ISSN 2348 – 7968

Secret Sharing Scheme: Securing data in multi-clouds using systmatic Secret sharing methods

Shalini K B[1], Shashikala S V[2] Shruthika C A[3]

[1] P.G.Scholor,Dept.Of CSE, B.G.S Institute of Technology,
B.G.Nagar,Karnataka,India

[2] Professor,Head Dept.Of CSE, B.G.S Institute of Technology,
B.G.Nagar,Karnataka,India

[3] Assistant Professor Dept.of CSE,B.G.S Institute of Technology,
B.G Nagar,Karnataka,India

Abstract

Cloud storages have many exciting features that attract many individuals and organizations for storing and sharing data over the cloud. However, security and key management are still remaining the highlighted concerns in cloud storage. Managing/protecting keys is a problem for existing approaches, and opens the risk of attackers working to offline brute-force crack the decryption and/or surreptitiously obtaining the key and using it offline. To address these issues, we propose the CloudStash scheme, a system that applied the secret-sharing scheme directly on the file to store multi-shares of a file into multi-clouds. CloudStash utilizes secret-sharing, low cost cloud storages and multi-threading to improve confidentiality, availability, performance and fault tolerance. CloudStash achieves this improvement by splitting a file into multi-shares of secret and distributing these multi-shares into multi-clouds simultaneously where threshold shares are required to reconstruct the file. Our experiments show that CloudStash is statistically significantly faster for small files, and even for large files the added cost is not statistically worse. So the added security benefits are nearly free from the users' perspective.
.

Keywords:Cloud storage security; secret-sharing; multi-clouds; performance; key management,cloudstash;

1. Introduction

Cloud storage has many attractive features such as low cost, ease of use, and unbounded resources for storing and sharing data among multiple individuals and organizations. However, there is a highlighted concern about security in the clouds that reluctant many individuals and organizations form utilizing the cloud [1] [2]. Protecting data in the cloud is generally accomplished with standard encryption, which leads to issues with key management and insider/brute force attacks [3]. Insider attacks remain a critical issue for the cloud, where the employees working on the cloud storage can illegally access and obtain copies of the data stored in the cloud even though it is encrypted [4] [5]. Using weak encryption keys may allow an attacker to apply brute force attacks on encrypted data and get the data [6] [3]. If insiders or attackers can brute-force guess the key (or password protect it), or if they can obtain a key that is reused, they can get the data [3]. Key management is also an issue for encryption-based protection in cloud computing and must deal with key generation, key distributions, key storing, key disabling, key protection and key regeneration among multi-users [7] [8]. The security of current cloud

This storage schemes depends on how users protect their keys. If a user leaves his key without protection or if he uses weak password, this may allow an attacker to compromise it.

Beyond confidentially, there are also issues of availability [9]. Incidents occur in operational cloud storages, for exam-ple, in June 2011, the clients' authentications were corrupted when the Dropbox system was updated [10]. Another example was when customers' data were corrupted due to a disposition of new defective load balancer in Amazons S3 [11].

To protect the clouds from such incidents, many secure cloud storage schemes appear in the literature. However, these schemes still face many security challenges. One of the biggest challenges of these schemes is the reliance on the availability of the cloud itself. For example, first, a single cloud storage might be down for maintenance, or due to internet connections failure [6]. Second, a single cloud storage might be attacked by the denial-of-service attack [2]. Finally, one cloud storage might become dominant with data lock-in where clients become dependent on a single cloud storage provider, and it is expensive to move to another cloud storage provider [12].

In this paper, we propose CloudStash, a multi-cloud scheme that utilizes the secret-sharing scheme [13] to directly protect data for low cost cloud storage and multi-threading to address confidentiality, availability and perfor-mance in cloud storage. Specifically, our contribution is to design, implement, and evaluate a CloudStash scheme comparing it with the approach used in state of the art system wherein data is encrypted and stored in the multi-cloud and the standard encryption key is protected via secret sharing. Using shares to separately protect the key initially seems as if might be faster. However, as we shall show, it is often slower, as it requires multiple rounds of cloud access. Furthermore, it also leaves the data open to an attack on the key.

First, CloudStash provides confidentiality by applying Secret-Sharing Scheme directly on a file that is split into multi-shares. Then it distributes these multi-shares over multiple clouds. CloudStash guarantees that no single cloud stores the shares needed for recovery. Thus, an in-sider/attacker would have to compromise multiple clouds to have the data necessary to even launch a brute-force attack. If desired, the user can combine this with encryption while the share ensures no insider attacks and no brute-force attacks against partial data.

Second, CloudStash bypasses key management issues [7] [8]. Instead of using storing/managing keys in untrusted clouds, or relying on the users protections of keys, Cloud-Stash directly applies the secret-sharing scheme [13] to provide confidentiality and avoid key management issues.

Finally, CloudStash provides availability by distributing multi-shares over multi-clouds and not depending on a single cloud’s availability. CloudStash provides performance by us-ing multi-threading to manage multi-shares into multi-clouds in parallel. CloudStash provides integrity and fault tolerance by hashing and signing each share and then distributing these signed multi-shares into multi-clouds. When the downloaded share is corrupted, CloudStash can regenerate a new share from different cloud.

This paper’s content is organized as follows: section II discusses the background, section III evaluates CloudStash goals, and section IV describes CloudStash design and algorithm. Section V discusses experimental evaluation and conclusion in section VI.

2. Background

This section discusses the prior art work done in the area of confidentiality, key management, and availability in cloud storage. Then we compare the previous art work with the CloudStash scheme.

2.1 Confidentiality

To provide confidentiality, single cloud schemes such as CloudProof [14], Cryptonite [15], and CloudSeal [16] rely on symmetric encryption tools. However, multi-clouds schemes such as DepSky [17], Intercloud [18], and N-Cloud [19] not only use symmetric encryption tools, but also distribute encrypted data over multi-clouds. In this distribu-tion, each cloud stored part of the encrypted data provides a high level of confidentiality. DepSky [17] used DepSky-CA protocol to encrypt the data with random secret key. They encoded the encrypted data and shared key, then distributed them into four clouds where each cloud stored block of data along with key share. Intercloud [18] performed symmetric encryption on the data, and split the key into shares based on the secret-sharing scheme, then they attached key shares to piece of data in order to distribute them to clouds. N-Cloud [19] performed symmetric encryption on each chunk of a file and then distributed multi-chunks into multi-clouds with different combination where each cloud is missing one chunk. CloudStash provides confidentiality by applying secret-sharing scheme directly on a file to split it into multi-shares and then distributing these multi-shares over multiple clouds in parallel. CloudStash guarantees that no single cloud stores the shares needed for recovery.

2.2. Key Management

Secure Cloud storage scheme in general relies on en-cryption/decryption keys to protect users’ data in the cloud. Dealing with these keys raises key management issues such as how to store the keys, how to distribute the keys between multi-users and how to protect the keys. Zissis et al. [7] argued that the main key management issues are regarding key generation, key distributions, key storing, key disabling, key protection and key regeneration. Bjorkgvist et al. [8] proposed key lifecycle key management system (KLMS) to manage the task for keys and form strict access control. Intercloud [18] used KLMS for its key management. CloudProof [14] introduced family block that contains data block and key block. CloudProof [14] relied on broadcast encryption [20], and key rotation [21] to distribute keys to users who are listed in ACL. Users with read access key permissions can read the data block while users with write access keys permissions can update the data block. Cryptonite [15] introduced strongbox file to manage users keys in the Azure Web. Strongbox file used broadcast encryption to distribute read keys to authorized readers and distribute shared signature keys to authorized writers. CloudSeal [16] integrated symmetric encryption, proxy-based re-encryption [22] and the secret-sharing scheme to manage users access control and protect content in the cloud. CloudSeal [16] provided both, forward security and backward security; forward security where clients cannot access the content in the cloud before joining the group, and backward security where clients cannot access the content in the cloud after leaving the group. DepSky [17] and Intercloud [18] utilized the secret-sharing scheme and multi-clouds to distribute encrypted data with shares of key over multi-clouds without managing keys services. N-Cloud [19] manages the encrypted keys at client-side, and does not save the encrypted keys in the cloud. However, CloudStash bypasses key management issues [7] [8]. Instead of using storing/managing keys in untrusted clouds, or relying on the users protections of keys; CloudStash directly applies the secret-sharing scheme [13].

2.3. Availability

Single cloud schemes such as CloudProof [14], Cryp-tonite [15], and CloudSeal [16] relied on a single cloud's availability. However, multi-clouds schemes such as DepSky [17], HAIL [23], and N-Cloud [19] provided a high level of availability by distributing data over multi-clouds. DepSky [17] used four commercial clouds and distributed data over four clouds where each cloud stored half of the data. HAIL [23] utilized RAID-like techniques [24] to distribute data over multiple clouds in order to provide more availability and integrity of the data stored in the cloud. N-Cloud [19] divided a file into small chunks and distributed these small chunks over multi-clouds where each cloud is missing at least one chunk, and each chunk is being replicated over N-1 out of N cloud storages. CloudStash provides availability by distributing multi-shares over multi-clouds and not depending on a single cloud's availability.

3.CloudStash Scheme Objectives

CloudStash utilizes the secret-sharing scheme [13] and low cost cloud storages to provide confidentiality. Cloud-Stash splits a file into multi-shares and distributes these multi-shares over multi-clouds. Each share is a part of the secret, and it is not giving any full information about the file. CloudStash ensures that no single cloud stored threshold shares to address insider attacks [4] [5]. An attacker cannot break the confidentiality of CloudStash un-less he compromises M cloud where M is secret sharing threshold M which may be up to the number of shares N. This approach is quite different from the previous work of a single cloud schemes. If an attacker compromises the cloud in a single cloud, he can get the whole encrypted file and run brute force attack trying to break the weak keys encryption. In pervious multi-cloud approaches, if an attacker compromises the cloud, he can obtain part of the data. However, in CloudStash an attacker cannot get any information unless he can compromise all clouds and get the threshold shares. Moreover, insider attacks are less harmful for CloudStash in the sense that each cloud storage only stores one share, which is meaningless for an attacker to get. In other schemes, an insider attack can try to brute force break the key for getting the whole file in case of a single cloud schemes or part of file in case of multi-clouds schemes.

CloudStash utilizes the secret-sharing scheme and low cost cloud storages, not only to provide confidentiality but also to avoid key management issues. In CloudStash, an attacker must compromise all clouds in order to obtain the file, making it very difficult and costly to achieve.

CloudStash provides availability by utilizing the secret-sharing scheme and low cost cloud storages to split a file into multi-shares and distribute these multi-shares over multi-clouds. When one cloud storage is either attacked or unavailable [6], clients can retrieve their data from other available clouds by reconstructing threshold shares. CloudStash ensures that the number of threshold shares is less than the total number of clouds; in fact CloudStash does not rely on a single cloud's availability.

CloudStash utilizes multi-threading in uploading and downloading operations in order to provide a high perfor-mance.

CloudStash provides integrity and fault tolerance by sign-ing each share and distributing each share into a differ-ent cloud storage. CloudStash can then verify each share before reconstructing the file. If one or more shares get corrupted, CloudStash can regenerate those shares from different clouds.

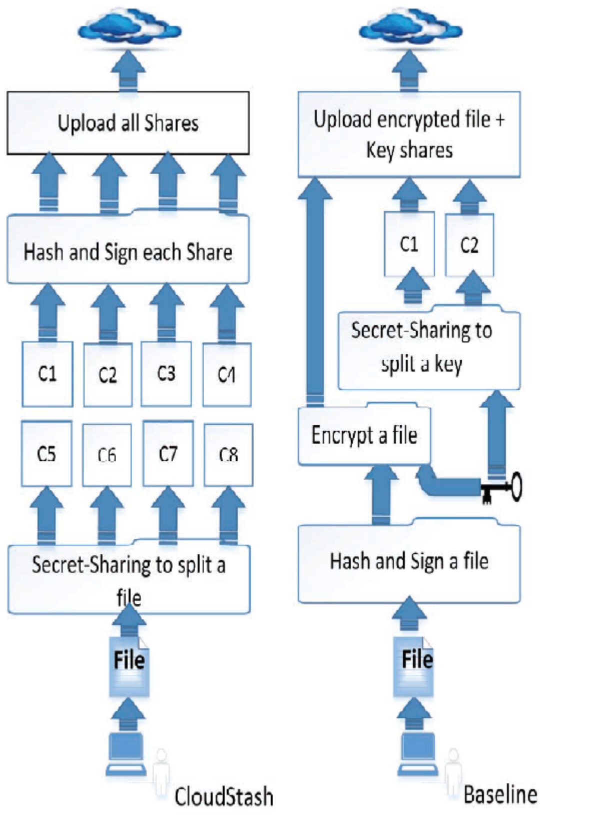

Figure 1. Architecture of CloudStash and baseline in uploading operation.

4. Design and Algorithm Analysis

4.1. Upload Operation

```
Data: A file as a plaintext
Result: Shares Ni stored in cloud storages Si
 for (each file) do
      apply secret-sharing scheme to split a file into shares
      Ni where i=1,2,. . . ,N;
      determine number of cloud storage Si;
      determine number of shares Ni;
      determine number of threshold Mi;
      for (each shares Ni) do
            hash and sign each share Ni;
       end
      for (all shares Ni and signatures of each share) do
      upload all shares Ni and signature into cloud storage
      Si in parallel;
      end
end
```

Algorithm 1: Algorithm of uploading operation of Cloud-Stash.

4.2 Download Operation

5.Experimental Evaluation

5.1. Implementation

The experiments compare two systems: our baseline and CloudStash. We conducted the experiments in python with Amazon AWS S3 API and used different sizes of files as

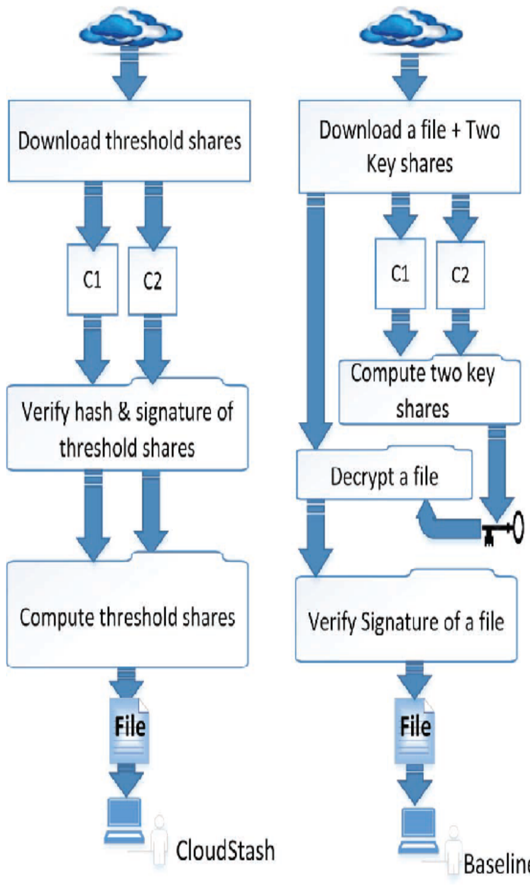
Download threshold shares
C1
C2
Verify hash & signature of threshold shares
Compute threshold shares
File
CloudStash
Download a file + Two Key shares
C1
C2
Compute two key shares
Decrypt a file
Verify Signature of a file
File
Baseline

Figure 2. Architecture of CloudStash and baseline in downloading operation.

```
Data: Shares Ni stored in cloud storage Si
Result: An original file as a plaintext
for all (cloud storage Si and shares Ni ) do
          choose two cloud storages S1 and S2 out of Si;
          for (each cloud storage S1 and S1) do
                    download all threshold shares Mi and
                    signatures in parallel;
         end
     for (all threshold shares Mi) do
          verify hash and signature of all threshold shares
```

```
            Mi;
            if (threshold shares Mi  is corrupted) then
                    Re download that threshold shares Mi
                    and signatures;
            end
        end
        apply secret-sharing scheme to reconstruct the
        original file by usingl threshold shares Mi;
end
```

Algorithm 2: Algorithm of downloading operation of CloudStash

(1KB, 10KB, 100KB, 1MB, and 10MB). We used eight cloud storages centers of Amazon AWS S3 (N. Virginia, Oregon, N. California, Ireland, Singapore, Tokyo, Sydney, and Sao Paulo), and we conducted the two experiments on a machine with processor Intel core i5 CPU 2.40 GHz, 4GB RAM and 64 bit Linux Operating System. The results of these experiments are the average of at least twenty runs. For internet connection, we used Comcast home internet, with a measured upload bandwidth of 11.65 Mbps and download bandwidth is 37.12 Mbps.

1)Baseline Setup: The baseline algorithm used symmet-ric encryption AES for encryption with CFB mode, SHA512 for hashing, and RSA 1024 for signature on the whole file (without splitting). Also, we used the secret-sharing scheme on the key to split it into shares [13]. During an upload operation, we singned, hashed and encrypted the file and then split the key into shares by using the secret-sharing scheme. Then we use multi-threading, to upload the file along with each share of the key in parallel into eight cloud storages, where each cloud storage stored one file along with one share of the key. During a download operation, we downloaded a file from one cloud storage then downloaded the two key shares (threshold shares of the key) from two cloud storages. We then computed the two shares of key and decrypted the file. To achieve fair comparison to the CloudStash, we used the baseline experiment to compare it againt the CloudStash, since this baseline is similar to some multi-clouds storage schemes found in literature (i.e. DepSkay [17]).

2)CloudStash Setup: The CloudStash algorithm used the secret-sharing scheme [13] directly on the file itself instead of the key to split the file into multi-shares. The experiment then used SHA512 for hashing and RSA for signature on each share. Mainwile, multi-threading program was used to upload the multi-shares in parallel into eight cloud storages, where each cloud storage was used to store one share of the file. During a download operation, multi-threading was used to download two shares from two cloud storages that the threshold shares, and then computed two shares to reconstruct the file back.

5.2 Evaluation

First, we seek to prove that applying the secret-sharing scheme directly on files will not cost more than applying symmetric encryption on files and applying the secret-sharing scheme on the keys.

Then we needed to analyze if CloudStash achieves its goal by measuring the total time (includes computation time and communication time) for upload operation and download operation and compared it to state-of-the-art baseline. In the experiment, we use multi-threading to upload/download in parallel.

The computation time for an upload operation in Cloud-Stash includes the time for applying the secret-sharing scheme [13] on the file and split it into shares and calculating the hash and signature for each share. The computation time for a download operation in CloudStash, includes the time for verifying the hash, and signature for each share. Also, include the time for applying the secret-sharing scheme on the threshold shares to combine them to get the file back.

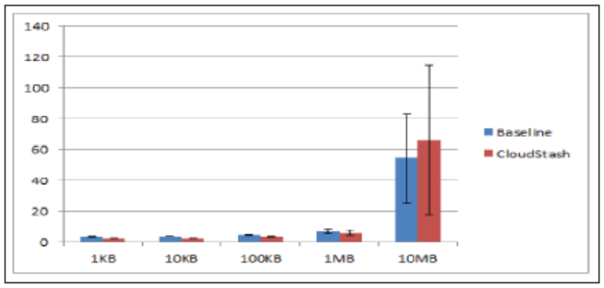

Figure 3. The average total uploading times (with standard deviation as error bars) comparison between two algorithms (Baseline and CloudStash) for (1KB, 10KB, 100KB, 1MB, and 10MB) files.

The computation time for upload operation in the baseline includes the time for calculating the hash and signature for each file. It also includes the time for encrypting the file using AES symmetric encryption, and applying the secret-sharing scheme on the key to split into multi-shares. The computation time for download operation in the baseline includes the time for verifying the hash and signature for each file. It also includes the time for applying the secret-sharing scheme on the key threshold shares to combine them to get the key back. Finally, use the key to decrypt the file using AES symmetric decryption to get the plaintext file back.

The communication time for the upload operation in CloudStash is the time to transfer each share form the client machine to the cloud storage. The communication time for download operation in CloudStash is the time to transfer the share from the cloud storage to the client machine.

The communication time for upload operation in the baseline, is the time to transfer each file and the key shares form the client machine to the cloud storage. The communication time for download operation in the baseline is the time to transfer the file and the key shares from the cloud storage to the client machine.

Table I

THE AVERAGE UPLOADING TIMES, STANDARD DEVIATION AND T-TEST COMPARISON BETWEEN TWO ALGORITHMS (BASELINE, AND CLOUDSTASH) FOR (1KB, 10KB, 100KB, 1MB, AND 10MB) FILES. THE P-VALUE FOR THE TEST FOR ALL BUT 10MB ARE STATISTICALLY SIGNIFICANT, REJECTING THE NULL-HYPOTHESIS AND SUPPORTING THE CLAIM.

File / Scheme	1KB (sec)		10KB (sec)		100KB (sec)		1MB (sec)		10MB (sec)	
Measure	Ave.	Std.	Ave.	Std.	Ave.	Std.	Ave.	Std.	Ave.	Std.
Baseline	3.34	0.19	3.40	0.10	4.55	0.22	6.76	1.45	54.25	28.83
CloudStash	2.34	0.07	2.41	0.06	3.32	0.17	5.84	1.71	66.09	48.56
T-test P-value	5.95305E-18		1.05836E-27		5.55771E-20		0.036294		0.177958	

Figure 3 and Table I show the average total upload time for all files (1KB, 10KB 100KB, 1MB, and 10MB) and standard deviation over the 20 runs. CloudStash achieved improvements in all small/mid files sized compared to the baseline. We observed that the improvement of CloudStash achievement in total upload times for all small/mid files.

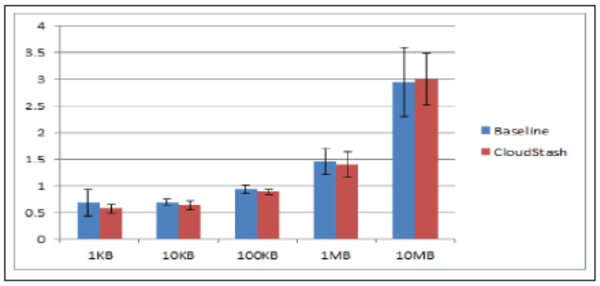

Figure 4. The average downloading times (with standard deviation as error bars) comparison between two algorithms (Baseline, and CloudStash) for (1KB, 10KB, 100KB, 1MB, and 10MB) files.

size is 42.73% improvement in 1KB file size, 41.07% improvement in 10KB file size, 37.04% improvement in 100KB file size, and 15.75% improvement in 1MB file size. However, CloudStash incurred 21.82% overhead in big file like 10MB.

For formal testing, the null hypothesis Ho is that the time for the baseline is less than or equal to CloudStash, i.e. the baseline is better. Table I shows the P-values from T-test rejecting the null hypothesis over 20 run, using a one-sided two-sample unequal variance (heteroscedastic) t-test. We can reject the null hypothesis for files less than or equal to 1MB, and we conclude that CloudStash outperforms the baseline in all small/mid files sizes in upload operation. For larger files, we cannot reject the null hypothesis and for big files sizes, such as 10MBfile. The baseline used symmetric encryption on files, and the secret-sharing scheme on keys which increase overhead time in small/mid files sizes compared to the CloudStash while has lower overhead for big files.**Title:** The title should be centered across the top of the first page and should have a distinctive font of 18 points Century. It should be in a bold font and in lower case with initial capitals.

Table II

THE AVERAGE DOWNLOADING TIMES, STANDARD DEVIATION AND P-VALUES FROM T-TEST COMPARISON BETWEEN TWO ALGORITHMS (BASELINE, AND CLOUDSTASH) FOR (1KB, 10KB, 100KB, 1MB, AND 10MB) FILES. THE P-VALUE FOR THE TEST FOR 1KB, 10KB, AND 100KB FILES ARE STATISTICALLY SIGNIFICANT, REJECTING THE NULL-HYPOTHESIS AND SUPPORTING THE CLAIM.

File / Scheme	1KB (sec)		10KB (sec)		100KB (sec)		1MB (sec)		10MB (sec)	
Measure	Ave.	Std.	Ave.	Std.	Ave.	Std.	Ave.	Std.	Ave.	Std.
Baseline	0.67	0.25	0.69	0.06	0.94	0.08	1.46	0.24	2.95	0.65
CloudStash	0.57	0.08	0.64	0.08	0.89	0.05	1.39	0.24	3.00	0.48
T-test P-value	0.025543041		0.022457642		0.030924002		0.186557		0.379437	

Figure 4 and Table II show that CloudStash achieved improvements in all small/mid files sizes in total download time compared to the baseline. We observed significant improvement for CloudStash in small files. CloudStash achieved the following improvements: 17.54% in total down-load times in 1KB file size, 7.8% in 10KB file size, 5.61% in 100KB file size, and 5.03% in 1MB file size. However, CloudStash incurred 1.69% overhead in 10Mb file size. Table II shows the p-value for t-test result. From the results above, we conclude that CloudStash outperforms the state- of-the-art baseline for small/mid file downloads and is not significantly different the baseline in all small/mid files with no statistically difference for larger files.

The conclusion that we be drawn from this experi-ment is that CloudStash presents statistically significant improvements in small/mid files sizes during both an upload operation and a download operation. On larger files, the algorithms large variance in communication time results in the baseline is failing to be statistically significantly better. Together these supports our claim that applying the secret-sharing scheme directly on the files will not cost more than applying symmetric encryption on the files and applying the secret-sharing scheme on the keys.

6. Conclusions

In conclusion, CloudStash shows a significant performance improvement in addition to security improvement and fault tolerance. Our experiment shows that applying the secret-sharing scheme directly on the file instead of the key was statistically faster for small/mid files sizes comparing to the state-of-the-art approach that use the secret-sharing scheme on the key after decrypted the file using symmet-ric encryption. In addition to performance improvements, CloudStash also addressed insider/ brute force attacks and avoided key management issues that occurred on other schemes.

Acknowledgments

This research is a combined effort of me and our guide of BGS Institute of Technology. I would like to gratefully acknowledge my organization, HOD, and our guide for their valuable suggestion and support.

References

[1] G. Anthes, “Security in the cloud,” Communications of the ACM, vol. 53, no. 11, pp. 16–18, 2010.

[2] S. Subashini and V. Kavitha, “A survey on security issues in service delivery models of cloud computing,” Journal of Network and Computer Applications, vol. 34, no. 1, pp. 1–11, 2011.

[3]K.-K. R. Choo, “Cloud computing: challenges and future directions,” Trends and Issues in Crime and Criminal Justice, no. 400, p. 1, 2010.

[4]C. Cachin, I. Keidar, and A. Shraer, “Trusting the cloud,” Acm Sigact News, vol. 40, no. 2, pp. 81–86, 2009.

[5]F. Rocha and M. Correia, “Lucy in the sky without diamonds: Stealing confidential data in the cloud,” in Dependable Sys-tems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on. IEEE, 2011, pp. 129–134.

[6]M. A. AlZain, E. Pardede, B. Soh, and J. A. Thom, “Cloud computing security: from single to multi-clouds,” in System Science (HICSS), 2012 45th Hawaii International Conference on. IEEE, 2012, pp. 5490–5499.

[7]D. Zissis and D. Lekkas, “Addressing cloud computing se-curity issues,” Future Generation Computer Systems, vol. 28, no. 3, pp. 583–592, 2012.

[8]M. Bjorkqvist,¨ C. Cachin, R. Haas, X.-Y. Hu, A. Kurmus, R. Pawlitzek, and M. Vukolic,´ “Design and implementation of a key-lifecycle management system,” in Financial Cryp-tography and Data Security. Springer, 2010, pp. 160–174.

[9]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.

[10]A. Ferdowsi, “Yesterday’s Authentication Bug,” 2011. [Online]. Available: https://blog.dropbox.com/2011/06/ yesterdays-authentication-bug/

[11]A. Ferdowsi, “Amazon Discussion Forums: S3 data corruption?” 2008. [Online]. Available: https://forums.aws. amazon.com/thread.jspa?threadID=22709#

[12]H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, “Racs: a case for cloud storage diversity,” in Proceedings of the 1st ACM symposium on Cloud computing. ACM, 2010, pp. 229–240.

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

[14]R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang, “Enabling security in cloud storage slas with cloudproof,” in Proc. USENIX ATC, 2011.

[15]A. Kumbhare, Y. Simmhan, and V. Prasanna, “Cryptonite: a secure and performant data repository on public clouds,” in Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012, pp. 510–517.

[16]H. Xiong, X. Zhang, D. Yao, X. Wu, and Y. Wen, “Towards end-to-end secure content storage and delivery with public cloud,” in Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, 2012, pp. 257– 266.

[17]A. Bessani, M. Correia, B. Quaresma, F. Andre,´ and P. Sousa, “DepSky: dependable and secure storage in a cloud-of-clouds,” in Proceedings of the sixth conference on Computer systems. ACM, 2011, pp. 31–46.

[18]C. Cachin, R. Haas, and M. Vukolic, “Dependable storage in the intercloud,” IBM Research, vol. 3783, pp. 1–6, 2010.

[19]F. Alsolami and C. E. Chow, “N-cloud: Improving perfor-mance and security in cloud storage,” in High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on. IEEE, 2013, pp. 221–222.

[20]A. Fiat and M. Naor, “Broadcast encryption,” in Advances in CryptologyCRYPTO93. Springer, 1994, pp. 480–491.

[21]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.

[22]G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved proxy re-encryption schemes with applications to secure distributed storage,” ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 1, pp. 1–30, 2006.

[23]K. D. Bowers, A. Juels, and A. Oprea, “Hail: a high-availability and integrity layer for cloud storage,” in Pro-ceedings of the 16th ACM conference on Computer and communications security. ACM, 2009, pp. 187–198.

[24]D. A. Patterson, G. Gibson, and R. H. Katz, A case for redundant arrays of inexpensive disks (RAID). ACM, 1988, vol. 17, no. 3.

Shalini.K.B received her B.E degree in Information Science from SJBIT under Visvesvaraya Technological University Karnataka in 2013 and currently she is a post graduate student pursuing M.Tech in Computer Science and Engineering from B.G.S Institute of Technology under Visvesvaraya Technological University Karnataka.
She has presented 2 papers National Conference level and 1 paper intrenational coneference level. Her main research interests include computer networks and cloud computing as well as wireless sensor network and storage area network.

Shashikala.S.V received her B.E degree in Computer Science from Mysore University, Karnataka in 1990 and M. Tech in Computer Science and Engineering from Visvesvaraya Technological University in 2005; she is currently working towards her PhD degree in the area of Big data over a multipath TCP.
She is a Professor and Head in the department of computer science at B.G.S Institute of Technology; B.G.Nagar Mandya and having totally 22 years of teaching experience. She has presented 4 papers in National level conferences and also presented 2 papers in International conferences, she has published 2 papers in International Journals. Her main research interests include computer networks, Big data and data mining.

Shruthika.C.A received her B.E degree in Computer Science and Engineering from Visvesvaraya Technological University Karnataka in 2008 and M. Tech in Computer Networks Engineering from Visvesvaraya Technological University Karnataka in 2012;
She has presented 2 papers in national conference level and 1 paper in international conference level She is an Assistant Professor of the department of computer science at B.G.S Institute of Technology; B.G.Nagar and having 4 years of Teaching Experience. Her main research interests include computer networks and wireless sensor networks.