International Journal of Recent Trends in Engineering & Research
ISSN : 2455-1457 Impact Factor : 3.344

ENHANCING INTEGRITY USING FMSR-DIP SCHEME OVER CLOUD DATA

Seenivasan. D[1], Latha. G[2], Revathy. V[3], SeemaNasrin. S[4]
[1]Assistant Professor,Department of CSE, K.S.Rangasamy College of Technology, Tiruchengode, Tamil Nadu, India
[2,3,4] Student, Department of CSE, K.S.Rangasamy College of Technology, Tiruchengode, Tamil Nadu, India

Abstract: Data de-duplication is a technique for eliminating duplicate copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth. Promising as it is, an arising challenge is to perform secure deduplication in cloud storage. Although convergent encryption has been extensively adopted for secure deduplication, a critical issue of making convergent encryption practical is to efficiently and reliably manage a huge number of convergent keys.

such a baseline key management scheme generates an enormous number of keys with the increasing number of users and requires users to dedicatedly protect the master keys. To this end, this project proposes Dekey, a new construction in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers.

The original data copy is first encrypted with a convergent key derived by the data copy itself, and the convergent key is then encrypted by a master key that will be kept locally and securely by each user. The encrypted convergent keys are then stored, along with the corresponding encrypted data copies, in cloud storage.

In addition, the project also considers the revocation of users in the given group. If the original (first) user of the group intimates the server with a user's (B) revocation, then the server rejects the proof of ownership submitted by that user (B).

I. INTRODUCTION

Cloud computing means that instead of all the computer hardware and software you're using sitting on your desktop, or somewhere inside your company's network, it's provided for you as a service by another company and accessed over the Internet, usually in a completely seamless way. Exactly where the hardware and software is located and how it all works doesn't matter to you, the user—it's just somewhere up in the nebulous "cloud" that the Internet represents.

Types of cloud computing

1.Infrastructure as a Service (IaaS) means you're buying access to raw computing hardware over the Net, such as servers or storage. Since you buy what you need and pay-as-you-go, this is often referred to as utility computing. Ordinary web hosting is a simple example of IaaS: you pay a monthly subscription or a per-megabyte/gigabyte fee to have a hosting company serve up files for your website from their servers.

2.Software as a Service (SaaS) means you use a complete application running on someone else's system. Web-based email and Google Documents are perhaps the best-known examples. Zoho is another well-known SaaS provider offering a variety of office applications online.

3.Platform as a Service (PaaS) means you develop applications using Web-based tools so they run on systems software and hardware provided by another company. So, for example, you might develop your own ecommerce website but have the whole thing, including the shopping cart, checkout, and payment

mechanism running on a merchant's server. Force.com (from salesforce.com) and the Google App Engine are examples of PaaS.

II. EXISTING SYSTEM

General Symmetric Encryption process includes

1. KeyGenSE: is the key generation algorithm that generates K using security parameter.
2. EncryptSE(K, M) C is the symmetric encryption algorithm that takes the secret and message M and then outputs the ciphertext C.
3. DecryptSE(K, C): M is the symmetric decryption algorithm that takes the secret K and ciphertext C and then outputs the original message M.

In the existing system, a user (or data owner) derives a convergent key from each original data copy and encrypts the data copy with the convergent key. In addition, the user derives a tag for the data copy, such that the tag will be used to detect duplicates.

Apart from step 1, 2 and 3 it also contains

TagGenCE(M):T(M) is the tag generation algorithm that maps the original data copy M and outputs a tag T(M). It allows TagGenCE to generate a tag from the corresponding ciphertext, by using T(M)=TagGenCE(C), where C=EncryptCE(K, M).

This tag will be used to reuse the key after some time by the same user if the proof of ownership is satisfied.

Instead of encrypting the convergent keys on a per-user basis, Dekey constructs secret shares on the original convergent keys (that are in plain) and distributes the shares across multiple KM-CSPs (Key Management-Cloud Service Provider). If multiple users share the same block, they can access the same corresponding convergent key.

III. PROPOSED SYSTEM

All the existing system methods are implemented in proposed system. Using the proposed algorithms, the KM-CSP can efficiently manage the group of users.

In addition, the project also considers the revocation of users in the given group. If the original (first) user of the group intimates the server with a user's (B) revocation, then the server rejects the proof of ownership submitted by that user (B).

Session based deduplication is considered. Here if the user provides the session duration i.e, front date and to date, then only with the data range, proof of ownership can be allowed in server on those dates. This increases the security if the outsourced data need to be safely accessed on the given duration.

The advantages of the proposed system are,

- Session based outsource data access is provided to increase the security.
- User Revocation management is also implemented.
- Key Storage cost is less when compared to existing system.

IV. MODULE DESCRIPTION

The project contains the following modules.

- **SYMMETRIC ENCRYPTION**
- **CONVERGENT ENCRYPTION**
- **PROOF OF OWNERSHIP**
- **PROOF OF OWNERSHIP BASED ON SESSION**
- **REVOCATION OF USERS**

1. SYMMETRIC ENCRYPTION

In this module, general symmetric encryption/decryption technique is implemented.
Symmetric encryption uses a common secret key *K* to encrypt and decrypt information. A symmetric encryption scheme consists of three primitive functions:
KeyGenSE (1□): K is the key generation algorithm that generates *K* using security parameter 1□;
EncryptSE (*K*, M): C is the symmetric encryption algorithm that takes the secret *K* and message M and then outputs the ciphertext C;
DecryptSE (*K*, C): M is the symmetric decryption algorithm that takes the secret *K* and ciphertext C and then outputs the original message M.

2. CONVERGENT ENCRYPTION

Convergent encryption provides data confidentiality in deduplication. A user (or data owner) derives a convergent key from each original data copy and encrypts the data copy with the convergent key. In addition, the user derives a tag for the data copy, such that the tag will be used to detect duplicates.
To detect duplicates, the user first sends the tag to the server side to check if the identical copy has been already stored. Note that both the convergent key and the tag are independently derived and the tag cannot be used to deduce the convergent key and compromise data confidentiality. Both the encrypted data copy and its corresponding tag will be stored on the server side.

3. PROOF OF OWNERSHIP

The verifier derives a short value from a data copy M. To prove the ownership of the data copy M, the prover needs to send and run a proof algorithm with the verifier. It is passed if and only if and the proof is correct.

4. PROOF OF OWNERSHIP BASED ON SESSION

In this module, session based deduplication is considered. Here if the user provides the session duration i.e, front date and to date, then only with the data range, proof of ownership can be allowed in server on those dates. This increases the security if the outsourced data need to be safely accessed on the given duration.

5. REVOCATION OF USERS

In this module, we consider the revocation of users in the given group. If the original (first) user of the group intimates the server with a user's (B) revocation, then the server rejects the proof of ownership submitted by that user (B).

V. CONCLUSION

The proposed system attempts to formally address the problem of achieving efficient and reliable key management in secure deduplication. It introduces a baseline approach in which each user holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud.
Such a baseline key management scheme generates an enormous number of keys with the increasing number of users and requires users to dedicatedly protect the master keys. It proposes Dekey, a new construction in which users do not need to manage any keys on their own but instead securely distribute the convergent key shares across multiple servers. Security analysis demonstrates that Dekey is secure in terms of the definitions specified in the proposed security model. In addition, the users can revocate from the given group at any time. To do so, if the original (first) user of the group intimates the server with a user's (B) revocation, then the server rejects the proof of ownership submitted by that user (B). Likewise, session based deduplication is considered. Here if the user provides the session duration i.e,

front date and to date, then only with the data range, proof of ownership can be allowed in server on those dates. This increases the security if the outsourced data need to be safely accessed on the given duration.

At present, the project involves General Symmetric Encryption process which includes tag generation. Here Group based user management is not provided. Also Session based outsource data access is not provided. In addition User Revocation management is not implemented. In future these options can be included so that Session based outsource data access can be provided to increase the security. User Revocation management can also implemented. Key Storage cost can be reduced when compared to existing system.

REFERENCES

i. P. Anderson and L. Zhang, ‘‘Fast and Secure Laptop Backups with Encrypted De-Duplication,’’ in Proc. USENIX LISA, 2010, pp. 1-8.
ii. S. Kamara and K. Lauter, ‘‘Cryptographic Cloud Storage,’’ in Proc. Financial Cryptography: Workshop Real-Life Cryptograph. Protocols Standardization, 2010, pp. 136-149.
iii. Mac OS X Time Machine [cited 2nd April 2010].
iv. W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur.Single instance storage in Windows 2000.In Proceedings of 4th USENIX Windows Systems Symposium.Usenix, 2000.
v. C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, J. Szczepkowski, C. Ungureanu, and M. Welnicki. Hydrastor: a scalable secondary storage. In Proc. 7th USENIX Conference on File and Storage Technologies, 2009.
vi. W. Bolosky, S. Corbin, D. Goebel and J. Douceur.Single instance storage in Windows 2000.In Proc. 4th USENIX Windows Systems Symposium, 2000.
vii. B. Zhu, K. Li, and H. Patterson.Avoiding the disk bottleneck in the Data Domain deduplication file system.In Proc. 6th USENIX Conference on File and Storage Technologies, 2008, pp. 1-14.
viii. N. Agrawal, W. Bolosky, J. Douceur and J. Lorch.A five-year study of file-system metadata.In Proc. 5th USENIX Conference on File and Storage Technologies, 2007.
ix. Microsoft Corporation. Volume Shadow Copy Service.MSDN.[Online] 2010.[Cited August 31, 2010.]
x. M. Rabin. Fingerprinting by Random Polynomials. Harvard University Center for Research In Computing Technology Technical Report TR-CSE-03-01, 1981. Boston, MA.
xi. ShaiHalevi, Danny Harnik, Benny Pinkas and Alexandra Shulman-Peleg: Proof of ownership in remote storage systems. eprint, IACR, 2011/207.
xii. M. Vrable, S. Savage, and G. Voelker. Cumulus: Filesystem backup to the cloud. In Proc. of USENIX FAST, 2009.
xiii. Amazon. SmugMug Case Study: Amazon Web Services.
xiv. A. Shamir. How to Share a Secret. CACM, 22(11):612–613, Nov 1979.
xv. R. Geambasu, T. Kohno, A. Levy, and H. M. Levy. Vanish: Increasing Data Privacy with Self-Destructing Data. In Proc. of USENIX Security Symposium, Aug 2009.

