TPM Based Schema for Reinforcing Security in IBE's Key Manager

Zakaria Igarramen[1](✉), Ahmed Bentajer[2], and Mustapha Hedabou[1,3]

[1] Cadi Ayyad University, ENSA of Safi, Safi, Morocco
z.igarramen@gmail.com, m.hedabou@gmail.com
[2] SIGL Laboratory, Abdelmalek Essaadi University, ENSA of Tetouan, Tetouan, Morocco
a.bentajer@gmail.com
[3] University Mohammed VI Polytechnique, Benguerir, Morocco

Abstract. Confidentiality and secure deletion are among the most important security concerns about public cloud storage. Client-side cryptography is widely recognized as a robust approach for addressing these issues. However, managing keys related to encrypted files and their policies is a very challenging task for cloud storage's users. Studies have shown that even the most achieved solution, namely FADE design, suffers from a leak of cryptographic key due to authentication mechanism and a heavy key management infrastructure. In this paper, we propose a new design that brings the use of Identity Based Encryption (IBE) and Trusted Platform Module (TPM) together. The use of the IBE leads to ease the key management since the security of the entire system is mainly based only on one master key that will be in turn stored and managed securely trough the TPM facilities. The analysis of the prototype implementation of our design have demonstrated that it offers a significant value-added of security with a negligible overhead amount of computing time.

Keywords: TPM · IBE · Security · Confidentiality · Trusted computing

1 Introduction

In the last decade, the use of cloud-based services gained an expanding interest. For the general public, the cloud is materialized using storage solutions for storing, sharing and massive computation of data. Outsourcing sensitive data to a Cloud Service Provider (CSP) removes the burden of maintaining internal infrastructures. Customers only need to pay the allocated resources.

Therefore, outsourcing sensitive data storage to a third party manager raises many security issues [1–3]. It is commonly agreed that client-side cryptography is a good alternative to reduce the risk of data confidentiality leakage [1,4,10,14]. However, it is the client who must securely manage the security decryption key.

Therefore, when data need to be shared with a group of users, maintaining the security of decryption keys become more difficult even if a key manager system (KMS) is set up [12].

Using ID-based cryptography (IBE) leverage this issue since the Private Key Generator (PKG) compute the corresponding private key (d_{ID}) each time the user needs it for decryption process [1,8]. As, the PKG knows all privates keys it may allow a global key escrow if it is compromised. Thus, this requires more trust to the PKG since the confidentiality of data is based on the secret of the master secret (*msk*) key used to compute the d_{ID}.

In this paper, we describe a model based on the use of Trusted Platform Module (TPM) for improving the security of PKG and its *msk*. Our model relies on a decentralized PKG used in conjunction with TPM for managing the security of the *msk*.

The originality of our model is double benefits. First, it increases the confidentiality of the *msk*, since it remains encrypted when it is not used by the PKG. Second, processes that need to secure secrets, such as digital signing, can be made more secure with a TPM. For example, if at boot time it is determined that the server is not trustworthy because of unexpected changes in configuration, access to the *msk* can be blocked until the issue is remedied. Besides the prototype can work atop any cloud storage solution since no engineering changes are needed on the CSP side.

The reminder of this paper is as follow. The second section presents a background of key management security issues in cloud storage and IBE. The third section presents TPM. In the fourth section, we present our design, which will be discussed in the fifth section. Finally, we come-up with our conclusions and assumptions.

2 Background

Cloud storage confidentiality issues have been discussed in many research articles [1,3,4,14], the problem is that when customers outsource their data they lose control over them. Many cryptographic solutions have been proposed to bypass this issue, most of them are client-side encryption based [1,3,4]. The aim is to encrypt data before outsourcing them and delegate the cryptographic keys to a trusted third party. Perlman [14] proposed the Ephemerizer that securely delete data after an expiration time through ephemeral key. Data are encrypted by an ephemeral key managed by the Ephemerizer which destroy it at time expiration, as a result data remain unreadable. Tang et al. [10] extend the Ephemerizer solution, in FADE each key corresponds to a combination, using logical OR/AND, of atomic policies that can be revoked.

However, FADE focused more on secure deletion operations on cloud and the PKG's security has been almost not discussed. Ranjan et al. [5] proved that the PKG can present key escrow. A malicious user may intercept a policy P_i and its corresponding private key.

2.1 ID-Based Encryption

IBE is a public key cryptosystem that does not rely on a Certificate Authority (CA). The idea was first proposed by Adi Shamir in 1984 [6]. The system enables any pair of users to communicate securely with no need for certificates and CA deployment. This means that each entity, which has access to public parameters of the system, may generate a public key generated from publicly identifiable information denoted (ID) in order to simplify cryptographic key management. To get the d_{ID}, the user must authenticates him self to PKG through his ID. In 2001, Boneh and Franklin [8] proposed a full scheme that achieve IBE based on bi-linear function defined in elliptic curves such as Weil and Tate Pairing [7].

The IBE scheme as proposed by Boneh and Franklin uses four algorithm:

- **Setup:** Run once by the PKG to generate the whole system parameters (*msk*, the Master Public Key (*mpk*) and *params*). While *params* and *mpk* are made available for the public the *msk* is kept secret and used to compute d_{ID}
- **KeyGen:** Used by the PKG in order to compute the d_{ID}
- **Encrypt:** Used by the entity willing to encrypt data using its ID
- **Decrypt:** Used by the entity willing to decrypt data using the d_{ID}

Boneh and Franklin scheme is based on a bilinear pairing e on $\mathrm{G}_1, \mathrm{G}_\mathrm{T}$ and two hash function $\mathrm{H}_1 : 0, 1^* \rightarrow \mathrm{G}_1 \setminus \{\infty\}$ and $\mathrm{G}_\mathrm{T} \rightarrow \{0, 1\}^1$ where `l` is the bi-length of plain text. The PBC is a free portable C [9] library allowing the rapid prototyping of Pairing-Based Cryptosystems. It is designed to be the backbone of implementations of PBC providing elliptic curve generation, elliptic curve arithmetic and pairing computation.

Nowadays, IBE is widely implemented in different applications including emails, due to its simplified key management. A commercial implementation of IBE is published by Voltage Security to enable enterprises to send and receive secure communications.

2.2 Trsuted Platform Module

TPM is a cryptoprocessor promoted by the Trusted Computing Group (TCG) to promote Trusted Computing Platform (TCP). Computers that incorporate TPM can provide cryptographic-based function. TPM (Fig. 1) is a chip that contain a cryptographic co-processor, secure memory, I/O components and other components. It also implements a validated FIPS key size generation for computing digital signature key (512 to 2024 bits). The TPM chip becomes part of new released PC motherboard. The chip is able to store an Endorsement Key (EK) that identify the TPM (Thus device authentication). The chip is tamper resistant through its multiple physical security mechanism. Besides, the chip enables platform integrity, this is done during the boot process, the firmware and operating system components (integrity measurement) are measured and stored in the TPM $\rightarrow$ BIOS $\rightarrow$ BootLoader $\rightarrow$ OS $\rightarrow$ Application. The integrity measurements can be used as evidence for how a system started. Once we need to attest the platform, a trusted third party sends a token to the platform. The platform takes both the ML and token encrypted by the TPM's EK and send the encrypted information to the third party which decrypt it using EK's corresponding public key (thus ensuring authentication) and checks the integrity of data (ML and Token) to verify that the configuration it deems trusted.

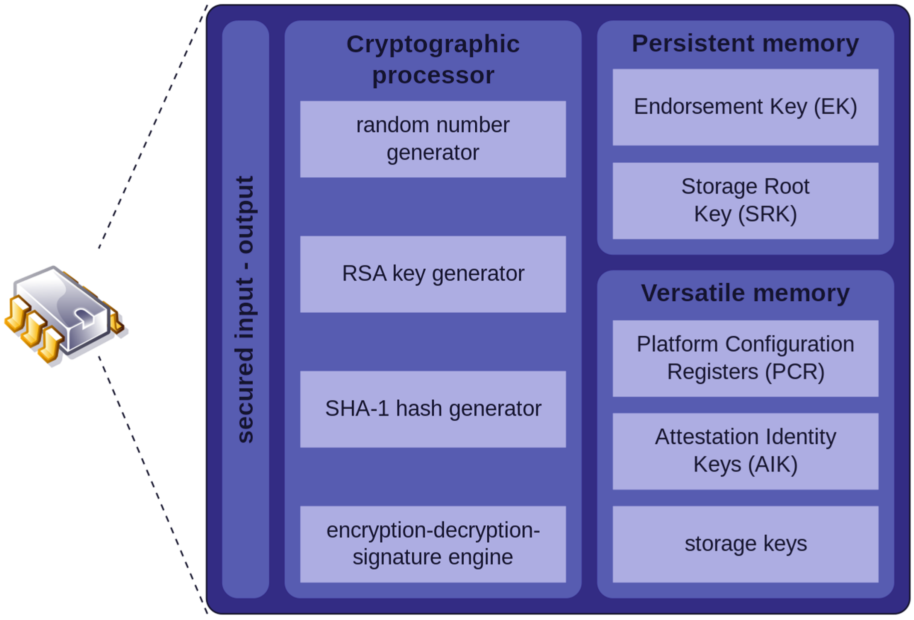

Fig. 1. Components of a Trusted Platform Module complying with the TPM version 1.2 standard

3 Proposed Design

In this section, we present our design for secure deletion and secure key management through IBE and TPM. To encrypt a file the user generates a data key (AES-256 CBC mode) and uses his ID as a public key to encrypt it. For the PKG, once the **Setup** algorithm executed, the *msk* is encrypted by the TPM and kept secret until a user ask for a d_{ID}. The fully IBE client-side cryptography is described in [1]. In this paper we focus more on PKG security and its interaction with TPM (Fig. 3).

We note that we are still developing the code based on *TrouSerS's C API* [11]. In order to test our design, we have used an embedded Linux platform with TPM chip. We wrote small Bash files in order to enable the PKG interact with the TPM.

3.1 Architectural Solution

In this section, we present our secure deletion model based on IBE. Each file is associated to the ID of the user. The file is encrypted with a data key

(256-bit AES key), and the data key is further encrypted by IBE mechanism. The components of our design are:

- **Cloud user:** Generates the data key, performs cryptographic operations on data and the data key.
- **Cloud storage service:** A non-trusted third party storage provider
- **PKG:** A server with TPM integrated. It generates the d_{ID} and performs cryptographic operation through TPM for the *msk*

In the **Upload** (Fig. 2) process, the user does not need to interact with the PKG, he executes the **POST** function which generates the data key K and IBE public key D_{id}. **At a second stage, the function encrypts the file F with K ($Enc\{F\}_K$) then the K with D_{id} ($Enc\{F\}D_{id}$) and upload data to the cloud storage.**

For the **Download** process (Fig. 3), the user get the data from the cloud storage, authenticates himself at the PKG and asks for the d_{ID}. At this stage, the PKG will ask the TPM to unseal the *msk* in order to compute the d_{ID} then it will be resealed.

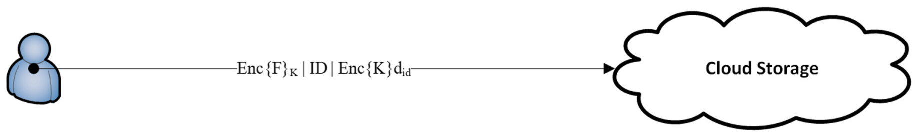

Fig. 2. File upload process

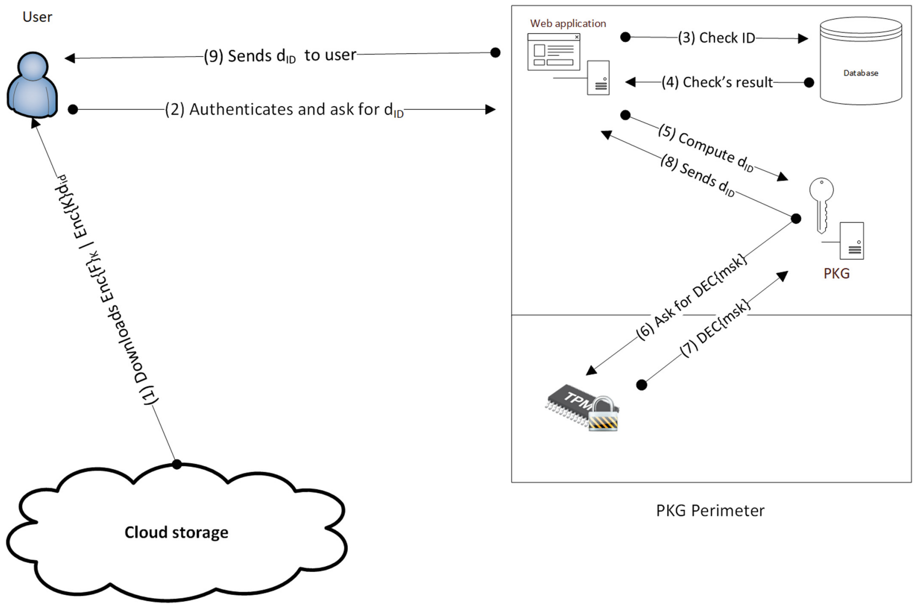

Fig. 3. File download process

3.2 Securing the Msk

In upload process, the user does not need to interact with the PKG (Fig. 2). In Download process, the user authenticates him self at the PKG which verify the authorization and grant access to a computation of d_{ID}. At this stage, PKG asks the TPM to unseal the *msk* then it will be resealed once the computation done:

```
# tpm_unsealdata -i msk.enc -o msk -z
```

Experiments has been conducted to measure the running time of plain upload/download and cryptographic operations between the TPM and the PKG. We dropped the overhead cost time of cryptographic operations between TPM and PKG since it was almost zero. The only overhead cost time is related to client-side cryptography using IBE mechanism.

4 Discussion

First, we talk about prerequisites that we have used to test our design. The **tcsd** daemon from the *trouSerS* API must be running and *tpm-tools* installed in Linux distribution. We used a well-known password (20 bytes of 0s) for owner and Storage Root Key (SRK) in order to facilitate the test environment (so that it can be non-interactive), for this purpose the command `tpm_takeownership -y -z` is used. However, other password should be used in production environment. For IBE client-side cryptography we used the same design as described in [1].

Tests are conducted using a computer with an i5 1.6 GHZ processor and 16 GB RAM. Experiments are performed on files of the following sizes: 10 KB, 1 MB, 5MB, 10 MB and 15 MB.

Once the Setup algorithm executed and global parameters generated the *msk* is sealed by TPM:

```
# tpm_sealdata -i msk -o msk.enc -z
```

We measured the overhead cost time of our design for download (Table 1) operations on files of different sizes (Upload operation measurements have been dropped since they do not require interactions with TPM). The measurements concerns the cryptographic operations using IBE design and operations for unsealing the *msk*. For sealing operation we dropped the measurements since it does not affect our system.

The experiment have proved that the overhead cost time of our IBE design with TPM does not imply a remarkable overload time, and that the transfer of the plains files remains a dominant factor. Also, the design enable to reinforce the PKG security with a zero-time cost. Our model leverage the burden of key management infrastructure and reinforce the confidentiality of the IBE PKG. By using the TPM we add a new security layer with a zero overhead cost time. Also, the use of TPM enables more trust in the service of PKG and resists to software attacks.

Table 1. Proposed design download operation

File size	Plain download	Design download	Decryption	Key generation	TPM unseal	Overhead
100 KB	1.339	1.3929	0.0454	0.0085	≈0.0	4.03%
1 MB	2.417	2.4728	0.0473	0.0085	≈0.0	2.31%
5 MB	7.334	7.4504	0.1079	0.0085	≈0.0	1.59%
10 MB	13.152	13.3608	0.2003	0.0085	≈0.0	1.59%
15 MB	21.089	21.3842	0.2867	0.0085	≈0.0	1.40%

Security: Our model makes it possible to keep the confidentiality of the outsourced data, since they are encrypted on the client side. So, it is difficult for malicious users or the storage service provider to have access to these data even in case of a data leak or system disaster. The nature of hardware-based cryptography ensures that the information stored in hardware is better protected from external software attacks [13].

Besides, our model addresses the problem tied to Key Management by using the TPM to seal the *msk*, using hardware cryptography the performance is increased because the host system does not support encryption operations. Also, it protects against the most common risks, such as startup attacks, malicious code and force attacks. Unlike software security that shares host resources and it is vulnerable to force attack since malicious users can access the memory of the host and reset the counter of decryption attempts.

5 Conclusion and Perspectives

In this paper we surveyed the security concerns of PKG in IBE. We proposed a design for reinforcing the security of PKG. Compared to traditional PKI, the use of IBE does not imply a repository of public keys and simplify the key managements and there is no need to request certificate of the public key every time for an establishment of connection. Our solution takes benefits from the use of TPM that protect the *msk* from corruption and unauthorized access by sealing it when it is not used by the PKG, also it enables the authentication capabilities inherent to TPM, so only the PKG is able to get the unsealed *msk*.

As a perspective, we aim to continue the development of our design with *TrouSerS's C API*. Also we aim to enable the integrity check function provided by the TPM through setting up a model for Root of Trust Reporting (RTR).

References

1. Bentajer, A., Hedabou, M., Abouelmehdi, K., Igarramen, Z., El Fezazi, S.: An IBE-based design for assured deletion in cloud storage. Cryptologia 1–12 (2019). https://doi.org/10.1080/01611194.2018.1549123
2. Wheeler, A., Winburn, M.: Privacy challenges. In: Wheeler, A., Winburn, M. (eds.) (Cloud Storage Security) Computer Science Reviews and Trends 2015, Chap. 3, pp. 57–74. Elsevier, Boston (2015). https://doi.org/10.1016/B978-0-12-802930-5.00003-4

3. Sun, X.: Critical security issues in cloud computing: a survey. In: 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS), Omaha, pp. 216–221. IEEE (2018) https://doi.org/10.1109/BDS/HPSC/IDS18.2018.00053
4. Rajeswari, S., Kalaiselvi, R.: Survey of data and storage security in cloud computing. In: IEEE International Conference on Circuits and Systems (ICCS), Thiruvananthapuram, pp. 76–81. IEEE (2017). https://doi.org/10.1109/ICCS1.2017.8325966
5. Ranjan, A. K., Kumar, V., Hussain, M.: Security analysis of cloud storage with access control and file assured deletion (FADE). In: Second International Conference on Advances in Computing and Communication Engineering, Dehradun, pp. 453–458. IEEE (2015). https://doi.org/10.1109/ICACCE.2015.10
6. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
7. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography, 1st edn. Springer, New York (2004). https://doi.org/10.1007/b97644
8. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
9. PBC Library. https://crypto.stanford.edu/pbc/about.html. Accessed 2 Mar 2019
10. Tang, Y., Lee, P.P.C., Lui, J.C.S., Perlman, R.: FADE: secure overlay cloud storage with file assured deletion. In: Jajodia, S., Zhou, J. (eds.) SecureComm 2010. LNICST, vol. 50, pp. 380–397. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16161-2_22
11. The open-source TCG Software Stack. http://trousers.sourceforge.net/. Accessed 24 Feb 2019
12. Barker, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for Key Management, Part 2: Best Practices for Key Management Organization. NIST Special Publication. NIST (2005). https://doi.org/10.6028/NIST.SP.800-57p2
13. Regenscheid, A.: Platform Firmware Resiliency Guidelines. NIST Special Publication 800-193. NIST (2018). https://doi.org/10.6028/NIST.SP.800-193
14. Perlman, P.: The Ephemerizer: making data disappear. Technical report. Sun Microsystems, Inc., Mountain View (2005)