International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 11 No. 1 Oct. 2014, pp. 19-26

http://www.ijisr.issr-journals.org/

SECURITY STORAGE SYSTEM FOR CLOUD USER USING OSD WITH A SELF-DESTRUCTING DATA

Ms. Punam U. Lambat[1], Prof. Mr. Parag Jawarkar[2], and Mr. G. Rajesh Babu[1]

[1]Department of Computer Science & Engineering,
Tulsiramji Gaikwakd –Patil college of Engineering & technology mohgaon wardha road Nagpur,
Maharashtra, India

[2]Department of Electronics and Telecommunication,
Tulsiramji Gaikwakd –Patil college of Engineering & technology mohgaon wardha road Nagpur,
Maharashtra, India

ABSTRACT: Cloud computing focuses on maximizing the effectiveness of the sharing of resources. It is not only shared by multiple users but can also dynamically reallocating as per demand. Cloud computing is the notion of outsourcing on-site available services and data storage to an off-site. Personal data stored in the cloud may contain account number, password, notes and other important information that could be used and misused by a miscreant, a competitor or a court of law. These data are cached, copied and archived by Cloud Service Providers (CSPs), often without user authorization and control. To overcome this problem to propose a Self Destruction method is protecting the user data privacy through Shamir Secret sharing algorithm, which can generate a pair of keys. Self Destruction method is associated with Time to Live (TTL) property to specify the life time of the keys. TTL trigger the Self Destruction operation, then the keys becomes destructed or unreadable after a user specified period. User can decrypt after timeout, either the user give correct keys. Shamir algorithm generates new keys to the user. Self Destruction mechanism reduces the overhead during upload and download file in the cloud. The result demonstrates that Self Destruction is practical to use and meet all privacy preserving goals. In this paper, active storage framework provides virtualization environment to run client's application and data is treated as objects to increase the throughput and decrease the latency.

KEYWORDS: Cloud Computing, confidential data privacy, object based active storage, Vanishing data, Authentication, Self-estructing Data, Distributed hash table (DHT).

1 INTRODUCTION

Development of Cloud computing and popularization of mobile Internet, Cloud services are becoming more and more important for people's life. People are more or less requested to submit or post some personal private information to the Cloud by the Internet. When people do this, they subjectively hope service providers will provide security policy to protect their data from leaking, so others people will not invade their privacy. Distributed hash tables (DHTs): DHTs discard data older than a certain age. The key is permanently lost, and the encrypted data is permanently unreadable after data expiration. This system is built upon standard cryptographic techniques and assuredly deletes files to make them unrecoverable to anyone upon revocations of file access policies. A model of load-managed active storage, which strives to integrate computation with storage access in a way that the system can predict the effects of offloading computation to Active storage Units (ASU).

We implemented a proof-of-concept *SeDas* prototype. Through functionality and security properties evaluations of the *SeDas* prototype, the results demonstrate that *SeDas* is practical to use and meets all the privacy-preserving goals described.

Corresponding Author: Corresponding Author (email)

Compared to the system without self-destructing data mechanism, throughput for uploading and downloading with the proposed *SeDas* acceptably decreases by less than 72%, while latency for upload/download operations with self-destructing data mechanism increases by less than 60%.

The problem of vanish discussed above, in our previous work, we proposed a new scheme, to prevent hopping attack, which is one kind of the Sybil attacks , by extending the length range of the key shares to increase the attack cost substantially, and did some improvement on the Shamir secret sharing algorithm implemented in the vanish system. The previous work aimed at some special applications, e.g., database, multimedia, etc., there is no general system level self-destructing data in the literature

There are multiple storage services for a user to store data. Meanwhile, to avoid the problem produced by the centralized "trusted" third party, the responsibility of SeDas is to protect the user key and provide the function of self-destructing data. Fig. 1 shows the brief structure of the user application program realizing storage process. In this structure, the user application node contains two system clients: any third-party data storage system (TPDSS) and Self Destructing data system. The user application program interacts with the server through client, getting data storage service. The way to attain storage service by client interacting with a server depends on the design of TPDSS. We do not need a secondary development for different TPDSS. The process to store data has no change, but encryption is needed before uploading data and the decryption is needed after downloading data.

1.1 SHAMIR SECRET SHARING ALGORITHM

Shamir algorithm are ideal for storing information that is highly sensitive and highly important. A secret sharing method can secure a secret over multiple servers and remain recoverable despite multiple server failure. The dealer may act as several district participants, distributing the shares among the participants. Each share may be stored on a different server, but the dealer can recover the secret even if several servers break down as long as they can recover.

1.2 SELF DESTRUCTION USING TIME TO LIVE PROPERTY

Time to Live is a mechanism that limit the lifespan or lifetime of keys stored in cloud. TTL may be implemented as a counter or timestamp attached to or embedded in the keys. Once the prescribed event occur or timespan has elapsed, keys can be self destructed without any user intervention. In computing application, TTL is used to improve performance of caching or to improve privacy from the leakages.

Our objectives are summarized as follows:

1) We focus on the related key distribution algorithm, Shamir's algorithm. We use these methods to implement a safety destruct with equal divided key.
2) Based on active storage framework, we use an object-based storage interface to store and manage the equally divided key.
3) In this, self-destructing data system supports security erasing files and random encryption keys stored in a hard disk drive (HDD) or solid state drive (SSD), respectively.
4) Through functionality and security properties evaluation of the SeDas prototype, the results demonstrate that SeDas is practical to use and meets all the privacy-preserving goals. The prototype system imposes reasonably low runtime overhead.
5) Protect the privacy of past, archived data against accidental, malicious and legal attacks (e.g. Emails, Trash Bin).
6) Ensure that all copies of certain data become unreadable after a user-specified time, without any specific action on the part of a user.
7) Even if an attacker obtains both a cached copy and the user's cryptographic keys and passwords.

The rest of this paper is organized as follows. We review the related work in Section II. We describe the proposed methodology in Section III, and we conclude this paper in Section IV

2 RELATED WORK

R. Geambasu , T. Kohno , A. Levy and H. M. Levy [1] Proposes a system that meets this challenge through a novel integration of cryptographic techniques with global-scale, P2P, distributed hash tables (DHTs). We implemented a proof-of-concept Vanish prototype to use both the million-plus-node Vuze Bit- Torrent DHT and the restricted-membership OpenDHT.

We evaluate experimentally and analytically the functionality, security, and performance properties of Vanish, demonstrating that it is practical to use and meets the privacy-preserving goals described above.

S. Wolchok , O. S. Hofmann , N. Heninger , E. W. Felten , J. A. Halderman , C. J. Rossbach , B. Waters and E. Witchel [3] Proposes a two Sybil attacks against the current Vanish implementation, which stores its encryption keys in the million-node Vuze Bit Torrent DHT. These attacks work by continuously crawling the DHT and saving each stored value before it ages out. They can efficiently recover keys for more than 99% of Vanish messages

LingfangZeng, Zhan Shi, ShengjieXu, Dan Feng[4]They Discuss the existing state-of-the-art self-destructing data schemes (Vanish) exhibit fragile for hopping attack and sniffing attack in realistic application. Propose a new scheme called SafeVanish

Y. Xie, K.-K. Muniswamy-Reddy, D. Feng, D. D. E. Long, Y. Kang, Z. Niu, and Z. Tan [10] Proposed a system that is aimed to be practically used:

- Small modifications to the existing T10 OSD standard
- Four kinds of critical characteristics in terms of user case
- System demonstration on three real world applications in terms of performance, scalability, language, etc

Y. Tang , P. P. C. Lee , J. C. S. Lui and R. Perlman [11] propose a cloud storage system called FADE, which aims to provide assured deletion for files that are hosted by today's cloud storage services. We present the design of policy-based file assured deletion, in which files are assuredly deleted and made unrecoverable by anyone when their associated file access policies are revoked. We present the essential operations on cryptographic keys so as to achieve policy-based file assured deletion. We implement a prototype of FADE to demonstrate its practicality, and empirically study its performance overhead when it works with Amazon S3. Our experimental results provide insights into the performance-security trade-off when FADE is deployed in practice.

Y. Kang , J. Yang and E. L. Miller [13] provide portability and efficiency by delegating storage management to SCM devices and eliminating duplicate mapping tables on both host and device. Further, this approach allows systems to immediately utilize new SCM technologies with no change to host systems, providing flexibility to system designers without the need to optimize the file system for many different types of SCMs. Moreover, this approach can also provide new functionality such as object-level reliability and object level compression by leveraging the semantics of object-based requests.

3 PROPOSED METHODOLOGY

A. Existing system

Personal data stored in the Cloud may contain account numbers, passwords, notes, and other important information that could be used and misused by a miscreant, a competitor, or a court of law. These data are cached, copied, and archived by Cloud Service Providers (CSPs), often without users' authorization and control. Self-destructing data mainly aims at protecting the user data's privacy. All the data and their copies become destructed or unreadable after a user-specified time, without any user intervention. Besides, the decryption key is destructed after the user-specified time.

Disadvantages:

These data are cached, copied, and archived by Cloud Service Providers (CSPs), often without users' authorization and control. Self-destructing data mainly aims at protecting the user data's privacy. All the data and their copies become destructed or unreadable after a user-specified time, without any user intervention. Besides, the decryption key is destructed after the user-specified time.

B. Proposed system

We present SeDas, a system that meets this challenge through a novel integration of cryptographic techniques with active storage techniques based on T10 OSD standard. We implemented a proof-of-concept SeDas prototype. Through functionality and security properties evaluation of the SeDas prototype, the results demonstrate that SeDas is practical to use and meets all the privacy-preserving goals described above. Compared with the system without self-destructing data mechanism, throughput for uploading and downloading with the proposed SeDas acceptably decreases by less than 72%, while latency for upload/download operations with self-destructing data mechanism increases by less than 60%.

Advantages:

Compared with the system without self-destructing data mechanism, throughput for uploading and downloading with the proposed SeDas acceptably decreases by less than 72%, while latency for upload/download operations with self-destructing data mechanism increases by less than 60%.

c. Module 1: Storage Architecture :

There are three parties based on the active storage framework. i)Metadata server (MDS): MDS is responsible for user management, server management, session management andfile metadata management. ii) Application node: The application node is a client to use storage service of the SeDas.iii) Storage node: Each storage node is an OSD. It contains two core subsystems: key value store subsystem and active storage object (ASO) runtime subsystem. The key value store subsystem that is based on the object storage component is used for managing objects stored in storage node: lookup object, read/write object and so on. The object ID is used as a key. The associated data and attribute are stored as values.

d. Module 3:Active Storage Object:

An active storage object derives from a user object and has a time-to-live (ttl) value property. The ttl value issued to trigger the self-destruct operation. The tll value of a user object is infinite so that a user object will not be deleted until a user deletes it manually.

e. Module 4:Self-Destruct Method Object:

Kernel code can be executed efficiently; however, a service method should be implemented in user space with these following considerations. Many libraries such as libc can be used by code in user space but not in kernel space. Mature tools can be used to develop software in user space. It is much safer to debug code in user space than in kernel space.

f. Module 5:Data Process:

To use the SeDas system, user's applications should implement logic of data process and act as a client node. There are two different logics: uploading and downloading.

- Uploading file process

When a user uploads a file to a storage system and stores his key in this SeDas system, he should specify the file, the key and *ttl* as arguments for the uploading procedure. Fig. presents its pseudo-code. In these codes, we assume data and key has been read from the file. The ENCRYPT procedure uses a common encrypt algorithm or user-defined encrypt algorithm. After uploading data to storage server, key shares generated by *ShamirSecretSharing* algorithm will be used to create active storage object (ASO) in storage node in this system.

- Downloading file process

Any user who has relevant permission can download data stored in the data storage system. The data must be decrypted before use. The whole logic is implemented in code of user's application.

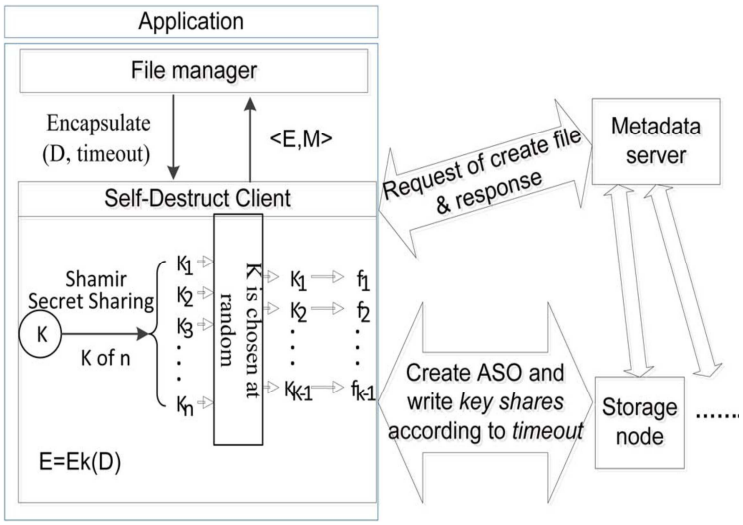

Fig 1. Uploading and Downloading file process

g. Module 6:Data Security Erasing in Disk

Our implementation method is as follows:

- The system prespecifies a directory in a special area to store sensitive files.
- Monitor the file allocation table and acquire and maintain a list of all sensitive documents, the logical block address (LBA).
- LBA list of sensitive documents appear to increase or decrease, The update is sent to the OSD.
- OSD internal synchronization maintains the list of LBA, the LBA data in the list updates.

4 SYSTEM ARCHITECTURE

Shamir algorithm implemented in cloud server to generate a pair of keys to the user. The main components are key generation, encryption and decryption using Message Digest Algorithm. The user can also specify the lifetime of each keys. These keys associated with Time to Live (TTL) property. When the keys can be self destructed after user specified time without any user intervention. The encrypt and decrypt procedure uses a message digest algorithm (MD5) for encryption and decryption. MD5 has been utilized in wide variety of security application, and is also commonly used to check data integrity. Keys are stored in DHT, before download a data in cloud the Shamir algorithm check the keys match with hash table. Then only the given user is authorized to download data in cloud.

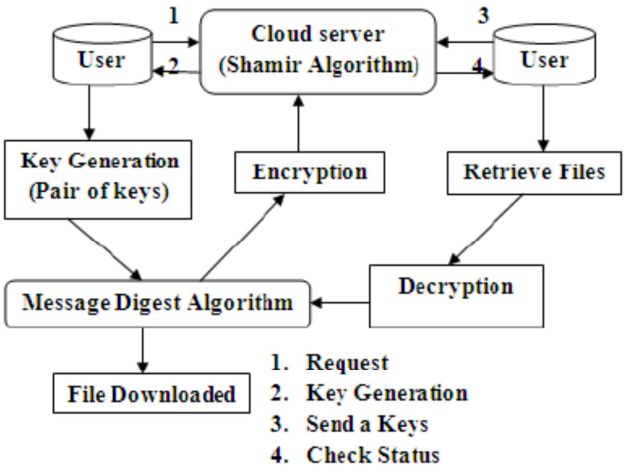

Fig 2. System Model

In Proposed system, sedas method implement automatic self destruction of keys to avoid explicit delete or modification of data in cloud environment by any third party or attackers. It leads several advantages are reduces communication overhead, network delay, generate pair of keys to single file (keys length is 2bytes) , increase processing speed, increase I/O performance and also it will meets all privacy preserving goals.

5 MODULES

User Authentication
Object Storage Devices (OSD)
Secret Key Part
File Uploading
Self Destruction Method
Downloading File
Performance Evaluation

5.1 MODULES DESCRIPTION

User Authentication: - A new user has to first create a profile. This is done by registration. A user id and password are submitted by the user. The user can login successfully only if user id and password are entered correctly. The login is a failure if the incorrect user id or wrong password is entered by the user. This helps in preventing unauthorized access.

Object Storage Devices (OSD):- An OSD is a computer storage device, similar to disk storage but working at higher level. Instead of providing a block oriented a block oriented interface that reads and writes fixed sized block of data. Each object has both data (uninterpreted sequence of bytes) and metadata (an extensible set of attributes describing the object).The OSD is responsible for managing the storage the storage of objects and their metadata. OSD implements a security mechanism that provides the user data privacy. N extensible set of attributes describe objects. Some attributes are implemented directly by the OSD, such as the number of bytes in an object and the modified time of object.

Secret Key Part:- Shamir Secret Sharing is an algorithm in cryptography. It is a form of secret sharing. Where a secret is divided into parts, giving each participant its own unique part. Where some parts or all of them are needed in order to reconstruct the secret. Each of these pieces of information must be kept highly confidential. Secret sharing are ideal for storing information that is high sensitive and highly important and also allow arbitrarily high levels of confidentiality and reliability to be achieved.

File Uploading:- Before upload file in cloud the user perform encryption using pair of keys generated by Shamir algorithm through MD5.When a user upload only a encrypted file in cloud and stores his keys using sedas method, it should specify the file, keys and TTL as the arguments for uploading procedure. When the keys are self destructed after a user specified time.

Self Destruction Method:- Self destruction mainly aims at protecting the user data privacy. All the keys become self destructed or unreadable after user specified time. The result demonstrate that the sedas is practical to use and meet all privacy preserving goals described. Sedas does not affect the normal use of storage system and can also meet the requirement of self destructing data under a survival time by user controllable keys. These are multiple storage services for a user to store data. Meanwhile, to avoid problem produced by the centralized “trusted” third party, the responsibility of sedas is to protect the user keys and provide the function of self destructing data.

Downloading File:- Any user who has relevant permission can download data stored in the cloud. The data must be decrypted before use. If the self destruct operation has nottriggered, the client can get enough key shares to reconstruct the keys successfully. During download process, Shamir algorithm checks the given keys are expired or not. If the keys are not expiring, the user can easily download. Otherwise, Shamir algorithm generates a new pair of keys to the authorized user.

Performance Evaluation:-Compare with the native system without self destructing data mechanism, throughput for uploading and downloading with the proposed sedas acceptably decreased by less than 72%, While latency for upload/download operations with sedas data mechanism increases by less than 60%.

6 RESULT

Here it an evaluation of latency for the files of different sizes. The time taken for uploading and downloading file determines the latency. In this system during file access there is high speed from the first bit to last bit arrival rate of the file. It is evaluated with the file size. Downloaded time is divided according to the file size and makes it to the multiplication of the bytes. To the previous result overhead of sender and receiver is added.

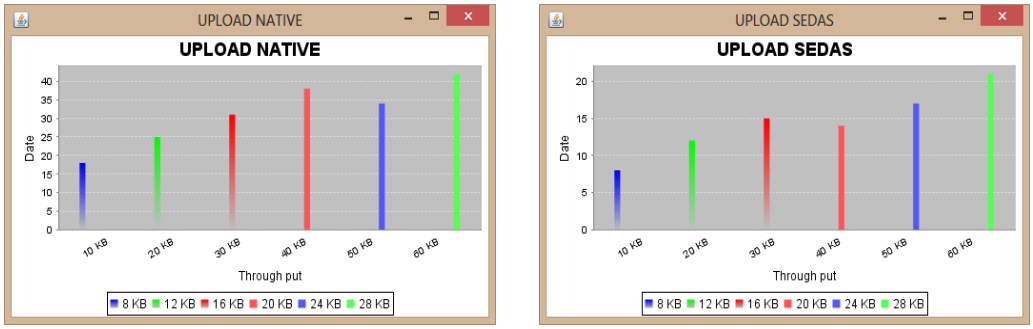

Fig.:3 Comparisons of throughput in the upload operations.

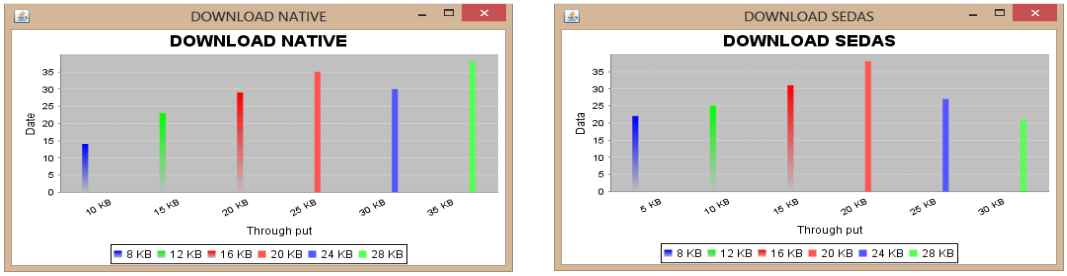

Fig.:4Comparisons of throughput in the download operations.

Fig. shows the throughput results for the different schemes. The throughput decreases because upload/download processes require much more CPU computation and finishing encryption/ decryption processes in the SeDas system, compared with the Native system. From Fig. 13, we can see that SeDas reduces the throughput over the Native system by an average of 59.5% and up to 71.67% for the uploading

From Fig. 17, we can see that SeDas reduces the throughput over the *Native* system by an average of 30.5% and up to 50.75% for the downloading.

Compared with the Native system without self-destructing data mechanism, throughput for uploading and downloading with the proposed SeDas acceptably decreases by less than 72%, while latency for upload/download operations with self-destructing data mechanism increases by less than 60%.

7 CONCLUSION

This paper introduced a new approach for protecting data privacy from attackers who retroactively obtain, through legal or other means, a user's stored data and private decryption keys. .We demonstrated the feasibility of our approach by presenting SeDas, a proof-of-concept prototype based on object-based storage techniques. Its causes sensitive information, such as account numbers, passwords and notes to irreversibly self-destruct, without any action on the user 's part. Our measurement and experimental security analysis sheds insight into the practicability of our approach. Our plan to release the current system will help to provide researchers with further valuable experience to inform future object-based storage system designs for Cloud services.

8 FUTURE WORK

Data and its associated key are destroyed after the expiration time of the key. Data such as file is treated as objects. When a single object is destroyed then any object referenced to that particular object gets free. Thus the reference count decreases. Instead of destroying an object as soon as its reference count falls to zero, it is added to the unreferenced list objects and periodically destroyed from the list.

REFERENCES

[1] R. Geambasu , T. Kohno , A. Levy and H. M. Levy "Vanish: Increasing data privacy with self-destructing data", *Proc. USENIX Security Symp.*, pp.299 -315 2009

[2] A. Shamir "How to share a secret", *Commun. ACM*, vol. 22, no. 11, pp.612 -613 1979

[3] S. Wolchok , O. S. Hofmann , N. Heninger , E. W. Felten , J. A. Halderman , C. J. Rossbach , B. Waters and E. Witchel "Defeating vanish with low-cost sybil attacks against large DHEs", *Proc. Network and Distributed System Security Symp.*, 2010

[4] L. Zeng , Z. Shi , S. Xu and D. Feng "Safevanish: An improved data self-destruction for protecting data privacy", *Proc. Second Int. Conf. Cloud Computing Technology and Science (CloudCom)*, pp.521 -528 2010

[5] L. Qin and D. Feng "Active storage framework for object-based storage device", *Proc. IEEE 20th Int. Conf. Advanced Information Networking and Applications (AINA)*, 2006

[6] Y. Zhang and D. Feng "An active storage system for high performance computing", *Proc. 22nd Int. Conf. Advanced Information Networking and Applications (AINA)*, pp.644 -651 2008

[7] T. M. John , A. T. Ramani and J. A. Chandy "Active storage using object-based devices", *Proc. IEEE Int. Conf. Cluster Computing*, pp.472 -478 2008

[8] A. Devulapalli , I. T. Murugandi , D. Xu and P. Wyckoff *Design of an intelligent object-based storage device*, 2009 [online] Available: http://www.osc.edu/research/network_file/projects/object/papers/istor-tr.pdf

[9] S. W. Son , S. Lang , P. Carns , R. Ross , R. Thakur , B. Ozisikyilmaz , W.-K. Liao and A. Choudhary "Enabling active storage on parallel I/O software stacks", *Proc. IEEE 26th Symp. Mass Storage Systems and Technologies (MSST)*, 2010

[10] Y. Xie , K.-K. Muniswamy-Reddy , D. Feng , D. D. E. Long , Y. Kang , Z. Niu and Z. Tan "Design and evaluation of oasis: An active storage framework based on t10 osd standard", *Proc. 27th IEEE Symp. Massive Storage Systems and Technologies (MSST)*, 2011

[11] Y. Tang , P. P. C. Lee , J. C. S. Lui and R. Perlman "FADE: Secure overlay cloud storage with file assured deletion", *Proc. SecureComm*, 2010

[12] C. Wang , Q. Wang , K. Ren and W. Lou "Privacy-preserving public auditing for storage security in cloud computing", *Proc. IEEE INFOCOM*, 2010

[13] R. Weber "Information Technology—SCSI object-based storage device commands (OSD)-2", *Technical Committee T10, INCITS Std., Rev. 5*, 2009

[14] Y. Kang , J. Yang and E. L. Miller "Object-based SCM: An efficient interface for storage class memories", *Proc. 27th IEEE Symp. Massive Storage Systems and Technologies (MSST)*, 2011

[15] M. Wei , L. M. Grupp , F. E. Spada and S. Swanson "Reliably erasing data from flash-based solid state drives", *Proc. 9th USENIX Conf. File and Storage Technologies (FAST)*, 2011

[16] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou, “Scalable performance of the panasas parallel file system,” in *Proc. 6th USENIX Conf. File and Storage Technologies (FAST)*, 2008.

[17] S. A.Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, “Ceph: A scalable, high-performance distributed file system,” in *Proc. 7th Symp Operating Systems Design and Implementation (OSDI)*, 2006.