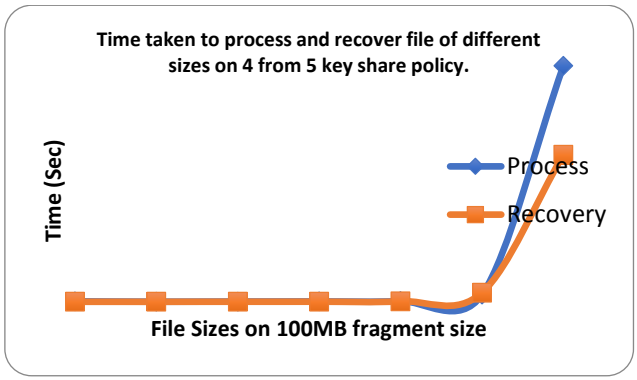

Fig. 13: Process and Recover of a file using 100MB fragment size on 4 from 5 Policy.

VI. References

[1] K. Bill, *New Study: Cost of Data Center Outages – 2016*. 2016.

[2] M. Pokharel, S. Lee, and J. S. Park, "Disaster recovery for system architecture using cloud computing," in *Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on*, 2010, pp. 304–307.

[3] J. I. Khan and O. Y. Tahboub, "Peer-to-Peer Enterprise Data Backup over a Ren Cloud," in *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*, 2011, pp. 959–964.

[4] Z. Jian-Hua and Z. Nan, "Cloud computing-based data storage and disaster recovery," in *Future Computer Science and Education (ICFCSE), 2011 International Conference on*, 2011, pp. 629–632.

[5] S. R. Patil, R. M. Shiraguppi, B. P. Jain, and S. Eda, "Methodology for Usage of Emerging Disk to Ameliorate Hybrid Storage Clouds," in *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2012, pp. 1–5.

[6] "The Great Bank Robbery: Carbanak cybergang steals $1bn from 100 financial institutions worldwide | Kaspersky Lab." .

[7] T. Loruenser, A. Happe, and D. Slamanig, "ARCHISTAR: towards secure and robust cloud based data sharing," in *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, 2015, pp. 371–378.

[8] T. Ermakova and B. Fabian, "Secret sharing for health data in multi-provider clouds," in *Business Informatics (CBI), 2013 IEEE 15th Conference on*, 2013, pp. 93–100.

[9] E. Ukwandu, W. J. Buchanan, L. Fan, G. Russell, and O. Lo, "RESCUE: Resilient secret sharing cloud-based architecture," in *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, 2015, vol. 1, pp. 872–879.

[10] F. Alsolami and T. E. 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.

[11] B. Fabian, T. Ermakova, and P. Junghanns, "Collaborative and secure sharing of healthcare data in multi-clouds," *Inf. Syst.*, vol. 48, pp. 132–150, 2015.

[12] A. Shamir, "How To Share a Secret," *Commun. ACM*, vol. 22, no. 1, pp. 612–613, 1979.

[13] A. Abdallah and M. Salleh, "Analysis and comparison the security and performance of secret sharing schemes," *Asian J. Inf. Technol.*, vol. 14, no. 2, pp. 74–83, 2015.

[14] W. Buchanan, D. Lanc, E. Ukwandu, L. Fan, and G. and, "The Future Internet: A World of Secret Shares," *Futur. Internet*, vol. 7, no. 4, pp. 445–464, 2015.

[15] E. Ukwandu, W. J. Buchanan, and G. Russell, "TCloud: Availability at Zero Downtime." Retrieved from http://thecyberacademy.org/wp-content/uploads/2016/05/PGCS-symposium_2016_paper_4.pdf, 2016.