Providing Security to Data in Cloud environment with Key Managers

A.Preethy
Department of Computer Science and Engineering
M.E student , Meenakshi Engineering College Anna University Chennai

Abstract-*Storage of data at off-site in cloud helps the customers to focus on data storage system. In order to help customer, the data has been outsourced to third-party administrative control. This has encountered serious security concerns. To provide a security for data in cloud and also to propose a proper key management system using key sharing. To technique with multiple key managers. Data security provides key management, access control, and file assured deletion.The data security utilizes shamir's(k,n) threshold scheme to manage keys. Cryptographic keys must be stored in a robust manner and a single point of failure should not affect the availability of data. To avoid man-in-the-middle attack user can access their key and data is ensured through a policy file that states policies under which access is granted to the keys. The DaSCE makes use of both symmetric and asymmetric keys. The confidentiality and integrity services for data are provided through symmetric keys that are secured by using asymmetric keys. Asymmetric key pairs are generated by third party KM's. Out of the key pair, only public key is transmitted to the client.*

Keywords-Cloud computing, High Level Petri Nets (HLPN), Key Managers, cloud computing

I. INTRODUCTION

CLOUD computing has emerged as a promising com- puting paradigm and has shown tremendous potential in managing the hardware and software resources located at third-party service providers. On-demand access to the computing resources in a pay-as-you-go manner relieves the customers from building and maintaining complex infrastructures. Cloud computing presents every computing component as a utility, such as software, platform, and infrastructure. The economy of infrastructure, maintenance, and flexibility makes cloud computing attractive for organizations and individual customers. Despite benefits, cloud computing faces certain challenges and issues that hinder widespread adoption of cloud. For instance, security, performance, and quality are a few to mention.

The development and operation of data storage sites is an ongoing process in organizations. Off-site data storage is a cloud application that liberates the customer from focusing on data storage systems. Representing system characteristics and capabilities as utility, causes the user to focus on aspects directly related to data (security, transmission, processing). However, moving data to the cloud, administered and operated by certain vendors requires high level of trust and security. Multiple users, separated through logical barriers of virtual machines, share resources including storage space. Multitenancy and virtualization generate risks and underpins the confidence of users to adopt the cloud model.

Armbrust et al. ranked data confidentiality and auditing at number three in the list of top ten obstacles impeding widespread cloud adoption. The cloud service providers can access the data without authorization from the user and other machines in cloud can also access the data. Data being the principal asset for organizations, needs to be secured. Especially, when data must enter a public cloud. To avoid unauthorized access to cloud data, access control mechanism must be enforced. Moreover, data leakage and data privacy strategies must be employed so that only authorized users can access and utilize data.

Refraining cloud service providers from utilizing the customer data requires high preventive measures. Encryption techniques provide a solution to ensure privacy and confidentiality of stored data. However, key management becomes a prime issue in the case of encryption. Compromise or failure of a key storage facility may lead to the loss of data. Therefore, cryptographic keys must be stored in a robust manner and a single point of failure should not affect the availability of data.

The security concerns of outsourcing data to public clouds, serves as our motivation to work for the development of data security technique. We aim for a technique capable of addressing the aforementioned critical issues. We propose a data security scheme that uses key manager servers for the management of cryptographic keys. Shamir's (k, n) threshold scheme is used for the management of keys that uses k shares out of n to rebuild the key.

II. FILE ASSURED DELETION (FADE)

The FADE protocol provides privacy, integrity, access control, and assured deletion to outsourced data. The FADE uses both symmetric and asymmetric keys. Sym-metric keys are protected by using Shamir's (k, n) scheme to ample the trust level in the key. The FADE works with a group of key managers (KM). Following keys are used by FADE protocol.

The variable K is termed as data key and is used to encrypt file F of the client. The S is a secret key that is used to encrypt K. The public/private key pair generated by KMs is represented by (ei, di) and is used to encrypt S. The K and S are symmetric keys. The operations supported by FADE are: (a) File upload, (b) File download, (c) Policy Revocation, and (d) Policy Renewal. The aforementioned operations are explained below.

2.1 File Upload

When data must be uploaded to the cloud, the client re- quests the KM to generate a public/private key pair. The said is done by sending a policy file, Pi, to the KM. The KM generates the key pair, associate that with the Pi, and sends the public part of the key (ei, ni) to the client. After receiving the public key for Pi, the client performs the following cryptographic operations. The client encrypts the F with the K to generate {F}K (F encrypted with K). The K is then encrypted with Si to get{K}Si. Subsequently, Si is encrypted with the public key generated by the KM with Pi. The Si is encrypted using asymmetric encryption (Siei mod n). The Pi, {F}K, {K}Si, and(Sie mod n) are uploaded to the cloud afterwards. The hashed MAC (HMAC) of data file is also uploaded with the encrypted file. The client deletes all of the symmetric keys through secure overwriting.

When FADE works with full quorum of KMs, Si is di- vided into n shares and each share is encrypted with a public key generated by one of the KMs. The key is divided based on Shamir's (k, n) threshold scheme. To get back the Si, k shares are needed. The FADE protocol does not authenticate the client for the file upload process.

2.2 File Download

The client requests cloud to download the file and en- crypted keys. The client checks for the integrity of the file through the HMAC. Afterwards, the client generates a secret number R and calculates Rei and then generates SieRei = (SiR)ei. The (SiR)ei is then sent to KM for decryption. The KM decrypts (SiR)ei with corresponding di and sends back SiR. At this point, ABE comes into the play. The KM sends SiR with ABE, where the attributes used are based on Pi. The client extracts Si from the received message and decrypts K that in turn is used to decrypt F. The process is highlighted.

2.3 Policy Revocation

If Pi needs to be revoked, then the client requests the KM by sending the Pi. The KM generates a random number r and sends it to the client after encryption with ABE. The authentic client decrypts r, calculates the hash value, and sends it back to the KM. After verification, the KM revokes Pi and acknowledges the client.

2.4 Policy Renewal

If Pi needs to be renewed as Pj, client downloads all of thekeys and sends Pi and encrypted Si to KM along with Pj.The KM decrypts Si. Moreover, KM sends new public keyparameters (ej, nj) to client. Wewill now formally analyze FADE in the following section.

III. DaSCE

The security of FADE depends on the key exchange between the client and the KM. if the key exchange is compromised, then si is compromised, that in turn leak all the keys and the data. We observed that the reason for the said attack is the independence of communication steps between the client and the KM that allows the attacker to launch the attack and subvert the whole process.

3.1 File Upload

For establishing a session key, we assume that the required parameters are fixed and publically available to all of the users. We call these parameters as α and p, where α is a large number known as the primitive root and p is a large prime number. The process comprises of following steps.

The client generates a random number x and calculates αx mod p and sends to the KM.

- The KM generates a random number y and calculates
- αy mod p. The KM also calculates (αx)y as a session key,
- EK, between client and KM.
- The KM generates digital signature over {αy, αx}
- (SKM{αy, αx}) and encrypts it with the generated ses-
- sion key to generate EK(SKM{αy, αx}).
- □□The KM sends (αy , EK(SKM{αy, αx})) to the client.
- □□The client verifies the signature using the public

key

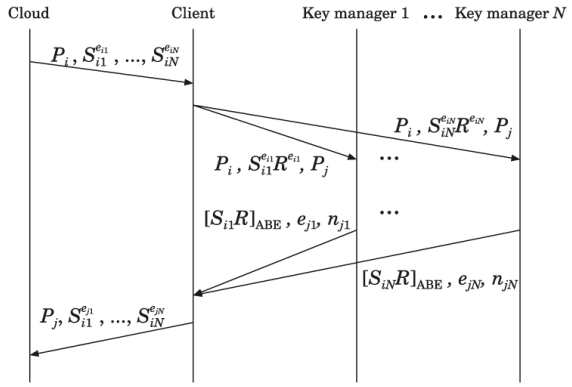
- of the KM and calculates the session key as (αy)x.
- The client calculates EK(SCli{ αx , αy }) and encrypts Pi
- with EK and sends both of the values to the KM. Thesent message contains EK(SCli{ αx , αy }), EK(Pi).
- The KM verifies the signature of the client. Upon suc-cessful verification, the KM decrypts Pi and generates(ei, ni) with Pi. The KM stores the decrypted Pi.
- The KM encrypts (ei, ni) with the EK to generate(EK(ei, ni)), which is sent to the client.
- The client encrypts the file F with key K, calculatesMAC with IK; and encrypts K and IK with Si. Afterwards Si is encrypted with ei. Subsequently, the clientsends all the encrypted data to cloud.
- The client erases all of the keys except public key parameters received from the KM.

3.2 File Download

The file download process of DaSCE is depicted. The process starts with the client downloading the data from the cloud. To decrypt F, we need the K that is encrypted with Si. The Si is encrypted with (ei, ni), received from KM. The client establishes the session key with the KMs and during the process both the client and the KMs authenticate each other using digital signatures.

The process of key establishment and authentication is the same as discussed. In the third step, after verifying the authenticity of the KMs, the client generates a random number R and encrypts it with the public key of the corresponding KM. The client then calculates SieiRei and sends it along with its own signature and encrypted Pi. We combine these steps to minimize the communication overhead. The KM after verifying the digital signature of the client, decrypts Pi and checks whether the policy still holds or otherwise. If the policy is valid, then the KM decrypts SieiRei with the corresponding di to generate SiR. The purpose of R is to mask the actual value of Si. The KM encrypts SiR with the session key, which is sent to the client. It is noteworthy, that in FADE, SiR is returned by applying ABE. However, in the DaSCE, we do not use ABE, instead session key is used to send SiR to the legitimate user. Therefore, the access control is being managed by the aforementioned technique.

The client after receiving SiR extracts Si from SiR. It isimportant to remember that with multiple KMs, a share of Si will be received from at least k KMs. Consequently k number of Sis will be used to generate Si. The client decrypts K and IK using Si. It verifies the integrity of F using IK and decrypts F upon successful verification.



DaSCE file download

The file download process of DaSCE is depicted. The process starts with the client downloading the data from the cloud. To decrypt F, we need the K that is encrypted with Si. The Si is encrypted with (ei, ni), received from KM. The client establishes the session key with the KMs and during the process both the client and the KMs

3.3 Policy Revocation

The same process of key establishment, as discussed is used for the policy revocation in DaSCE. The client encrypts Pi with the session key and sends to KM. The KM after performing decryption on Pi revokes the keys generated with Pi. The deleted keys include the private key di and associated prime numbers pi and qi. It also sends acknowledgement to the client. When di associated with Pi is deleted, the corresponding Si cannot be decrypted. This results in logical deletion of F as K cannot be decrypted without Si. Therefore, we say that F is assuredly deleted. It is noteworthy that assured deletion does not correspond to physical deletion of data. It is difficult to get assurance of file deletion from system outside the administrative control of data owner.

To boost the level of trust in the proposed scheme, the key generation and management is not dependent on a single KM. Shamir's secret sharing scheme is applied to counter any malicious KM. Any malicious KM cannot get hold of Si independently. At least k number of KMs needs to be compromised in order to get access of enough di's that can be used to decrypt Si. It is also noteworthy that for decryption process Si is sent to KM. However, Si is not sent in plain as discussed in Section 5.3. The Si is masked by multiplication with R. Therefore, even if malicious KM

Keeps the resultant decrypted information, the extraction of Si will remain a challenge. Moreover,

aforementioned case of malicious KM seems hard to be translated into successful attack. If we build a case of a malicious user that somehow has got hold of some other user's encrypted Si, the malicious user has to go through the authentication process of at least k number of KMs to decrypt the Si. Moreover, the system can tolerate the failure of n – k server while providing the correct functionality. The aforesaid fact makes the system robust to the point of working of n – k servers.

3.4 Policy Renewal

The policy renewal does not involve any operation on F. The client downloads Si and Pi; establishes session key with the KM; and sends Pi, SieiRei, and Pj to KM by session key encryption. The KM decrypts SieiRei to obtain SiR and generates new public/private key pair for Pj. Therefore, the KM sends SiR and new public parameters (ej, nj) to the client. The client extracts Si and re-encrypts it with (ej, nj). Finally, the client sends Pj and encrypted Si to the cloud.

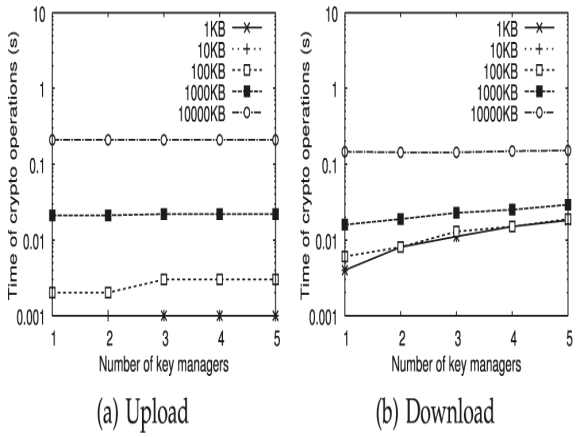
IV. FILE UPLOAD/DOWNLOAD WITH SINGLE KEY MANAGER

We used files of nine different sizes (0.3 KB, 1 KB, 10 KB, 30 KB, 50 KB, 100 KB, 500 KB, 1 MB, and 10 MB) to measure time consumption in file upload and download process. The results are provided in Fig. 15. In general, file transmission time increased with the increase in file size. However, in some cases the change in file transmission time was small that may be causeddue to network conditions at various times. Nevertheless, file transmission time was dependent on the network. In case of file upload, cryptographic operations time varied between 0.037 sec and 0.201 sec. The cryptographic operations time increases with the increase in the file size. In the case of 10 MB file, the cryptographic operations time makes 2.35% of total file upload time and 2.45% of file transmission time. The time for session key establishment almost remained constant (having slight changes). The largest time taken during key establishment was noted to be 0.0898sec that constituted 2.67% of the total upload time. Thepercentage for key establishment time was 2.39% for 10MB file. Similarly, in case of file download operations the cryptographic operations time varied from 0.039 sec to 0.211 sec.

The cryptographic operations time was dependent on the size of the file. Therefore, the time increases with the file size. However, it made lower percentage of total upload time and file transmission time. The key establishment time does not depend on the file size; therefore, it remains almost constant. Slight changes were possible, due to network transmission conditions. The DaSCE and FADE takes same amount of time for cryptographic operations. However, unlike FADE, we perform additional steps for key establishment in DaSCE that makes an additional overheads. Therefore, key establishment process increases the time consumption of DaSCE as compared to the protocols that run without establishing the session keys. It is noteworthy that the increase in time consumption upturns the security level for policy files, symmetric, and asymmetric keys used in the DaSCE.

V. FILE UPLOAD/DOWNLOAD WITH MULTIPLE MANAGERS

Subsequently, we evaluated the performance of DaSCE by using multiple KMs. The file sizes we used were 0.3 KB, 1 KB, 10 KB, 50 KB, 100 KB, 500 KB, and 1MB. The number of KMs used was one, three, five, seven, fifteen, 25, and 50. Fig. 16 revealed the key establishment time and cryptographic operation time for the aforementioned files sizes and the KMs. The key establishment time increased with the increase in the number of KMs. This is because the client had to complete all the message passing steps necessary to establish the key with all the KMs. The key establishment time varies between 0.069 (single KM) seconds and 0.24 seconds (50 KMs). It must be noted that there was slight increase in the key establishment up to ten KMs. However, with higher number of KMs the increase followed a higher trend. As discussed earlier, the increase in time consumption due to key establishment augments the security level. Therefore, we say that user has to select the number of KMs judicially. A balance between tolerate able timeconsumption and security level in needed while deciding the number of KMs. In the coming discussion we will also see that the key establishment time constitutes low percentage of total time. The cryptographic operation time remained constant for the file of same size as final symmetric encryption is done on client with generated keys (symmetric key, K). It depicts the key establishment time and cryptographic operation time taken by file download with multiple KMs. It must be noted that the key establishment constituted a low percentage of the total consumed time.



VI. IMPLEMENTATION AND EVALUATION

We used C# for implementing a working prototype of DaSCE. The .Net cryptographic packages were used for the involved cryptographic operations. Large prime numbers were handled by using the BigInt class. Policies were uploaded as a separate file to the cloud and the KM.

The system consists of two servers (the cloud and the KM) and a client (work station). Multiple policies were combined using OR and/or AND operations. The policy and data files were not merged into a single file, to keep the policy renewal operation light weight. According to the processes described in Section 5, we also implemented the client side software functions, such as file upload, down-load, revocation, and renewal.

In our prototype, the client interacts with the KM (s) and the cloud for setting up the keys, and uploading/downloading data. The KM sets up the keys, revokes, and/or renews policies and manages the keys accordingly. We evaluated the DaSCE on the basis of: (a) Key(s) establishment time, (b) Key Transmission time, (c) File transmission time, and (d) Cryptographic operations time. It is noteworthy, that the time required for key establishment is the time for setting up a session key be-tween the involved parties. The cryptographic operations time is the time taken by AES and MAC operations. Above given parameters collectively make up total file upload/download time. Moreover, the aforesaid parameters are evaluated using single and multiple KMs.

VII. CONCLUSIONS AND FUTURE ENHANCEMENT

We proposed the DaSCE protocol, a cloud storage security system that provided key management, access control, and file assured deletion. Assured deletion was based on policies associated with the data file uploaded to cloud. On revocation of policies, access keys are deleted by the KMs that result in halting of the access to the data. There-fore, the files were logically deleted from the cloud. The key management was accomplished using (k, n) threshold secret sharing mechanism. We modeled and analyzed FADE. The analysis highlighted some issues in key management of FADE. DaSCE improved key management and authentication processes. The working of the DaSCE protocol was formally analyzed using HLPN, SMT-Lib, and Z3 solver. The performance of the DaSCE was evaluated based on the time consumption during file upload. The direction in which this project can be enhanced by generating policy files when user upload their files in cloud. Inside the policy file contains username, filename which is upload by the user and access permission. This policy file will be encrypted by Key Manager by using user attributes. In future, the DaSCE methodology can be extended to secure group shared data and secured data forwarding

REFERENCES

[1] Ali M, Bilal K, Khan S.U, Veeravalli B, Li K, and Zomaya A.Y. (2015), “DROPS: Division and Replication of Data in the Cloud for Optimal Performance and Security,” IEEE Transactions on Cloud Computing, DOI: 10.1109/TCC.2015.2400460.

[2] Ali M, Khan S.U, and Vasilakos A.V.(2015), “Security in cloud computing: Opportunities and challenges,” Information Sciences, Vol. 305, pp. 357-383.

[3] Armbrust M, Fox A, R. Griffith.(2010), “A View of Cloud Computing,” Communications of the ACM, Vol. 53, No. 4, pp. 50-58.

[4] Blumenthal M.S.(2011), “Is Security Lost in the Clouds?” Communications and Strategies, No. 81, pp. 69-86.

[5] Cachinand C and Schunter M. (2011), "A cloud you can trust," IEEE Spectrum, Vol. 48, No. 12, pp. 28-51.

[6] CloudSecurityAlliancesecurityalliance.org/initiatives/cdg/CSA_CCAQIS_Survey.pdf (accessed March 24, 2013).

[7] Cremers C. (2008), "The Scyther Tool: Verification, falsification, and analysis of security protocols." In Computer Aided Verification, Springer Berlin Heidelberg, pp. 414-418.

[8] Diffie W, Oorschot P.C.V, and Wiener M.J. (1992), “Authentication and authenticated key exchanges,” Designs, Codes and Cryptography, Vol. 2, No. 2, pp. 107-125.

[9] En N and Srensson N. (2003), “An extensible SAT-solver,” Lecture Notes in Computer Science, vol. 2919, Springer, pp. 502-518.

[10] omes C.P, Kautz H, Sabharwa Al, and Selman B. (2007), “Satisfia-bility solvers,” In Handbook of Knowledge Representation, Elsevier.