ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 9, September 2015

Enhancing Security in Cloud by Self-Destruction Mechanism

Kshama Bothra[1], Sudipta Giri[2]

Student, Department of Information Technology, MIT College of Engineering, Pune, India [1]

Professor, Department of Information Technology, MIT College of Engineering, Pune, India [2]

Abstract: Cloud computing, a recent computing technology is preferred by most of the users to store their data. Large amount of data can be stored in cloud. It is possible to access data stored in cloud by any third party. Providing security to the data stored in cloud is the prime concern. Data stored in cloud may contain personal notes, account information, password. To overcome the problem self-destruction method is proposed. All the data and their copies become self-destructed after user specified time period. Shamir secret sharing algorithm is used which generate key shares. Self-Destruction mechanism is conjoined with time-to-live field to specify the life time of the keys. After user specified time period key should be destructed or become unreadable. Any legitimate user can download file till the timeout. The file should be encrypted before upload and decrypted before download. Self-destruction mechanism indicates that it is practical to use and meet all preserving goals. Self-destruction mechanism reduces the time taken to upload and download file as compared to native system.

Keywords: Cloud computing, self-destruction, Active Storage Object, Time to live (ttl), data privacy.

I. INTRODUCTION

Due to easy access and availability cloud services are becoming very important in people’s life. People are desired to post personal private information to the cloud by the internet. People hope that service provider will provide security to their data stored in cloud. Security and privacy are the main concern for the data stored in cloud. Data stored in cloud is replicated in many nodes and authorized user does not have information about the storage of these copies. Unauthorized users can access these data and can store it for their future use. Cloud Service Providers negligence, hacker’s intrusion or legal action is also responsible for imparting the privacy.

Vanish [1] supplies a new idea for sharing and protecting privacy of data. In Vanish secret key is divided and stored in a P2P system with distributed hash tables (DHTs). Vanish integrates the cryptographic techniques with global scale, P2P and distributed hash tables. DHT have property of discarding data older than certain age. Since one cannot get enough parts of a key, decrypting data encrypted with the key won’t be possible.

Characteristics of P2P are challenges of Vanish, duration of key survival is also not known in Vanish, attack like Sybil attack and hopping attack are possible in Vanish. Sybil attacks can recover about 99% of the Vanish messages.

Length range of key shares was increased in SafeVanish. This increased the attack cost but was unable to control the attack to large extent.

This paper presents a solution to implement self-destructing data system. Self-destruct method defines two new modules, self-destruct method that is associated with each secret key part and survival time parameter for each secret key part. Shamir secret sharing algorithm is used to generate key shares.

1. Shamir secret sharing algorithm is used to generate key shares. Data is divided into parts and each part is stored in different nodes to reconstruct the original data some or all parts are needed.
2. We split encrypted data and store them in different nodes of Hadoop.
3. Self-destructing data supports security by erasing files and encrypted data stored in different nodes.
4. By evaluating functionality and security properties it is found that proposed method is practical to use.
5. Self-destruction method protects the privacy of against accidental, malicious and legal attacks.
6. Ensures that all the copy of data stored in different nodes becomes unreadable after user-specified time without users involvement.

The rest of this paper is organized as follows. We review the related work in Section II. We describe the architecture, design and implementation of Self-destruction in Section III. The extensive evaluations are presented in Section IV, and we conclude this paper in Section V.

II. LITERATURE SURVEY

Cloud computing is use to outsource data to third party. It is very important to provide security to the data stored in cloud. Tang et al. [1] proposed FADE which is built upon standard cryptographic techniques. FADE is readily deployable cloud technique used to protect deleted data with policy based file based assured deletion. FADE is built upon standard cryptographic techniques it firsts encrypts data files to guarantee privacy and assuredly deletes files to make them inaccessible to anyone. Each file is associated with single atomic file access policy. When the policy of the file is revoked the corresponding key will be removed from the key manager. Thus when the policy of file is revoked user won’t be able to access the file. Perlman et al. [12] presented three types of implicit delete: expiration time known at file creation, on-demand deletion of single file, and custom keys for classes of data.

Vanish [2] is a system that automatically self-destruct data after a period of time. It integrates cryptographic techniques with global-scale, P2P, distributed hash tables (DHTs). DHT have a property of discarding data older than certain age. Key with which data is encrypted is permanently lost hence encrypted data becomes permanently unreadable. Vanish takes a data object D, and encapsulates it into VDO V. To encapsulate the data D, Vanish picks random data key, K, and encrypts D with K to obtain a ciphertext C. Vanish uses threshold secret sharing to split data key K into N pieces $K_1,\ldots,K_N$. A parameter of secret sharing is a threshold that can be set by the user or by an application using Vanish. Threshold determines how many of the N shares are required to reconstruct the original key. Sybil attack [6] may endanger the system by continuously crawling, the DHT and saving value before it ages out. It is seen that about 99% of Vanish messages can be recovered easily. Wolchok et al. [6] inferred that public DHTs like VuzeDHT cannot provide enough security to Vanish.

Using both OpenDHT and VuzeDHT will help in providing more security as compared to using only one. Vanish is novel approach to provide security, but, in its current form it is found to be insecure.

Since Vanish is vulnerable to hopping attack and sniffer attack. SafeVanish[3] is proposed by L. Zeng et al., in SafeVanish length range of the key shares is increased to increase the attack cost substantially. Improved approach against sniffing attack is proposed in SafeVanish by using public key cryptosystem. In SafeVanish sender will first encrypt message with private key before pushing them to DHT node to store. Then will use the receiver's public key to encrypt the key shares and completes encryption process. Encrypted cipher text key shares will be sent by sender to the DHT node, to save shares safely. Attacker won't be able to access data, since they don't have knowledge of private key corresponding to public key. Receiver will first decrypt text using receiver's private key, than decrypt it using sender's public key.

III. PROPOSED WORK

Self-destructive system describes two new modules, a self-destruct method that is linked with each secret key part and each secret key part is associated with survival time parameter.

A. *Self-destructive architecture*

Figure 1, shows the architecture of self-destructive (self-des). In the architecture there are mainly four blocks.

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

ii) User layer: User layer is for the authorized client who uses the service of self-destructive architecture. New user is required to fill his/her basic information and registers. If user is already registered than he/she needs to validate himself/herself.

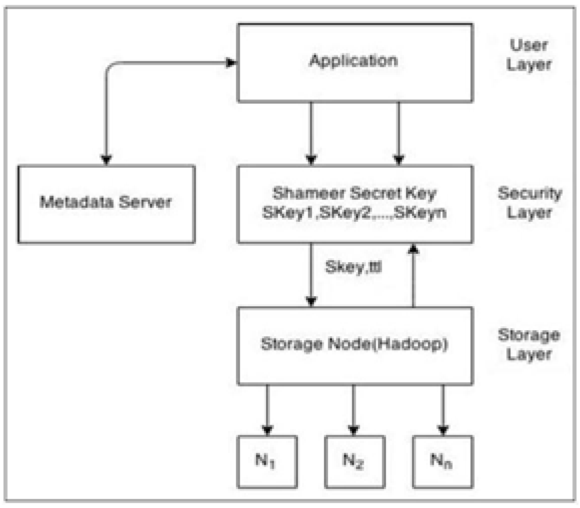

Figure 1: Proposed Architecture

iii) Security layer: This is the core part of the architecture. To generate key shares Shamir secret sharing algorithm is used. If the key is divided into n pieces then knowledge of at least k pieces will be required to decrypt data. If user has knowledge of less than k pieces than it would be hard to recover data. AES algorithm is used to encrypt and decrypt data. AES algorithm is not susceptible to attack and is faster than traditional algorithm.

iv) Storage node: Hadoop is used as a storage node. Hadoop can process large amount of data in very less time interval. Encrypted data is divided and stored in different nodes in Hadoop along with key. Storing data in different nodes in Hadoop will provide more security. Attacker is required to collect all the parts of data stored in different nodes in Hadoop. Since attacker won't have knowledge where data is stored it will be difficult to download it. Data stored in different nodes will be destructed after time-to-live value expires, no user will be able to download data.

B. *Proposed plan of work:*

Proposed plan of work describes different modules used in the proposed system.

1. Development of Login System: Login page is created in this module. Registration form contains field like First Name, Last Name, Address, Login ID and password. If the user is already registered than they need to validate themselves by filling login detail like login ID and Password. If the entries filled by user are incorrect then message ―Sorry! Entered details are incorrect‖ message will pop up. If entered details are correct then user will be able to perform Upload and Download operation.
2. Secret Key Part: Shamir Secret Sharing algorithm is used in secret key part. It is a type of secret sharing, where a secret is divided into parts. Each part is stored in different nodes in Hadoop. To reconstruct the data some or all parts are required. User provides key and Shamir Secret Sharing algorithm is used to generate key shares. Data is encrypted with the key shares and divided into parts before uploading it into different nodes in Hadoop. Each of the secret part must be kept highly confidential.

3. File uploading: User has to browse and select the location of the file which is to be uploaded. User has to first encrypt the file by using pair of keys generated by Shamir algorithm. File is encrypted using AES algorithm. File, key and ttl should be supplied as argument for uploading procedure. Below are the steps for uploading file.

```
Step1: UploadFile(data,key,ttl)
    data: data read from file to be uploaded
    key: data read from the key
    ttl: time-to-live of the key
Step2: encrypt the input data with the key
    buffer=ENCRYPT(data,key)
Step3: connect to data storage server if failed
    then return fail;
Step4: create file in the data storage server and write
    buffer into it;
Step5: use ShamirSecretSharing algorithm to get key
    Shares
Step6: Connect to DS[i];
    if successful then create_object
    (sharedkeys[i],ttl);
    else
    for j from 1 to i then
    delete key shares created before this one
```

4. Self-Destruction Method: Self-destruction mechanism mainly aims at protecting user's data privacy against malicious attack. Data stored in different node will be destructed after user specified time period. Results demonstrate that self-destruction mechanism is practical to use and it meets all the privacy preserving goals described.

5. Downloading File: Authorized user can access the data uploaded by downloading it. Data must be decrypted before using. If the time-to-live field is not expired than user can download file. User won't be able to download file after time-to-live value expires. After the timeout if user is trying to access file then it gives message —Sorry! You cannot access the file.

C. *Mathematical Model*

The system is modeled as S = {s, e, X, Y, t, F| ϕ}

- Where, s is the initial state
  - ✓ The user will input the data (*D*)
  - ✓ Provides key (*k*) and
  - ✓ timespan (t_m)
- e is the end state of the system which comprise of two states :
  - ✓ If t < t_m: Data will be retrieved from nodes (*N*)
  - ✓ If t > t_m: Data will be deleted from all nodes
- X = Set of inputs in the system
  X = {D, *k*, t_m}
  - ✓ D = {d_1, d_2, d_3,.,.,.,d_n}
  - ✓ k = encryption key
  - ✓ t_m = time for which the data is present
- Y = Set of outputs
  Y = {R_D}
  - ✓ R_D = Retrieved data after decryption from various nodes using key (*k*)
- t = the time for which the data is present in the database.
- Function F = {Enc_k(D), $Decr_k$(E)}
  - ✓ Where, Enc_k(D) = Encryption of data using key (k) for storing data in encrypted format
  - ✓ Dec_k(E) = Decryption of data for retrieving original data
- Φ = Constraint on whole system.
  - ✓ Constraint on the whole system is network connection

IV. EVALUATION

A. *Functional Testing*

Authorized user can access Cloud service provided by the service provider. User can perform two operations file uploading and file downloading. To upload file user has to first enter key and then select the location of the file which is to be uploaded. Time-to-live (ttl) value is also provided by the user. The system encrypts data and split encrypted data and store in different nodes. File is uploaded successfully. Sample text file was also downloaded successfully before time-to-live (ttl) value expires.

B. *Performance Evaluation*

We uploaded file using two method native system and proposed system. Results show that uploading time by using proposed system decreases by 43%. Figure 2, Represents time taken to upload file using native and proposed scheme.

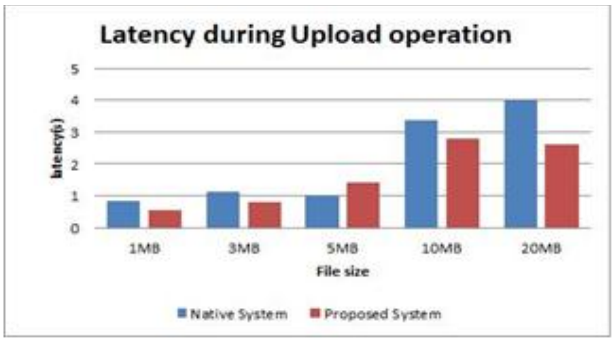

Figure 2: Time taken to Upload file

Figure 3, represents time taken to download file using existing method and proposed method.

We downloaded file using two method native system and proposed system. Found that downloading time by using proposed system decreases by 25%.

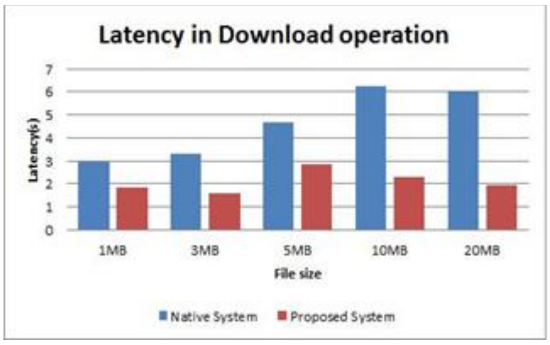

Figure 3: Time taken to Download file

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 9, September 2015

Figure 4, represents the throughput during file upload and download process. Throughput is calculated in MB/s. Throughput decreases because upload/download process require much more CPU computation.

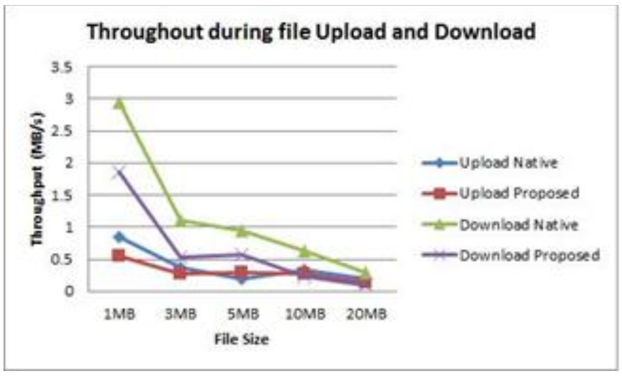

Figure 4: Throughput during file Upload and Download

In summary, the time taken to upload and download file using self-destructing system is less as compared to with native system.

V. CONCLUSION

Data privacy is one of the main aspects in cloud environment. Different approaches are used to protect privacy of data. Self-destruction is a mechanism to protect data from the user who retroactively obtain another user's stored data and key. Hadoop is used to store data in different nodes. Hadoop allows storing huge amount of data, and processing it in much more efficient manner and faster manner. File is encrypted and stored in different nodes in Hadoop. To decrypt file user will not only require key but also all the encrypted parts of a file. Authentic user will be able to download file. Also time-to-live field is provided, once time-to-live value expires user won't be able to download file. It is found that uploading time decreases by 43% and downloading time by 25% as compared to using native system.

ACKNOWLEDGEMENT

I am indeed thankful to my internal guide **Prof. Sudipta Giri** for his able guidance and assistance to complete this paper. I am grateful for their valued support and faith on me. I extend my special thanks to Head of Department of Information Technology, **Prof. Anil S. Hiwale** who extended the preparatory steps of this paper-work.

REFERENCES

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

[2] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy, ―Vanish: Increasing data privacy with self-destructing data,‖ in Proc. USENIX Security Symp., Montreal, Canada, Aug. 2009, pp. 299–315.

[3] L. Zeng, Z. Shi, S. Xu, and D. Feng, ―Safevanish: An improved data self-destruction for protecting data privacy, in Proc. Second Int. Conf. Cloud Computing Technology and Science (CloudCom), Indianapolis, IN, USA, Dec. 2010, pp. 521–528.

[4] Lingfang Zeng , Shibin Chen , ―SeDas: A Self-Destructing Data System Based on Active Storage Framework,‖ IEEE Transactions On Magnetics, Vol. 49, No. 6, June 2013

[5] A. Shamir, ―How to share a secret,‖ *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[6] 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, in *Proc. Network and Distributed System Security Symp.*, 2010.

[7] 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, in *Proc. 27th IEEE Symp. Massive Storage Systems and Technologies (MSST)*, 2011.

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

[9] R. Perlman, ―File system design with assured delete, in *Proc. Third IEEE Int. Security Storage Workshop (SISW)*, 2005.

[10] R. Geambasu, J. Falkner, P. Gardner, T. Kohno, A. Krishnamurthy, and H. M. Levy, ―Experiences building security applications on DHTs‖, technical report, UWCSE- 09-09-01, 2009.

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

[12] R. Perlman, ―File system design with assured delete, in *Proc. Third IEEE Int. Security Storage Workshop (SISW)*, 2005.

BIOGRAPHIES

Kshama Bothra Research Scholor, MIT college of Engineering, University of Pune. He has received the B.E. degree in Information Technology from Raisoni College of Engineering, Nagpur in 2012 and currently pursuing M.E from M.I.T College of Engineering, Pune.

Prof. Sudipta Giri done MTech from IISc Banglore. He is currently working as Assistant Professor in MIT College of Engineering, Information Technology Department, Pune Has received the BTech degree from IIT Kanpur.