A Way of Key Management in Cloud Storage Based on Trusted Computing

Xin Yang, Qingni Shen, Yahui Yang, Sihan Qing

A Way Of Key Management In Cloud Storage Based On Trusted Computing

Xin Yang[1,2,3], Qingni Shen *[1,2,3], Yahui Yang [1], Sihan Qing[1,4]

1 School of Software and Microelectronics,
Peking University, Beijing, China
2 MoE Key Lab of Network and Software Assurance,
Peking University, Beijing, China
3 Network & Information Security Lab., Institute of Software,
Peking University, Beijing, China
4 Institute of Software, Chinese Academy of Sciences, Beijing 100086, China
E-MAIL: yangxin@pku.edu.cn, {qingnishen, yhyang, qsihan}@ss.pku.edu.cn

Abstract. Cloud security has gained increasingly emphasis in the research community, with much focus primary concentrated on how to secure the operation system and virtual machine on which cloud system runs on. We take an alternative perspective to consider the problem of building a secure cloud storage service on top of a public cloud infrastructure where the service provider is not completely trusted by the customer. So, it is necessary to put cipher text into the public cloud. We describe an architecture based on Trusted Platform Module and the client of cloud storage system to help manage the symmetric keys used for encrypting data in the public cloud and the asymmetric keys used for encrypting symmetric keys. The key management mechanism includes how to store keys, how to backup keys, and how to share keys. Based on the HDFS (Hadoop Distributed File System), we put a way of key management into practice, and survey the benefits that such an infrastructure will provide to cloud users and providers, and we also survey the time cost it will bring to us.

Keywords: cipher text; key management; symmetric keys; asymmetric keys; backup; public cloud

1 Introduction

With the development in networking technology and the increasing need for computing resources, many companies have been prompted to outsource their storage and computing needs. This new economic computing model is commonly regarded as cloud computing and includes various types of services[1] such as: infrastructure as a service (IaaS), where a customer makes use of a service provider's computing, storage or networking infrastructure; platform as a service (PaaS), where a customer

*Corresponding Author

leverages the provider's resources to run custom applications; and finally software as a service (SaaS), where customers use software that is run on the providers infrastructure.
Cloud can be commonly classified as either private or public. In a private cloud, the infrastructure is managed and owned by the company or individual which means accessing to data is under its control and is only granted to parties they trusts. However, in a public cloud the infrastructure is owned and managed by a cloud service provider, which means that customer's data is beyond of its control and therefore could be acquired by illegal parties [7, 10].

Storage services based on public clouds such as Microsoft's Azure storage service and Amazon's S3 provide customers with scalable and dynamic storage. By moving their data to the cloud customers can avoid the costs of building and maintaining a private storage infrastructure, opting instead to pay a service provider as a function of its needs. For most customers, this provides several benefits including availability (i.e. we cannot access data from anywhere) and reliability (i.e. we do not have to worry about backups) at a relatively low cost.

Although the benefits of using a public cloud infrastructure are apparent, it introduces significant security and privacy risks. In fact, it seems that the biggest difficulty is how to guarantee the confidentiality and the integrity of data [4, 10]. While, so far, consumers have been willing to trade privacy for the convenience of software services (e.g., for web-based email etc), this is not the case for enterprises and governments, and especially for military departments. Any customers do not want to store mission-critical data or private data like personally identifiable information, or medical and financial records [7, 8, 9]. So, unless the problem of confidentiality and integrity are addressed properly, many potential customers are reluctant to use the public cloud instead of the traditional computing facility.

As a customer, there are many reasons to consider that the public cloud is unsafe [8]. Importing our data into public cloud enables us to face at least the following threats.

First, as a result of the limit of our law and technology, while we can trust the cloud providers, they cannot guarantee if one of the operators of the cloud will steal our data directly, because they can have the capacity and the authority to get data easily.

Second, the traditional network attack. When we upload data into public cloud, it is totally possible that attackers can intercept or destroy our data.

Third, unlike data stored in our personal computer, the data stored in the cloud is not isolated from other people's data. As a result, the risk of being attacked of our data in the cloud is increasing.

Last but not least, the same to the traditional storage system, the public cloud is also faced the threat of attacker from all over the world.

To address the concerns stated above and increase the usage of cloud storage. We are suggesting designing *Private Data Protecting Service (PDPS)* based on the architecture of Trusted Computing and the client of public cloud. Such a service should aim to encrypt the data that will be uploaded into cloud and protect the keys used for encryption. More precisely, such a service should provide:

- Confidentiality: the encryption before data is uploaded to the public cloud and decryption service after data is downloaded from the cloud.
- Key management: be responsible for the protection the keys from various kinds of attack, at the same time, the relationship between data and keys will also be maintained in this part.
- Data sharing: customers can share data by ways sharing their keys safely which are related to the data.
- Key backup: it is possible that our client which stores keys will collapse suddenly. So there must be a mechanism for key backup, and more importantly the backup must not be used by illegal customer.

PDPS is built based on the client of public cloud, the coupling factor between the public cloud and *PDPS* is nearly zero, so we can retain the main benefits of a public cloud:

- Availability: customer's data is accessible from any machine with a TPM and at all times.
- Reliability: customer data is reliably backed up by backing up the keys.
- Data sharing: customers can share their data with trusted parties by sharing keys.

2 Background and Motivation

With the development of Trusted Computing Organization, TPM 1.2 they produced (Trusted Platform Module) serves a very important part to key storage, data signing, IMA and remote attestation.

2.1 Key Management and Cloud

As shown in [5], there are tremendously large amount of files in the cloud, if we encrypt them one by one, there must be the same amount of keys we should manage safely. If we encrypt thousands of files by the same key, it is very easy for attackers to get the key by analysis of the encrypted text [6]. One of techniques that TPM use for storing data and keys in a secure fashion is to use encryption to store data, and to use encryption and decryption to control access to that data. Although a TPM is somewhat similar to a smart card chip, one difference is that it is attached (through the PC) to a large amount of persistent storage. One of the design goals for secure storage is to take advantage of the persistent storage to store a large amount of private keys and symmetric keys [2]. As a result, we can store a large amount of encrypted data that is associated with the TPM. However, TPM does not help us to manage the map between the Keys and the data. So we should add the mapping service into the private data protecting service.

2.2 RNG

As stated in [2], in order to generate keys internally, it is necessary for the TPM to have an internal random number generator (RNG). Typically, instead of having a true random number generator (it is difficult to do), many TPMs have pseudo random number generators (PRNGs) that are periodically fed with entropy from timing measurements or other sources of entropy within the TPM itself. This entropy is then distilled into the current seed, so entropy is always increasing. The output of the PRNG or RNG is used to generate keys, nonce, and seeds in the PKCS#1 V2.0 blob creation. Hence, in the solution, the keys are provided by the TPM instead of customs. At the same time TPM provide a series of interfaces for legal customs to manage the keys in a simple way.

2.3 Key Structure

Figure 1 shows the basic setup of keys inside a TPM. In the figure, if Key 2 is directly below Key 1, then Key 2 is encrypted (Asymmetric Encryption) with the public key corresponding to Key 1. Thus, we see that User 1 migratable storage is wrapped with the platform migratable key, which is in turn wrapped with the SRK. User 1's non-migratable storage key is also wrapped with the SRK. Hence, the architecture of keys is a tree structure.

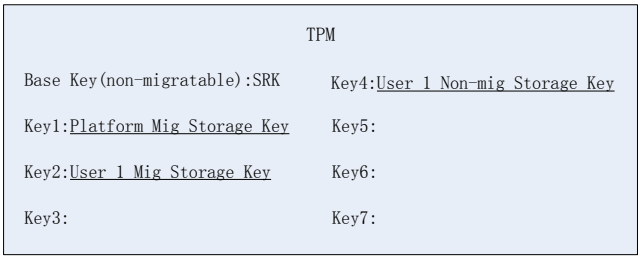

Figure 1: Basic key structure of TPM

2.4 Migratable Keys Versus Non-migratable Keys

Migratable keys are meant for two purposes. As shown in [3], the first is to provide a quick way to transfer keys from one platform to another in the event that a user wants to change his or her facilities, either because of upgrading or other reasons. The second is to provide the capability for more than one system to use a key. Keys can be valuable if they have associated certificates, or store a lot of data. For example, if a key is used for signing e-mail, a person might want to use it at home and at work or on multiple systems. A key that is used to store symmetric keys might be shared among a group, so only members of that group have access to the keys necessary to decrypt a shared file.

2.5 Motivation

First, using persistent storage appears to have a major problem. It is possible that the computer that stores key collapses or some other disasters happen, our solution needs to take into account disaster recovery. So we add *Backup Service* into the architecture. Second, as a customer in the cloud, it is totally possible that customers need to share data with some other ones safely. In other words, he should send related keys to others. In our solution, we provide *Key Sharing Service* to help customers to share their keys safely by constructing a backup structure, and the whole process for sharing keys and backing up keys is maintained by the structure. Third, as to data that has high confidentiality, customers do not wanted them to be shared, so in this solution, we provide a interface to enable customer to decide the confidential level of their data neatly.

3 Architecture of a private data protecting service

We now describe a possible architecture for PDPS, as shown in Figure 2, the architecture includes four sub services (also called functional modules): *data encryption service*, which processes data before it is sent to the cloud; *data decryption service*: which processes data after it is received from the cloud; *backup service*, which backs up keys in a safe way; *data sharing service*, which enables customers to share data among trusted parties. In the figure, the right side shows four basic modules, among these modules RNG (random number generator), the structure of key tree, and the operation of migration are maintained by the function of TPM. The structure of *key data mapping* is implemented outside TPM. As shown in Figure 2, the thick lines represent the key flow between modules, while the thin ones indicate the call of function between functional modules.

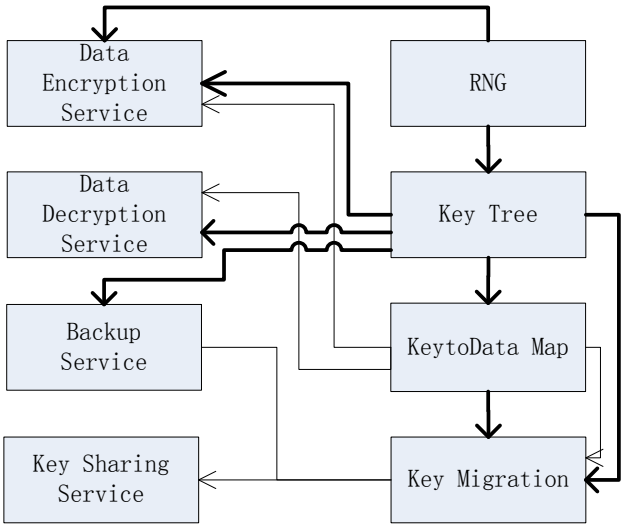

Figure 2: Architecture of PDPS

3.1 Create Key Structure of Cloud

As show in Figure 3, we design a logical structure of keys we will use in the client end of cloud storage system. The characters shown in every square represents in which step the key will be generated. Every square in Figure 3 represents a key. The process of creating such a structure includes (the creation process is based on the assumption that the TPM in the cloud client have not been used, it is just a new TPM):

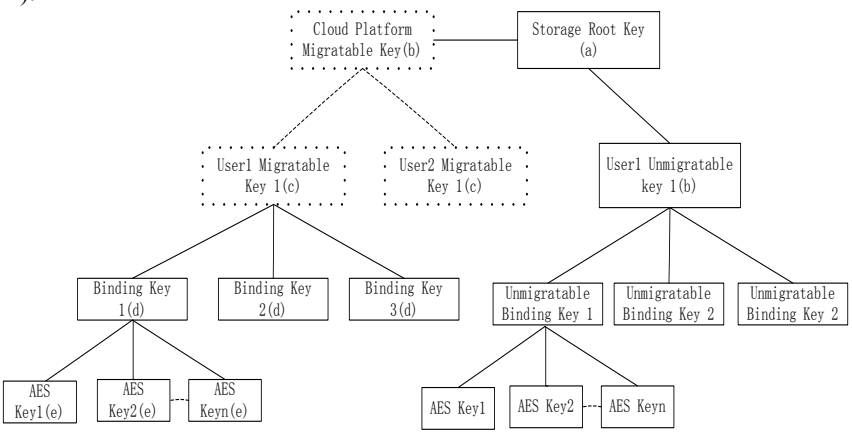

Figure3：Key Structure of Cloud

- Upon activating the TPM (or upon reset), the TPM will use its random number generator to create a new base 2048-bit SRK key. The SRK is required to be non-migratable and to be a 2048 (or greater) storage key, but other parameters are left to the end user. The parameters include whether authorization is required to use the key and what that authorization is.
- Create EK (Endorsement Key), especially this key does not involve in the Key Tree, When a TPM comes to a new user, it typically does not include anything other than an endorsement key (EK), EK is used for authentication when the user needs to share some keys or use others' keys.
- Create a Cloud Platform Migratable Key which is wrapped by its father key—SRK, the construction of a RSA Key contains the following steps: first, call RNG method to generate a 1024 bits or 2048 bits pair that includes public key and a private key p(we choose 1024 bits); second, construct a structure that can be used by TPM, in this structure, we can configure a series of attributes like "migratable" or "non-migratable", after the construction, we will move the key from the inner of the TPM to persistent storage.
- Create User1 Migratable Key which is wrapped by its father Platform Key. If the client end has another user, then create User2 Migratable Key.
- User1 creates a Binding Key (used for encrypt symmetric key) which is wrapped by User1 Migratable Key.
- If user1 want to encrypt a file, he will create a AES Key(a leaf in the key structure) and wrap the key by one of his binding key.
- If user1 have some classified data and he is sure that he does not need to share it

with some other ones. hence, he can create a User1 Non-migratable Key, the rest steps are same to a Use1 Migratable Key.

3.2 Data Processing Service

Data Processing Service contains Data Encryption Service and Data Decryption Service.

A user Alice stores her data in the cloud. Before it is sent to the cloud, the data encryption service will call the function of RNG to generate a symmetric key (we adopt a 256 bits AES key), then the service uses the symmetric key to encrypt the upload filter of the data. After the encryption, the service encrypt the symmetric key by one of the user's binding key, then input the relation between the symmetric key and the binding key and the relation between the encrypted file and binding key into a map.

When Alice gets a file from the cloud, after the authentication of TPM, the client will open a download filter. The Data Decryption Service will get the key related to the file according to the map stated in the 3.2, then use the Key to decrypt the download filter, then Alice get the plaintext.

3.3 The Backup Service

As stated in Specification of Trusted Computing, the non-migratable key cannot be migrated out from the TPM. So, in this section, we talk about the backup of migratable keys. In addition, as all we know, public key in the migratable key is not sensitive. The goal we will achieve is to ensure the private key must not be got by illegal parties when we move this private part of key out of TPM.

1. We design a structure to describe the migration authentication. In Figure 4, TPM owner authorize three kind of authority to the migratable key. The first is use authority that means the key can be used by legal customers. The second is migration authority that means the key can be migrated by the owner of the key. The third is encryption authority that means the key can be encrypted by trusted third parties (generally, trusted third party indicates the customer whom you want to share your key with). Combine the three authorization we get a migration authentication structure:

```
Struct TPM_MIGRATIONKEYAUTH{
        TPM_PUBKEY migrationKey;
        TPM_MIGRATE_SCHEME migrationScheme;
        TPM_DIGEST digest;} TPM_MIGRATIONKEYAUTH;

Digest= SHA-1(migrationKey || migrationScheme
                    ||TPM_PERMANENT_DATA -> tpmProof)
```

Figure 4: Migration Authentication Structure

Migrationkey represents the public key provided by the trusted party. Migration Scheme represents the scheme in which key will be migrated. When TPM owner change, the TPM Proof changes too, hence the digest, at this case, the MIGRATIONKEYAUTH will be invalid.

2. Check the migration authentication structure to see if the key is migratable and if the wrapping key has the authority to wrap the migratable key. If both checks are passed, we can get the migratable key from key structure stated in Chapter 3.1.
3. Construct the backup structure, as shown in Figure 5, the construction contains three steps:

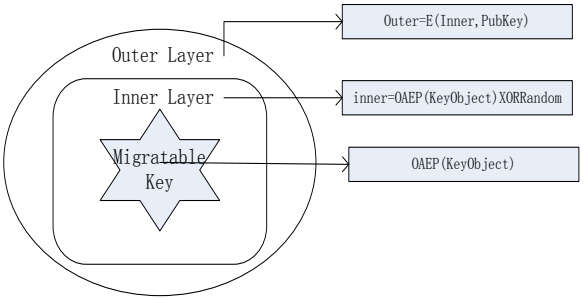

Figure 5: Backup structure

- Use OAEP (defined in RSA PKCS1 V2.1) to codify the key object that is attained in the step 2. The process protects the key from being tampered by vicious cloud storage manager.
- Call the RNG method to generate a random, and XOR it with OAEP(Key Object), then we get the inner layer.
- Use the authorized public key to encrypt the inner layer, then we get the outer layer

4. The backup object can be stored in many ways. When we need to inject the backup object, the process contains 4 steps.
   - Customer input password get the authority to communicate with TPM in the target platform.
   - The target platform receives the backup object, and then, the TPM gets the private key from the key structure.
   - Decrypt the backup object with the private key, then we get the inner layer
   - After the Key owner provides the random which is used in the backup construction process, we can use the random to XOR the inner layer to get the key object.

3.4 The Key Sharing Service

The key sharing service will use the backup structure stated in Chapter 3.3. The sharing process concludes:

- When customer Alice needs to share a file with Bob, first, Bob will send a public key (the corresponding private key has existed in the key tree in Bob's TPM) to

Alice.

- Alice use the public key to create a backup object by the process stated in Chapter 3.3, after the process, Alice sends the backup object to Bob.
- After Bob receives the backup object, he will use the process stated in the chapter 3.3 to inject the key object into Bob's TPM. After the injection, Bob can use the key freely.

4 Constructing the prototype and the performance evaluation

We implement PDPS in HDFS (Hadoop Distributed File System). Our experiment is constructed using four computers with an Intel Core 2 processor running at 1.86 GHz, 2048 MB of RAM, and a 7200 RPM Western Digital 250 GB Serial ATA drive with an 8 MB buffer – one namenode, two datanodes, and one client. As shown in Figure 6, they are all in the same Ethernet.

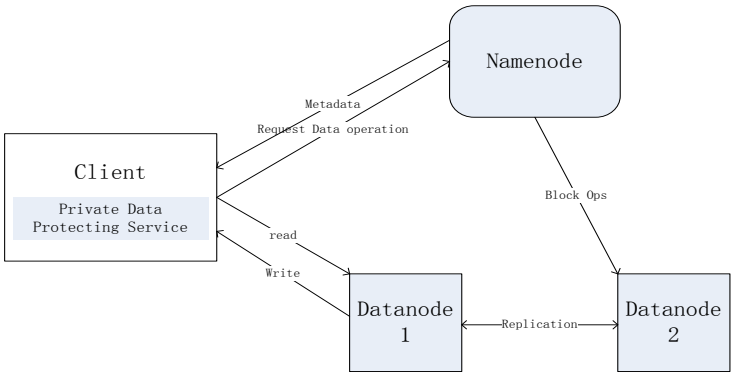

Figure 6: Architecture of HDFS with PDPS

Compared to HDFS without a PDSP, the time we will used for PDPS (when uploading a file) includes the time it costs to generate a symmetric key, the time it costs for encryption of the file upload filter, and the time it costs for encryption of the symmetric key and persistence of the symmetric key. Especially, the time cost for encrypting the file upload filter is offset by the time cost for uploading the file. Table1 shows the time we cost when we upload various kinds of data with the PDSP comparing the uploading without a PDSP.Table2 shows the time we cost when we download files of various amounts.

Table1: Time consumption of uploading a file

File size	1K	1M	32M	128M	1G
with PDSP (ms)	635	1322	3133	12227	93744
without PDSP (ms)	523	1088	2746	11601	92532
Cost percentage	21.4	21.1	14.1	4.54	1.21

Table2: Time consumption of downloading a file

File size	1K	1M	32M	128M	1G
with PDSP (ms)	1456	3056	10056	35452	3065256
without PDSP (ms)	1178	2532	8999	32669	2955235
Cost percentage	23.6	20.7	11.7	8.5	3.7

From Table1 and Table2, we can see the cost percentage decreases when the file size is growing. So we can make a conclusion that the PDSP suits for the cloud storage system that cope with files with large size like HDFS.

5. Conclusions and Future Work

In this paper, we have presented the architecture of private protecting service to keep the data in cloud safe. In the service, we explored a new fashion of data backup in a more flexible way. If user1 want to backup all his keys, he just needs to construct only one backup object—user1 migratable key. If the cloud system platform needs backup all its keys, it just needs to construct only one backup object—cloud platform migratable key. So, the same cost to the key sharing service. In the further study, we will implement the smaller unit of data process like Chunk in GFS, or block in HDFS and propose detailed designs for the authentication framework that will be used in key sharing service based on signature, remote attestation. The private data protecting service can be adopted by almost all of the cloud storage system, because it is based on the client end and have no interaction with the server or cloud provider.

Acknowledgement. This work is supported by National Natural Science Foundation of China under Grant No. 60873238, 61073156, 61070237, 60970135, and our IBM SUR 2009 Project.

References

1. S.Kmara, K.Lauter, “Cryptographic Cloud Storage”, Proceedings of Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization 2010, pp. 111-116, January 2010.
2. D.Challener, K.Yoder, A Practical Guide To Trusted Computing, IBM Press, PP.17-18, 2009.
3. B.Balacheff, L.Chen, Trusted Computing Platforms, Prentice Hall PTR, PP.166-167.2009.
4. Wang, C., Wang, Q., Ren, K., Lou, W.: Ensuring data storage security in cloud computing. In: Proc. of IWQoS 2009, Charleston, South Carolina, USA (2009)
5. Ateniese, G., Di Pietro, R., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: Proc. of SecureComm 2008 (2008)
6. W.Stallings, Cyptography and Network Security Principles and Practices, Fourth Edition, PP. 59-60, 2006

7. M. Christodorescu, R. Sailer, D. L. Schales, D. Sgandurra, and D. Zamboni. Cloud Security is not (just) Virtualization Security. In Proc. CCSW, 2009.
8. J. Dean and S. Ghemawat. Mapreduce: Simplied data processing on large clusters. In Proc. OSDI, 2004.
9. A. Haeberlen, P. Kuznetsov, and P. Druschel. PeerReview:Practical Accountability for Distributed Systems. In Proc. SOSP, 2007.
10. M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono. On Technical Security Issues in Cloud Computing. In Proc. CLOUD, 2009.