ISSN: 2321-7782 (Online)
ISSN: 2347-1778 (Print)
Impact Factor: 6.047

Volume 4, Issue 9, September 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Convergent Key Management in Hybrid Cloud

Pagadala Rajesh[1]
M.Tech Scholar
St.Marys group Of Institutions Guntur
Chebrole(V&M),Guntur(dt)
Andhra Pradesh – India

Subhani Shaik[2]
Associate Professor
St.Marys group Of Institutions Guntur
Chebrole(V&M),Guntur(dt)
Andhra Pradesh – India

Abstract: ***To protect confidentiality of the sensitive data while supporting de-duplication data is encrypted by the proposed convergent encryption technique before out sourcing. Problems authorized data duplication formally addressed by the first attempt of this paper for better protection of data security. This is different from the traditional duplication systems. The differential privileges of users are further considered in duplicate check besides the data itself. In hybrid cloud architecture authorized duplicate check supported by several new duplication constructions. Based on the definitions specified in the proposed security model, our scheme is secure. Proof of the concept implemented in this paper by conducting test-bed experiments. Convergent encryption [8] provides a viable option to enforce data confidentiality while realizing deduplication. It encrypts/decrypts a data copy with a convergent key, which is derived by computing the cryptographic hash value of the content of the data copy itself [8]. After key generation and data encryption, users retain the keys and send the ciphertext to the cloud. Since encryption is deterministic, identical data copies will generate the same convergent key and the same ciphertext.***

Keywords: hybrid cloud, authorized duplicate check, confidentiality, encryption.

I. INTRODUCTION

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 studies [3]. One critical challenge of today's cloud storage services is the management of the ever-increasing volume of data. According to the analysis report of IDC, the volume of data in the wild is expected to reach 40 trillion gigabytes in 2020 [9]. 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. From a user's perspective, 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 cipher texts, making deduplication impossible. However, the baseline approach suffers two critical deployment issues. First, it is inefficient, as it will generate an enormous number of keys with the increasing number of users. Specifically, each user must associate an encrypted convergent key with each block of its outsourced encrypted data copies, so as to later restore the data copies. Although different users may share the same data copies, they must have their own set of convergent keys so that no other users can access their files. As a result, the number of convergent keys being introduced linearly scales with the number of blocks being stored and the number of users.

II. Related Work

Convergent encryption [5], [8] 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. Here, we assume that the tag correctness property [5] holds, i.e., if two data copies are the same, then their tags are the same. 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. Formally, a convergent encryption scheme can be defined with four primitive functions:

Identification protocol: With two phases we can describe the identification protocol, **Proof** and **Verify.** In the stage of proof, a user U demonstrates his identity to a verifier by performing the some identification proof related to his identity. Private is the input of the prover/user i.e sensitive information such as private key of a public key in his certificate, credit card number, etc. These types of numbers cannot share with others. With the help of input of public information related to , the verifier perform the verification. At the end of protocol, the verifier output either accept or not to denote whether the proof is passed or not. Different types of identification protocols are there like, certificate based and identification based identification [5], [6]. This involves deleting data on clouds that have become slower and creating new data on ones that have become faster. Because this adaptation should be done frequently, new coded packets should be created without transferring a lot of data. However, the less data is transferred, the less information the new packets will contain. We propose a sparse scheme that strikes a good balance between the two. For creating performance enhancing packet number r on a cloud, we propose gathering only one critical packet from all other clouds, specifically number r counting from the end of the critical packets. The idea is to build performance enhancing packets from rarely used critical packets, thus lowering the probability of accessing linearly dependent packets during retrieval.

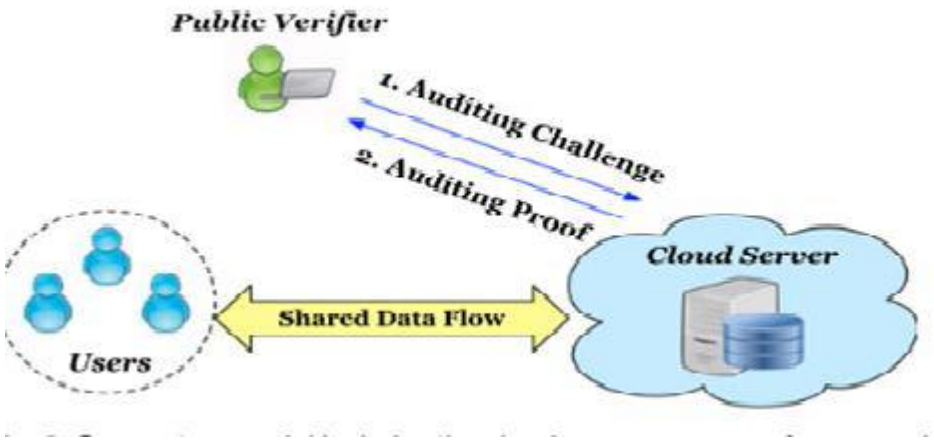

Fig: Architecture Diagram for Hybrid Cloud

III. System Model

3.1 Convergent key management with Hybrid Architecture:

By using the duplication technique, to store the data who will use S-CSP are consisted as group of affiliated client at high level. The main aim is enterprise all the network. To set the data back up and disaster recovery applications for reduce the storage space. We frequently go for de-duplication. Such systems are widespread and are often more suitable to user file backup and synchronization applications than richer storage abstractions.

3.2 Adversary Model:

Both the public and private clouds are "honest-but-curious" try to find out as much secret information as possible based on their possessions either within or out of the limits of privileges users would try to access data. In this one, all the files are sensitive and needed to be fully protected against both public and private cloud. Assume two kinds of adversaries are considered.

- External adversaries which aim to extract secret information as much as possible from both public cloud and private cloud.
- Internal adversaries who aim to obtain more information on the file from the public cloud and duplicate-check token information from the private cloud outside of their scopes.

3.3 Design Description:

The detailed architecture of the design is showed in figure2. We can get the processing details from this architecture. Four different types of modules are present in the architecture. Data Owner Module, Encryption and Decryption Module, Remote User Module, Cloud Server Module. User login details are required to upload or download a file and the details of modules mentioned below.

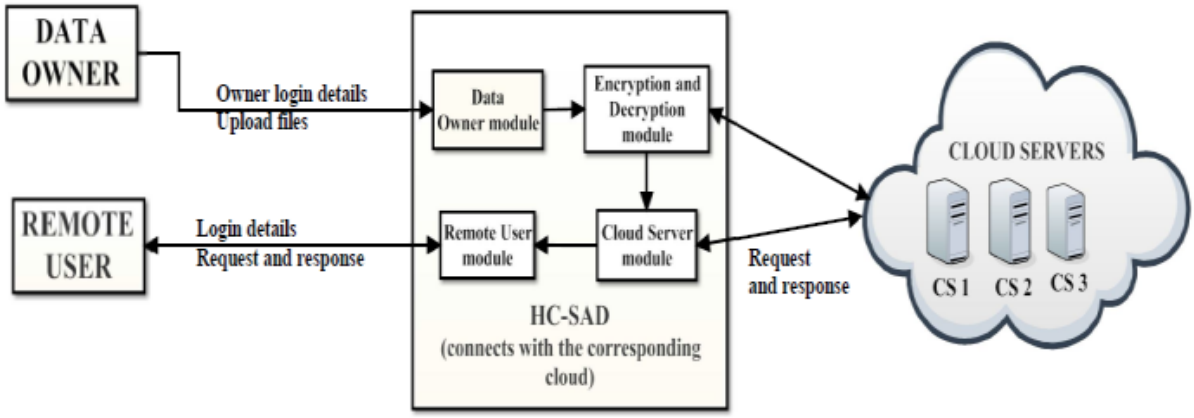

Fig: Control Flow on How the Data is passed to the sever

IV. METHODOLOGY

4. ALGORITHMS USED:

In this section, we use two types of algorithms,

1. For file uploading.
2. For file downloading.

FOR UPLOADING A FILE

BEGIN

Step –1 Read file

Step –2 Cloud server checks for duplication

Step –3 Sends duplication response whether the file already exists or not

Step – 4 If the file does not exist

4.1 Display "file does not exist"

Step – 5 Then it uploads the file

Step – 6 If the file already exist

6.1 Display "file already exist" END

FOR DOWNLOADING A FILE

BEGIN

Step –1 Read file

Step –2 Cloud server checks for duplication

Step –3 Sends duplication response whether the file already exists or not

Step –4 If the file exist -4.1 Display "file exist"

Step –5 then it downloads the file

Step –6 If the file does not exist

6.1 Display "file does not exist" END

V. IMPLEMENTATION

We first formulate a data outsourcing model used by Dekey. There are three entities, namely: the user, the storage cloud service provider (S-CSP), and the key management cloud service provider (KM-CSP), as elaborated below. User. A user is an entity that wants to outsource data storage to the S-CSP and access the data later. To save the upload bandwidth, the user only uploads unique data but does not upload any duplicate data, which may be owned by the same user or different users. S-CSP. The S-CSP provides the data outsourcing service and stores data on behalf of the users. To reduce the storage cost, the S-CSP eliminates the storage of redundant data via deduplication and keeps only unique data.KM-CSP. A KM-CSP maintains convergent keys for users, and provides users with minimal storage and computation services to facilitate key management. For fault tolerance of key management, we consider a quorum of KM-CSPs, each being an independent entity. Each convergent key is distributed across multiple KM-CSPs using RSSS. In this work, we refer a data copy to be either a whole file or a smaller-size block, and this leads to two types of deduplication:

1) file-level deduplication, which eliminates the storage of any redundant files, and 2) block-level deduplication, which divides a file into smaller fixed-size or variable-size blocks and eliminates the storage of any redundant blocks. Using fixed-size blocks simplifies the computations of block boundaries, while using variable size blocks (e.g., based on Rabin fingerprinting [22]) provides better deduplication efficiency. We deploy our deduplication mechanism in both file and block levels. Specifically, to upload a file, a user first performs the file level duplicate check. If the file is a duplicate, then all its blocks must be duplicates as well; otherwise, the user further performs the block-level duplicate check and identifies the unique blocks to be uploaded. Each data copy (i.e., a file or a block) is associated with a tag for the duplicate check.

5.1 RSSS with Pseudorandomness:

In Dekey, the RSSS secret is the hash key H0 of a data block B, where H0 ¼ hashðBÞ. Recall from Section 2 that the Share function of the ðn; k; rÞ-RSSS embeds r random pieces to achieve a confidentiality level of r. One challenge is that randomization conflicts with deduplication, since the random pieces cannot be deduplicated with each other. Instead of directly adopting RSSS, we here replace these random pieces with pseudorandom pieces in our Dekey implementation. We generate the r pseudorandom pieces as follows. Let m ¼ d r ðk_rÞe. We first generate m additional hash values as H1 ¼ hashðB þ 1Þ;H2¼ hashðB þ 2Þ; . . .;Hm¼hashðBþ mÞ. We then fill in the r pieces with the generated m additional hash values H1;H2; . . .;Hm. These r pieces are pseudorandom because 1. H1;H2; . . .;Hm cannot be guessed by attackers as long as the corresponding data

block B is unknown; and 2. H1;H2; . . .;Hm together with H0 cannot be deduced from each other as long as the corresponding data block B is unknown. The parameters n, k, and r determine the following four factors, whose effects are evaluated.

Confidentiality level: It is decided by the parameter r.

Reliability level: It depends on the parameters n and k, and can be defined by n _ k.

Storage blowup: It determines the key management overhead and depends on the parameters n, k, and r. It can be theoretically calculated by n k_r.

Performance: It refers to the encoding performance and decoding performance when using the k-of-n merasure code in the Share and Recover functions, respectively.

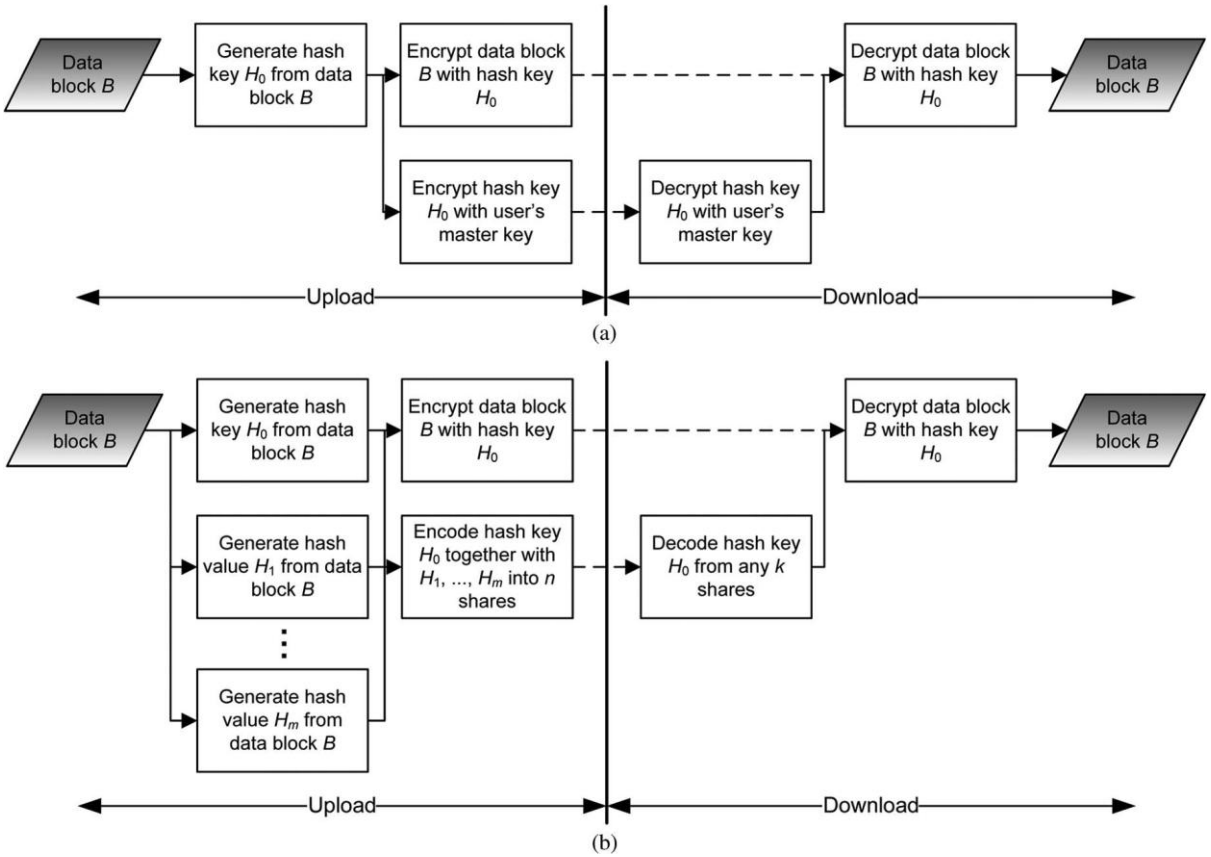

Fig: Key mapping For Uploading and Down loading

5.2 Implementation Details

Presents the flow block diagrams of core modules in the baseline approach and Dekey that we implement. In this figure, we omit the ordinary file transfer and deduplication modules for simplification. To make full use of the multi-core feature of contemporary processors, we assume that these modules running in parallel on different cores in a pipeline style. In the baseline approach, we simply encrypt each hash key H0 with the user's master key, while in Dekey, we generate n shares of H0. We choose 4 KB as the default data block size. A larger data block size (e.g., 8 KB instead of 4 KB) results in better encoding/decoding performance due to fewer chunks being managed, but has less storage reduction offered by deduplication [7], [15], [16], [29]. For each data block, a hash key of size 32 bytes is generated using the hash function SHA-256, which belongs to the family of SHA-2 that is now recommended by the US National Institute of Standards and Technology (NIST) [2]. In addition, we adopt the symmetric-key encryption algorithm AES-256 in Cipher-Block Chaining (CBC) mode as the default encryption algorithm. Both SHA-256 and AES-256 are implemented using the EVP library of OpenSSL Version 1.0.1e [1].

VI. EXPERIMENTAL RESULTS

The final results of the designed system are given below. From those results we get the detailed information to Check de-duplication and upload the files, Fetching the Signs using Hashing Algorithm, Checking for Duplication, file uploading, file downloading and attacker trying to attack(block) the cloud. Detailed procedure of the proposed system is given. Based on this we confirm that securely authorized de-duplication is successfully achieved with hybrid cloud approach. The output images given as below.

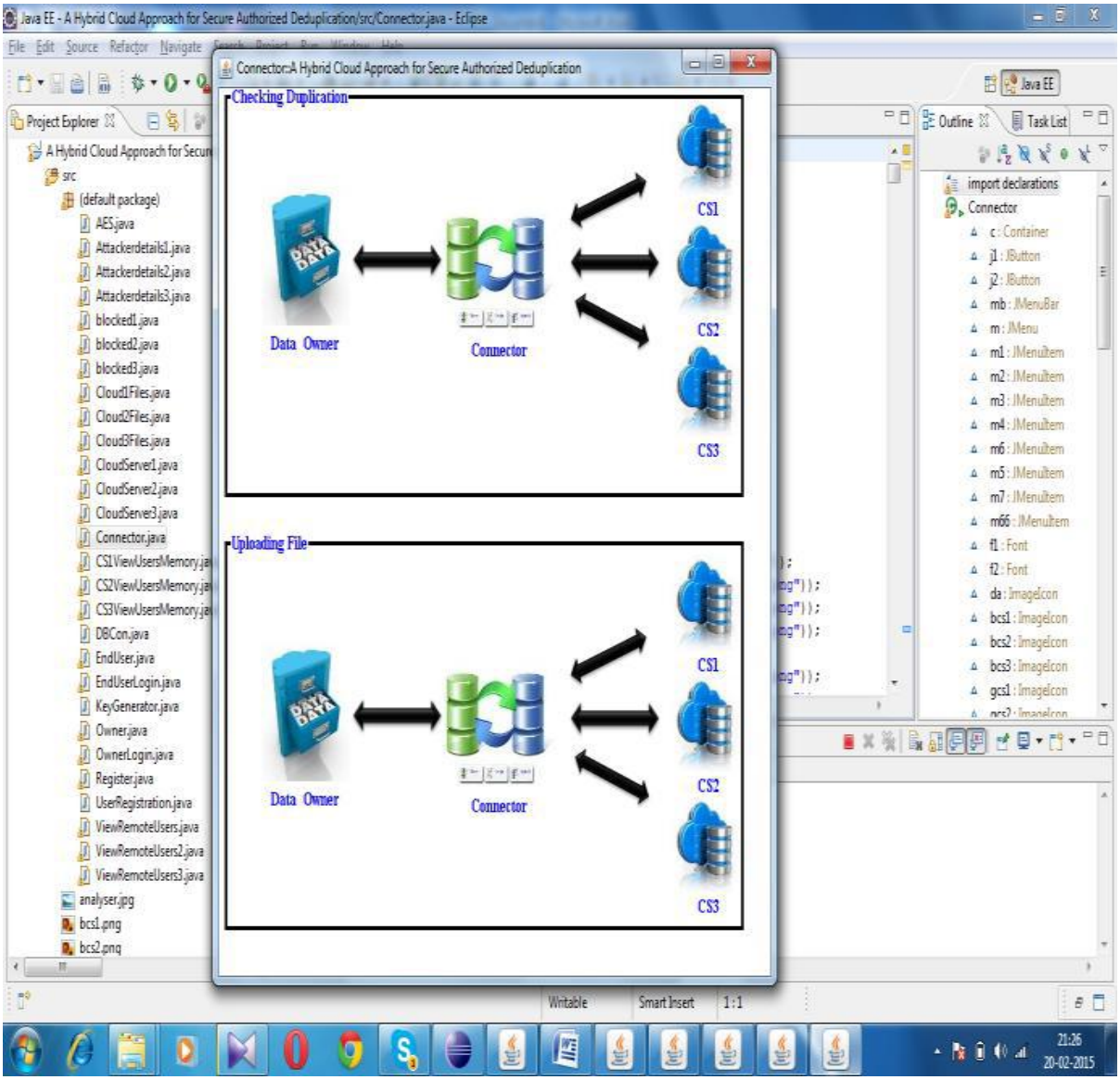

Fig: Key generation for uploading and downloading

VII. CONCLUSION

We propose Dekey, an efficient and reliable convergent key management scheme for secure deduplication. Dekey applies deduplication among convergent keys and distributes convergent key shares across multiple key servers, while preserving semantic security of convergent keys and confidentiality of outsourced data. We implement Dekey using the Ramp secret sharing scheme and demonstrate that it incurs small encoding/decoding overhead compared to the network transmission overhead in the regular upload/download operations. Notion of authorized data de-duplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new de-duplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, we implemented a prototype of

our proposed authorized duplicate check scheme and conduct test-bed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

References

1. Open SSL Project. [Online]. Available: http://www.openssl.org/.
2. NIST's Policy on Hash Functions, Sept. 2012. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/policy.html.
3. Amazon Case Studies. [Online]. Available: https://aws.amazon. com/solutions/case-studies/#backup.
4. P. Anderson and L. Zhang, ''Fast and Secure Laptop Backups with Encrypted De-Duplication,'' in Proc. USENIX LISA, 2010, pp. 1-8.
5. M. Bellare, S. Keelveedhi, and T. Ristenpart, ''Message-Locked Encryption and Secure Deduplication,'' in Proc. IACR Cryptology ePrint Archive, 2012, pp. 296-3122012:631.
6. G.R. Blakley and C. Meadows, ''Security of Ramp Schemes,'' in Proc. Adv. CRYPTO, vol. 196, Lecture Notes in Computer Science, G.R. Blakley and D. Chaum, Eds., 1985, pp. 242-268.
7. A.T. Clements, I. Ahmad, M. Vilayannur, and J. Li, ''Decentralized Deduplication in San Cluster File Systems,'' in Proc. USENIX ATC, 2009, p. 8.
8. J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M. Theimer, ''Reclaiming Space from Duplicate Files in a Serverless Distributed File System,'' in Proc. ICDCS, 2002, pp. 617-624.
9. J. Gantz and D. Reinsel, The Digital Universe in 2020: Big Data, Bigger Digital Shadows, Biggest Growth in the Far East, Dec. 2012. [Online]. Available: http://www.emc.com/collateral/analystreports/ idc-the-digital-universe-in-2020.pdf.
10. R. Geambasu, T. Kohno, A. Levy, and H.M. Levy, ''Vanish: Increasing Data Privacy with Self-Destructing Data,'' in Proc. USENIX Security Symp., Aug. 2009, pp. 316-299.
11. S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, ''Proofs of Ownership in Remote Storage Systems,'' in Proc. ACM Conf. Comput. Commun. Security, Y. Chen, G. Danezis, and V. Shmatikov, Eds., 2011, pp. 491-500.
12. D. Harnik, B. Pinkas, and A. Shulman-Peleg, ''Side Channels in Cloud Services: Deduplication in Cloud Storage,'' IEEE Security Privacy, vol. 8, no. 6, pp. 40-47, Nov./Dec. 2010.
13. Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, Wenjing Lou" A Hybrid Cloud Approach for Secure Authorized De-duplication" in vol: pp no-99, IEEE, 2014 [2] OpenSSL Project. http://www.openssl.org/.
14. P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.
15. M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In USENIX Security Symposium, 2013.
16. M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure eduplication. In EUROCRYPT, pages 296– 312, 2013.
17. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. J. Cryptology, 22(1):1–61, 2009.

