International Journal of Advanced Technology in Engineering and Science
Vol. No.5, Issue No. 03, March 2017
www.ijates.com
ijates
ISSN 2348 - 7550

SECURE AUDITING AND DATA DEDUPLICATING IN CLOUD USING SECCLOUD AND SECCLOUD+

Shaik Abdul Yunus Basha[1] , Reddy Harika[2],
K.Varalakshmi[3],Syed Iliyaz[4]

ABSTRACT

Cloud Computing is an emerging field of technology that relies on sharing of resources over the Internet to achieve economy of scales. It has gained momentum among both individuals and corporate sectors due to a variety of applications such as ease of administration, mobility and collaboration, reduction of hardware and licensing cost, and scalability .Cloud computing technology provide a large number of services. One of the attractive service is storing huge amount of data in cloud. But the data stored in cloud is not fully trustworthy. To provide secure concerns , we propose two systems seccloud and seccloud+.

Sec cloud will provide integrity audit and prevent duplicate files uploading into cloud. Seccloud+ enables integrity auditing and secure deduplication along with encrypting the data before uploading the file. These two systems will prevent multiple copies of a file uploading into the file and provide data integrity . seccloud will maintain map reduce cloud to create a tag to the file before uploading into cloud. Here we prevent the problem of integrity auditing and deduplication . seccloud+ is the advancement of seccloud. A third party has been introduced which involves MD5 algorithm to provide users a convergent encryption before uploading as well as auditing the integrity of data by means of hash key that have being uploaded and stored in cloud. When Relate to earlier work, user computation is greatly reduced in the process of uploading and auditing file. The other expertise called Secured cloud plus is used to encrypt the data before uploading, and auditing The trusted 3rd party security service provider would not storeany data at its end, and its only connected to providing security service. The application or software will provide data integrity verfication by using hashing algorithm like SHA-1, provided encryption/decryption using symmetric algorithm like AES

Keywords: Cloud computing, Data integrity, Auditing, Data deduplication.

I. INTRODUCTION

Now a days vendors and organizations centring mainly on cloud due to its flexibility and tranquillity of uploading and attainment of the data. Many Companies have undertaken different and exclusive innovations on concept of cloud .There by, cloud has become a trending technology in the process of storing and manipulating the data for the huge number of user and organizations. Data processing is carried out by login scenario. In case of sensitivity of the information, authentication is carried out to give access to right user. This

process will helps in process of providing rights, in the sense some of the files are read only, write and some are both in authenticated way by the authenticated person can precede over the data. There are many mechanisms to have Files interpretation called encryption techniques which help to convert plain text to cypher text in order to have security of the data in the file to be upload. Cloud possesses benefits such as efficiency, storage capacity, source provider, data getter ,resource pooling etc. All this capabilities made cloud as magnificent storage zone from individual user to huge organizations. Cloud is able to store bulk amount of data like Tera Byte or peta bytes etc., even then if duplication of redundant will cause over heading effect on cloud server so some mechanisms has to be introduced to avoid the duplication and also enable the integrity of the data

1.1 What is cloud computing?

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

1.2 How Cloud Computing Works?

The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer game The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing

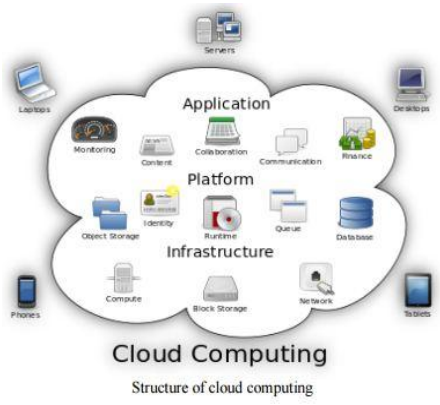

Fig: Structure of Cloud Computing

II PREPHASE

At first idea of avoiding redundant duplication of data in cloud was illustrated by „At eniese‟ which involves in heavy computational over heading at both client (user) side and server side. This idea involves comparing of Meta data of the files stored either client end or server end before uploading. Both server and client contains this

Meta data, it gets compared to the Meta data generated during each upload. During this process of generation of the Meta data client has to give privacy terms and credentials to each and every file access. No issue will arrive if the cloud is trusted one, In case of malicious or scrawny it may leads to hacking and distraction of the uploaded data or files. That‟ s why trust on the cloud has become a big issue. Therefore avoiding of redundant duplicates by this phenomenon is risky. The other

issue related with this phenomenon is lot of Meta data generation which leads in increase of size of Meta data more than plain text, this result in wastage and increase in cost of cloud. Hence the phenomenon is not significant for avoiding redundant duplication of data. In addition, cause in risk of preservation and storing of data also involves in unreliability ,over heading, degradation of performance, time consuming and slow down of process. Apart from this, mechanism stands as detector and eliminator of the duplicates To have impact on the above issues method call "secureauditing" has been introduced which acts a third party trusted server. And also act as a mediator between client and server named as „auditor‟ . This idea concentrate on correctness property called "name of correctness", that means two files namely f1, f1 should contain same data i.e., if data in file f1has "Cloud is an outstanding technology", the other file of same name f1 should also contain same data i.e., "Cloud is an outstanding technology". This method will not work practically, In case of large organizations the names of files may be on the bases of subject of the data stored in the file, in some conditions it may be same. When coming to individual client the percentage of storing files with same name is high so that this idea fails in single client prospective. Every individual have their own perception end view in choosing name of a files. Under those conditions this method gives worst case performance in avoiding duplicate data, which in

turn results in wastage of space in cloud zone. Here sever which act as auditor, fails drastically to stop the entering of duplicate data. Even then this auditor plays a vital role for the large organization due to subject based naming to the files. So this method of avoiding duplicates is greatly admired in cloud for the technology of duplicate eliminator. Here one more problem will come across i.e, files with different name contain same data also stores in cloud which in turn results in wastage of memory and also causes time waste in the process of storing. To have brief knowledge about this lets have a good example, if a user has uploaded a file named as f1 and contain a content "hello welcome to computer", at the same if another user uploaded a file named as f2 with the same content "hello welcome to computer", auditor will audit filenames and store the files because their names are different here wastage of memory in cloud will takes place. There by it provides service which is unreliable based on names .This auditing involves two technique called secured cloud and securedcloudplus

2.1 Advantages

Integrity Auditing

Cloud Storage move the user data to large data centres, which are remotely located ,which causes constant data movements. During this process of uploading and downloading, the data could deliberately be tampered or get corrupt in between. Due to this there is a lot of security issues as the user don't have control over it. Here with the help of our system, we will check the correctness of the data

De-Duplication

De-Duplication identifies and manages duplicate files in the cloud [6].As it happens with everybody; we at times upload the same file multiple times [2] Also the same file can be uploaded by different users[4]. Due to this there is the duplication of file, resulting in wastage the scarce storage resource [4] De-Duplication ensures duplicate data [6] is physically stored only once, and the proof of ownership of the with complete transparency provided to genuine owners

Process of De-Duplication:

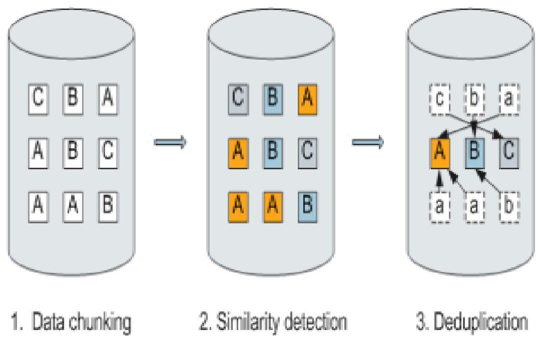

Fig: Data Deduplication Process

Secured Data

The data being stored in the cloud isin an encrypted form to ensure standard security being provided [4].This helps to resist unauthorized users the access to sensitive and personal data [2] [7]. Also security is the major and important factor while storing data in cloud because the user once uploaded the data don't have any rights or authority on that file [7]

III LITERATURE SURVEY

1. A Survey on **"Secure and Constant Cost Public Cloud Storage Auditing with Deduplication"**

Data integrity and storage efficiency are two important requirements for cloud storage. Proof of Retrievability (POR) and Proof of Data Possession (PDP) techniques assure data integrity for cloud storage. Proof of Ownership (POW) improves storage efficiency by securely removing unnecessarily duplicated data on the storage server. However, trivial combination of the two techniques, in order to achieve both data integrity and storage efficiency, results in non-trivial duplication of metadata (i.e., authentication tags), which contradicts the objectives of POW. Recent attempts to this problem introduce tremendous computational and communication costs and have also been proven not secure. It calls for a new solution to support efficient and secure data integrity auditing with storage deduplication for cloud storage. In this paper we solve this open problem with a novel scheme based on techniques including polynomial-based authentication tags and homomorphic linear authenticators. Our design allows deduplication of both files and their corresponding authentication tags. Data integrity auditing and storage deduplication are achieved simultaneously. Our proposed scheme is also characterized by constant realtime communication and computational cost on the user side. Public auditing and batch auditing are both supported. Hence, our proposed scheme outperforms existing POR and PDP schemes while providing the additional functionality of deduplication. We prove the security of our proposed scheme based on the Computational Diffie-Hellman problem, the Static Diffie-Hellman problem and the t-Strong Diffie-Hellman problem. Numerical analysis and experimental results on Amazon AWS show that our scheme is efficient and scalable.

2. A Survey on “DupLESS: Server-Aided Encryption for Deduplicated Storage”

Cloud storage service providers such as Drop box, Mozy, and others perform deduplication to save space by only storing one copy of each file uploaded. Should clients conventionally encrypt their files, however, savings are lost. Message-locked encryption (the most prominent manifestation of which is convergent encryption) resolves this tension. However it is inherently subject to brute-force attacks that can recover files falling into a known set. We propose an 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. It enables clients to store encrypted data with an existing service, have the service perform deduplication on their behalf, and yet achieves strong confidentiality guarantees. We show that encryption for deduplicated storage can achieve performance and space savings close to that of using the storage service with plaintext data.

3. A Survey on **“Scalable and Efficient Provable Data Possession”**

Storage outsourcing is a rising trend which prompts a number of interesting security issues, many of which have been extensively investigated in the past. However, Provable Data Possession (PDP) is a topic that has only recently appeared in the research literature. The main issue is how to frequently, efficiently and securely verify that a storage server is faithfully storing its client’s (potentially very large) outsourced data. The storage server is assumed to be untrusted in terms of both security and reliability. (In other words, it might maliciously or accidentally erase hosted data; it might also relegate it to slow or off-line storage.) The problem is exacerbated by the client being a small computing device with limited resources. Prior work has addressed this problem using either public key cryptography or requiring the client to outsource its data in encrypted form.

In this paper, we construct a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk encryption. Also, in contrast with its predecessors, our PDP technique allows outsourcing of dynamic data, i.e, it efficiently supports operations, such as block modify-cation, deletion and append

IV SYSTEM ARCHITECTURE

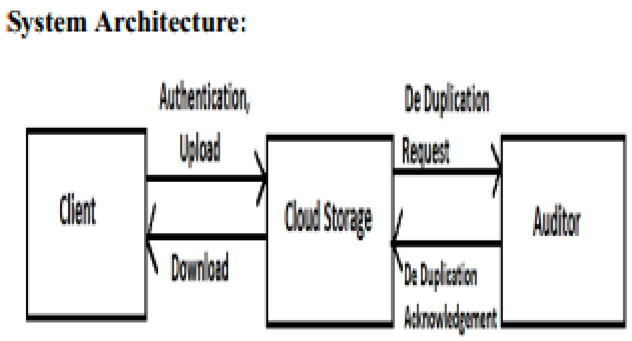

Fig: Sequence Flow of System Architecture

There are three main entities in the system

Client user, Cloud Server and the Auditor. The user directly interacts with the Cloud by registration, authentication [11], upload and download. The Cloudsystem generates hash value for each file being uploaded and stores the same. [9] The Cloud system checks for duplicate data items in cloud on the basis of their hash value and notifies and sends deduplication request to the Auditor [11] The Auditor interacts with the Cloud system only [11]. Upon receiving the deduplication request, it performs suitable action over it. Through GUI the user will interact with the system. It GUI allows the use of icons or other visual indicators to interact with users. There is a database in the back end with the cloud which handle all thMD5 algorithm: Hashing is done to generate a unique hash value for each data item being uploaded [12]. This hash value is used for auditing purpose to identify duplicate files[8] The same hash value is used as checksum to ensure crucial

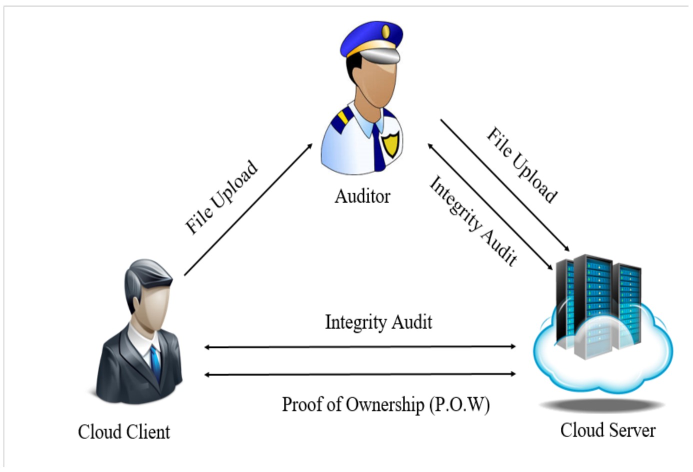

Fig:System Architecture

V ALGORITHM

AES AGORITHM

Broadly speaking the encryption/decryption can be done via symmetric key or asymmetric key. In symmetric algorithms, both parties share the secret key for both encryption/decryption, and from privacy perceptive it is important that this key is not compromised, because cascading data will then be compromised. Symmetric encryption/decryption require less power for computation. On the other hand asymmetric algorithms use pairs of keys, of which one key is used for encryption while other key is used for decryption.

Generally the private key is kept secret and generally held with the owner of data or trusted 3rd party for the data, while the public key can be distributed to others for encryption. The secret key can't be obtained from the public key. In our case since the encryption/decryption is performed on trusted 3rd party server, symmetric key is used, and it delegates the burden of key management to the trusted 3rd party. If key management where to be done at clients end it would mean,

1. either they have to remember the big key
2. store the key in all devices/machine which will be used to access the cloud services, which make user device a bottleneck.
3. individual owner has to take the responsibility of sharing the key with specic authorized group of user which he/she dene.

Outline of the AES Algorithm

```
Constants: intNb = 4; // but it might change someday
int Nr = 10, 12, or 14; // rounds, for Nk = 4, 6, or 8
Inputs: array in of 4*Nb bytes // input plaintext
array out of 4*Nb bytes // output ciphertext
array w of 4*Nb*(Nr+1) bytes // expanded key
Internal work array: state, 2-dim array of 4*Nb bytes, 4 rows and Nb cols
Algorithm:
void Cipher(byte[ ] in, byte[ ] out, byte[ ] w) f
byte[ ][ ] state = new byte[4][Nb];
state = in; // actual component-wise copy
AddRoundKey(state, w, 0, Nb - 1); // see Section 4 below
for (int round = 1; round < Nr; round++) f
SubBytes(state); // see Section 3 below
ShiftRows(state); // see Section 5 below
MixColumns(state); // see Section 5 below
AddRoundKey(state, w, round*Nb, (round+1)*Nb - 1); // Section 4
gSubBytes(state); // see Section 3 below
ShiftRows(state); // see Section 5 below
```

20 AddRoundKey(state, w, Nr*Nb, (Nr+1)*Nb - 1); // Section 4out = state; // component-wise copy

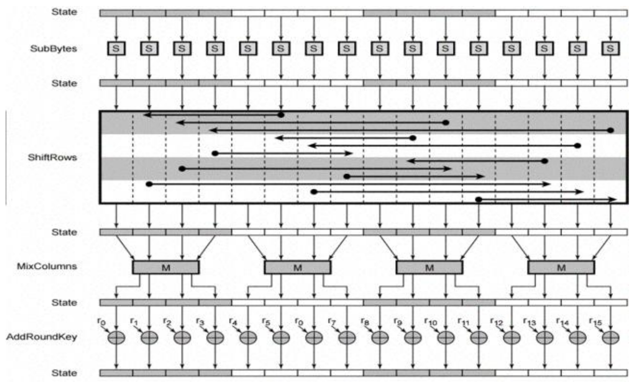

Fig: Sequence of AES Algorithm

MD 5 ALGORITHM

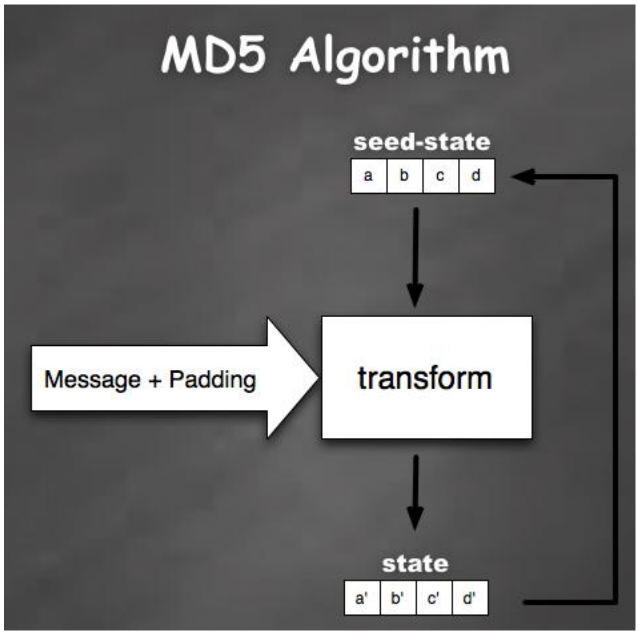

Fig: MD5 Algorithm sequence

MD5 algorithm , It generates 16bit hash key to the file positioned for encryption. At the same time user generates a master key for the file to have authenticated access by which file can be downloaded and decrypted. The successful encrypted file is uploaded to the auditor. Insegment1 this job is carried out. Coming to redundant duplication, hash code generated by MD5 in background for every encryption is sorted and maintained by the auditor. Here the speciality of MD5 is generating hash code for the file irrespective of names. This hash code helps the manager to cross check hashes for each and every time. This method also use to store some hash code even then it doesn‟ t leads to over heading on manager i.e, auditor end, this is because the hash code generated by MD5 algorithm is 16 characters key, are same for even 1000 characters file. This hash code even differs even if one character in the file differ. So MD5 is a highly strengthened idea for redundant duplication. There by this hash stored in auditor for the each file will decide whether to traverse the file or not to the cloud. Here by it reduces and replaces the traditional idea of comparing the file names for the existence in cloud. Here vital role is played by the auditor to compare and cross check content of the file data. In regular interval of time the auditor often logs in to have a check for replication to avoid duplicates. According to auditor observation if the same content of the files are keep on outsourced then it leads to wastage of cloud space which cause impact on over heading and performance of cloud and drain the efficiency of the cloud. All the current hash code of the file is cross check with the existing code . Memory constraints will not be affected by the key stored. Memory consumed by just128 bit will not occupy heavy space in the audit dynamo, Therefore MD5 hashing technic gives very less over heading which is not a very big deal. Here hash code comparison is not at all carried out by the client end nor does cloud server, it is done separate by the third party server called auditor .No the client server or cloud server is responsible for the generation of the hash code even for storing and comparing. Consequently this method is more accurate and reliable way of auditing and monitoring the redundant duplicates, nonredundant in cloud. As cloud cost for storing the data with repetition of file leads to wastage of money to the client. If file is very small then repetition is considerable, if it is in terabytes or peta bytes repetitions is not suggestible

A. Public Key Generation

For each and every current input of client a random unique key is generated known as public key. This key is generated for every login and key is very much essential for access of cloud. Next client upload the file f1 and call for encryption to encrypt, let the encrypted file be f2, cliental ways want to secure the data so uploads the encrypted file. Input (f1, f2) Upload ( )

In this process cypher texted file taken as input and ask the client for desired name to upload the file, The name of the file is displayed in the list of file

B. Hash Key Generation

Here MD5 is used which runs on back ground of auditor.This MD5 generates 16 char hash key Auditor will performall the process of finding duplications and store the key in the log for future checking.

C. Redundant Duplication

Here comparing of hash key for the new hash

```
INPUT (F1, F2)
DES (F1, F2) =£1,£2;
MD5 (F1, F2) =f1#,f2#;
UPLOAD ((£1, f1#) ∩ (£2, f2#));
```

V SECCLOUD

In this subsection, we respectively describe the three protocols including file uploading protocol, integrity auditing protocol and proof of ownership protocol in SecCloud. Before our detailed elaboration, we firstly introduce the system setup phase of SecCloud, which initializes the public and private parameters of the system.

ijates
ISSN 2348 - 7550

System Setup: The auditor working as an authority picks a random integer _ ∈ R Zp as well as random elements g; u1; u2; : : : ut ∈ R G, where t specifies the maximum number of sectors in a file block. The secret key sk is set to be _ and kept secret, while the public key pk = (g_; {ui}t i=1) is displayed to other entities

File Uploading Protocol: Based on the public and private keys generated in system setup, we then describe the file uploading protocol. Suppose the uploading file F has s blocks: B1;B2; : : : ;Bs, and each block Bi for i = 1; 2; : : : ; s contains j sectors: Bi1;Bi2; : : : ;Bij. Let n be the number of slave nodes in the MapReduce cloud. The client runs the deduplication test by sending hash value of the file Hash (F) to the cloud server. If there is a duplicate, the cloud client performs Proof of Ownership protocol with the cloud server which will be described later. If it is passed, the user is authorized to access this stored file without uploading the file. Otherwise (in the second phase), the cloud client uploads a file F as well as its identity IDF to the distributed file system in MapReduce auditing cloud, and simultaneously sends an “upload” request to the master node in MapReduce, which randomly picks {_i}ni =1 such that Σn i=1 _i = _ and assigns the ith slave node with _i. When each slave node (say the ith salve node) receives the assignment _i, it does two steps: 1) Pick up (IDF;F) in the distributed file system in MapReduce, and build a Merkle hash tree on the blocks {Bj}sj=1 of F. 2) Let hroot denote the hash of the root node of Merkle hash tree built on F. This slave node uses _i to sign hroot by computing _i = h_iroot. Finally, the signature _i is sent to the the slave node which is specified by master node for executing the reducing procedure. The specified slave node for reducing procedure gathers all the signatures {_i} ni =1 from the other slave nodes, and computes _ = Πn i=1 _i. The “reduced” signature _ is finally sent back to client as receipt of the storage of file F. In the third phase, the MapReduce auditing cloud starts to upload the file F to cloud server. To allow public auditing, the master node builds file tags of F. Specifically, master node firstly writes and arranges all the sectors of F in a matrix (we say S), and computes a homographic signature for each row of the matrix S (highlighted red in Fig. 3). Notice that the tag generation procedure also follows the computing paradigm with MapReduce. That is, for the ith (i = 1; 2; : : : ; s) row of S, the jth (j = 1; 2; : : : ; n) slave node computes _ij = [Hash(IDF||Bi) Πt k=1 uBik k]_j , where Σn j=1 _j = _. Accordingly, all the signatures {_ij} nj =1 are then multiplied into the homomorphic signature _i = Πn j=1 _ij at a specified reducing slave node. The homomorphic signature allows us to in future aggregate the signatures signed on the sectors in the same column of S using multiplication. Finally, the master node uploads (ID;F; {_i}si =1) to cloud server.

Integrity Auditing Protocol:

In the integrity auditing protocol, either the MapReduce auditing cloud or the client works as the verifier. Thus, without loss of generality, in the rest of the description of this protocol, we use verifier to identify the client or MapReduce auditing cloud. The auditing protocol is designed in a challenge-response model. Specifically, the verifier randomly picks a set of block identifiers (say IF) of F and asks the cloud server (working as prover) to response the blocks corresponding to the identifiers in IF. In order to keep randomness in each time of challenge, even for the same IF, we introduce a random coefficient for each block in challenge. That is, for each identifier i ∈ IF, the coefficient ci for the block identified by i is computed as ci = f (tm||IDF||i), where f(·) is a pseudorandom function and tm is the current time period. Finally, C = {(i; ci)}i∈ IF is sent to cloud server for challenge.

Proof of Ownership Protocol:

The POW protocol aims at allowing secure deduplication at cloud server. Specifically, in deduplication, a client claims that he/she has a file F and wants to store it at the cloud server, where F is an existing files having been stored on the server. The cloud server asks for the proof of the ownership of F to prevent client unauthorized or malicious access to an unowned file through making cheating claim. In SecCloud, the POW protocol is similar to [3] and the details are described as follows. Suppose the cloud server wants to ask for the ownership proof for file F. It randomly picks a set of block identifiers, say IF ⊆ {1; 2; s} where s is the number of blocks in F, for challenge. Upon receiving the challenge set IF, the client first computes a short value and constructs a Merkle tree. Note that only sibling-paths of all the leaves with challenged identifiers are returned back to the cloud server, who can easily verify the correctness by only using the root of the Merkle tree. If it is passed, the user is authorized to access this stored file.

VI SECCLOUD+

We specify that our proposed SecCloud system has achieved both integrity auditing and file deduplication. However, it cannot prevent the cloud servers from knowing the content of files having been stored. In other words, the functionalities of integrity auditing and secure deduplication are only imposed on plain files. In this section, we propose SecCloud+, which allows for integrity auditing and deduplication on encrypted files.

A. System Architecture

Compared with SecCloud, our proposed SecCloud+ involves an additional trusted entity, namely key server, which is responsible for assigning clients with secret key (according to the file content) for encrypting files. This architecture is in line with the recent work [4]. But our work is distinguished with the previous work [4] by allowing for integrity auditing on encrypted data. SecCloud+ follows the same three protocols (i.e., the file uploading protocol, the integrity auditing protocol and the proof of ownership protocol) as with SecCloud. The only difference is the file uploading protocol in SecCloud+ involves an additional phase for communication between cloud client and key server. That is, the client needs to communicate with the key server to get the convergent key for encrypting the uploading file before the phase 2 in SecCloud. Unlike SecCloud, another design goals of file confidentiality is desired in SecCloud+ as follows.

File Confidentiality:

The design goal of file confidentiality requires to prevent the cloud servers from accessing the content of files. Specially, we require that the goal of file confidentiality needs to be resistant to “dictionary attack”. That is, even the adversaries have preknowledge of the “dictionary” which includes all the possible files, they still cannot recover the target file [4].

B. SecCloud+ Details We introduce the system setup phase of SecCloud+ as follows.

System Setup:

As with SecCloud, the auditor initializes the public key pk = (g_; {ui}t i=1) and private key sk = _, where g; u1; u2; : : : ; ut ∈ R G. In addition, to preserve the confidentiality of files, initially, the key server picks a random key k for further generating file encryption keys, and each client is assigned with a secret key ck for encapsulating file encryption keys. Based on the initialized parameters, we then respectively describe the three protocols involved in SecCloud+.

File Uploading Protocol:

Suppose the uploading file F has s blocks, say B1;B2; : : : ;Bs, and each block Bi for i = 1; 2; : : : ; s contains t sectors, say Bi1;Bi2; : : : ;Bit.Client computes hF = Hash(F) by itself. In addition, for each sector Bij of F where i = 1; 2; : : : ; s and j = 1; 2; : : : ; t, client computes its hash hBij = Hash(Bij). Finally (hF; {hBi}i=1;:::;s;j=1;:::;t) is sent to key server for generating the convergent keys for F. Upon receiving the hashes, the key server computes sskF = f(ks; hF) and sskij = f(ks; hBij) for i = 1; : : : ; s and j = 1; : : : ; t, where ks is the convergent key seed kept at the key server, and f(·) is a pseudorandom function. It is worthwhile nothing that, 1) We take advantage of the idea of convergent encryption [21][22][23] to make the deterministic and "content identified" encryption, in which each "content" (file or sector) is encrypted using the session key derived from itself. In this way, different "contents" would result in different cipher texts, and deduplication works. 2) Convergent encryption suffers from dictionary attack, which allows the adversary to recover the whole content with a number of guesses. To prevent such attack, as with [4], a"seed" (i.e., convergent key seed) is used for controlling and generating all the convergent keys to avoid the fact that adversary could guess or derive the convergent key just from the content itself. 3) We generate convergent keys on sector-level (i.e., generate convergent keys for each sector in file F), to enable integrity auditing. Specifically, since convergent encryption is deterministic, it allows to compute homomorphic signatures on (convergent) encrypted data as with on plain data, and thus the sector-level integrity auditing is preserved. Client then continues to encrypt F sector by sector and uploads the ciphertext to auditor. Specifically, for each sector Bij of F, i = 1; 2; : : : ; s and j = 1; 2; : : : ; t, client computes ctBij = Enc(sskBij ;Bij), and sends (IDF; {ctBij }i=1;:::;s;j=1;:::;t) to auditor, where Enc(·) is the symmetric encryption algorithm. The convergent keys sskij are encapsulated by client's secret key ck and directly stored at the cloud servers.

Integrity Auditing Protocol:

The integrity auditing protocol works in the same way of that in SecCloud, but imposed on encrypted data. Specifically, the verifier (could be either the client or the auditor) submits a set of pairs {(i; ci)}i∈ IF where IF ⊆ {1; 2; : : : ; s} and ci ∈ R Z. Upon receiving {(i; ci)}i∈ IF , the cloud servers then computes !j = Σ i∈ IF cictBij for each j = 1; 2; : : : ; t, as well as the aggregated homomorphic signature _ = Π i∈IF _ci i . In addition, the cloud server constructs a Merkle hash tree on encrypted blocks ctBi of F and attempts to prove retrievability at block-level. Precisely, for each i ∈ IF, the cloud server computes a pair (Hash(ctBi);Ωi), where ctBi = [ctBi1 ; : : : ; ctBit] and Ωi includes the necessary auxiliary information for reconstructing the root node using {ctBi }i∈IF . Finally (_; {!j}t j=1; {(Hash(ctBi);Ωi)}i∈IF) is sent to verifier for auditing

VII CONCLUSION

Aiming at achieving both data integrity and deduplication in cloud, we propose SecCloud and SecCloud+. SecCloud introduces an auditing entity with maintenance of a MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud. In addition, SecCoud enables secure deduplication through introducing a Proof of Ownership protocol and preventing the leakage of side channel information in data deduplication. Compared with previous work, the computation by user in SecCloud is greatly reduced during the file uploading and auditing phases. SecCloud+ is an advanced construction motivated by the fact that customers always want to encrypt their data before uploading, and allows for integrity auditing and secure deduplication directly on encrypted data

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A view of cloud computing,” *Communication of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] J. Yuan and S. Yu, “Secure and constant cost public cloud storage auditing with deduplication,” in *IEEE Conference on Communications and Network Security (CNS)*, 2013, pp. 145–153.

[3] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, “Proofs of ownership in remote storage systems,” in *Proceedings of the 18th ACM Conference on Computer and Communications Security*. ACM, 2011, pp. 491–500.

[4] S. Keelveedhi, M. Bellare, and T. Ristenpart, “Dupless: Serveraided encryption for deduplicated storage,” in *Proceedings of the 22Nd USENIX Conference on Security*, ser. SEC’13. Washington, D.C.: USENIX Association, 2013, pp. 179–194. [Online]. Available: https://www.usenix.org/conference/usenixsecurity13/technicalsessions/presentation/bellare

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS ’07. New York, NY, USA: ACM, 2007, pp. 598–609.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, “Remote data checking using provable data possession,” *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 12:1–12:34, 2011.

[7] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, “Scalable and efficient provable data possession,” in *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, ser. SecureComm ’08. New York, NY, USA: ACM, 2008, pp. 9:1–9:10.

[8] C. Erway, A. K¨upc¸ ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 213–222.

[9] F. Seb´e, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, no. 8, pp. 1034–1038, 2008.

[10] H. Wang, "Proxy provable data possession in public clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 551–559, 2013.

[11] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[12] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ser. ASIACRYPT '08. Springer Berlin Heidelberg, 2008, pp. 90–107.

[13] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Computer Security – ESORICS 2009*, M. Backes and P. Ning, Eds., vol. 5789. Springer Berlin Heidelberg, 2009, pp. 355–370.

[14] J. Xu and E.-C. Chang, "Towards efficient proofs of retrievability," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '12. New York, NY, USA: ACM, 2012, pp. 79–80.

[15] E. Stefanov, M. van Dijk, A. Juels, and A. Oprea, "Iris: A scalable cloud file system with efficient integrity checks," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 229–238.

[16] M. Azraoui, K. Elkhiyaoui, R. Molva, and M. O¨ nen, "Stealthguard: Proofs of retrievability with hidden watchdogs," in *Computer Security -ESORICS 2014*, ser. Lecture Notes in Computer Science, M. Kutyłowski and J. Vaidya, Eds., vol. 8712. Springer International Publishing, 2014, pp. 239–256.

	Mr.S.Abdul Yunus Basha pursued his B.Tech from JNTA,Anantapur and M.Tech JNTA,Anantapur . He worked as professor in RGMCET, Nandyal and presently working as Placement officer in BITS. He has published 6 International Journals .His research areas of interest are Computer Netwoks, Cryptography, Cloud computing, Bigdata

	Miss Reddy Harika is pursuing B.Tech from Brindavan Institute of Technology & Science in the department of CSE.
	Miss K.Varalakshmi is pursuing B.Tech from Brindavan Institute of Technology & Science in the department of CSE
	Mr.Syed Iliyaz is pursuing B.Tech from Brindavan Institute of Technology & Science in the department of CSE