Journal of Theoretical and Applied Information Technology
31st March 2017. Vol.95. No 6
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A USER FRIENDLY AND IMPROVED DESIGN FOR SECURE DELETION IN CLOUD STORAGE

[1]BENTAJER AHMED, [1]ENNAAMA FAOUZIA, [2]HEDABOU MUSTAPHA and [1]ELFEZAZI SAID

[1]Cadi Ayyad University, High School of Technology of Safi, LAPSSII Laboratory, Morocco
[2]Cadi Ayyad University, National School of Applied Sciences of Safi, MTI Laboratory, Morocco

E-mail: [1](a.bentajer, faouzia.ennaama,selfezazi)@gmail.com, [2]mhedabou@gmail.com

ABSTRACT

Assured file deletion is a major concern in cloud computing security. The countermeasures proposed by Cloud Service Providers (CSPs) are not totally satisfactory from the data owner's perspective. Because, CSP do not offer an irrefutable proof of an assured file deletion. Other proposed approaches for achieving assured deletion in cloud storage are based on the use of encryption operations. File Assured Deletion (FADE) is most efficient amongst them. The system is built upon cryptographic techniques to guarantee privacy and integrity of outsourced files. This paper illustrates a secure yet user friendly update for FADE. The proposed update, is easy to implement, require less computational resources and ensure assured file deletion. Furthermore, we implemented a prototype of our update to validate the model, and verify that it offers assured deletion with a minimal trade-off of performance for large files upload and download operations.

Keywords: *Cloud Computing, Secure Deletion, Confidentiality, Cloud Storage.*

1. INTRODUCTION

Cloud Computing is a new wind of change in IT's world. SMEs (Small and Medium Companies) considers it as a great solution to be competitive [1]. The concept has significant trend with a potential of increasing agility, flexibility, and lowering the costs. Moving to the cloud need to be especially cognizant of the various obstacles that organizations should be aware of to take some preventive measures [2, 31]. The reason is that security in the cloud differs from that in in-house IT infrastructure [3]. Cloud Storage is a delivery model proposed by CSPs (Cloud Service Providers) which offers an abstraction of infinite storage on-demand (e.g. Dropbox [4]) that helped SMEs saving millions of dollars [3, 5]. However, the consumer's perspective of cloud storage is still moderated. The concern is about the CIA triad (Confidentiality, Integrity and Availability) [5, 6]. The reason behind this, is that the consumer does not have a clear vision of CSPs procedures to ensure data confidentiality in term of storage location or assured deletion. As mentioned by NIST guide for assured deletion and Garfinkel et al. in their study of disk sanitization [9, 8] deleting a file only remove its entry from the table and does not delete it from the physical media until new data overwrites it. Likewise, CSPs are not offering a clear vision to their consumers about the procedures for file's assured deletion, what causes them to be

worried about the remnant of replicated files[8, 9, 10].

The confidentiality concern prompted some organizations to take some management preventive solutions [11] as the deletion of replicated files after a finite number, of year and files should not exceed the country's bound [3]. But it stills not enough as secure delete in the cloud depends on a million different variables. In their analysis of data remnants in cloud storage services, Quick et al. have identified that potential data resides even when anti-forensic process was undertaken, the study showed that some data remain on the computer hard drive after using a storage software (Dropbox, Google drive) or when using the web browser to access cloud storage service, also uninstalling the storage client from the virtual machine does not really remove the synchronized folder then the files are exposed to confidentiality leakage and are not assuredly deleted[12, 13, 14]. Thereby, the implication of CSPs, researchers and engineers become more and more needed to save files confidentiality since proposed conventional methods [8, 9, 15] are no longer effective and viable for assured deletion. A recommended approach for file's assured deletion in the cloud is the use of encryption mechanism, the aim is to encrypt files before outsourcing them. Vanish's solution is adopted for this purpose [16], the system implements policy-based approach to encrypt files before outsourcing them and assuredly delete them by deleting the corresponding cryptographic keys after a defined time period ensuring that files remain unreadable. Vanish is time-based policy, Tang et al. proposed an evolved implementation called FADE (File Assured DEletion) which is a generalization of time based file assured deletion [17]. FADE combines one- or two-level Boolean expression to generate the policy's corresponding key and encrypt files before outsourcing them, yet maintaining and protecting keys leverage another security issue: can we trust the Key Manager? In our proposed update, we try to reinforce the trust between the KM and file owner by splitting the encryption duty, because if we can not trus the cloud storage it have to be the same for KM. Besides the proposed update use stronger control key, since the AES-128 key used in FADE can be now recovered through a biclique attack [29].

The rest of the paper is organized as follows: in the "Background" section, we present a survey about methods and mechanisms used for secure deletion and policy based secure deletion. In the "Security Analysis of FADE" section, we present FADE and discuss the Key Manager's security concern because it is considered as a third party in FADE's design and being that security of files is more centralized on it. In the "Proposed Design" section, we present our design and the prototype implementation, which will be validated by a statistical study, that is illustrated in the "Evaluation and discussion" section. Finally, we come to end with our conclusions.

2. SECURE DELETION

Secure deletion refers to a set of operations and processes to assuredly delete data from a storage media and make the file recovery infeasible for a given level of efforts. Referring to NIST Guide [9] and Garfinkel et al. study [8], assured deletion can be achieved in multiple ways clearing, purging/sanitizing, degaussing, media destruction and encryption the most common are secure overwriting and encryption based deletion [10, 18]. In the cloud using secure overwriting through a pattern is an abstract operation. Since deleting files are due to CSP's praxis, nothing guarantee that all replicated files will be simultaneously overwritten. Protection of outsourced data through encryption has been considered by researchers in the field, Darren et al. analyzed popular cloud storage service with different case study in different devices (laptop and mobile) and showed that there is some leaked information when using storage service [19, 12, 13, 14]. Besides they showed that it was possible to gain full access to files in Google drive and Dropbox when the client software is installed without the need to have credential information (username and password), also in previous version of Dropbox client software, it was possible to copy the Sync file to another computer and synchronize it to an account. Besides, user privacy concerns in OSN (online social network) is considered as a big problem since deleted pictures may not be immediately removed from the OSN servers which can compromise the user's privacy [20].

For all this reasons and others, we can be sure that file's privacy in the cloud should not be a one side responsibility, but the consumer needs to be part in order to be sure that the confidentiality of its data is preserved. Ateniese et al. [21] proposed an auditing system that verifies the integrity of outsourced data. Wang et al. proposed also a secure and efficient access to large scale outsourced data mechanisms that support changes in user access rights [22], but those solutions involve some engineering change and their implementation is more challenging. NIST guide and researchers [6, 8, 9] agreed that the optimal solution for assured deletion is encryption; because, in contrast of alternative methods, cryptographic based deletion techniques can be easily deployed with all kind of storage system regardless of their physical location and with no need to make engineering changes, besides the control of data is maintained by the consumer and security properties are derived from cryptography. That is why many storage manufacturers started releasing their Self-Encrypted Drives (SEDs) [8], but this does not involve the consumer in encryption process since cryptographic operations have to be done on off-premise side (at CSP). Besides, the technology is not immune to attack, thing that have been demonstrated by Daniel Boteanu at Black hat Europe 2015 in Amsterdam through the hot plug attack. However, the development of encryption algorithms and schemes played an important role in the evolution of secure deletion in the cloud [23]. The operation has taken another meaning and the implication of consumer play an important role. The aim is the use of policy based encryption, meaning that each file is associated with a set of policies in order to make the delete operation easy and flexible when the policy meets the need by deleting the associated key.

2.1 Policy Based Deletion

The policy-based secure deletion scheme, aims to maintain files on storage media and selectively delete some of them when a policy meets the needs, the general idea is similar to access control [24] but the aim here is the delete and not the access operation. A user will be able to decrypt a file, if and only if his attributes satisfy the policy of the respective file. Policies may be defined over attributes using conjunctions, disjunctions and (k,n)-threshold gates, i.e., ***k*** out of ***n*** attributes have to be present. For example, in Figure. 1, File1 is associated to policies Alice and Exp2017. Meaning that the decryption key will be deleted when Alice leaves the organization or the policy ”***Exp2017***” is satisfied (the date is 01-01-2018 unless it was renewed). Perlman [25], introduced time-based file assured deletion which mean that, after a predefined time duration, keys are deleted and makes associated files inaccessible. The operation consists of encrypting the file with a Data Key which in its turn encrypted by a Control key that is maintained by a separate third party (Key Manager) and when the predefined period expire the Key Manager remove the Control Key. This design was later prototyped in Vanish [16].

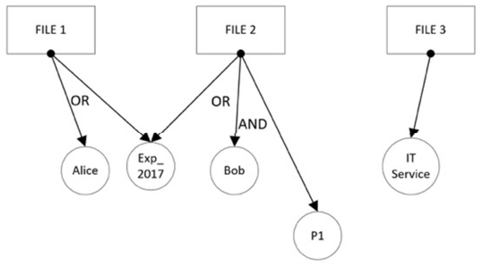

Figure 1: Example of policy graph

3. FILE ASSURED DELETION (FADE)

FADE’s popularity increased quickly, researchers and engineers starts studying its capabilities [26] and security concerns [27]. FADE is built upon standard cryptographic techniques, the encrypted files remain on, an untrusted, cloud storage and encryption keys are independently maintained by, a trusted, Key Manager (KM). The delete operation is similar to ABE (attribute based encryption)[23] paradigm whereby access rights are granted to users through the use of policies. FADE’s upload scenario is as follow (Figure. 2):

- For each policy, P_i the Key Manager generates large RSA prime number ***pi*** and ***qi***.
- Calculate ***ni*** = ***pi*** × ***qi***
- Then the Key Manager choose RSA Private/Public pair control key (***di,ei***)/(***ni,ei***).
- Key Manager sends its public key (***ni,ei***) to data owner.
- Data owner generates a data key K and a secret key Si (Both K and Si are generated using symmetric-key encryption AES-128).
- Data owner sends to cloud the encrypted file ***F***. [***Enc{K}Si, Siei , Enc{F}K***] and drop ***K*** and ***Si*** since they are stored at cloud storage.

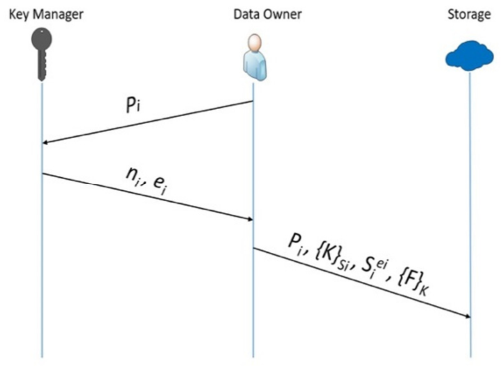

Figure 2 : FADE upload operation

For file download scenario and more information about FADE design, we refer the reader to [17].

3.1 Security Analysis of FADE

As mentioned earlier FADE’s design is based on blinded RSA, which have shown some side channel leakage [28], meaning that data owner encrypts the file through a Data Key, and this data key is further encrypted by key manager’s control key. Since CSP is an untrusted third party, we believe that it has to be the same for KM. In the upload operation (Fig. 2), data owner’s cryptographic keys are stored at the CSP, while KM store its cryptographic key locally. Then, if CSP colludes with KM they can decrypt sensitive files following the same download operation described in FADE.

Ranjan et al. [27] have shown, in their network security study of FADE, that some

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

information (policy, public key and private key) can be easily leaked by sniffing the network flow between data owner and key manager. If there is a policy belonging to many users e.g. P1: members of IT group. A range of files satisfy this policy (P1) so if one client captures the Pi of some ith client he can get client's secret key and access all files that satisfy the same policy even if the client asks for key deletion. Also, Habib et al. [26] pointed that FADE's design has complex system architecture for storing keys at the Key Manager.

FADE uses symmetric encryption (AES-128) for cryptographic operations. A study [29] proved that a biclique attack could recover an AES-128 keys with a computational complexity of 2126.13 and data complexity 256. Thus, with the rise of computational machines, the recovery of such key will become easier in the coming decades.

3. PROPOSED UPDATE

We would like to inform the reader that our update benefits from all security aspects in FADE, our main contributions are: (i) adding a secure channel between data owner and KM for key exchange. The Control Key is encrypted before it is sent to data owner. (ii) The design is lightweight through the use of XOR operation, which makes it more suitable for personal use (iii) cryptographic process is split between data owner and KM. The operation is described in this section.

In our case study, we consider that KM is an untrusted third party since it is the same for CSP. Our proposed update splits encryption's duty. The cryptographic key is divided into two parts: KM's key ***Ke*** and data owner's key ***Kc***. The combination of the two keys is used for cryptographic operations. ***Ke*** is an AES-256 key, the choice was made based on the study presented in [29] because AES-256 needs more level effort to be discovered and does not threaten the practical use of AES due to its high computational complexity. In client's side we use the XOR operation, a random key is generated based on the length read from the file. The idea behind XOR encryption is that it is impossible to reverse the operation without knowing the initial value of one of the two arguments which is the case in our proposed solution. For example, if we XOR two variables of unknown values, we cannot know from the output what the values of those variables are. If (A □ B) returns TRUE, we cannot know whether A is FALSE and B is TRUE, or whether B is FALSE and A is TRUE. However, if KM security is compromised and the attacker get access to ***Ke***, or the key manager colludes with cloud storage it will be impossible to get access to the file because Kc is needed, and it is encrypted with the client's Public key.

File upload and keys exchange scenario (Figure. 3):

- For each policy ***Pi***, KM generates secret AES-256 key $\boldsymbol{K_{ei}}$.
- KM sends Kei encrypted with data owner's public key Enc$\{\boldsymbol{K_{ei}}\}_{\boldsymbol{PubC}}$
- Data owner generates a data key ***Kc***.
- Data owner encrypts the file ***F*** with its data key **Enc{F}Kc**
- Data owner sends to cloud [P_i, Enc{Enc{F}Kc}K_{ei}]
- Data owner sends its data key encrypted with its public key to KM Enc$\{Kc\}_{PubC}$

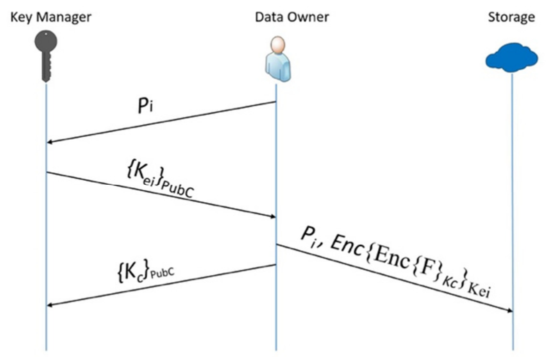

Figure 3 : Proposed Update Upload Operation.

File Download and key exchange scenario (Figure. 4):

- Data owner fetch [$\boldsymbol{P_i}$,***Enc{Enc{F}***$_{\boldsymbol{Kc}}$***}***$\boldsymbol{K_{ei}}$] from cloud storage
- Data owner sends $\boldsymbol{P_i}$ to KM.
- KM sends the corresponding control key ***Enc{Kei}***$_{\boldsymbol{PubC}}$ and data key ***Enc{Kc}***$_{\boldsymbol{PubC}}$.
- Data owner decrypts $\boldsymbol{K_c}$ and $\boldsymbol{K_e}$ using its private key.
- Data owner decrypts the file ***F***.

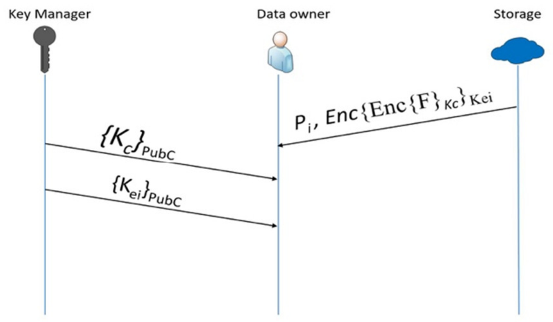

Figure 4 : Proposed Update Download Operation

4. EVALUATION AND DISCUSSION

We used the Dropbox API [4, 30] to download/upload plain files from/to Dropbox for different sizes.
All the tests have been done on a computer with i7-5500 2.40 Ghz processor and 16Go of Ram. It is important to note that the performance results depend on the deployment environment. Nevertheless, we emphasize that our experiments can prove the feasibility of the update in reinforcing security of FADE.

We implement a prototype of our update atop Dropbox, evaluate the experimental results and performances. It is crucial that the modification does not introduce substantial performance overhead that will lead to a big increase in data management costs and significant computational overhead. Therefore, our empirical results aim to answer the following issue: Does the update improve security of KM on behalf of time overhead performance? We average each of our measurement results over 5 different trials.

We measured the time performance of the design using our developed prototype. We divided the overhead time of each measurement into three components:

- Data transmission time: The download/upload time between file owner and cloud storage.
- Cryptographic operations: The total computational time used for performing AES and XOR encryption operation.
- Key manager: The interaction time between KM and file owner for generating and downloading cryptographic key and policy.

The experiments aim to measure the running time (in second) of file upload and download operations for different sizes (Table 3 and 4) and to calculate the overhead cost time (table 1 and 2). Our experiments showed that plain file transmission is a dominant factor and the cryptographic operations time increases linearly with the file size and remains negligible compared to time upload and download operations.

Table 1: Overhead Cost Time for Upload Operation

File Size	Proposed update overhead cost time
1KB	21.02%
10KB	20.28%
100KB	11.69%
1MB	3.44%
5MB	1.21%
10MB	0.97%

Table 2: Overhead Cost Time for Download Operation

File Size	Proposed update overhead cost time
1KB	89.86%
10KB	66.95%
100KB	36.91%
1MB	27.21%
5MB	8.97%
10MB	7.80%

First, the proposed update is a security improvement of FADE, which is considered to be the best known scheme for secure deletion in cloud storage.

We note that when the file size is small, the cryptographic operations and data transmission could be equal to plain file upload and download. However, the whole operation time is negligible and data owner could not feel the overhead time.
In FADE's download operation [17], the data owner fetches [$\boldsymbol{Enc\{K\}_{Si}}$, $\boldsymbol{S_i^{ei}}$, $\boldsymbol{Enc\{F\}_K}$] from the storage cloud. Then the data owner generates a secret random number $\boldsymbol{R}$, computes $\mathbf{R^{ei}}$, and sends $\boldsymbol{R^{ei}.S_i^{ei}}$ to KM to request for decryption. The KM then computes and returns $\boldsymbol{((R.S_i)^{ei})^{di}}$ to the data owner. The data owner can now remove R and obtain Si, and decrypt $\boldsymbol{Enc\{K\}_{Si}}$ and hence $\boldsymbol{Enc\{F\}_K}$.

Following the same procedure, KM and CSP can collude to get access to data owner files. However, the proposed update splits the cryptographic operation between KM and data owner. The file is encrypted first by $\boldsymbol{K_c}$ then by $\boldsymbol{K_e}$. And since $\boldsymbol{K_c}$ is stored encrypted by data owner's public key at KM, it is hard for KM and CSP to collude and read sensitive data. Moreover, even if KM's security is compromised, data owner's file will remain protected since the attacker can't get $\boldsymbol{K_c}$.

The key leakage inherent to FADE [26] is no longer a problem, because the flows between key manager and client are reduced and reinforced through encryption (each key exchange is encrypted). Also, the problem of AES-128 key-recovery attack [29] is no longer a concern for coming decades, because the updated design uses an AES-256 key.

The empirical study of the proposed update shows that the overhead cost is slightly negligible when we deal with files of large size. Namely more than 1MB, the overhead tends to a negligible value, which lead us to conclude that the update is a user friendly solution because it does not involve a complex system architecture for generating cryptographic keys as FADE. that reinforce the security of FADE and data privacy. In a nutshell, the proposed update enhances the security of FADE without penalizing significantly its performance (Fig. 5 and 6).

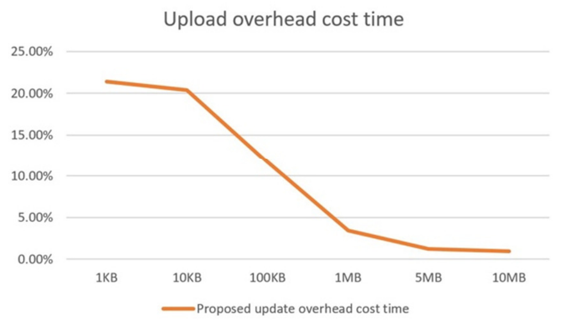

Figure 5 : Upload overhead cost time

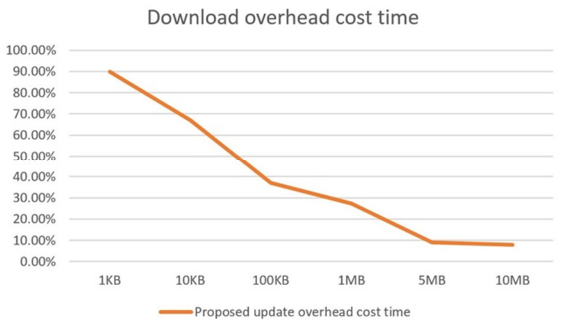

Figure 6 : Download overhead cost time

5. ASSUMPTION

Our main design goal is to provide assured deletion of files with a high security insurance. When the file owner asks for a file to be, the KM delete the corresponding Control Key. In case that he does not, and the KM colludes with the cloud storage in order to get sensitive data, it will be hard for both of them since the file is encrypted by the file owner's data key which is stored encrypted at the KM. However, we stress that this design is not intended to be a formal specification (indeed many important business and engineering questions would need to be addressed). However, it's only meant reinforce the security concern of FADE and emphasis it security concern as described in [26,27,28,29] . We believe that at such stage it is still important to improve KM security robustness in order to minimize the chance of being compromised since it's considered as a high target for external attacker.

6. CONCLUSION

Nowadays, data confidentiality become more challenging and sensitive in public cloud storage solutions. While its benefit is well understood by consumers, its security concern in term of confidentiality and assured deletion are not. Our contribution presents an introduction to some confidentiality concerns about public storage solution in term of assured deletion and proposed mechanism to mitigate them as FADE.

Although, FADE proved its capability to ensure assured file deletion, but the design showed some leakage that we tried to improve in order to benefit from the security advantage of cryptography by involving the consumer in encryption process and increasing the security of key management. When FADE design focused on files assured deletion our proposed update focused on keys security by splitting the duty of encryption between the consumer and key manager. We noticed that our new design's performance stays insignificant when the file size increases. Thus, it is more suitable for organizations that aim to archive large files. In the other hand individual customers who manipulates small file sizes can still get best result. So we can say that the new design improves security of key management without affecting the overhead performance.

In our future work, we will try to design a solution similar to ABE mechanism, in order to have a provably secured and optimized system that will benefit from all the advantage of cryptography security with less interaction with the key manager in order to reduce the risks of web flow's leakage and to leverage the Key Manager trust's problem.

ACKNOWLEDGEMENT

Authors are thankful to SEF and MH for their helpful discussions and for providing several useful references. They wish also to thank Miss. Hajar Chiba for pointing out first layer English corrections. They would also like to thank the reviewers and Editor for the time that they will consider for the work.

REFRENCES:

[1] Rai, R., Sahoo, G., Mehfuz, S.: Exploring the f actors influencing the cloud computing adoptio n: a systematic study on cloud migration. Sprin gerPlus 4, 1-12 (2015). doi:10.1186/s40064-01 5-0962-2

[2]
[3] A. Bentajer , K. Abouelmehdi, L. Dali, S. Elfez

azi, M. Hedabou, and F. ElAmrani : An assessing approach based on fmeca methodology to evaluate security of a third party cloud provider. Journal of Theoretical and Applied Information Technology. vol 74, 336-344 (2015)
[4] L.K. Ronald and R.D. Vines: Cloud Security: A Comprehensive Guide to Secure Cloud Computing. Wiley Publishing, Hoboken (2010)
[5] Dropbox for .NET Developers. https://www.dropbox.com/developers/documentation/dotnet
[6] K. Hashizume and Rosado, D.G., Fernandez-Medina, E., Fernandez, E.B.: An analysis of security issues for cloud computing. Journal of Internet Services and Applications, vol. 4, p.1-13 (2013). doi: 10.1186/1869-0238-4-5
[7] Y.A.A.S. Aldeen and M. Salleh and M.A. Razzaque: A comprehensive review on privacy preserving data mining. SpringerPlus, vol. 4, p. 1-36 (2015). doi:10.1186/s40064-015-1481-x
[8] S. Aljawarneh. Advanced Research on Cloud Computing Design and Applications, IGI Global. Hershey, doi:10.4018/978-1-4666-8676-2
[9] S.L. Garfinkel and A. Shelat: Remembrance of data passed: a study of disk sanitization practices. IEEE Security Privacy, pp.17-27, (2003). doi:10.1109/MSECP.2003.1176992
[10] NIST, Computer Security Division, Information Technology Laboratory: NIST Special Publication 800-88 Revision 1 Guidelines for Media Sanitization. Gaithersburg (2014). NIST, Computer Security Division, Information Technology Laboratory
[11] S. Skorobogatov: Data remanence in flash memory devices. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2005: 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005. Proceedings, vol. 3659, pp. 339-353. Springer, Berlin, Heidelberg (2005)
[12] D. Yimam and E.B. Fernandez: A survey of compliance issues in cloud computing. Journal of Internet Services and Applications, vol. 7, pp. 1-12, (2016). doi:10.1186/s13174-016-0046-8
[13] Q. Darren and R.C. Kim-Kwang: Dropbox analysis: Data remnants on user machines. Digit. Investig., p. 3-18, (2013). doi:10.1016/j.diin.2013.02.003
[14] Q. Darren and R.C. Kim-Kwang: Google drive: Forensic analysis of data remnants. Journal of Network and Computer Applications, pp. 179-193 (2014). doi:10.1016/j.jnca.2013.09.016
[15] Q. Darren, B. Martini, and R.C. Kim-Kwang: Microsoft skydrive cloud storage forensic analysis. In: Cloud Storage Forensics, 1st Edition, 1st edn., pp. 23-61. Syngress, Boston (2014)
[16] K. Munadi, F. Arnia, M. Syaryadhi, M. Fujiyoshi, and H. Kiya: A secure online image trading system for untrusted cloud environments. SpringerPlus, Vol. 4, pp. 1-12, (2015). doi:10.1186/s40064-015-1052-1
[17] R. Geambasu, T. Kohno, A.A. Levy, and H.M. Levy: Vanish: Increasing data privacy with self-destructing data. In: Proceedings of the 18th Conference on USENIX Security Symposium, pp. 299-316 (2009)
[18] Y. Tang, P.P.C. Lee, J.C.S. Lui, and R. Perlman.: Secure overlay cloud storage with access control and assured deletion. IEEE Transactions on Dependable and Secure Computing, Vol. 9, pp. 903-916, (2012). doi:10.1109/TDSC.2012.49
[19] P. Gutmann: Secure deletion of data from magnetic and solid-state memory. In: Proceedings of the 6th Conference on USENIX Security Symposium, Focusing on Applications of Cryptography – Vol. 6, pp. 8-8 (1996). USENIX Association
[20] Q. Darren, B. Martini, and R.C. Kim-Kwang: Cloud storage forensic framework. In: Cloud Storage Forensics, 1st Edition, 1st edn., pp. 13-21. Syngress, Boston (2014)
[21] S. Lai, J.K. Liu, K-K.R. Choo, and K. Liang.: Secret picture: An efficient tool for mitigating deletion delay on osn. In: Qing, S., Okamoto, E., Kim, K., Liu, D. (eds.) Information and Communications Security: 17th International Conference, ICICS 2015, Beijing, China, December (2015), Revised Selected Papers, pp. 467-477. Springer, Cham (2016)
[22] G, Ateniese, R. Di Pietro, L.V. Mancini, and G. Tsudik: Scalable and efficient provable data possession. In: ACM (ed.) Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, pp. 1-10 (2008). doi:10.1145/1460877.1460889. http://doi.acm.org/10.1145/1460877.1460889
[23] W. Wang, Z. Li, R. Owens, and B. Bhargava: Secure and efficient access to outsourced data. In: ACM (ed.) Proceedings of the 2009 ACM Workshop on Cloud Computing Security, pp. 55-66 (2009). doi:10.1145/1655008.1655016. http://doi.acm.org/10.1145/1655008.1655016
[24] H.C.A. van Tilborg, and S. Jajodia: Encyclopedia of Cryptography and Security. Springer, Boston, MA (2011). doi:10.1007/978-1-4419-5906-5

[25] M. Habiba, M.R. Islam, A.B.M.S Ali, and M.Z . Islam: A new approach to access control in cl oud. Arabian Journal for Science and Engineeri ng, vol. 41, pp.1015-1030 (2016). doi:10.1007/ s13369-015-1947-8
[26] R. Perlman: File system design with assured de lete. In: Third IEEE International Security in St orage Workshop (SISW'05), pp. 6-88 (2005). d oi:10.1109/SISW.2005.5
[27] A.B. Habib, T. Khanam, and R. Palit: Simplifie d file assured deletion (sfade) - a user friendly overlay approach for data security in cloud stor age system. In: IEEE (ed.) International Confer ence on Advances in Computing, Communicati ons and Informatics (ICACCI), pp. 1640-1644 (2013). doi:10.1109/ICACCI.2013.6637427. I EEE
[28] A.K. Ranjan, V. Kumar, and M. Hussain: Secu rity analysis of cloud storage with access contr ol and file assured deletion (fade). In: IEEE (ed .) Second International Conference on Advance s in Computing and Communication Engineeri ng (ICACCE), pp. 453-458 (2015). doi:10.110 9/ICACCE.2015.10. IEEE
[29] C. D. Walter. Longer randomly blinded RSA keys may be weaker than shorter ones. In Proceedings of the 8th international conference on Information security applications (WISA'07), Sehun Kim, Moti Yung, and Hyung-Woo Lee (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 303-316. (2007).
[30] B. Tao, H. Wu, "Improving the biclique crypta nalysis of AES", Proc. Australas. Conf. Inf. Se curity Privacy, vol. 9144, pp. 39-56, Jun. 2015. http://dx.doi.org/10.1007/978-3-319-19962-7_ 3
[31] Dropbox Core API. https://www.dropbox.com/developers-va/core/docs
[32] Jayaprakash Kar, and Manoj Ranjan Mishra, "Mitigating Threats and Security Metrics in Cloud Computing," Journal of Information Processing Systems, vol. 12, no. 2, pp. 226~233, 2016. DOI: 10.3745/JIPS.03.0049.

ANNEXURE:

Table 3: Overhead Cost Time for Upload Operation

File Size	Plain upload	Proposed update upload	Policy transmission	XOR(second)	AES(second)
1KB	0.365	0.693	0.292	0.012	0.024
10KB	0.466	0.778	0.271	0.015	0.026
100KB	0.978	1.339	0.322	0.013	0.026
1MB	1.9	2.417	1.9	0.187	0.029
5MB	6.73	7.334	6.73	0.13	0.084
10MB	8.12	8.753	8.12	0.23	0.093

Table 4: Overhead Cost Time for Download Operation

File Size	Plain upload	Proposed update download	Policy transmission	XOR(second)	AES(second)
1KB	1.41	1.712	0.262	0.015	0.025
10KB	1.57	1.89	0.279	0.015	0.026
100KB	3.25	3.63	0.341	0.013	0.026
1MB	15.6	16.137	0.321	0.187	0.029
5MB	51.6	52.225	0.411	0.13	0.084
10MB	75.5	76.233	0.41	0.23	0.093