A Physical Security Based Mailing System for Self Destructive Databases

Sarita Deshmukh[3], Rahul Kapse ,Varunakshi Bhojane
Department of Computer Engineering ,
Pillai HOC College of Engineering and Technology, Rasayani.

Abstract:
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, authentication and control. First we present SeDas, 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. In addition, the decryption key is destructed after the user-specified time. Second a system is providing security by using Kerberos for self-destructing e-mail messages. An e-mail client application provides a user interface through which the sender of an e-mail message can enter the message and a time period for destruction of the message. Once the sender has provided this information, the e-mail client application sends a request to an e-mail server application to transmit the self-destructing e-mail message. The e-mail client application may also receive self-destructing e-mail messages. When a self-destructing e-mail message is received, the destruction time date associated with the e-mail message is identified and the message is destroyed at the specified time.

Keywords: Cloud computing, Self Destructing, Kerberos, One-Time-Password

I. Introduction

With the development of cloud computing and popularization of mobile web, cloud online services are becoming more useful for people's life. People are requested to submit their personal information to the cloud by the internet. When people do this, they hope the service provider will provide security to protect their data from leaking, so other people will not invade their privacy. With more and more services and applications are emerging in the internet, it becomes easier to expose the user's sensitive data in the internet. Without much attention to this problem, it will result in the break out of leaking messages. For e.g. Emails can be leaked via cloud service provider, negligence or hackers. One of the most important reasons for exposing sensitive user data is that the user data will be stored in the cloud environment for a long time, which may not be controlled by the user himself. These are the major challenges to protect people's privacy. Self-destructing data systems are designed to address these concerns. Their goal is to destroy data after a pre-specified timeout. Self destruction is implemented by encrypting data with a key and then escrowing the information needed to reconstruct the decryption key with one or more third parties. Assuming that the key reconstruction information disappears from the escrowing third parties at the intended time, encrypted data will become permanently unreadable: (1) even if an attacker obtains a copy of the encrypted data and the user's cryptographic keys and passphrases after the timeout, (2) without the user or user's agent taking any explicit action to destroy it, (3) without need to change any stored or archived copies of that data, and (4) without the user relying on secure hardware. Once the key-renovation details disappear, data owners can be confident that their data will remain inaccessible to powerful attacks, whether from hackers who obtain copies of [1].

Assuming that the key reconstruction information disappears from the escrowing third parties at the intended time, encrypted data will become permanently unreadable: 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.

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. Some special attacks to characteristics of P2P are challenges of Vanish uncontrolled in how long the key can survive are also one of the disadvantages for Vanish. In

considering these disadvantages, this paper presents a solution to implement a self-destructing data system, or SeDas, which is based on an active storage framework. The SeDas system defines two new modules [3], 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, SeDas can meet the requirements of self-destructing data with controllable survival time while users can use this system as a general object storage system. Numerous applications could benefit from such self-destructing data. The self-destructing data is broadly applicable in today's Web centered world, where users sensitive data can persist in the cloud indefinitely(sometimes even after the user's account termination).With self-destructing data, users can regain control over the lifetimes of their Web objects, such as private messages on Facebook, documents on Google Docs, or private photos on Flickr.

Electronic mail ("e-mail") has become a pervasive method of communication for many computer users worldwide. Because e-mail provides a quick and easy method of communication that was not previously possible, e-mail is now commonly used in the home and in the workplace to send all types of communications ranging from trivial notes to highly sensitive business communications. However, although conventional e-mail systems are fast and convenient, these systems are not suitable for transmitting all types of information.

For instance, the ease with which e-mail messages may be forwarded, saved, and otherwise distributed make conventional e-mail systems inappropriate for highly confidential or proprietary information. Moreover, in conventional e-mail systems the sender of an e-mail message cannot restrict the operations that may be subsequently performed on an e-mail message. Therefore, once an e-mail message has been sent, the message may be subsequently forwarded to other e-mail users, printed, saved, copied, moved, and otherwise replicated. The inability to control the number and type of operations that may be subsequently performed on a sent e-mail message makes conventional e-mail systems unsuitable for sending confidential information for which absolute control of distribution is a necessity.

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. The Internet is an insecure place. Many of the protocols used in the Internet do not provide any security. Tools to "sniff" passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Worse yet, other client/server applications rely on the client program to be "honest" about the identity of the user who is using it. Other applications rely on the client to restrict its activities to those which it is allowed to do, with no other enforcement by the server. Kerberos was created by MIT as a solution to these network security problems. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. It has the following characteristics:

i) It is secure: it never sends a password unless it is encrypted.

ii) Only a single login is required per session. Credentials defined at login are then passed between resources without the need for additional logins.

iii) The concept depends on a trusted third party – a Key Distribution Center (KDC). The KDC is aware of all systems in the network and is trusted by all of them.

iv)It performs mutual authentication, where a client proves its identity to a server and a server proves its identity to the client.

Kerberos introduces the concept of a Ticket-Granting Server (TGS). A client that wishes to use a service has to receive a ticket – a time-limited cryptographic message – giving it access to the server. Kerberos also requires an Authentication Server (AS) to verify clients. The two servers combined make up a KDC. Active Directory performs the functions of the KDC shown in Fig-1.

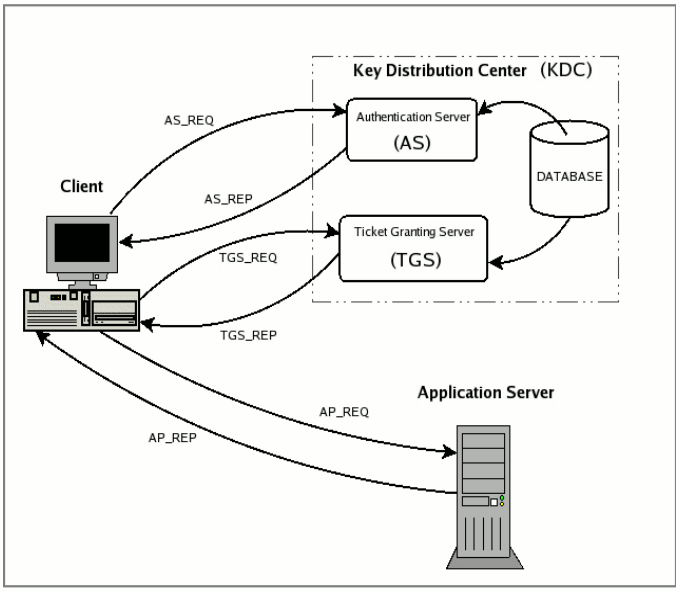

Fig- 1. Kerberos protocol system

One-Time Password (**OTP**): OTP is a password that is valid for only one login session or transaction, on a computer system or other digital device. OTPs avoid a number of shortcomings that are associated with traditional (static) password based authentication; a number of implementations also incorporate two factor authentication by ensuring that the one-time password requires access to something a person has (such as a small key ring fob device with the OTP calculator built into it, or a smartcard or specific cell phone) as well as something a person knows.

The most important advantage that is addressed by OTPs is that, in contrast to static passwords, they are not vulnerable to replay attacks. This means that a potential intruder who manages to record an OTP that was already used to log into a service or to conduct a transaction will not be able to abuse it, since it will be no longer valid. A second major advantage is that a user, who uses the same (or similar) password for multiple systems, is not made vulnerable on all of them, if the password for one of these is gained by an attacker. A number of OTP systems also aim to ensure that a session cannot easily be intercepted or impersonated without knowledge of unpredictable data created during the previous session, thus reducing the attack surface further.

II. RELATED WORK

Vanish [1] is a system for creating messages that automatically self-destruct after a period of time. Vanish encapsulates data objects so that they ―self-destruct‖ after a specified time, becoming permanently unreadable. It encrypts the data using a randomly generated key and then uses Shamir secret sharing [2] to break the key into n shares where k of them are needed to reconstruct the key. Vanish stores these shares in random indices in a large, pre-existing distributed hash table. After they expire, the key is permanently unavailable, and the encrypted content is permanently unreadable. However, Sybil attacks [3] may compromise the system 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. The public DHTs like Vuze DHT probably cannot provide strong enough security for Vanish. So, Geambasu *et al.* [4] proposes two main countermeasures, although using both OpenDHT and VuzeDHT can provide the maximum security that is derived from either system: if both DHTs are insecure, then the hybrid will also be insecure. Vanish is an interesting approach, it provide security to people's confidential data but, in its current form, it is insecure. To address the problem of Vanish discussed above, in our previous work [7], we proposed a new scheme, called Safe Vanish, 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, and did some improvement on the Shamir Secret Sharing algorithm and is implemented in the Vanish system.Tang *et al.* [8] proposed FADE which is built upon standard cryptographic techniques and assuredly deletes files to make them unrecoverable to anyone upon revocations of file access policies. Perlman *et al.* present three types of assured delete: expiration time known at file creation, on-demand deletion of individual files, and custom keys for classes of data.

Conventional e-mail systems may also be inappropriate for sending confidential or proprietary information because these systems do not allow the sender of an e-mail message to control the lifespan of the e-mail message. E-mail messages may, therefore, languish in a recipient's e-mail "in-box" or on an e-mail server computer for months or even years. Some e-mail systems will allow an e-mail recipient to specify that messages should be deleted after a certain amount of time. However, these systems do not allow the sender to specify a time for destruction of the sent e-mail message. Therefore, an e-mail sender cannot be certain that a sent e-mail message containing time sensitive information will ever be deleted.

The proposed SeDas does not affect the normal use of storage system and can meet the requirements of self-destructing data under a survival time by user controllable key.Therefore, in light of the above-described problems, there is a need for a system for providing self-destructing e-mail messages that allows a user to specify a time for the destruction of a sent e-mail message and that will destroy all instances of the e-mail message when the specified time arrives.

III. ARCHITECTURE OF SEDAS

Fig-2 shows the architecture of SeDas. 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 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.

a) The key value store subsystem: This is based on the object storage component is used for managing objects stored in storage node. The object ID is used as a key. The associated data and attribute are stored as values.

b) The ASO runtime subsystem: This is 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.

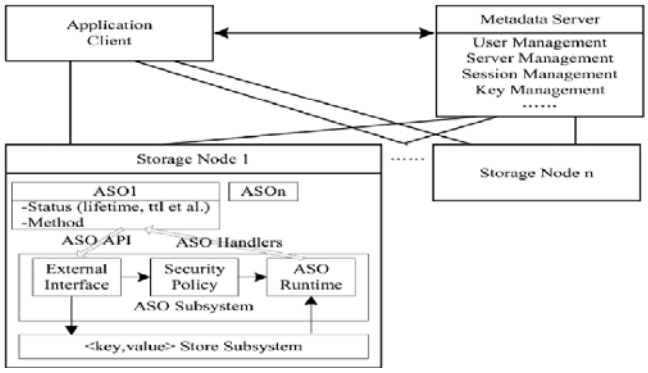

Fig- 2: SeDas system architecture

IV. IMPLEMENTATION

The procedure of the self destruction is given below step by step that how the data file will be upload and stored in the cloud databases and how the file get encrypted and download after downloading how it will get decrypted. Explain how the file will get self destructed from the databases.

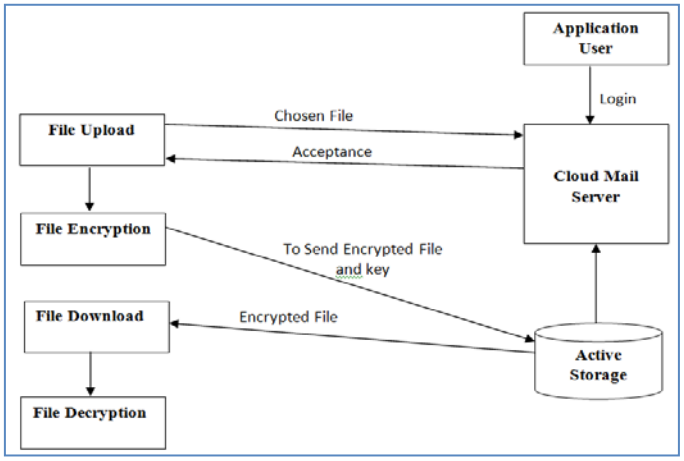

Fig-3 Self Destruction Process

From the Fig-3, The Self Destruction Process is as follows:
Step 1: Confidential file has chosen by the application user of the file. The file chosen by the user alone stored
in the cloud because in the private cloud service providers offers few services which are needed by the users.
Step2: The file is divided and stores each portion of the file in Active Storage. Now the encrypted file is stored in the Active storage.
Step3: The mail server verifies the authorized access by the user or not. During downloading the file verification procedure is completed by the mail server and assurance the user to access the service. Otherwise service is denied.
Step4: Along with the key the existence parameter of the key is defined. It provides the lifetime for the file. As far as the key is alive user can downloads the file. It becomes facilitates the file to be more confidential. To upload the file there is needto first to create the account and obtain the unique identity from the service provider. No need to convert the format of the file in cloud storage.
Step5: User who has relevant authorization can download data stored in the data storage system. The data must be decrypted before use. The whole process is implemented in user's application. These processes suppose that encrypted data and Meta information of the key has been read from the downloaded file.
Step6: Before decrypting, client should try to get key shares from storage nodes in the SeDas system. If the time lapse, then the client cannot get enough key shares to rebuilt the key successfully. If the connected ASO of the key has been destroyed, then client cannot rebuild the key, so he can only read the encrypted data.
The project implementation is divided into following modules:

1. Development of login tracking of the user
To use the SeDas system, users first have to sign up successfully to the system by filling all information mention in the form. And then only user can login to the system by enter registered Email Id and Password.

2. Providing security to mail server and mail client login via Kerberos
Kerberos will make sure that only authenticated users are logged in to the system, and in case the user is not authorized or does some sort of intrusion, then Kerberos automatically logs the user out of the system like if the user is a spoofer and changes it's PC's IP address, then the Kerberos system monitors this and automatically logs out the user. Also we set the login period, if the login period is over user can not access the account.
Authentication process:
i. The client sends a user name and server name to the KDC
ii. The KDC replies with a ticket and session key, encrypted with the user's password
-This ticket is known as the "Ticket Granting Ticket"(TGT):Yes, it is a ticket used to grant other tickets .
-The client uses the TGT with the user's password.
iii. Now client can use the SeDas by using the ticket and session key.

3. Providing One- Time Password to the personal registered mobile number.
In this module, once the user login to the system successfully then the ticket and the secret key will generate by Kerberos and that ticket and secret key will send to the user personal mobile number in digital form as a One Time Password and then the user can enter that One Time Password which he have on his registered number and use the SeDas system. The One Time Password will be time limited. Every login the OTP will be generated by the system.

4. Development of Active Storage Framework

An active storage object obtains from a user object and has a time-to-live (TTL) property. To start the self-destruct operation TTL value is being used. The TTL value of a user object is unlimited so that a user object will not be deleted until a user deletes it by hand. The TTL value of an active storage object is limited so an active object will be deleted when the value of the connected policy object is true. Interfaces extended by Active Storage Object class are used to manage TTL value.

In active storage framework object based interface to store and manage the evenly divided key. Evaluation of self-destructing data based on its practicality and security policies. The result demonstrates that self destructing data meets all the secrecy- preserving goals. Co-ordination between application server and storage devices is achieved by object based interface.

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.

i) Uploading file process: The user uploads a file to a storage system and stores the key in the key management system, the user should specify the file, the key and *TTL* as arguments for the uploading procedure. This process assumes data and key has been read from the file. The procedure uses a common encryption algorithm or user-defined encryption algorithm. After uploading data to storage server, key shares generated by Shamir Secret Sharing algorithm will be used to create active storage object (ASO) in storage node in the SeDas system.

ii) 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 process is implemented in user's application. This process assumes that encrypted data and Meta information of the key has been read from the downloaded file. Before decrypting, client should try to get key shares from storage nodes in the SeDas system. If the time expires been, the client cannot get enough key shares to rebuilt the key successfully. If the associated ASO of the key has been destructed, the client cannot rebuild the key so he can read only the encrypted data.

6. Development of deletion module in case user logs out

There is no need of explicit delete actions by the user, or any other third- party storing that data. No need to modify any of the stored or archived copies of that data. The time of key attainment is determined by Active storage system and not controllable for the user. In distributed object-based storage system self destructing data function is used. Extensive experiments show that the proposed Self-destructing data system does not affect the normal use of storage system. The requirements of self-destructing data under a survival time by user controllable key.

V. CONCLUSION

Data privacy has become increasingly important in the Cloud environment. 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 leveraging of the essential properties of active storage framework based on T10OSD standard. We demonstrated the feasibility of our approach by presenting SeDas, a proof-of-concept prototype based on object-based storage techniques. SeDas causes sensitive information, such as account numbers, passwords and notes to irreversibly self-destruct, without any action on the user's part.

We are providing security to the clients and the server via Kerberos**.** Kerberos provides the security to the login user and avoid the hacker from hacking the password and from steeling the valuable information. Also we can send the self destruct mails which store and providing the security to the mails server. We can store any valuable information for decided time to the mail server by defining the ttl value for the respective mails. So that we can aware to the mail server for the important mails and files with the time limit they can access the mails.

Also we are providing more security to the account by using OTP. For every login the new OTP will be created so that nobody can guess the OTP and access the system and the data. So this system is become more secure.

REFERENCES

[1] R. Geambasu , T. Kohno , A. Levy and H. M. Levy "Vanish: Increasing data privacy with self-destructing data", Proc.USENIX Security Sym, 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 ilarge DHEs", Proc. Network and Distributed System Security Symp, 2010,pp.1-20.

[4]Lingfang Zeng , Shibin Chen , Qingsong Wei , and Dan Feng,"SeDas: A Self-Destructing Data System Based on Active Storage Framework",IEEE TRANSACTIONS ON MAGNETICS, VOL. 49, NO. 6, JUNE 2013.

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

[6]T. M. John, A. T. Ramani, and J. A. Chandy, “Active storage usingobject-based devices,” in Proc. IEEE Int. Conf. Cluster Computing,2008, pp. 472–478
[7] 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.
[8] Y. Tang, P. P. C. Lee, J. C. S. Lui and R. Perlman "FADE: Secure overlay cloud storage with file assured deletion", Proc.Secure Comm, 2010,pp 1-20.
[9]R. Perlman, “File system design with assured delete,” in Proc. Third IEEE Int. Security Storage Workshop (SISW), 2005
[10] C. Wang, Q. Wang, K. Ren and W. Lou "Privacy-preserving public auditing for storage security in cloud computing", Proc. IEEE INFOCOM, 2010, pp.1-20
[11]Z. Niu, K. Zhou, D. Feng, H. Chai, W. Xiao, and C. Li, “Implementing and evaluating security controls for an object-based storage system,” in Proc. 24th IEEE Conf. Mass Storage Systems and Technologies(MSST), 2007.