Poster: RESCUE: The New Approach to Cloud-based Disaster Management

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

Context: There are many risks in moving data into public cloud environments, along with an increasing threat around large-scale data leakage. This includes weaknesses around the loss of encryption keys and also for a large-scale data loss on an outage. The current practice in cloud-based disaster management see disaster management as a way of bringing system back online after the disasters not as a way of mitigating such occurrence by providing a full-proof resilient system that can withstand disaster scenarios thereby providing a zero system downtime. Overall this work aims to apply secret sharing methods as used in cryptography alongside data fragmentation technique to create robust and secure Cloud-based data storage, including mitigating disaster and providing self-healing.

Related work: Nojoumian *et al* defined a principle of a social share, which is a scheme used to break data into shares and distribute to hosts dynamically based on reputation and their interactions with other hosts. Their work applied the method in cloud computing to create a self-organising system. A drawback of this approach is that the distribution is based on a calculated trust function and in disaster management each host is contracted based on service level agreement (SLAs), which cannot be fully trusted, and moreso, disaster cannot be judged most often using social function as some are natural while others are man-made. We therefore tend to modify the system by removing the aspect of trust function and replacing with host monitoring for efficient and timely information on the host performances during system interactions. The essence is to implement a break-glass system in a case of adverse host failures.

Contribution: This work outlines a method to split large sized data into chunks based on a specified chunks size, encrypt the chunks, and use secret sharing to share encryption key based on the secret sharing policy of say 3 from 5 and distribute equal number of encrypted chunks and shares to each host and using a host monitor provide information on access denial rate, which helps to implement a break-glass mechanism in a case of a predefined adverse denial rate. This aims to improve on data availability, mitigate losses, eliminate key management problem and provide self-healing. In the case the data size is not large, the data is broken into shares using a secret sharing policy as stated above and equal shares distributed to each host based on the number of host's subscription. The policy determines the number of shares made out of secret (data/key) and the required number of hosts (threshold) capable of reconstructing the secret, whereas any number less than the threshold cannot. Using the concept of modified social secret sharing scheme, the share hosting will be dynamic based on certain predetermined factors. This implies that though shares are distributed evenly, accessing each share from a host will be dynamic based on prevailing factors as reported by the host monitoring.

Implementation: The implementation involves providing data security in a keyless manner, with in-built failover protection, to build on self-healing, consistent data availability using a dynamic share hosting and resilience. The system will scale up the experiment using public clouds and with a key metrics of effects of latency on performance, keyless security and availability.

Results

Table 1: Share Creation against Policy

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

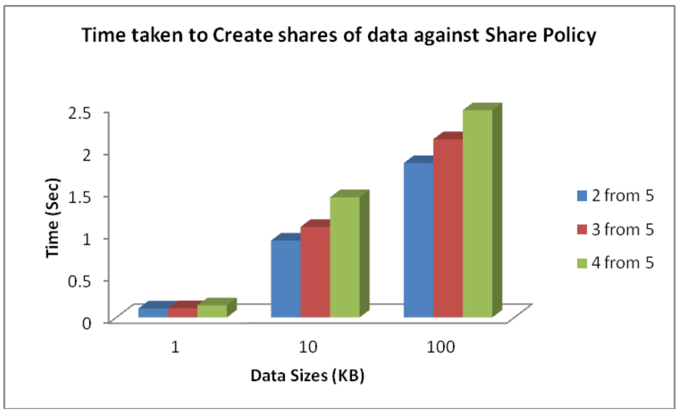

Fig. 1: Time taken to Create shares of data against Share Policy

Table 2: Writing shares against Policy

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

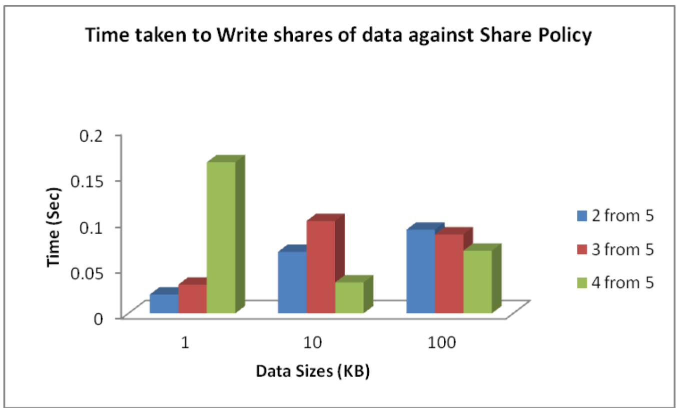

Fig. 2: Time taken to Write shares of data against Share Policy

Table 3: Share recovery against Policy

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

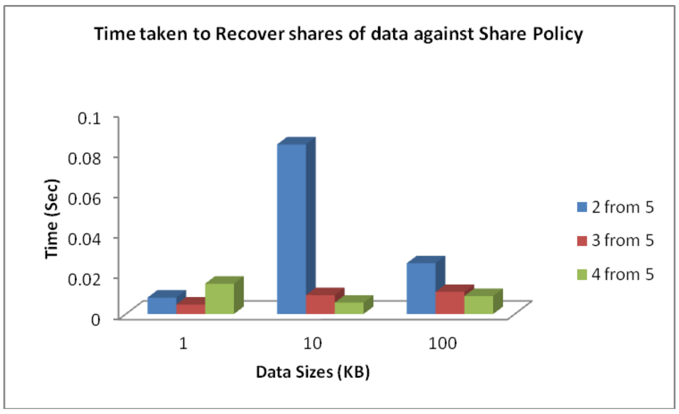

Fig. 3: Time taken to Recover shares of data against Share Policy

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

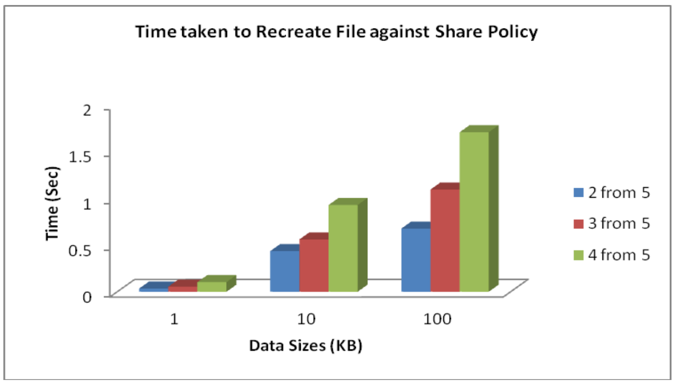

Fig. 4: Time taken to Recreate File against Share Policy

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 data sharing, every data is treated as sequence of bytes so file types does not matter and file size must not exceed certain threshold as Secret sharing algorithm is built on finite field arithmetic known as Galois field (2^{16}). The evaluator in this case is the performance overhead at an increasing thresholds and data sizes. The results in figures 1 and 4 show normal distribution, but that of 2 and 3 did not, indicating the effect of share policy on the system. The experiment shows that decreasing the threshold rather adds more overhead to the system than expected as indicated in figure 3. More experiments have been performed on using Data Splitting and Recreation for large sized data in conjunction with Secret sharing scheme in safe guarding encryption key but cannot be used here for space. Increasing the size of data adds more performance overhead while key sharing shows slightly higher overhead in key recovery and file decryption than file encryption and key sharing. These depict their strengths and weaknesses at different application scenarios.

The aim of the experiment is to demonstrate the implication variance in data size, and key sharing policy have on the performance of secret sharing scheme (SSS) algorithm in terms of share creation and share recreation in case one wants to apply any in cloud-based designs and varied data sizes were evaluated. 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.

Four primary sets of results were presented which use the parameters of *N=2, M=5; N=3, M=5; and N=4, M=5*. The variable *N* relates to the number of shares required for recreation of the original arbitrary data (using each SSS algorithm), while *M* relates to the number of shares to be created. Results are presented in seconds for Time taken, while KB for Data sizes. From the figures and tables presented, it can be clearly demonstrated that key share policy has effect on the time taken to write and recover shares and therefore important factors in choosing location of cloudlets to host shares.