Safe De-duplication with Capable and Reliable Convergent Key Organization

[1]C.Mani, N.Arthi[2]
Associate professor, Final year student, Master of computer Application, Department of computer application, Nandha engineering college (Anna University), Erode-52, Tamilnadu, India.

Abstract — *Data reduplication is a process for eliminating imitation print of statistics, and has been extensively used in cloud storeroom to lessen storage room and upload bandwidth. Although convergent encryption has been far adopted for secure reduplication, a vital issue of making convergent encryption practical is to powerful and constantly run a huge quantity of convergent keys. We first begin a baseline advance in which each client holds an independent master key for encrypting the convergent keys and outsourcing them to the cloud. Though, such a baseline key group method generate an huge number of keys with the high number of users and fulfil users to dedicatedly guard the master keys. To this end, we plan De key, a new construction in which users do not need to handle any keys on their personal but as a substitute strongly dispense the convergent key shares across several servers. Security analysis demonstrates that De key is safe in terms of the definition particular in the future security model. As a evidence of concept, we apply De-key using the Ramp secret sharing scheme and reveal that De-key incurs partial overhead in practical environments.*

Keywords — *Reduplication, proof of rights, convergent encryption, key management*

1. INTRODUCTION

The initiation of cloud storeroom motivate enterprise and organization to bond out data storage to third-party cloud provider, as evidence by many real-life case studies [3].To make data model scalable, reduplication has been a familiar method to cut storage gap and upload bandwidth in cloud storeroom. Instead of observance many data copies with the same content, reduplication eliminate superfluous data by keeping only one objective copy and referring other disused facts to that copy. Each such copy can be distinct based on dissimilar granularities: it may transfer to either a total file (i.e., file-level reduplication), or a more fine-grained fixed-size or variable-size data block (i.e., block-level reduplication).Today's trade shade storeroom forces, such as Drop-box, Mozy , and Memopal, have been applied reduplication to addict facts to keep upholding fee [12].

From a user's view, facts outsourcing raise safety and privacy concern. We must trust third-party cloud provider to right impose privacy, truth checking, and contact run mechanism beside some incast and outcast attack. Thus, identical data copies of dissimilar user will guide to different code text, make reduplication impractical.

Convergent encryption [8] provide a feasible choice to impose data secrecy while realize reduplication. 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]. That is, 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 equivalent encrypted facts copies, in cloud storage. The master type can be used to improve the encrypted keys and thus the encrypted records. In this way, each client only wants to be the master key and the metadata about the outsourced data.

However, the baseline advance suffer two vital employment issue. Earliest, it is incompetent, as it will cause an vast quantity of key with the rising quantity of user. Exclusively, each user must connect an encrypted convergent key with all mass of its outsourced encrypted data copies, so as to soon return the data copies. Even though many users may allocate the equal facts copy, they must have their hold set of convergent keys so that no other user can right to use their records. For model, assume that a client supplies 1 TB of facts with all sole block of mass 4 KB every, and that each convergent key is the hash rate of SHA-256, which is used by Drop box for reduplication [17]. Then the total size of keys will be 8 GB. The quantity of key is further multiply by the quantity of users. The resultant severe key organization overhead leads to the huge storage fee, as users must be owed for storing the great quantity of keys in the cloud below the pay-as-you-go model.

Second, the baseline advance is erratic, as it need each user to dedicatedly be his own master key. If the master type key is by fault gone, then the user data cannot be better; if it is compromised by attackers, then the user data will be leaked. To this conclusion, we direct a new manufacture called De-key, which provide competence and reliability guarantees for convergent key organization on both user and cloud

storage side. Our plan is to apply de-duplication to the convergent keys and force secret attachment method.. To improve data copies, a user must contact a least number of key servers through confirmation and acquire the secret shares to rebuild the convergent keys.. This significantly reduce the storage overhead of the convergent keys and make the key management reliable beside failures and attacks. To our facts, none of alive studies officially address the trouble of convergent key management.

2 . PRELIMINARIES

In this section, we formally define the cryptographic primitives used in our secure de-duplication.

.

2.1 *Symmetric Encryption*

Symmetric encryption uses a general secret key _ to encrypt and decrypt information. A symmetric encryption scheme consists of three primitive functions:

- $KeyGen_{SE}$ð1–Þ! is the key production algorithm that generate _ using safety parameter 1–;
- $Encrypt_{SE}$ð_; MÞ! C is the symmetric encryption algorithm that takes the furtive and memo M and then output the ciphertext C;
- $Decrypt_{SE}$ð_; C Þ! M is the symmetric decryption algorithm that takes the covert and ciphertext C and then outputs the unique message M.

2.2 Encryption Convergent

Convergent encryption [5], [8] provide facts privacy in de-duplication. A client derives a convergent key from each unique data replica and encrypts the facts copy with the convergent key. In addition, the client derives a tag for the data copy, such that the tag will be used to detect duplicates. Now, we guess that the tag exactness goods [5] hold, i.e., if two facts copies are the equal, 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 privacy. Equally the encrypted data copy and its equivalent tag will be stored on the server side.

2.3 Proof of Ownership

The view of resistant of rights (PoW) is to explain the trouble of a tiny confusion worth as a deputy for the full file in client-side reduplication [11], where the opponent might use the storage check as a comfortable sharing network. This shows mechanism in PoW gives a solution to keep the safety in client-side reduplication. In this mode, a client can establish the server that it really has the file. Dekey supports client-side reduplication with PoW to permit users to verify their control of data copies to the storage server. Purposely**,** PoW is accepted as an interactive algorithm run by a prover and a verifier (i.e. storage server).The verifier derives a small charge_ðMÞ from a data copy M. To confirm the privileges of the data copy M, the prover needs to post $_^{0}$ and run a evidence algorithm with the verifier. It is agreed if and only if $_^{0}$ ¼ _ðMÞ and the proof is exact. Specifically, the notation of $PoW_{F;j}$ will be used to denote a PoW protocol with respect to T_jðF Þ ¼ $TagGen_{CE}$ðF; jÞ.

2.3 Ramp Secret Sharing

Dekey use the Ramp secret sharing scheme (RSSS) [6], [25] to shop convergent keys. Specially, the ðn; k; rÞ-RSSS generates n shares from a secret such that 1) the secret can be recovered from any k shares but cannot be improved from less than k shares, and 2) no data about the secret can be deduced from any r shares. It is identified that when r ¼ 0, the ðn; k;0 Þ-RSSS become the ðn; kÞ Rabin's Information Dispersal Algorithm (IDA) [23]; when r ¼ k _ 1, the ðn; k; k _ 1 Þ-RSSS becomes the ðn; kÞ Shamir's Secret Sharing Scheme (SSSS) [26]. The ðn; k; rÞ-RSSS build on two ancient functions.

- Share divides a secret S into ðk_rÞ part of equal mass, generates r casual pieces of the same size, and encodes the k pieces with non-systematic k-of-n removal code into n shares of identical size;
- Recover takes any k out of n shares as inputs and then output the unique covert S.

To create the generated share correct for reduplication, we return the above chance piece with pseudorandom piece in the performance of Dekey.

3. PROBLEM FORMULATION

3.1 System Model

We initially plan a data outsourcing copy used by Dekey. There are three entity, namely: the user, the storage cloud service provider (S-CSP), and the key-management cloud service provider (KM-CSP), as elaborated below.

>User- A client is an entity that needs to subcontract data storage to the S-CSP and access the data soon. To keep the upload bandwidth, the client only uploads sole data but do not upload any spare data, which may be owned by the same user or diverse users.

>S-CSP- The S-CSP provides the data outsourcing facility and supplies data on behalf of the users. To decrease the storage cost, the S-CSP eliminate the storage of redundant data via reduplication and keeps merely single facts.

>KM-CSP- A KM-CSP covers convergent keys for clients, and helps users with minimal storage and addition forces to make easy key management.

For mistake tolerance of key management, we believe

a quorum of KM-CSPs, each initiate with a self entity. Each convergent key is spreaded across many KM-CSPs using RSSS. In this effort, we refer a data copy to be either a total file or a tiny-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 shares a file into small fixed-size and eliminates the storage of any disused blocks. Using fixed-size blocks simplify the computation of block limits, while using variable-size blocks provides best deduplication competence. We organize our deduplication mechanism in both file and block levels. Specially, to upload a file, a client performs the file-level duplicate check. If the file is a duplicate, then all its blocks must be duplicates as well; or else, the user add performs to the block-level duplicate verify and check the unique blocks to be uploaded. Each data copy is connected with a tag for the spare check .All data copies and tags will be stored in the S-CSP.

3.2 Threat Model and Security Goals

Our hazard form considers two types of attackers:
1) An outer foe may obtain some facts of the data copy of interest via public channels. It plays a task of a client that interacts with the S-CSP. 2) An indoor attacker is honest-but-curious, and it is referred by S-CSP or any of the KM-CSPs. It's aim is to remove helpful data or convergent keys. We need the inner attacker to follow the protocol perfectly .Here, we agree to the approval among the S-CSP and KM-CSPs. However, we require that the quantity of colluded KM-CSPs is not other than a predefined threshold r if the ðn; k; rÞ-RSSS is used, such that a convergent key cannot be guessed for an unpredicted note by a power attack from the colluded KM-CSPs.

We aim to achieve the following security goals:

>Semantic safe of convergent keys- We need the convergent keys distributed stored among the KM-CSPs stay semantically safe, still if the opponent controls a predefined quantity of KM-CSPs. In addition, these KM-CSPs are also approved to collude with S-CSP and users. The goal of the rival is to recover and improve the convergent keys for files that do not go to them.

>Data confidentiality- We need that the encrypted data copies be semantically safe when they are irregular. In fact this must has freshly been formal in [5] and called the privacy beside selected supply attack. This also imply that the data is safe beside the opponent who does not own the data. That is, the user cannot get the possession of the data from the S-CSP and KM-CSPs by running the PoW protocol if the user does not have the file.

4. CONSTRUCTIONS

In this part , we present a baseline advance that realize convergent encryption in deduplication, and consider the limits of the baseline approach in key management. To this end, we present our creation Dekey, which aims to mitigate the key management overhead and provide fault tolerance guarantees for key management, while preserving the required security properties of secure deduplication.

4.1. Baseline Approach

The baseline approach involves only the user and the S-CSP (i.e., no KM-CSPs are required). Its idea is that each user has all his data copies encrypted by the corresponding convergent keys, which are then further encrypted by an independent master key. The encrypted convergent keys are outsourced to the S-CSP, while the master key is securely maintained by the user.

4.1.1System Setup

The scheme setup stage initializes the necessary parameters in the following two steps:

> S1: The following entity are started:1) a symmetric encryption plan with the primitive functions ðKeyGen$_{SE}$; Encrypt$_{SE}$; Decrypt$_{SE}$Þ and the user's master key _ ¼ KeyGen$_{SE}$ð1–Þ for some security parameter 1–; 2) a convergent encryption scheme with the primitive functions ðKeyGen$_{CE}$; Encrypt$_{CE}$; Decrypt$_{CE}$; TagGen$_{CE}$Þ; and 3) a PoW algorithm PoW$_F$ for the file and a PoW algorithm for the block, which is denoted by Po W$_B$.

>S2: The S-CSP initializes two types of storage system: a fast storage plan for storing the tags for capable copy checks, and a case storage system for storing equally encrypted data copies and encrypted convergent keys.

4.1.2 Uploading File

Suppose that a user uploads a file F. Foremost, it performs file-level reduplication as follows.

- S1 - On effort file F , the client computes and sends the file tag TðF Þ ¼ TagGen$_{CE}$ðF Þ to the S-CSP.
- S2- Upon receiving TðF Þ, the S-CSP checks whether there exists the parallel tag on the S-CSP .Thus, the S-CSP replies the client with a reply ''file duplicate,'' or ''no file duplicate'' otherwise.
- S3- If the client receives the reaction ''no file duplicate'', then it jumps to S5 to carry on with block level reduplication. If the reply is ''file duplicate,'' then the user runs PoW$_F$ on

F with the S-CSP to show that it really owns the identical file F that is stored on the S-CSP.

- S4- If Po W_F is accepted, the S-CSP merely returns a file pointer of F to the client, and no more information will be uploaded. If PoW_F fails, the S-CSP aborts the upload process. The client then performs block-level reduplication to further reduce any disused block, as described below.

- S5- On input file F and the master key _, the user performs the next computations: 1) Divide F into a group of blocks fB_ig (where i ¼ 1; 2; . . .); 2) for each block B_i, total block tag $TðB_iÞ$ ¼ $TagGen_{CE}ðB_i$ Þ; and 3) Send the set of block tags fT $ðB_iÞg$ to the S-CSP for fake checks.

- S6: Ahead receiving block tags fT $ðB_iÞg$, the S-CSP computes a mass signal vector $_B$ in the following manner: For all i, if there exists diverse stored block tag that matches T $ðB_iÞ$, the S-CSP sets $_B½i& ¼ 1$ to indicate ‘‘block duplicate’’; or else it sets $_B½i& ¼ 0$ to indicate ‘‘no block duplicate’’ and also stores T $ðB_iÞ$ in its rapid storage. Then, the S-CSP returns $_B$ to the user.

- S7: Up on delivery the signal $_B$, the client performs the next operations: for each i, if $_B½i& ¼ 1$, the user runs PoW_B on B_i with the S-CSP to show that it owns the block B_i. If it is passed, the S-CSP simply returns a block pointer of B_i to the user. Next, the user keeps the block pointer of B_i and do not want to upload B_i; otherwise it computes the encrypted block C_i ¼ .

- S8- For all blocks f B_ig, the user also computes the encrypted convergent keys fCK_ig,where CK_i ¼ $Encrypt_{SE}ð_;$ $K_iÞ$ with the master key and convergent key K_i.

- S9- The user uploads the unique blocks B_i’s with $_B½i& ¼ 0$, all encrypted convergent keys fCK_i g and T ðF Þ to the S-CSP, which then supplies them in the file storage system.

4.1.3 Case Download

Assume a client needs to a file F . It initially sends a request and the file name to the S-CSP and performs the subsequent steps.

- S1- If failed, the S-CSP sends back an end signal to the user to specify the download failure. Otherwise, the S-CSP returns the corresponding ciphertexts fC_ig and the encrypted convergent keys fCK_ig to the client.

- S2: Upon delivery the encrypted data from the S-CSP, the user first uses master key to recover each convergent key. Then it uses Ki to recover the original block B_i ¼ $Decrypt_{CE}ðK_i;$ $C_iÞ$. Finally, the user can obtain the original file F ¼ fB_ig.

4.1.4Boundaries

The baseline advance suffers two foremost trouble. The first crisis is the vast storage slide in key organization. In exacting, each user should link a convergent key with every data copy that he owns and encrypts all convergent keys with his own master key. The encrypted convergent keys are diverse across users due to the dissimilar master keys. Thus, the quantity of convergent keys raise linearly with the unique data copies being stored and the quantity of users, so imposing heavy storage overhead. A extra trouble is that the master key presents the Sole point-of-failure and wants to be securely and constantly maintained by the client.

4.2 Dekey

Dekey is planned to efficiently and consistently maintain convergent keys. Its plan is to enable reduplication in convergent keys and issue the convergent keys across many KM-CSPs. Instead of encrypting the convergent keys on a per-user basis, Dekey constructs secret shares on the original convergent keys and distributes the shares across many KM- CSPs. If several user share the same block, they can access the related convergent key. This reduces the storage overhead for convergent keys. Now elaborate Dekey details as follows.

4.2.1. System Setup

The system setup phase in Dekey is alike to that in the baseline approach, but an additional step for starting the key storage in KM-CSPs. In Dekey, we assume that the number of KM-CSPs is n.

>S1-On enter security parameter 1–, the user initializes a convergent encryption scheme, and two PoW protocols POW_F and POW_B for the file rights verification and block ownership evidence, correspondingly.

>S2- The S-CSP initializes both the fast storage system and the file storage system .

>S3- Every KM-CSP initializes a rapid

storage system for block tags and a trivial storage system for holding convergent key shares .

4.2.2.Uploading File

To upload file F , the user and the S-CSP achieve both file-level and block-level reduplications. The file level reduplication process is equal to the baseline approach. Specifically, the user sends the file tag TðF Þ to the S-CSP for the file copy check. If a file duplicate is found, the client will run the PoW protocol POW_F with the S-CSP to show the file rights. It skips the block level duplicate check and jumps to the key supply stage. If no duplicate exists, then block-level reduplication will be performed as equal as S5-S7 of the baseline method. Finally, the S-CSP stores the ciphertext C_i with _B½i& ¼ 0 and proceeds equivalent pointers back to user for local storage.

Once both file-level and block-level copy checks, an spare phase called key sharing is performed. As reverse to the baseline approach, this step enables Dekey for not keeping a master secret key for all user, but instead split each convergent key among many KM-CSPs. If a file print is found S-CSP, the user will run the PoW protocol $POW_{F;j}$ for the tag T_jðF Þ ¼ $TagGen_{CE}$ðF; jÞ with j-th KM-CSP to prove the file rights. All the pointers for key shares of F stored on the j-th KM-CSP will be returned to the user if the proof is agreed. If no file duplicate is found, the subsequent steps will be taken.

>S1- On input file F ¼ fB_ig, for all block B_i, the user computes and sends the block tag T ðB$_i$Þ ¼ $TagGen_{CE}$ðB$_i$Þ to each KM-CSP.
>S2- For each received T ðB$_i$Þ, the j-th KM-CSP checks whether an extra same tag has been stored: if so, a PoW for block $POW_{B;j}$ will be performed between the user and j-th KM-CSP with respect to T_jðB$_i$ Þ ¼ $TagGen_{CE}$ðB$_i$; jÞ. If it is passed, the j-th KM-CSP will return a pointer for the secret share stored for the convergent key Ki to the user or else it keeps TðBiÞ and sends back a sign for the safe share on the convergent key.
>S3- Upon receiving results for a block B_i returned from KM-CSPs, if it is a legal pointer, the client stores it locally, otherwise the secret shares K_{i1}; K_{i2}; K_{ik} by running ShareðK$_i$Þ using the on k; rÞ- RSSS. It sends the share K_{ij} and T_jðB$_i$Þ ¼ $TagGen_{CE}$ðB$_i$; jÞ to j-th KM-CSP for j ¼ 1,2..; n >S4- Upon receiving K_{ij} and T_jðB$_i$Þ, the j-th KM-CSP stores them and sends back the pointer for K_{ij} to the user for future access.

4.2.3 File Download

To download file F, the user downloads the encrypted blocks fC_ig from S-CSP as described in the baseline method.. After gathering all the shares, the user continues to redo the convergent key K_i ¼ Recover ðfK_{ij}gÞ for B_i. Finally, the encrypted blocks fC_ig can be decrypted with fK_ig to obtain the original file F .

4.3 Safety Analysis

Dekey is designed to solve the key organization problem in safe reduplication where the files have been encrypted by utilizing convergent encryption. Some vital tool have been used to create the safe reduplication and key organization protocols. So, we guess that the basic building blocks are safe , which contain the convergent encryption scheme, symmetric encryption scheme, and the PoW system. Based on this statement, we prove that Dekey is secure with respect to the safety of keys and data, as detailed below.

4.3.1Discretion of Outsourced Data at S-CSP

The files have been fixed by the convergent encryption plan before being outsourced to the S-CSP. Thus, the privacy of data can be achieved if the adversary cannot get the secret keys in convergent encryption or crack the safety convergent encryption. Some safety notions, for example, privacy beside chosen sharing attack, have been defined for the confidentiality. So ,our construction can also reach the safety for data based on a safe convergent encryption scheme if the encryption key is securely kept by the client.

4.3.2 Safety of Convergent Encryption Key

In our creation, the convergent encryption keys are strongly stored at the KM-CSPs in a dispersed way. Thus, the semantic refuge of convergent keys could be assured even if any r KM-CSPs collude. This could be easily achieved because RSSS is a semantically safe secret part method even if any r of n shares have been leaked. Remind that it requires the user to make a PoW protocol for the equivalent shares stored at diverse KM-CSPs. We want that the ideals used in PoW with various KM-CSPs are sovereign and the rival cannot compute the other tiny value even if he has the awareness of r values T_jðB$_i$Þ. Really, in our execution, T_jðB$_i$Þ is implemented with a cryptographic hash function H_jð _Þ and the over statement will be held clearly. In this way, if the adversary wants to get the j-th key share that it does not have, then he has to convince the j-th KM-CSP by running a PoW protocol. But, the values used to perform PoW with diverse KM-CSPs are dissimilar and the adversary cannot gain the other key shares even if he could obtain r share from dishonest KM-CSPs forced by him.

5. Functioning

In this part, we discuss the implementation details of Dekey. Dekey builds on the Ramp secret sharing scheme (RSSS) [6], [25] to distribute the shares of convergent keys across multiple key servers (see Section 2).

5.1.1RSSS with Pseudo randomness

In Dekey, the RSSS secret is the hash key H_0 of a data block B, where H_0 ¼ hashðBÞ. Remind from part 2 that the Share role of the ðn; k; rÞ-RSSS embeds r random pieces to achieve a confidentiality level of r. One dispute is that randomization conflicts with reduplication, as the casual pieces cannot be deduplicated with each other. Instead of straight adopting RSSS, we return these random pieces with pseudorandom pieces in our Dekey performance.

We generate the r pseudorandom pieces as follows. Let m ¼ $d_{ðk\_r}$ ^{r} $_{rÞ}$e. We first generate m additional hash values as H_1 ¼ hashðB þ 1Þ; H_2 ¼ hashðB þ 2Þ;; H_m ¼hashðB þ mÞ. We then fill in the r pieces with the generated m additional hash values H_1; H_2; . . . ; H_m.

These r pieces are pseudoran-dom because

1. H_1; H_2; . . . ; H_m cannot be guessed by attackers as lengthy as the equivalent data block B is unknown; and
2. H_1; H_2; . . . ; H_m jointly with H_0 cannot be deduced from each other as elongated as the parallel data block B is unknown. Implementation Details

To build full use of the multicore aspect of modern processors, we suppose that these modules running in similar on different cores in a pipeline style. In the baseline approach, we just encrypt each hash key $_0$ with the user's master key, while in Dekey, We prefer 4 KB as the evasion data block size. A larger data block size(e.g., 8 KB) results in better encoding/decoding act due to less chunks being managed, but has less storage reduction offered by reduplication [7], [15], [16], [29]. In count, we accept 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 store of OpenSSLVersion1.0.1e [1].

We relate the RSSS based on Jerasure Version 1.2 [20]. Concerning to the encoding and decoding modules in Fig. 1b, the choice of code symbol size w (in bits) deserves our discussion here. For an erasure system, a code symbol of size w bits refers to a vital part of encoding and decoding operations, both of which are performed in a finite field GF ð2^w Þ.. We thus often need to pay additional zeros to fill in the ðk _ rÞ pieces, resultant in different storage blow up ratios. We spot that for some w, the storage blow up ratio can be much higher than the theoretical value calculated by $_{ðk}$ ^{n} $_{rÞ}$. However, we find that if the minimum w is chosen, the practical storage blow up can often be strictly matched to the notional value. We locate that the encoding and decoding times raise with w. So, our Dekey implementation always chooses the minimum w that meets 2^w _ n *þ k*.

6. Overall Results

With ðn k; rÞ-RSS being used, we trial subsequent cases-3 _ n _ 8, 2 _ k _ n _ 1, and 1 _ r _ k _ 1, as exposed in Fig. 3. We can see that the encoding and decoding times of Dekey for each hash key are forever on the order of microseconds ,and hence are negligible compared to the data shift presentation in the Internet situation. Note that the encoding time in universal is higher than the decoding time, mainly as the encoding operation involves all n shares, while the decoding operation only involve a separation of k G n shares.

Specifically, during the file upload, the encoding time of Dekey is less than 20 usec in most cases, and is less than that of encrypting a data block .If we parallelize both the encoding and encryption modules, then the bottle-neck lies in the encryption part. Through the file download, Number of KM-CSPs n

6.1Confidentiality Level r

The encoding/decoding times versus the confidentiality level r, where we fix the number of KM-CSPs n ¼ 6 and the reliability level n _ k ¼ 2. The encoding/decoding times increase with r. Recall that the Share function of RSSS divides a secret into k _ r equal-size pieces (see Section 2). With a high r, the size of each piece raises, and this rise the encoding/decoding over-head as well. confidentiality level r ¼ 2. With the increase of n _ k, the encoding/decoding times decrease since fewer pieces (i.e., k) are being erasure-coded.

7. Related work

7.1 ***Traditional Encryption***

To protect the privacy of outsourced data, variety of cryptographic solutions have been future in the literature. Their plan builds on fixed encryption, in which each user encrypts data with an free safe key. Some studies [10],[28] propose to use threshold safe sharing [26] to keep the robustness of key management. However, the above studies do not consider reduplication. Using usual encryption, diverse users will simply encrypt equal data copies with their own keys, but this will lead to diverse ciphertexts and hence make reduplication impossible.

7.2 Convergent Encryption

Convergent encryption [8] ensures data privacy in reduplication. Bellare et al [5] formalize this primitive as message-locked encryption, and learn its application in space-efficient safe outsourced storage. There are also many implementations of convergent implementations of various convergent encryption variants for secure reduplication. It is called as commercial cloud storage providers, such as Bitcasa, also organize convergent encryption [5]. Though, as fixed before, convergent encryption lead to a major quality of convergent keys.

7.3 Proof of Ownership

Halevi et al. [11] offer ''proofs of ownership'' (PoW) for reduplication systems, such that a user can well prove to the cloud storage server that he/she owns a file without uploading the file itself. Some PoW constructions based on the Merkle Hash Tree are proposed [11] to enable

client-side reduplication, which comprise the bounded leakage setting. Pietro and Sorniotti [19] propose more competent PoW plan by choosing the projection of a file onto some randomly certain bit-positions as the file proof.

REFERENCES

[1] OpenSSL Project. [Online]. Available: http://www.openssl.org.

[2] NIST's Policy on Hash Functions, Sept. 2012. [Online]. Available:

[3] AmazonCase 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, ''Decen-tralized 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 Distrib-uted 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/analyst-reports/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] S. Kamara and K. Lauter, ''Cryptographic Cloud Storage,'' in Proc. Financial Cryptography: Workshop Real-Life Cryptograph. Protocols Standardization, 2010, pp. 136-149.

[14] M. Li, ''On the Confidentiality of Information Dispersal Algo-rithms and their Erasure Codes,'' in Proc. CoRR, 2012, pp. 1-4abs/ 1206.4123.

[15] D. Meister and A. Brinkmann, ''Multi-Level Comparison of Data Deduplication in a Backup Scenario,'' in Proc. SYSTOR, 2009, 1-12.

[16] D.T. Meyer and W.J. Bolosky, ''A Study of Practical Deduplica-tion,'' in Proc. 9th USENIX Conf. FAST, 2011, pp. 1-13.

[17] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl, ''Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space,'' in Proc. USENIX Security, 2011, p. 5.

[18] W.K. Ng, Y. Wen, and H. Zhu, ''Private Data Deduplication Protocols in Cloud Storage,'' in Proc. 27th Annu. ACM Symp. Appl. Comput., S. Ossowski and P. Lecca, Eds., 2012,
pp. 441-446.

[19] R.D. Pietro and A. Sorniotti, ''Boosting Efficiency and Security in Proof of Ownership for Deduplication,'' in Proc. ACM Symp. Inf., Comput. Commun. Security, H.Y. Youm and Y. Won, Eds., 2012,

[20] J.S. Plank and L. Xu, ''Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications,'' in Proc. 5th IEEE Int'l Symp. NCA, Cambridge, MA, USA, July 2006, 173-180.

[21] M.O. Rabin, ''Fingerprinting by Random Polynomials,'' Center for Research in Computing Technology, Harvard University, Cambridge, MA, USA, Tech. Rep. TR-CSE-03-01, 1981.

[22] M.O. Rabin, ''Efficient Dispersal of Information for Security, Load Balancing, Fault Tolerance,'' J. ACM, vol. 36, no. 2, pp. 335-348, Apr. 1989.

[23] A. Rahumed, H.C.H. Chen, Y. Tang, P.P.C. Lee, and J.C.S. Lui, ''A secure Cloud Backup System with Assured Deletion and Version Control,'' in Proc. 3rd Int'l Workshop Security Cloud Comput., 2011, pp. 160-167.

[24] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, ''Characteristics of Backup Work-loads in Production Systems,'' in Proc. 10th USENIX Conf. FAST, 2012, pp. 1-16.

[25] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, ''Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,'' IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847-859, May 2011.

[26] A.D. Santis and B. Masucci, ''Multiple Ramp Schemes,'' IEEETrans. Inf. Theory, vol. 45, no. 5, pp. 1720-1728, July 1999.
no. 11, pp. 612-613, 1979.

[27] M.W. Storer, K. Greenan, D.D.E. Long, and E.L. Miller, ''SecureDataDeduplication,'' in Proc. StorageSS, 2008, pp. 1-10
Cloud Storage with Access Control and Assured Deletion,''
vol. 9, no. 6, pp. 903-916, Nov./Dec. 2012.

[28] Yun, C. Shi, and Y. Kim, ''On Protecting Integrity and Confidentiality of Cryptographic File System for Outsourced Storage,'' in Proc. ACM CCSW,Nov. 2009, pp. 67-76.

[29] W. Wang, Z. Li, R. Owens, and B. Bhargava, ''Secure and Efficient Access to Outsourced Data,'' in Proc. ACM CCSW, Nov. 2009, pp. 55-66.

[30] A.D. Santis and B. Masucci, ''Multiple Ramp Schemes,'' IEEE
Trans. Inf. Theory, vol. 45, no. 5, pp. 1720-1728, July 1999,no. 11, pp. 612-613, 1979.