Journal of Theoretical and Applied Information Technology
15th December 2019. Vol.97. No 23
© 2005 – ongoing JATIT & LLS

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

A DATA CONFIDENTIALITY SYSTEM BASED ON TRUSTED PLATFORM MODULE IN CLOUD STORAGE ENVIRONMENT

[1]IGARRAMEN ZAKARIA, [2]HEDABOU MUSTAPHA, [3]BENTAJER AHMED

[1,2]ENSA School of Safi, Cadi Ayyad University, MTI Lab, Morocco

[2]Mohammed VI Polytechnic University, Ben Guerir, Morocco

[3]ENSA School of Tetouan, Abdelmalek Essaadi University, Morocco

E-mail: [1]zakaria.igarramen@ced.uca.ma, [2]mhedabou@gmail.com, [3]a.bentajer@gmail.com

ABSTRACT

Data confidentiality is a major concern in cloud storage environment security. A number of methodologies and algorithms are available to prevent privacy vulnerabilities and achieve data security. Existing solutions to protect the data mainly rely on cryptographic techniques. However, these cryptographic techniques add computational overhead, in particular when the data is distributed among multiple Cloud Service Provider (CSP) servers and more precisely Key Management Servers (KMS). File Assured Deletion (FADE) is a promising solution for addressing this issue. FADE achieves assured deletion of files by making them unrecoverable to anybody, including those who manage the cloud storage. The system is built by encrypting all data files before outsourcing, and then using a trusted party to outsource the cryptographic keys. But, this methodology remains weak since its security relies completely on the security of the key manager. In this paper, we propose a new scheme that aims to improve the security of FADE by using the Trusted Platform Module (TPM) and the Encrypted File System (EFS). A prototype implementation of the proposed scheme shows unique results, it provides a value-added security layer compared to FADE with a less overhead computational time.

Keywords: *Cloud Computing, FADE, TPM, VANISH, SSP, Ephemeriser. Cloud Storage, Secure Deletion, Confidentiality, Reliability, Integrity, Trusted Storage.*

1. INTRODUCTION

Cloud Computing is the on-demand delivery of compute power, applications, database, storage and other IT resources using the Internet. Cloud storage refers, commonly, to online space that we can use to store our data, as well as to keep a backup of our files. The most typical examples of cloud storage environments are Dropbox[1], Google Drive, Microsoft OneDrive and Amazon Drive. These services are proposed by Cloud Service Providers (CSPs) that offer obvious advantages, such as cost, accessibility, recovery and syncing.

However, clients wonder if these services are secure as the traditional ones based on in-house solutions. The concern is about the three key security objectives of any information system: confidentiality, integrity and availability, also known as the CIA triad [2,3]. These properties ensure that client's data is always secure and cannot be modified by unauthorized users and the data is always available at the latest versions when being retrieved by the user [4]. In general, the public perception is that, when some data is deleted, it no longer exists. Careful users take great precautions in protecting their data, by using strong passwords, two-factor authentication and encrypting their files. Unfortunately, they don't have any guaranty that there are no other copies of their deleted data. These concerns have prevented many consumers and enterprises from using widely the cloud despite its benefits [5]. The practical approach to render the data inaccessible on a cloud storage environment is to encrypt all the data before uploading them. The FADE system [6,32], is based on encrypting each message with a data key. It introduces a trusted third party to help managing the keys. This data key will be encrypted by an ephemeral public key witch is key managed by one or more trusted third parties, named the ephemerizers [7]. It should also be noted that risks arise when the third parties are compromised or down. In that case, certain operations will become not available. If we cannot trust the cloud storage, it must be the same for the key manager. To that end, instead of relying on centralized third parties to manage the keys, our proposed solution, design a decentralized approach, by using FADE system in conjunction with TPM and EFS, to protect data privacy.

The remainder of this paper is organized as follows: Section 2 presents the true meaning of a trusted cloud storage environment. Section 3 presents how FADE system provides a policy-based file assured deletion. In section 4, we discuss the limitations of FADE. In section 5, we review some related works on protecting outsourced data storage. Section 6, provides a brief description of our novel approach FADE-TPM-EFS to ensure privacy of deleted data. Finally, we come to end with our conclusions.

2. TRUSTED CLOUD STORAGE ENVIRONMENT

The main task of a "Trusted Cloud Storage Environment" is not only storing the data as well as it needs confidential storing and also integrity of the data would be maintained.

To achieve confidentiality and integrity of the data, cryptographic techniques can be used to encrypt data. For example, we can use the Encrypted File System (EFS) [8,9] to encrypt the client's data within the cloud. It is used to encrypt the user's data, manage and create keys which are used for data encryption and decryption [10].

EFS meant for encrypting stored files. Encryption procedures are transparent to the user and occurs at the file system level not at the application level. Diagram below illustrates the flow of the encryption process using EFS:

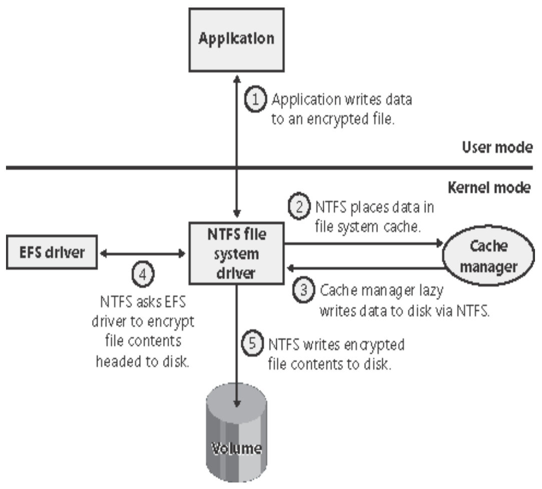

Application
1 Application writes data to an encrypted file.
User mode
Kernel mode
2 NTFS places data in file system cache.
EFS driver
NTFS file system driver
Cache manager
4 NTFS asks EFS driver to encrypt file contents headed to disk.
3 Cache manager lazy writes data to disk via NTFS.
5 NTFS writes encrypted file contents to disk.
Volume

Figure 1: Flow of EFS in an Encrypting File System (Microsoft Windows)

3. FILE ASSURED DELETION

File Assured Deletion (FADE) was discussed in many research articles [6,11,12,13]. It consists of encrypting the file with a DK (Data Key) which in its turn encrypted by a CK (Control Key) that is maintained by a separate third party KM (Key Manager) and when the predefined period expire the Key Manager remove the Control Key. This design was later prototyped in Vanish.

In order to make the deletion operation more flexible, File Assured Deletion system, combines several atomic boolean combination. The data owner will get the decryption key, if and only if his attributes satisfy the policy of the respective file. We can define policies over attributes using disjunctions, conjunctions and (k,n) threshold gates. As an example, in the figure below, the decryption key for the File X will be deleted if the date is the beginning of the year 2020 and when the applicant is an employee or a trainee in the Subsidiary A or B. The policy based deletion follows the same logic of Attribute Based Encryption (ABE) [14] where owner can get data only if several attributes are satisfied.

Figure 2: Example of Policy Based Deletion Scenario

3.1 Upload Scenario

- For each policy, Pi the Key Manager generates large RSA prime number pi and qi.
- Calculate

$$Pi \times qi = ni , \qquad (1)$$

- Then the Key Manager choose RSA Private/Public pair control key (ei,di)/(ni,ei)
- Key Manager sends its public key (ni,ei)to client
- Data owner generates a data key K and Si (Secret Key) of the policy
- Client sends to cloud Enc{K}Si – Siei Enc{F}kand drop Kand Si

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

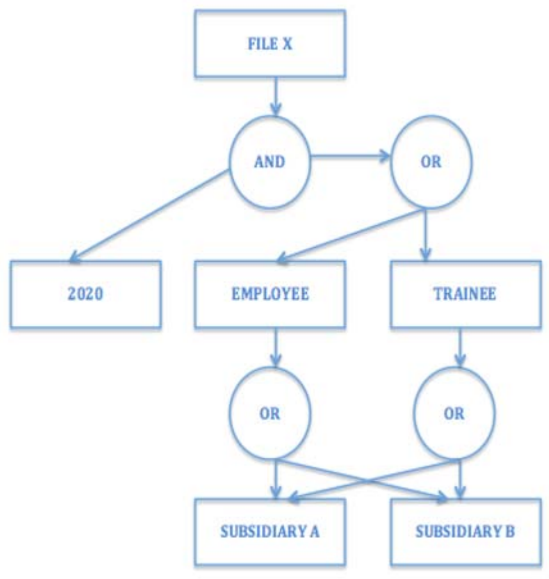
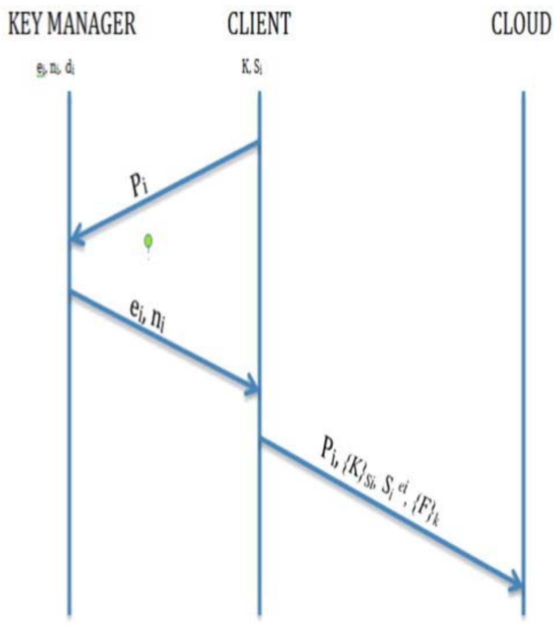


Figure 3: File Upload Architecture

3.2 Download Scenario

The data owner fetches Enc{K}Si ,Sieiand Enc{F}k from the storage cloud. Then the data owner generates a secret random number R, computes Rei, and sends Siei.Rei=(SiR)eito the key manager to request for decryption. The key manager then computes and returns ((SiR)ei)di = SiR to the data owner. The data owner can now remove and obtain Si, and decrypt Enc{K}Siand hence recovers Enc{F}k.



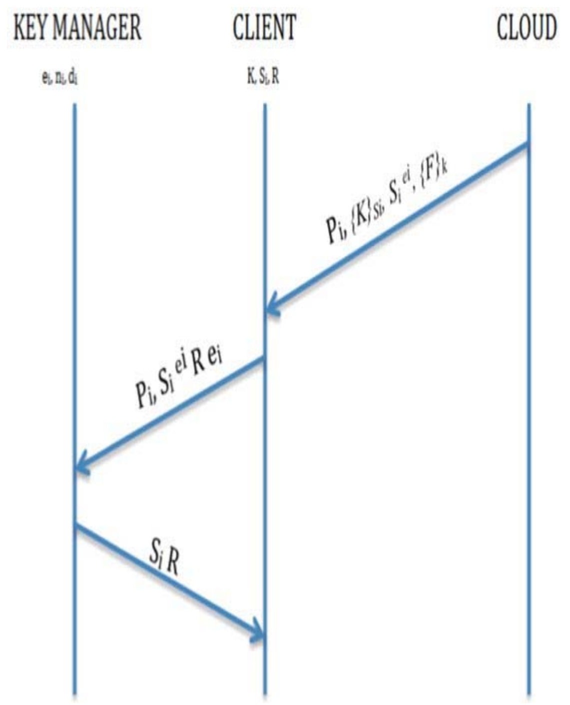


Figure 4: File Download Architecture

4. FADE'S LIMITATIONS

In the design of FADE, the encrypted files remain on the untrusted cloud storage and encryption keys are maintained by, a trusted key manager, which may be the subject of some side channel leakage [15,16]. Ranjan and Kumar [17] have shown, in their network security study of FADE, that some sensitive informations (policy, public and private key) can be leaked by sniffing the network flow between file owner and KM. Habib, Khanam and Palit [18] stated that FADEs design has a complex system architecture for storing keys at the KM, and this can lead to a leak of cryptographic key due to authentication mechanism and a heavy key infrastructure.

Also, if the key manager colludes with cloud storage, then cloud storage can decrypt the files of the data owner.

To avoid these limitations, we propose to add an additional layer of encryption to the data owner. The idea is that the data owner first encrypts a file with a long-term secret key, then encrypts this key with another secret key generated by the TPM (example AES key [19]). The entire process is conducted without involving any key manager. The overhead cost of time for the proposed scheme time for upload and download is reduced.

5. RELATED WORKS

The most relevant example of time-based scheme is the Firefox plugin for Gmail, which is an application prototyped on Vanish [20] system. It ensures that all copies of specific data become unreachable after a particular time without any action on the part of a user. This challenge is meets through a novel integration of cryptographic techniques with global-scale, P2P [21], DHTs [22]. This experience also reveals some limitations of existing DHTs, and the authors aimed to release the current Vanish system to provide further valuable experience to inform future DHT designs for privacy applications.

On the other side, there is also FADE System, as cited above in detail. That briefly, encrypts the data before storing in the cloud storage, using a third party key manager to store the keys which involves a relatively complex and unsafe system architecture.

6. OUR CONTRIBUTION: FADE-TPM-EFS SYSTEM

Nowadays, the search for confidentiality and data integrity in could environments, is more and more accentuated. Caeser proposed an algorithm to encrypt messages (En(x)=(x+n)mod26). In 1976 Diffie and Hellman proposed their solution to communicate on secure channel without the need of exchanging a common secret key. In 1984 Shamir [23] proposed the idea of identity-based cryptosystems. Also TPM was conceived to secure hardware through integrated cryptographic keys.

As well, the use of EFS (Encrypted File Systems) in Cloud Storage side, allows our system to grow and shrink automatically as we add and remove data. They can grow to petabytes in size, distributing data across an unconstrained number of storage servers in multiple Availability Zones.

That way, parallel to our FADE-TPM system that uses TPM in client side, for cryptographic operations, EFS in cloud storage side will bring us a new layer of security by encrypting encrypted data in the cloud storage.

6.1 TPM

A TPM [24] is a microchip designed to provide basic security-related functions, primarily involving encryption keys. The TPM is usually installed on the mother- board of a computer or laptop, and communicates with the rest of the system using a hardware bus.

Computers that incorporate a TPM have the ability to create cryptographic keys and encrypt them so that they can be decrypted only by the TPM. This process, often called ”wrapping” or ”binding” a key, can help protect the key from disclosure. Each TPM has a root ”wrapping” key, called the Storage Root Key (SRK) [25], which is stored within the TPM itself. The private portion of a key created in a TPM is never exposed to any other component, software, process, or person.

TPMs should support preventing attackers from being able to find information on a compromised client that can be used to compromise another system for which the client or its user has access. The information on clients could include encryption or signing keys, password, and personal or proprietary information. The TPM is designed to protect sensitive information on PC clients as well as the servers and networks they may connect to, in addition, some private RSA [26] keys never leave the TPM, so it is impossible to obtain them directly by software means.

Keys and other sensitive information may be stored outside the TPM. For data stored outside the TPM, the protection of the sensitive information is only as strong as the encryption algorithm by which it is protected. The TPM cannot increase the strength of an algorithm with respect to algorithmic attacks. For example, if a large file is encrypted with DES and the DES Key is encrypted with a 2048-bit RSA Key and stored in the TPM, the encrypted file is still subject to attacks on the DES encryption [27], which should be much easier than attacking the 2048-bit RSA Key. Here under the main cryptographic features that must be implemented in all TPMs:

- Random number generation (RNG)
- Asymmetric Key (RSA) and nonce generation
- Asymmetric encryption/decryption (RSA)
- Signing (RSA)
- Hashing (SHA-1)
- Keyed-Hash Message Authentication Code (HMAC)

There are two versions of trusted platform module:

- TPM 1.2
- TPM

Both TPM1.2 and TPM 2 offers same uses and functionality but only the com- ponents are different. TPM1.2 uses cryptographic algorithms like RSA, SHA1, and HMAC.
A TPM can take one of the following states:
- Without owner and disabled

- Without owner and activated
- Owner but disabled
- Owner and activated

The TPM must be enabled and have an owner to be used for securing your com- puter. To do this, the TPM must be initialized. During initialization, the TPM creates new root keys used by the TPM.

Computers manufactured to meet the requirements of this version of Windows include pre-boot BIOS functionality that makes it easy TPM computer to boot via the TPM initialization wizard. Normally, initialization of the TPM requires physical access to the computer to enable the module. This requirement helps protect the computer against malware threats able to initialize a TPM.

6.2 EFS

As we use Amazon as a cloud storage, we can easily create an encrypted file system so all our data and metadata is encrypted at rest using an industry-standard AES-256 encryption algorithm. That choice was made based on the study presented in [28] because AES-256 needs more level effort to be discovered

Encryption and decryption is handled automatically and transparently, so we don't have to modify our applications.

For Managing Keys, Amazon EFS is integrated with AWS KMS, which manages the encryption keys. AWS KMS also supports encryption by other AWS services such as Amazon Elastic Block Store (Amazon EBS), Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), Amazon Aurora, Amazon WorkMail, Amazon Redshift, Amazon WorkSpaces, etc.

It should be noted that, to create an encrypted file system we can use the AWS Management Console and the AWS CLI. For the later case, In the CreateFileSystem operation, the --encrypted parameter is a Boolean and is required for creating encrypted file systems. The --kms-keyid is required only when we use a customer-managed CMK and we include the key's alias or ARN:

```
$ aws efs create-file-system \
  --creation-token $(uuidgen) \
  --performance-mode generalPurpose \
  --encrypted \
  --kms-key-id user/customer-managedCMKalias
```

6.3.1 Proposed Design

In our prototype, we based our work on the java development language. As for the physical environment, it was a computer with i5-2500 CPU, 3,30 GHz (4 CPUs) and 16 Go of RAM. And the version of the TPM used on that computer is 1.2, with the IFX manufacturer.

Our application test was developed on java. So, we used JSR 321 [29] as a trusted computing API for java. It makes us able to develop a trusted computing API for java providing comparable functionality the TSS offers to the C world. We have installed the jTSS Core Services as a system service to enable our java applications to access the TPM.

About the cloud storage, we used Amazon S3 [30], and the AWS SDK [31] for Java for all the upload/download operations. As well, for the Encrypted File System, we used the Amazon EFS (Elastic File System).

In the experiments, we evaluate the system with an individual file of different sizes : 1KB, 10KB, 100KB, 1MB and 10MB. Diagram below represents, in a simplified way, our global approach:

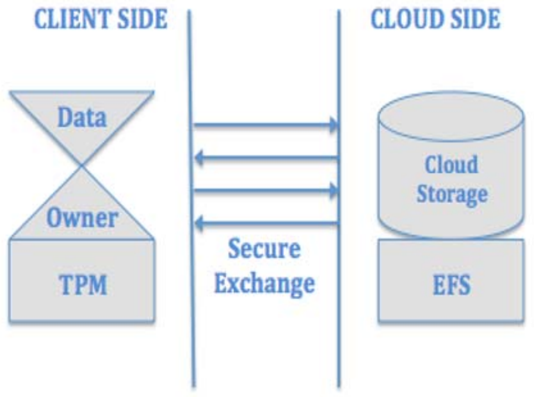


Figure 5: Proposed Diagram

6.3.1 FADE-TPM-EFS's Upload Scenario

Here we diagram the scenario of the file upload architecture.

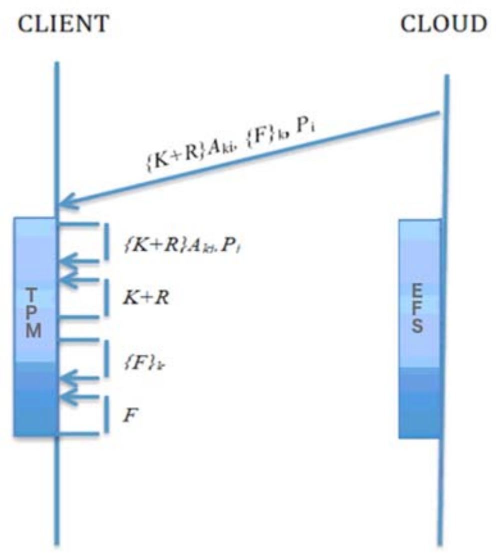


Figure 6: File Upload Architecture

6.3.2 FADE-TPM-EFS's Download Scenario

Here we diagram the scenario of the file download architecture.

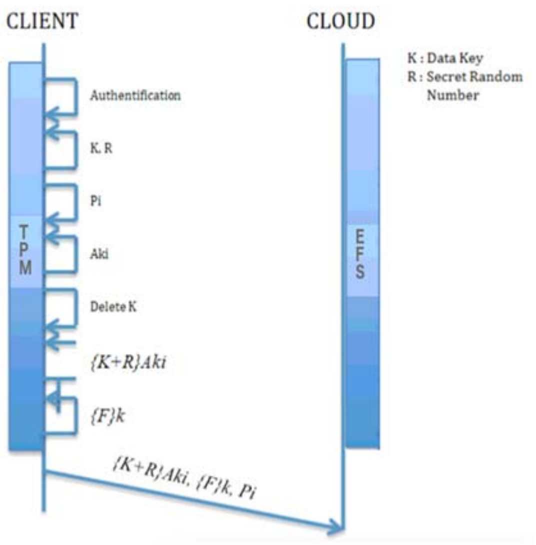


Figure 7: File Download Architecture

6.3.3 Results of time performance of FADE-TPM-EFS

Here we schematize and give figures in relation to the performance of upload and download operations.

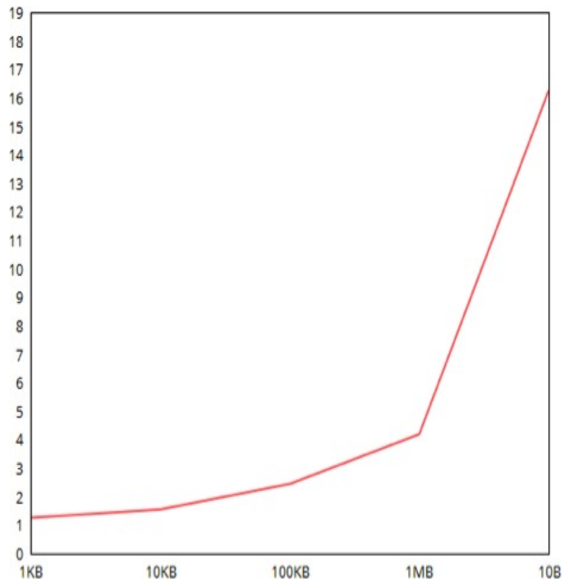


Figure 8: Proposed Design Upload Scenario

Table 1: Performance of Upload Operations.

File Size	Total Running Time (s)	Data Transmission (s)	AES+HMAC (s)	Key Management (TPM+EFS)
1KB	1.262	1.261	0.000	0.001
10KB	1.554	1.552	0.001	0.001
100KB	2.453	2.449	0.002	0.002
1MB	4.196	4.173	0.022	0.001
10MB	16.276	16.058	0.218	0.001

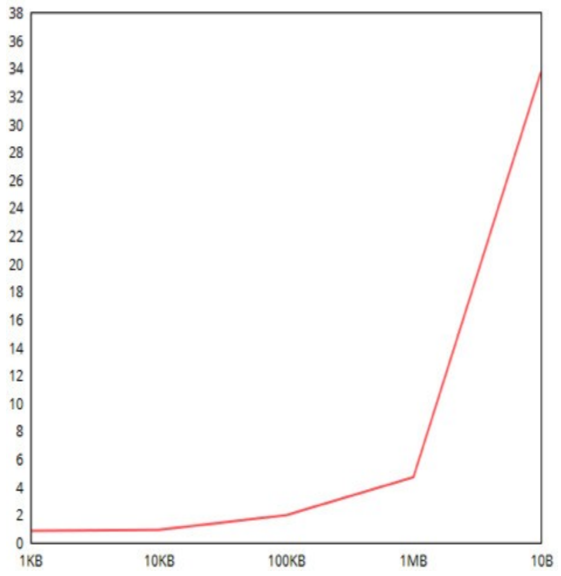


Figure 9: Proposed Design Download Scenario

Table 2: Performance of Download Operations.

File Size	Total Running Time (s)	Data Transmission (s)	AES+HMAC (s)	Key Management (TPM+EFS)
1KB	0.841	0.840	0.000	0.001
10KB	0.910	0.909	0.000	0.001
100KB	1.968	1.964	0.002	0.002
1MB	4.696	4.677	0.017	0.002
10MB	33.745	33.577	0.166	0.002

In the experiments, we evaluated our system when it operates on an individual file of different sizes, and we measured the time performance, by dividing the running time of each measurement into three components:

- Data transmission time, between the data owner and the cloud storage in upload/download process.
- AES and HMAC time, used for performing AES and HMAC on the file.
- Key Management time, for the data owner to coordinate with the Key Manager on operating the cryptographic keys.

Thereafter, we averaged each of our measurement results over dozen different trials. We measured the running time of file upload/download operations for different file sizes. We then compared the results between the three systems: FADE, FADE-TPM and FADE-TPM-EFS.

We should notice that the data transmission is divided into two components: the file component, which measures the transmission time for the file body and the file metadata, and the policy component, which measures the transmission time for the policy metadata.

Based on these results and in accordance with those calculated in the FADE [6], and FADE-TPM [24]; We note that the time of key management is almost the same with FADE in upload operation, but largely low in download. But basically, the time is negligible of both operations for our design, regardless of file size. This is almost logical, since our system is based on a client-side local TPM, and cloud storage-side local EFS, without any interaction with an external key generator.

As well, with the EFS, encryption has minimal effect on I/O latency and throughput. Encryption and decryption are transparent to users, applications, and services. All data and metadata is encrypted by Amazon EFS on our behalf before it is written to disk and is decrypted before it is read by data owner. We don't need to change data owner tools, applications, or services to access an encrypted file system.

7. DISCUSSION

The results are almost the same as compared to our previous work (FADE-TPM [33]). We measured the overhead cost time of our design for download/upload operations on files of different sizes. The measurement concerns the cryptographic operations using TPM and EFS. The experiment has proved that the overhead cost time of our design does not imply a remarkable overload time, and that the transfer of the plains files remains a dominant factor. Also, the design enables to reinforce the security with a zero-time cost. Our model leverages the burden of key management infrastructure and reinforces the confidentiality of the system. By using the TPM and EFS we add a new security layer with a zero overhead cost time. Also, the use of TPM and EFS enables more trust in the cloud storage service and resists to software attacks.

8. CONCLUSION AND FUTURE WORK

Cloud computing has become very promising paradigm. But at the same time several security problems can arise and await for means of neutralizing them. To this day, the state of the art presents several works whose aim is to ensure one property from the CIA triad. It's the data confidentiality. Most of these works relies on Key Managers to outsource the cryptographic keys. Also, it causes a large processing time, in proportion to file sizes. We already proposed FADE-TPM system that uses TPM in client side, for cryptographic operations, and now, the addition of the EFS module will bring a new layer of security by encrypting encrypted data in the cloud storage. Our final proposed scheme shows unique results: it provides a value-added security layer compared to previous works with a less overhead computational time.

Therefore, in this paper, we proposed our novel approach called FADE-TPM-EFS. It is a new design model for FADE, that envolve the consumer and the cloud storage in encryption process. It consists of using the client-side TPM for encryption operations, and the cloud storage-side EFS, instead of using a key manager that may not be fully trusted. This system has proven to be efficient and secure whatever the operation is upload or download and whatever the size of the file, without affecting the overhead performance. 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 results.

Our research can be extended in several directions. First, we are going to evaluate the performance of our design when multiple policies are associated with a file. Second, further study should be conducted to propose security modules for the customer, since in spite of that our system is performing, if a hacker spies the client or his identity was usurped, it could be critical.

In addition, we manage to combine our approach with Identity Based Encryption scheme (IBE) [34] in order to simplify certificate management. This scheme is based on the use of a pairing between elements of two cryptographic groups to a third group with a mapping to construct or analyze cryptographic systems.

REFRENCES:

[1] L.K Ronald and R.D. Vines, "Cloud Security: A comprehensive Guide to Secure Cloud Computing", Wiley Publishing, Hoboken, 2010.

[2] "Dropbox for .NET Developers", https://www.dropbox.com/developers/documentation/dotnet.

[3] 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.

[4] Jacob R. Lorch, David Molnar, Helen J. Wang, and Li Zhuang, "Enabling Security in Cloud Storage SLAs with CloudProof", Microsoft Research.

[5] http://technet.microsoft.com/en-us/library/cc700811.aspx

[6] Tang Y., Lee P, John C.S. Lui., Perlman R., "FADE: Secure Overlay Cloud Storage with File Assured Deletion", 2010.

[7] Tang Q., "From Ephemerizer to Timed-Ephemerizer: Achieve Assured Lifecycle Enforcement for Sensitive Data", 2005.

[8] Sharma S., Chugh A., "Survey Paper on Cloud Storage Security", International Journal of Innovative Research in Computer and Communication Engineering, vol. 1, Issue. 2, 2013, ISSN: 2320-9801.

[9] "Encrypting File Data withcAmazon Elastic File System. Encryption of Data at Rest and in Transit", Amazon Web Services, Inc. or its affiliates, 2018.

[10] http://www.trustedcomputinggroup.org.

[11] Garfinkel, S. L. and Shelat, A., "Remembrance of data passed: a study of disk sanitization practices", vol. 1, 2003, pp. 17-27.

[12] Philip McMichael, "The dangers of dead data, Computer Fraud Security", vol. 2014 Issue 3, 2014, p. 9-12. doi: 10.1016/S1361-3723(14)70470-1.

[13] Wasim Ahmad Bhat and Syed Mohammad Khurshaid Quadri, "After- deletion data recovery: myths and solutions, Computer Fraud Security", vol. 2012 Issue 4, 2012, p. 17-20, doi: 10.1016/S1361- 3723(12)70032-5.

[14] Bo Qin, Hua Deng, Qianhong Wu, Josep Domingo- Ferrer and David Nac- cache and Yunya Zhou, "Flexible attribute-based encryption applicable to secure e-healthcare records", International Journal of Information Security, Vol. 14, 2015, pp. 499-511, doi: 10.1007/s10207-014-0272-7.

[15] Walter Colin D, "Longer Randomly Blinded RSA Keys May Be Weaker Than Shorter Ones, Information Security Applications", 8th Interna- tional Workshop WISA 2007, Jeju Island Ko- rea, 2007, pp. 303-316.

[16] Igarramen Z., Hedabou M., "Protecting Co-resident VMs from Side-Channel Attack in Cloud Environment: SAFEPERIMETER System", Springer International Publishing Switzerland 2016, Lecture Notes in Electrical Engineering, 2016, p. 541-542, DOI: 10.1007/978-3-319-30298-0_55

[17] A. K. Ranjan and V. Kumar and M. Hussain, "Security Analysis of Cloud Storage with Access Control and File Assured Deletion (FADE)", Second International Conference on Ad- vances in Computing and Communication Engineering (ICACCE) Dehradun, 2015, pp. 453-458, doi: 10.1109/ICACCE.2015.10.

[18] A. B. Habib, T. Khanam and R. Palit, "Simplified File Assured Deletion (SFADE) - A user friendly overlay approach for data security in cloud storage system", International Conference on Advances in Computing Communications and Informatics (ICACCI), 2013, pp. 1640-

1644, doi: 10.1109/ICACCI.2013.663742.

[19] Kak A. AES, "The Advanced Encryption Standard", 2016.

[20] G eambasu R., Kohno T., A.Levy A., M.Levy H., "Vanish: Increasing Data Privacy with Self- Destruction Data", 2009.

[21] S.Alvi A., M.Bochare D., "Distributed Hash Table In Peer-To-Peer (P2P) System", 2014.

[22] Zhang H., Wen Y., Xie H., Yu N., "A Survey on Distributed Hash Table (DHT): Theory, Platforms, and Applications", 2013.

[23] Shamir Adi, "Identity-Based Cryptosystems and Signature Schemes, Advances in Cryptology", Proceedings of CRYPTO, pp.84, 47-53 , doi: 10.1007/3-540-39568-7_5.

[24] Dorwin D., "Cryptographic Features of the Trusted Platform Module", 2006, pp. 1- 6 21.

[25] Tomlinson A., "Smart Cards, Tokens, Security and Applications", Chapter: Introduction to the TPM, 2008, pp. 161-166.

[26] Evans M., "RSA Encryption. The university of Melbourne", on behalf of the Australian Mathematical Sciences Institute (AMSI), 2013.

[27] Singh G., Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security", 2013.

[28] 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).

[29] "JSR 321 Trusted Computing API for Java", 2009.

[30] "The Business Value of AWS", 2015.

[31] AWS Encryption SDK Developer Guide (2017).

[32] Bentajer A., Ennama F., Hedabou M. and Elfezazi S.: "A User Friendly and Improved Design for Secure Deletion in Cloud Storage", Journal of Theoretical and Applied Information Technology, vol. 95, No. 6, 2017, p. 1384-1385, ISSN: 1817-3195.

[33] Igarramen Z. and Hedabou M., "FADE-TPM: Novel Approach of File Assured Deletion Based on Trusted Platform Module", Lecture Notes in Networks and Systems, 2018, doi: DOI: 10.1109/CloudTech.2017.8284727

[34] Bentajer A., Hedabou M., Abouelmehdi K., Igarramen Z. and El fezazi S., "An IBE-based design for assured deletion in cloud storage", Cryptologia, Taylor & Francis Group, 2019, doi: 10.1080/01611194.2018.1549123