International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 12 Issue 1–DECEMBER 2014.

An Identity Destructing Records Organization Based On Energetic Storage Space Framework

M.Madhuri[*1] and T.Deepa[#2]

[*]*P.G. Student, QIS College of Engineering and Technology, India*

[#]*Assistant Professor, QIS College of Engineering and Technology, India*

[1]madhuri.mtechproj@gmail.com

Abstract— The Internet and Cloud technology, security of their privacy takes more and more risks. On the one hand, when data is being processed, transformed and stored by the current computer system or network, systems or network must cache, copy or archive it. These copies are essential for systems and the network. However, people have no knowledge about these copies and cannot control them, so these copies may leak their privacy. On the other hand, their privacy also can be leaked via Cloud Service Providers (CSPs') negligence, hackers' intrusion or some legal actions. These problems present formidable challenges to protect people's privacy. The secret key is divided and stored in a P2P system with distributed hash tables (DHTs). With joining and exiting of the P2P node, the system can maintain secret keys. According to characteristics of P2P, after about eight hours the DHT will refresh every node With Secret Sharing Algorithm.

Keywords: Cloud computing, Data privacy, Active Storage

I. INTRODUCTION

As people rely more and more on the Internet and Cloud technology, security of their privacy takes more and more risks. On the one hand, when data is being processed, transformed and stored by the current computer system or network, systems or network must cache, copy or archive it. These copies are essential for systems and the network. However, people have no knowledge about these copies and cannot control them, so these copies may leak their privacy. On the other hand, their privacy also can be leaked via Cloud Service Providers (CSPs') negligence, hackers' intrusion or some legal actions. These problems present formidable challenges to protect people's privacy. A pioneering study of Vanish [1] supplies a new idea for sharing and protecting privacy. In the Vanish system, a secret key is divided and stored in a P2P system with distributed hash tables (DHTs). With joining and exiting of the P2P node, the System can maintain secret keys. According to characteristics of P2P, after about eight hours the DHT will refresh every node. With Shamir Secret Sharing Algorithm [2], when one cannot get enough parts of a key, he will not decrypt data encrypted with this key, which means the key is destroyed.

Through functionality and security properties evaluation of the Identity Destructing prototype, the results demonstrate that Identity Destructing is practical to use and meets all the privacy-preserving goals. The prototype system imposes reasonably low runtime overhead. Identity Destructing supports security erasing files and random encryption keys stored in a hard disk drive (HDD) or solid state drive (SSD), respectively.

II. OBJECTIVES

The self destructive system defines two modules, a self-destruct method object that is associated with each secret key part and survival time parameter for each secret key part. In this case, System can meet the requirements of self-destructing data with controllable survival time while users can use this system as a general object storage system. Our objectives are summarized as follows.

1) We focus on the related key distribution algorithm, Shamir's algorithm, which is used as the core algorithm to implement client (users) distributing keys in the Object storage system. 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) Through functionality and security properties evaluation of this prototype, the results demonstrate that System is practical to use and meets all the privacy-preserving goals. The prototype system imposes reasonably low runtime overhead.

III. DATA PROTECTION IN DISK

We must secure delete sensitive data and reduce the negative impact of OSD performance due to deleting operation. The proportion of required secure deletion of all the files is not great, so if these parts of the file update operation changes, then the OSD performance will be impacted greatly.

Our implementation method is as follows:

i) The system pre-specifies a directory in a special area to store sensitive files.

ii) Monitor the file allocation table and acquire and maintain a list of all sensitive documents, the logical block address.

iii) Logical block address list of sensitive documents appear to increase or decrease, the update is sent to the OSD.

iv) OSD internal synchronization maintains the list of logical block address, the logical block address data in the list updates.

IV. EXPUNGE OF ENCRYPTION KEY

In this paper, erasing files, which include bits (Secret Shares [2]) of the encryption key, is not enough when we erase/ delete a file from their storage media; it is not really gone until the areas of the disk it used are overwritten by new information. With flash-based solid state drives (SSDs), the erased file situation is even more complex due to SSDs having a very different internal architecture [36]. For instance, different from erasing files which simply marks file space as available for reuse, data wiping overwrites all data space on a storage device, replacing useful data with garbage data. Depending upon the method used, the overwrite data could be zeros (also known as “zero-fill”) or could be various random patterns [41]. The ATA and SCSI command sets include “secure erase” commands that should sanitize an entire disk. Physical destruction and degaussing are also effective. SSDs work differently than platter-based HDDs, especially when it comes to read and write processes on the drive. The most effective way to securely delete platter-based HDDs (overwriting space with data) becomes unusable on SSDs because of their design. Data on platter-based hard disks can be deleted by overwriting it. This ensures that the data is not recoverable by data recovery tools. This method is not working on SSDs as SSDs differ from HDDs in both the technology they use to store data and the algorithms they use to manage and access that data.

V. SYSTEM ARCHITECTURE

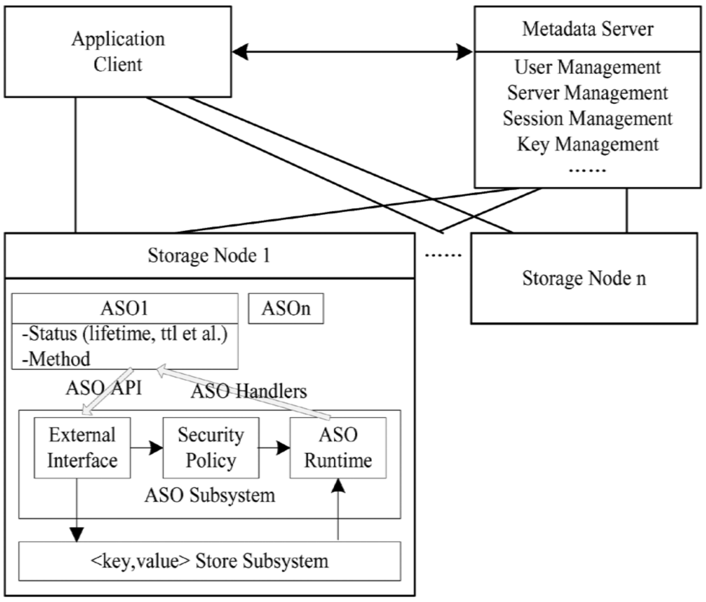

Figure 1 System Architecture

Fig. 1 shows the architecture of Identity Destructing. There are three parties based on the active storage framework.

i) Metadata server (MDS): MDS is responsible for user management, server management, session management and file metadata management.

ii) Application node: The application node is a client to use storage service of the Identity Destructing.

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. The ASO runtime subsystem based on the active storage agent module in the object-based storage system is used to process active storage request from users and manage method objects and policy objects.

Algorithm 1: Uploading the file using Shamir secret key algorithm for key splitting.

```
Procedure Upload File (data, key, TTL)
Data: data read from this file to be uploaded
Key: data read from the key
TTL: time-to-live of the key
Begin// encrypt the input data with the key
Buffer = Encrypt (data, key)
Connect to a data storage server;
if failed then return fail;
```

```
Create file in the data storage server and write buffer into it;
// use Shamir Secret Sharing algorithm to get key shares
// k is count of data servers in the storage system Shared keys
[1....k] = Shamir Secret Sharing Split (n, k, key)
For i from 1 to k then
Connect to DS[i]
If successful then create_object (shared keys[i], TTL);
Else
For j from 1 to i then
Delete key shares created before this one;
End for
Return fail;
End if
End for
Return successful;
End.
```

VI. ENERGETIC STORAGE

An active storage object derives from a user object and has a time-to-live (ttl) value property. The ttl value is used to trigger the self-destruct operation. The tll value of a user object is infinite so that a user object wills not be deleted until a user deletes it manually. The ttl value of an active storage object is limited so an active object will be deleted when the value of the associated policy object is true. Interfaces extended by Active Storage Object class are used to manage ttl value. The create member function needs another argument for ttl. If the argument is 1, User Object:: create will be called to create a user object, else, Active Storage Object::create will call User Object::create first and associate it with the self-destruct method object and a self-destruct policy object with the ttl value. The get TTL member function is based on the read_attr function and returns the ttl value of the active storage object. The set TTL, add Time and dec Time member function is based on the write_attr function and can be used to modify the ttl value.

VII. ASYMMETRIC CRYPTOGRAPHY

A pair of key namely public and private is used for encryption and decryption respectively. They are different key related to each other mathematically. Each user generates public and private keys. Public key is announced publicly whereas a private key is kept secretly. Each user also maintains a list of public keys of other users. Asymmetric cryptography need not to distribute key since all participants generate public and private keys locally. Plaintext, cipher text, encryption algorithm, decryption algorithm, public key and private key are the ingredients of asymmetric cryptography. Figure 4 explains the method of asymmetric cryptography. The encryption and decryption of input message is carried as follows:

Encryption: CT = E(KPw PT)

Decryption: PT = D(CT, Kpr) where, KPw and Kpr are the public and private key of a user respectively. CT denotes the cipher text;

Plaintext is expressed as PT. E and D denote encryption and decryption algorithms. A profound example of the asymmetric algorithm is RSA.

Algorithm 2: Uploading the file using short share secret key for content splitting.

```
Procedure Upload File (data, key, TTL)
Data: data read from this file to be uploaded
Key: data read from the key
TTL: time-to-live of the key
Begin
//encrypt the input data with the key
Buffer = Encrypt (data, key)
Connect to a data storage server;
if failed then return fail;
Create file in the data storage server and write buffer into it;
// use Short Share Secret Sharing algorithm to get key shares
// k is count of data servers in the SeDas system
Shared Content [1...k] = Block Split (n, k and buffer)
Shared keys [1....k] = Short Share Secret Sharing Split (n, k, key)
For i from 1 to k then
Connect to DS[i]
If successful then
create_object (sharedkyes[i], TTL);
create_object (sharedContent[i], TTL);
Else
For j from 1 to i then
Delete key shares created before this one;
Delete Content shares created before this one;
End for
Return fail;
End if
End for
Return successful;
End
```

VIII. CONCLUSION

Data privacy has become increasingly important in the Cloud environment. 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. A novel aspect of our approach is the lever-aging of the essential properties of active storage framework basedonT10 OSD standard. We demonstrated the feasibility of our approach by presenting Identity Destructing, a proof-of-concept prototype based on object-based storage techniques. Identity Destructing 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 re-lease the current Identity Destructing system will help to provide researchers with further valuable experience to inform future object-based storage system designs for Cloud services.

IX. FUTURE WORK

Short share secret scheme is mainly used for split the key and reconstruct the key for user. The key may be corrupted, and the malicious shares are also not identified and recovered. So future work extended to identify and recover the corrupted/malicious shares using Robust Secret Sharing scheme.

REFERENCES

[1] Geambasu, T. Kohno, A. Levy, and H. M. Levy, “Vanish: Increasing data privacy with self-destructing data,” inProc. USENIX Security Symp., Montreal, Canada, Aug. 2009, pp. 299–315.
[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.Hal-derman, C. J. Rossbach, B. Waters, and E.Witchel,“Defeatingvanish with low-cost sybil attacks against large DHEs,” inProc. 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,” inProc. Second Int. Conf. Cloud Computing Technology and Science (CloudCom), Indianapolis, IN, USA, Dec. 2010, pp. 521–528.
[5] L. Qin and D. Feng, “Active storage framework for object-based storage device,” inProc. IEEE 20th Int. Conf. Advanced Information Networking and Applications (AINA), 2006.
[6] Y. Zhang and D. Feng, “An active storage system for high perfor-mance computing,” inProc. 22nd Int. Conf. Advanced Information Networking and Applications (AINA), 2008, pp. 644–651.
[7] T. M. John, A. T. Ramani, and J. A. Chandy, “Active storage using object-based devices,” inProc. IEEE Int. Conf. Cluster Computing, 2008, pp. 472–478.
[8] A. Devulapalli, I. T. Murugandi, D. Xu, and P. Wyckoff, 2009, Design of an intelligent object-based storage device [Online]. Available: http://www.osc.edu/research/network_file/projects/ob-ject/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,” inProc. IEEE 26th Symp. Mass Storage Systems and Technologies (MSST), 2010.