www.ijiarec.com

ISSN:2348-2079

Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Secure Deduplication with Efficient and Reliable Convergent Key Management

[1]Mr. R. NavinKumar, M.C.A., M.Phil., Assistant Professor/MCA
[2]Mr. N. Sridharan, III MCA
Department of MCA, Nandha Engineering College(Autonomous), Erode-52.
E-mail Id : navinsoccer@gmail.com, bcasridhar9443@gmail.com

***Abstract* - Data deduplication 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. This paper makes the first attempt to formally address the problem of achieving efficient and reliable key management in secure deduplication. The project first introduces a baseline approach in which each user holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud. However, 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.**

In addition, the paper focuses on trade-offs between storage cost and rekeying cost for secure multicast based on hierarchical tree structure. Membership in secure multicast groups is dynamic and requires multiple updates in single time frame. The project presents a family of algorithms that provide a trade-off between the number of keys maintained by users and the time required for rekeying due to revocation of multiple users. It is shown that some well-known algorithms in the literature are members of this family. The project shows that algorithms in this family can be used to reduce the cost of rekeying when compared with previous solutions while keeping the number of keys manageable. It also describes a scheme to reduce the number of secrets further when revocations are periodic.

***Index Terms* – Deduplication, KeyGenSE, EncryptSE, DecryptSE, Encoding/Decoding.**

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.

The advent of cloud storage motivates enterprises and organizations to outsource data storage to third-party cloud providers, as evidenced by many real-life case studied. One critical challenge of today's cloud storage services is the management of the ever-increasing volume of data. A data management scalable, de-duplication has been a well-known technique to reduce storage space and upload bandwidth in cloud storage. Instead of keeping multiple data copies with the same content, de-duplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Each such copy can be defined based on different granularities: it may refer to either a whole file (i.e., file- level de-duplication), or a more fine-grained fixed-size.

This thesis study motivates us to explore how to efficiently and reliably manage enormous convergent keys, while still achieving secure de-duplication. To this end, we propose a new construction called Dekey, which provides efficiency and reliability guarantees for convergent key management on both user and cloud storage sides. Our idea is to apply de-duplication to the convergent keys and leverage secret sharing techniques. Specifically, we construct secret shares for the convergent keys and distribute them across multiple independent key servers.

Only the first user distribute such secret shares, while all following users who own the same data copy need not compute and store these shares again. To recover data copies, a user must access a minimum number of key servers through authentication and obtain the secret shares to reconstruct the convergent keys. In addition project, the secret shares of a convergent key will only be accessible by the authorized users who own the corresponding data copy. This project significantly reduces the storage overhead of the convergent keys and makes the key management reliable against failures and attacks.

One critical challenge of today's cloud storage services is the management of the ever increasing volume of data. To make data management scalable, deduplication has been a well-known technique to reduce storage space and upload bandwidth in cloud storage. Instead of keeping multiple data copies with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Each such copy can be defined based on different granularities: it may refer to either a whole file (i.e., filelevel deduplication), or a more fine-grained fixed-size or variable-size data block (i.e., block-level deduplication).Today's commercial cloud storage services, such as Dropbox, Mozy, and Memopal, have been applying deduplication to user data to save maintenance cost.

According to the user's view, data outsourcing raises security and privacy concerns. We must trust third-party cloud providers to properly enforce confidentiality, integrity checking, and access control mechanisms against any insider and outsider attacks. However, deduplication, while improving storage and bandwidth efficiency, is incompatible with traditional encryption. Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different ciphertexts, making deduplication impossible. A new construction Dekey is proposed to provide efficient and reliable convergent key management through convergent key deduplication and secret sharing. Dekey supports both file-level and blocklevel deduplications.

II. RELATED WORKS

This paper [1] describes an algorithm which takes advantage of the data which is common between users to increase the speed of backups, and reduce the storage requirements. This algorithm supports client-end per-user encryption which is necessary for confidential personal data. It also supports a unique feature which allows immediate detection of common subtrees, avoiding the need to query the backup system for every file. They describe a prototype implementation of this algorithm for Apple OS X, and present an analysis of the potential effectiveness, using real data obtained from a set of typical users. Finally, the author discusses the use of this prototype in conjunction with remote cloud storage, and present an analysis of the typical cost savings.

File systems often contain redundant copies of information: identical files or sub-file regions, possibly stored on a single host, on a shared storage cluster, or backed-up to secondary storage. Deduplicating storage systems take advantage of this redundancy to reduce the underlying space needed to contain the file systems (or backup images thereof). Deduplication can work at either the sub-file or whole-file level. More fine-grained deduplication creates more opportunities for space savings, but necessarily reduces the sequential layout of some files, which may have significant performance impacts when hard disks are used for storage (and in some cases necessitates complicated techniques to improve performance).

Alternatively, whole-file deduplication is simpler and eliminates file-fragmentation concerns, though at the cost of some otherwise reclaimable storage. Because the disk technology trend is toward improved sequential bandwidth and reduced per-byte cost with little or no improvement in random access speed, it's not clear that trading away sequentiality for space savings makes sense, at least in primary storage.

In this paper [4], a new notion which they call private data deduplication protocol, a deduplication technique for private data storage is introduced and formalized. Intuitively, a private data deduplication protocol allows a client who holds a private data proves to a server who holds a summary string of the data that he/she is the owner of that data without revealing further information to the server. Their notion can be viewed as a complement of the state-of-the-art public data deduplication protocols of Halevi et al [19]. The security of private data deduplication protocols is formalized in the simulation-based framework in the context of two-party computations. A construction of private deduplication protocols based on the standard cryptographic assumptions is then presented and analyzed. They show that the proposed private data deduplication protocol is provably secure assuming that the underlying hash function is collision-resilient, the discrete logarithm is hard and the erasure coding algorithm can erasure up to α-fraction of the bits in the presence of malicious adversaries in the presence of malicious adversaries. To the best their knowledge this is the first deduplication protocol for private data storage.

Cloud storage is an emerging service model that enables individuals and enterprises to outsource the

storage of data backups to remote cloud providers at a low cost. However, cloud clients must enforce security guarantees of their outsourced data backups. They present FadeVersion, a secure cloud backup system that serves as a security layer on top of today's cloud storage services. FadeVersion follows the standard version-controlled backup design, which eliminates the storage of redundant data across different versions of backups. On top of this, FadeVersion applies cryptographic protection to data backups. Specifically, it enables fine-grained assured deletion, that is, cloud clients can assuredly delete particular backup versions or files on the cloud and make them permanently inaccessible to anyone, while other versions that share the common data of the deleted versions or files will remain unaffected. They implement a proof-of-concept prototype of FadeVersion and conduct empirical evaluation atop Amazon S3. They show that FadeVersion only adds minimal performance overhead over a traditional cloud backup service that does not support assured deletion.

In this paper [5], the authors present FadeVersion, a secure cloud backup system that supports both version control and assured deletion. FadeVersion allows fine-grained assured deletion, such that cloud clients can specify particular versions or files on the cloud to be assuredly deleted, while other versions that share the common data of the deleted versions or files will remain unaffected. The main idea of FadeVersion is to use a layered encryption approach. Suppose that a file F appears in multiple versions. They first encrypt F with key k, and then encrypt key k independently with different keys associated with different versions. Thus, if they remove a key of one version, they can still recover key k and hence file F in another version.

In this paper [6], the author now outsource data backup to third-party cloud storage services so as to reduce data management costs, security concerns arise in terms of ensuring the privacy and integrity of out-sourced data. They design FADE, a practical, implementable, and readily deployable cloud storage system that focuses on protecting deleted data with policy-based file assured deletion. FADE is built upon standard cryptographic techniques, such that it encrypts outsourced data files to guarantee their privacy and integrity, and most importantly, assuredly deletes files to make them unrecoverable to anyone (including those who manage the cloud storage) upon revocations of file access policies. In particular, the design of FADE is geared toward the objective that it acts as an overlay system that works seamlessly atop today's cloud storage services. To demonstrate this objective, they implement a working prototype of FADE atop Amazon S3, one of today's cloud storage services, and empirically show that FADE provides policy-based file assured deletion with a minimal trade-off of performance overhead. Their work provides insights of how to incorporate value-added security features into current data outsourcing applications.

In this paper [7], the authors present a comprehensive characterization of backup workloads by analyzing statistics and content metadata collected from a large set of EMC Data Domain backup systems in production use. This analysis is both broad (encompassing statistics from over 10,000 systems) and deep (using detailed metadata traces from several production systems storing almost 700TB of backup data). They compare these systems to a detailed study of Microsoft primary storage systems, showing that backup storage differs significantly from their primary storage workload in the amount of data churn and capacity requirements as well as the amount of redundancy within the data. These properties bring unique challenges and opportunities when designing a disk-based file system for backup workloads, which they explore in more detail using the metadata traces. In particular, the need to handle high churn while leveraging high data redundancy is considered by looking at deduplication unit size and caching efficiency.

In this paper [7] the author first analyze statistics from a broad set of 10,000+ production EMC Data Domain systems. They also collect and analyze content-level snapshots of systems that, in aggregate, are to their knowledge at least an order of magnitude larger than anything previously reported. Their statistical analysis considers information such as file age, size, counts, deduplication effectiveness, compressibility, and other metrics. Comparing this to Meyer and Bolosky's analysis of a large collection of systems in Microsoft Corp. They see that backup workloads tend to have shorter-lived and larger files than primary storage. This is indicative of higher data churn rates, ameasure of the percentage of storage capacity that is written and deleted per time interval (e.g., weekly), as well as more data sequentiality. These have implications for the design requirements for purpose-built backup systems.

III. METHODOLOGY

General Symmetric Encryption process includes

- KeyGenSE: is the key generation algorithm that generates K using security parameter.
- EncryptSE(K, M) C is the symmetric encryption algorithm that takes the secret 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.

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.

- The group controller maintains a logical hierarchy of keys. To revoke multiple users, the KM-CSP distributes the new group key by using keys that are not known to the revoked users. However, this solution achieves a good rekeying cost only if the size of the revoked users is either very small or very large. In the above schemes, the logical key tree structure tends to become unbalanced after some membership changes.
- To handle multiple membership changes, the group controller repeats the process of revocation for each revoked user.
- Not hierarchical structure is used so that the number of keys is more.

Using the proposed algorithms, the KM-CSP can efficiently distribute the group key. The proposed system describes the family of key management algorithms for efficiently distributing the new group key when multiple users are revoked from the group.

In the algorithms, the storage at the group controller is linear and the storage at the users is logarithmic in the size of the group. The proposed key management algorithms for efficiently distributing the new group key when multiple users are revoked from the group. In the algorithms, the storage at the group controller is linear and the storage at the users is logarithmic in the size of the group.

- Hierarchical structure is used so that the number of keys is less.
- Encoding/Decoding overhead is low compared to existing system.
- Key Storage cost is less when compared to existing system

A. 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. In this module, four primitive functions are implemented to achieve the convergent encryption mechanism.

$KeyGen_{CE}$ (M): K is the key generation algorithm that maps a data copy M to a convergent key K;

$Encrypt_{CE}$(K, M): C is the symmetric encryption algorithm that takes both the convergent key K and the data copy M as inputs and then outputs a ciphertext C;

$Decrypt_{CE}$(K,C): M is the decryption algorithm that takes both the ciphertext C and the convergent key K as inputs and then outputs the original data copy M and

$TagGen_{CE}$(M): T(M) is the tag generation algorithm that maps the original data copy M and outputs a tag T(M). We allow TagGenCE to generate a tag from the corresponding ciphertext, by using $T(M)=TagGen_{CE}(C)$, where $C=Encrypt_{CE}(K,M)$.

B. PROOF OF OWNERSHIP

The verifier derives a short value □□M) 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 □' □□□□M) and the proof is correct.

C. PRROF 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.

D. 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).

IV. CONCLUSION

Data deduplication is a technique for eliminating duplicate copies of data, and has been widely used in cloud storage to reduce storage space and upload bandwidth. This project 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.

However, 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

[1] P. Anderson and L. Zhang, ''Fast and Secure Laptop Backups with Encrypted De-Duplication,'' in Proc. USENIX LISA, 2010, pp. 1-8.

[2] S. Kamara and K. Lauter, ''Cryptographic Cloud Storage,'' in Proc. Financial Cryptography: Workshop Real-Life Cryptograph. Protocols Standardization, 2010, pp. 136-149.

[3] Mac OS X Time Machine [cited 2nd April 2010]. http://www.apple.com/macosx/what-is-macosx/ time-machine.html.

[4] 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. http://research.microsoft.com/apps/pubs/ default.aspx?id=74261.

[5] 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.

[6] W. Bolosky, S. Corbin, D. Goebel and J. Douceur. Single instance storage in Windows 2000. In Proc. 4th USENIX Windows Systems Symposium, 2000.

[7] 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.

[8] 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.

[9] Microsoft Corporation. Volume Shadow Copy Service. MSDN. [Online] 2010. [Cited August 31, 2010.] http://msdn.microsoft.com/en us/library/bb968832(VS.85).aspx

[10] M. Rabin. Fingerprinting by Random Polynomials. Harvard University Center for Research In Computing Technology Technical Report TR-CSE-03-01,1981.Boston, MA.

[11] Shai Halevi, Danny Harnik, Benny Pinkas and Alexandra Shulman-Peleg: Proof of ownership in remote storage systems. eprint, IACR, 2011/207.

[12] M. Vrable, S. Savage, and G. Voelker. Cumulus: Filesystem backup to the cloud. In Proc. of USENIX FAST, 2009.