International Journal of Scientific Research and Engineering Development-– Volume3 Issue 1, Jan-Feb 2020

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Cloud Data Deduplication Using Threshold Convergent Cryptosystem

Sajin R Nair[1], Bibin Varghese[2]

[1]P G scholar, Dept. of CSE, Mount Zion College of Engineering, Kadammanitta, Kerala, India
[2]Assistant Professor, Dept. of CSE, Mount Zion College of Engineering, Kadammanitta, Kerala, India

----------**********----------

Abstract:

As a lot of company and personal users source their knowledge to cloud storage, recent knowledge breach incidents build end-to-end coding progressively fascinating. Data access control is an effective way to ensure the data security in the cloud. Due to data outsourcing and untrusted cloud servers, the data access control becomes a challenging issue in cloud storage systems.Architecture that provides secure deduplicated storage resisting brute-force attacks, and realize it in a system called DupLESS. In DupLESS, clients encrypt under message-based keys obtained from a key-server via an oblivious PRF protocol. Cipher text-Policy Attribute based Encryption (CP-ABE) is regarded as one of the most suitable technologies for data access control in cloud storage, because it gives data owners more direct control on access policies.Semantically secure coding renders numerous efficient storage improvement techniques, like knowledge deduplication, ineffective. Here introduced the conception of "data popularity" contestation that knowledged by several users doesn't need as robust protection as less-traveled data; supported this. Wherever it starts semantically secure ciphertext of a file is transparently downgraded to a convergent ciphertext that enables for deduplication as presently because the file becomes standard. During this paper propose associate increased version of the first theme. Specializing in usefulness, author tends to modify the first theme to enhance its efficiency and emphasize clear practicality. Author tends to analyze the efficiency supported quality properties of real datasets and supply a close performance analysis, together with comparison to various schemes in real-like settings. New theme moves the handling of sensitive decipherment shares and recognition state data out of the cloud storage, granting improved security notion, easier security proofs and easier adoption. Author tends to show that the new theme is more secure with External Diffie-Hellman assumption within the random oracle model.

Keywords **—Cipher Text-Policy Attribute Based Encryption(CP-ABE),DupLESS,Deduplication, Ciphertext,**

----------**********----------

I. INTRODUCTION

With the rapidly increasing amounts of data produced worldwide, networked and multi-user storage systems are becoming very popular. However, concerns over data security still prevent many users from migrating data to remote storage. The conventional solution is to encrypt the data before it leaves the owner's premises. While sound from a security perspective, this approach prevents the storage provider from e□ectively applying storage e□ciency functions, such as compression and deduplication, which would allow optimal usage of the resources and consequently lower service cost. Client-side data deduplication in particular ensures that multiple uploads of the same content only consume network bandwidth and storage space of a single upload. Deduplication is

actively used by a number of cloud backup providers as well as various cloud services (e.g. Dropbox). Unfortunately, encrypted data is pseudorandom and thus cannot be deduplicated: as a consequence, current schemes have to entirely sacrifice either security or storage e□ciency.

A scheme that permits a more fine grained security-to-efficiency trade-off is presented. The intuition is that outsourced data may require different levels of protection, depending on how popular the datum is: content shared by many users, such as a popular song or video, arguably requires less protection than a personal document, the copy of a pay slip or the draft of an unsubmitted scientific paper. This differentiation based on popularity also partly alleviates the user's need to manually sort data as common (deduplicable) and potentially sensitive (non-deduplicable) as every file is first treated as potentially sensitive and thus not deduplicated. Deduplication occurs only when the file becomes popular (i.e., is shared by many users). Note that strictly confidential files that must never be deduplicated should be explicitly marked as non-deduplicable (e.g., by the user or policy) and handled separately. Presented an encryption scheme, where the initially semantically secure ciphertext of a file is transparently downgraded to a convergent ciphertext that allows for deduplication as soon as the file becomes popular. In this paper here propose an enhanced version of the original scheme.

The author focus more on the practical application of the proposed scheme, providing an entirely new evaluation of the scheme performance together with efficiency analysis that enables potential adopters to easily predict the cost and efficiency of scheme deployment in their environments. Scheme internals and their description were modified to better reflect the individual processing steps and to achieve better performance, sparing unnecessary complexity where possible. Specifically, handling of the decryption shares was significantly modified to prevent the attacker from guessing whether two ciphertexts produced by the same user correspond to the same plaintext with a non-negligible advantage. This modification also enabled notable simplification of the security analysis. Additionally, removal of the file popularity information from the storage provider database enables the storage provider to handle both popular and unpopular files in the same way.

In this paper, compare different technologies or methods that can be used to find out the data deduplication. And also find out best one from the comparison.

II. LITERATURE SURVEY

J. Stanek, et al.[1] propose a a scheme that permits a more fine-grained trade-o□. The intuition is that outsourced data may require di□erent levels of protection, depending on how popular it is: content shared by many users, such as a popular song or video, arguably requires less protection than a personal document, the copy of a payslip or the draft of an unsubmitted scientific paper.

Storage e□ciency functions such as compression and deduplication a□ord storage providers better utilization of their storage backends and the ability to serve more customers with the same infrastructure. Data deduplication is the process by which a storage provider only stores a single copy of a file owned by several of its users. There are four di□erent deduplication strategies, depending on whether deduplication happens at the client side (i.e. before the upload) or at the server side, and whether deduplication happens at a block level or at a file level. Deduplication is most rewarding when it is triggered at the client side, as it also saves upload bandwidth. For these reasons, deduplication is a critical enabler for a number of popular and successful storage services (e.g. Dropbox, Memopal) that o□er cheap, remote storage to the broad public by performing client-side deduplication, thus saving both the network bandwidth and storage costs. Indeed, data deduplication is arguably one of the main reasons why the prices for cloud storage and cloud backup services have dropped so sharply.

Harnik,B.et.Al, [2] ly a single copy of redundant data, and provide links to that copy instead of storing other actual copies of this data. As storage services transition from tape to disk, data deduplication has become a key component in the backup process. By storing and transmitting

only a single copy of duplicate data, deduplication saves both disk space and network bandwidth. For vendors, it offers secondary cost savings in power and cooling achieved by reducing the number of disk spindles. These savings also translate to lower fees for service users. Deduplication's effectiveness depends on such factors as the type of data, the retention period, and the number of users. The percentage of space reduction is calculated as 100 percent less the inverse of the space reduction ratio. So, even a deduplication ratio of 1:3 results in a 66 percent saving.

Reported deduplication ratios in common business settings range from 1:10 to 1:500, resulting in disk and bandwidth savings of more than 90 percent. However, there's an inherent risk in entrusting data to the storage cloud. In doing so, the data owner releases control over the data. Yet, a wide range of users and applications are more than willing to hand over their data storage tasks to cloud providers. They put their trust in the cloud provider's integrity and in the security of its access control mechanisms. Setting these issues aside, we point out an additional threat: the privacy implications of cross-userdeduplication.

We demonstrate how deduplication in cloud storage services can serve as a side channel that reveals information about the contents of other users' files. Deduplication can also serve as a covert channel through which malicious software can communicate with a command-and-controlcenter, regardless of any firewall settings at the attacked machine. We analyzededuplication's security issues and propose a simple mechanism that allows cross-userdeduplication while reducing the risk of data leakage. Our mechanism states rules by which deduplication can be artificially turned off. This simple practice gives clients a guarantee that adding their data to the cloud has a limited effect on what an adversary might learn about this data. Thus, we can essentially assure clients of the privacy of their data.

The work proposed by S. Keelveedhi [3] DupLESS starts with the observation that brute-force ciphertext recovery in a CE-type scheme can be dealt with by using a key server (KS) to derive keys, instead of setting keys to be hashes of messages. Access to the KS is preceded by authentication, which stops external attackers. The increased cost slows down brute-force attacks from compromised clients, and now the KS can function as a (logically) single point of control for implementing rate-limiting measures. author can expect that by scrupulous choice of rate-limiting policies and parameters, brute-force attacks originating from compromised clients will be rendered less effective, while normal usage will remain unaffected.

Author start by looking at secret-parameter MLE, an extension to MLE which endows all clients with a systemwide secret parameter sk The rationale here is that if sk is unknown to the attacker, a high level of security can be achieved (semantic security, except for equality), but even if sk is leaked, security falls to that of regular MLE. A server-aided MLE scheme then is a transformation where the secret key is restricted to the KS instead of being available to all clients. One simple approach to get server-aided MLE is to use a PRF F, with a secret key K that never leaves the KS. A client would send a hash H of a file to the KS and receive back a message-derived key $K' \leftarrow F(K,H)$. The other steps are as in CE.

D. Meister and A. Brinkmann [4], propose a revocable multi authority CPABE scheme, where an efficient and secure revocation method is proposed to solve the attribute revocation problem in the system. In CP-ABE scheme, there is an authority that is responsible for attribute management and key distribution. Our attribute revocation method is efficient in the sense that it incurs less communication cost and computation cost, and is secure in the sense that it can achieve both backward security (The revoked user cannot decrypt any new cipher text that requires the revoked attribute to decrypt)and forward security (The newly joined user can also decrypt the previously published ciphertexts1, if it has sufficient. attributes). Our scheme does not require the server to be fully trusted, because the key update is enforced by each attribute authority not the server. Even if the server is not semi trusted in some scenarios, our scheme can still guarantee the backward security. Then, we apply our proposed

revocable multiauthority CP-ABE scheme as the underlying techniques to construct the expressive and secure data access control scheme for multi-authority cloud storage systems.

III. PROPOSED SYSTEM

In this paper propose is to provide a scheme guaranteeing semantic security for unpopular data (deduplication forbidden), and, transparently transitioning to convergent security offerings as soon as a file becomes popular (deduplication enabled). In this work first present the cryptosystem that forms the core of the proposed scheme. Next the author discuss the role of the identity provider IdP and index repository serviceThe main intuition behind the scheme is that there are scenarios in which data requires different degrees of protection that depend on how popular a datum is. Let us start with an example: imagine that a storage system is used by multiple users to perform full backups of their hard drives. The files that undergo backup can be divided into those uploaded by many users and those uploaded by one or very few users only.

Here propose a Threshold Convergent Cryptosystem $\mathcal{E}\mu\mathcal{E}_{\mu}$ is a special-purpose threshold cryptosystem that allows all users to encrypt arbitrary messages m of fixed length λ associated with some label ℓ in such a way that once enough (more than some threshold) of the users provide their decryption shares (created using the same label ℓ, all the messages associated with ℓ can be decrypted.

Encrypt interface now includes the added label ℓ and the decryption process is designed to be non-interactive. No interactivity requires modification of the DShare interface the decryption share is created using the label ℓ instead of using the ciphertext and is stored in some repository until required for decryption. For the purpose of this cryptosystem as well as in the remainder of this paper will make use of $\Lambda=\{\lambda,q,G_1,G_2,G_T g,\check{g},\hat{e},\}$ whereG_1,G_2,G_T are pairing groups satisfying the Symmetric external Diffie-Hellman (SXDH) assumption where$G1=\langle g\rangle, G_2=\langle\check{g}\rangle$, are of prime order q and $\hat{e}$: G1*G2 → G_T is an efficiently computable, non-degenerate bilinear pairing.

SXDH requires the decisional Diffie-Hellman problem (DDH) to be intractable in both G1 and G2. Bitsize of q is determined by the security parameter λ which corresponds to the bitlength of the exploited symmetric encryption scheme key (for 128-bit security one would set λ to 128 and bitsize of q two times larger i.e., $|q|=256$ it will also use two cryptographic hash functions. A semantically secure symmetric cryptosystem E and a convergent encryption scheme$\mathcal{E}_{c.}$

$\mathcal{E}_{\mu}$ Setup $(\lambda,n;t)$ → $(pk;sk,\{r_i,sk_i)\}^n_{i=1}$ as first generated and then the secret key is generated.

$\mathcal{E}_{\mu}$ Encrypt (pk, ℓ,m) → (c) and compose the cipher text c as c_1 and c_2.

$\mathcal{E}_{\mu}$Dshare $(r_i;sk_i; \ell)$ →$(r_i;ds_i)$to create a share key .

$\mathcal{E}_{\mu}$ Decrypt parse c as c_1 and c_2 . Using all decryption shares in S_t compute.

$\mathcal{E}_{\mu}$ has a few noteworthy properties:

i) The decryption algorithm is non-interactive, meaning that it does not require live participation of the entities that executed the $\mathcal{E}_{\mu}$:DShare algorithm.

ii) It mimics convergent encryption in that the decryption shares are deterministically dependent on the plaintext label. However, in contrast to plain convergent encryption, the cryptosystem provides semantic security as long as less than t decryption shares are collected.

iii) The cryptosystem can be reused for an arbitrary number of messages, i.e., the$\mathcal{E}_{\mu}$.Setup algorithm should only be executed once. Finally, note that it is possible to generate more shares $sk_j(j>n)$ any time after the execution of the $\mathcal{E}_{\mu}$. Setup algorithm, to allow new users to join the system even if all the original n key-shares Were already assigned.

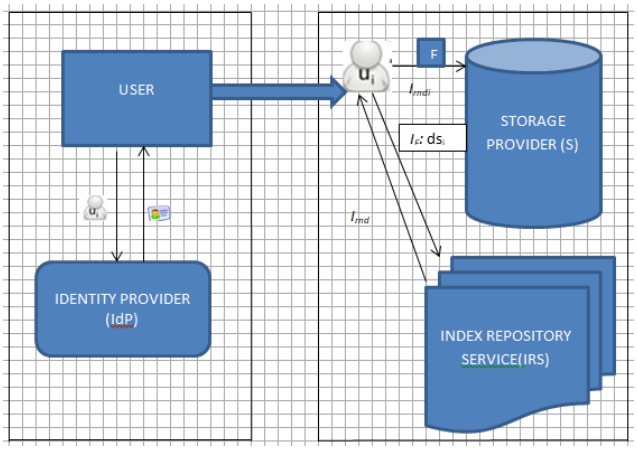

Fig 1 : Proposed Architecture

Proposed system consists of users, storage provider and two trusted entities, the identity provider, and the index repository service, as shown in Fig.1.The storage provider (S) offers basic storage services and can be instantiated by any storage provider (e.g., Bitcasa, Dropbox, etc.). Users U*i* € U own files and wish to make use of the storage provider to ensure persistent storage of their content. Users are identified via credentials issued by an identity provider IdP when a user first joins the system.File *F* is identified within S via a unique file identifier $I_F$of bitsize*k* ($|I_F| = k$)computed by a collision-resistantindexingfunction$I:\{0,1\}^* \rightarrow \{0,1\}^k$.$I_F$is issued by the index repository service IRS during the file upload process. The IRS also maintains a record of how many distinct users have uploaded a file.

IV. CONCLUSION

In this paper, implement a simple method, this work deals with the inherent tension between storage optimization methods and end-to-end encryption. In the previous work author introduced a novel approach that allows varying the security level of a file based on how popular that file is among the users of the system and presented an encryption scheme that implements this approach. In this work author modify and significantly enhance the originally proposed scheme, focusing on its practicality. Specifically, re-located storage of decryption shares and removal of file popularity information from the storage provider allowed us to strengthen the security of unpopular files and simplify the security proofs while modification of scheme internal processes led to decreased complexity and clear functionality isolation of individual algorithms. The resulting scheme prevents deduplication of unpopular data and allows their automatic transition to a popular state (once they are uploaded by enough users) where deduplication can be applied.

To evaluate space savings efficiency and the computational and communication overhead of the proposal, present an extensive performance evaluation using real datasets and real-like environment. To provide comparison to other state of the art secure deduplication schemes include their prototypes in the performance evaluation. Security-wise, the sceme is resilient to user-collusion attacks (up to a clearly defined point) and to an honest but curious storage provider. By automatically differentiating between popular and unpopular data, the scheme alleviates the user's need to handle low-entropy files differently (if their eventual deduplication is acceptable) and surpasses the other schemes in the fact that it never deduplicates unpopular files nor leaks whether there are any duplicates of unpopular files in the storage (unless the indexing server is compromised). These advantages come at the cost of lower space-saving efficiency compared to the other schemes

REFERENCES

[1] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage", *in Proc. 18th Int. Conf. Financial Cryptography Data ecur.*,pp: 99–118, 2014.

[2] D. Harnik, B. Pinkas, and A. Shulman - Peleg, "Side channels in cloud services:Deduplication in cloud storage", *IEEE Secur. Privacy,* vol. - 8, no. 6, pp : 40–47, Nov. /Dec. 2010.

[3] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Serveraided encryption for deduplicated storage", *in Proc. 22nd USENIX Secur. Symp.,* pp. 179–194, 2013.

[4] D. Meister and A. Brinkmann, "Multi-level comparison of data deduplication in a backup scenario", *in Proc. SYSTOR: Israeli Exp. Syst. Conf.*, Art. no. 8, 2009.

[5] N. Mandagere, P. Zhou, M. A. Smith, and S. Uttamchandani, "Demystifying data Deduplication", *in Proc. ACM/IFIP/USENIX Middleware Conf. Companion,*pp: 12–17, 2008.

