IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

Assuring Confidentiality with Minimum Data Redundancy in Hybrid Cloud

Mr R. V. Argiddi[1], Ms Prachi Sontakke[2]

Assistant Professor, Computer Science, Walchand Institute of Technology, Solapur, India[1]

Computer Science, Walchand Institute of Technology, Solapur, India[2]

Abstract: Cloud computing as the necessity of the ever-increasing volume of data and its storage, comes with the critical challenge of managing this large amount of data with scalability. A data compression technique called deduplication is used for eliminating the redundant data by deleting the duplicate copies so as to save the cloud space. Apart from data compression, we are also applying the security aspects using convergent encryption with some additional concepts viz., cryptographic tuning and domain separation. Maintaining the authorized access for the user's confidential data, the concept called 'Proof of Ownership' is used for recognizing the respective user along with his access privilege.

Keywords: secure deduplication, convergent encryption, cryptographic tuning, domain separation.

I. INTRODUCTION

The ubiquity of the cloud computing has resulted in the widespread availability of cluster-based services and applications accessible through the Internet. Examples include online storage services, big data analytics, and e-commerce websites. In such a cluster-based cloud environment, each and every physical machine runs a multiple virtual machines as instances of a guest operating system to contain different kind of user applications, and their data is stored in virtual hard disks which increases the security issues. Networking of computer tends to use several advancements in computing technologies like Distributed computing and Cloud computing. With the same underlying concept between these two terms, they even slightly differ from each other.

Pay as u go – Pay-as-u-go is a payment method of cloud computing that charges based on the usage. This practice is similar to that of various bills, which use only resources that are required. Users can select the CPU, memory, storage, OS, security, networking capacity, access controls, and any additional software that are required. Whereas, in distributed systems, the client needs to buy the particular service or the server for its storage. Broad Network access – Data over distributed systems can be accessed and exchanged within a specified range or location using LAN, MAN, WAN networks. However, data in cloud is stored across multiple servers all over the world. And these cloud storage services can be accessed through web service application programming interface (API) or co-located cloud computer service. Infrastructure Maintenance – In cloud, storage maintenance tasks, such as purchasing additional storage capacity, are handled by the service provider. Whereas in, distributed systems, it may affect the whole maintenance of the infrastructure even if a single server is damaged. Resource sharing –Cost of the resources, managing it, technical support, and in case of any resource failure, repairing it; all this has to be managed by the client itself in the distributed system. While in cloud, managing of resources and technical support is given by the cloud service provider (CSP) in the cloud. As described earlier, costing of any resources is as per the usage and no additional payment by the client is required in case of any repair or failure. This is done by the CSP itself.

A. Definitions

1) User: A non-technical end user who accesses services through a browser or via some cloud applications.
2) Cloud Service Provider: CSP is an organization that offers services to the customers of cloud using remote facility via Internet.
3) Cloud Storage: It is a service model in which data is maintained, managed and backed up remotely and made available to the users over the network

B. Deployment Model

Due to the increasing security issues in cloud computing, we are using strong encryption algorithms along with public and private key encryption. The following table I depicts the classification of various cloud deployment models according to the storage of data and its respective keys.

Table I: Storage of data and keys in various cloud

Data	Key	Advantages	Disadvantages
Private Cloud	Private Cloud	Highly Secured	High cost
Public Cloud	Public Cloud	No cost	Less secured
Private Cloud	Public Cloud	-	High cost Not secure
Public Cloud	Private Cloud	Less cost Highly secured	-

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

The model we are using is Hybrid cloud, with keys in the private cloud and data in the public cloud. Hence, being highly secured as the keys are in private cloud and service to the client at the less cost due to the large volume of data in a public cloud.

C. Service Model

In this paper, we are using Storage-as-a-Service model of cloud computing. This service is used by the large companies for buying the storage space on rent from the cloud service provider. Many enterprises promote Storage-as-a-service as a convenient way to manage backups. These enterprises sign a service level agreement (SLA) whereby the Storage-as-a-service provider agrees to rent storage space on a cost-per-gigabyte-stored and cost-per-data-transfer basis.

II. PRELIMINARIES

A. Security in Cloud

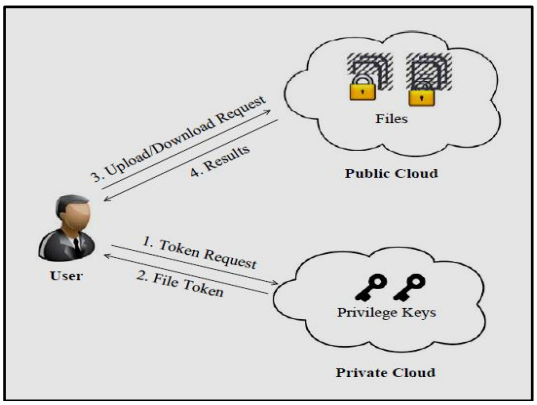

Fig 1. Security in Hybrid Cloud

Security of the data in a network can be achieved using cryptographic concepts. To protect the data from getting corrupted or hacked, each file that is being stored should be encrypted in unreadable format. Encryption comes with the concept of keys – Public and Private keys. While the public key accessible to everyone, private keys are very secured and available only to the corresponding user. Figure 1 illustrate the most convenient, cost reducing and secured approach of storing the keys in cloud.

1) Convergent Encryption:

Convergent encryption, also known as Content hash keying, is a cryptographic system. In the context of cloud computing, it is used to remove duplicate copies of the files from storage without the provider having access to the encryption keys. This encryption system computes the cryptographic hash value of the original file and then the encryption is performed on the file by using this particular obtained hash value as a key. Finally, the hash value itself s encrypted with the key chosen by the respective user and stored in the cloud.

B. Data Storage in Cloud

To address the problem of ever increasing volume of data on cloud and many a times which may store redundant data copies, as a result of which extra storage space is wasted; a well-known technique called data deduplication has attracted and grabbed more attention recently.

1) Data Deduplication:

The basic principle of data deduplication is to store a single copy of the repetitive data and a pointer to point to all the duplicate blocks. This can be achieved at File level or Block level as shown in Fig 2. The old and new data are compared and if they match, they are marked as duplicate. Data pointers are updated and the redundant copy is deleting. Thus, reducing the data storage volumes. Data deduplication engine gives an index of the digital signature to the data segment and the signature of a given repository to identify the data blocks. A pointer is provided by the index to determine the presence of the data block. In the copy operation, the data deduplication software finds the duplicate block of data, and if found, it inserts a link into the main original data block index location instead of repeating the data and storing the data block again. Occurrence of the same block more than once will cause the generation of more pointers to the indexing table.

Fig 2. Category of data deduplication strategies

C. Proof of Ownership

It is challenge-response protocol enabling a storage server to check if a requesting entity is the actual data owner, based on a some short value. That is, when a user wants to upload a data file (A) to the cloud, firstly hash value is computed and then the user sends a hash value hash = H(A) to the storage server. This latter maintains a database of hash values of all received files, and looks up hash. If the match is found, then A is already outsourced to cloud servers. Then there is no need for the cloud to upload the file to remote storage servers. If there is no match, then the user has to send the file data (A) to the cloud.

D. Confidentiality

Confidentiality is a set of rules or a promise that limits access or places restrictions on certain types of information.

III. SYSTEM MODEL

A. Architecture for authorized secure deduplication. Fig 3 shows the architecture of our system model which includes main components such as key request handler and tag generation, deduplication check, convergent encryption and decryption. A new deduplication system obtained for differential duplicate check is proposed under this hybrid cloud architecture where in the public cloud resides the S-CSP. The user is only allowed to perform the duplicate check for files marked with the corresponding

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

privileges**.** To support authorized deduplication, the tag of a file F will be recognized by the file F along with the privilege. To show the difference with traditional notation of tag, we call it file token instead.

To achieve authorized access, a secret key K_P will be bounded with a privilege p to generate a file token. Let $\emptyset_{F,P}$ = TagGen(F, K_P) be the token of F which is only allowed to access by user with privilege p. In another word, the token$\emptyset_{F,P}$, can only be computed by the users with the privilege p.

As a result of this, if a file is been uploaded by a user as a duplicate token $\emptyset_{F,P}$ then a duplicate check that is sent from another user will be delivered successful if and only if he also has the file F and privilege p. This kind of token generation function could be quickly implemented as H(F, K_P), where H(.) denotes a cryptographic hash function.

B. Design goals

The problem of privacy preserving deduplication in cloud computing and propose a new deduplication system supporting for,

1) Differential Authorization**:** Each individual who has been authorized can hold his/her individual token of his file to perform duplicate check based on his privileges. Under this assumption, no other user can generate a token for duplicate check which is out of his privileges or without the aid from the private cloud server.
2) Authorized Duplicate Check**:** Authorized user can use his/her corresponding private keys to generate query for specific file and the privileges he/she owned with the help of private cloud, while the duplicate check is performed by the public cloud directly and that informs the user if there is any duplicate.

The security aspects to be considered in this paper lie in two folds, that includes the file token security and data files security.

For the file token security, two aspects are defined as unforgeability and indistinguishability of file token. The details are given below.

1) Unforgeability of file token/duplicate-check token:
Users with unauthorized access and inappropriate privileges should be prevented from getting or generating the file tokens for any duplicate check of any file which stored at the S-CSP.

The users are not allowed to collude with the public cloud server so as to break the unforgeability of file tokens. In our system, the S-CSP is a true provider and curious and will truly perform the duplicate check on receiving the duplicate request from the users.

The duplicate check token of users should be issued from the private cloud server in our scheme.

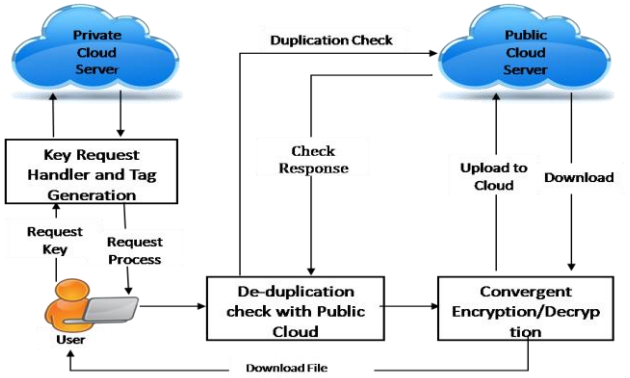

Fig 3. Architecture

2) Indistinguishability of file token/duplicate-check token: It requires that any user without querying the private cloud server for some file token, cannot get any useful information from the token that includes the file information or the privilege information.

3) Data Confidentiality:
Users with unauthorized access and inappropriate privileges or files, including the S-CSP and the private cloud server, have to be prevented from access to the underlying original file that is stored at S-CSP. In another word, the goal of the opponent is to access, retrieve and recover the files that does not belong to them. In our system, a higher level of confidentiality is achieved as compared to the previous definition of data confidentiality that was based on convergent encryption,.

C. Privacy and deduplication

The prior approaches do not provide any privacy. An adversary can gain access to the raw associative array and can gain access to the contents of all the stored files. It is possible to encrypt all the files before storing them, but if the same file is encrypted with a different key, the encrypted files will be different and the deduplication will fail. This could be solved by encrypting all the files with the same key, but then everyone with some access would immediately have full access. We need a solution with different encryption keys for every file but that still deduplicates files. This may seem paradoxical, until you bend the rules slightly to require different encryption keys for every different file. You can then have equal keys for equal files. Equal files, equal keys thus equal encrypted data and problem solved.

D. Convergent Encryption

In convergent encryption, a hash of the unencrypted file is taken as the encryption key. This is safe because you need the key to decrypt the file, unless you have already have the unencrypted file, but then there is nothing left to secure. First we need some ingredients. Take the cryptographic primitives

$H_A: \{0,1\}^* \rightarrow \{0,1\}^{lA}$
$H_B: \{0,1\}^* \rightarrow \{0,1\}^{lB}$
$E: \{0,1\}^{lK} X \{0,1\}^* \rightarrow \{0,1\}^*$
$D: \{0,1\}^{lK} X \{0,1\}^* \rightarrow \{0,1\}^*$

DOI 10.17148/IJARCCE.2016.5575

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

where H_A and H_B are cryptographic hash functions of length l_A and l_B respectively and E and D are symmetric encryption and decryption functions with key length l_K. This implies that $H_A(X1) = H_A(X2) \Leftrightarrow X1=X2$

$H_B(X2) = HB(X2) \Leftrightarrow X1=X2$
$D(K1,E(K2,X)) = X \Leftrightarrow K1 = K2$

Where the implications in the leftward direction are exact but in the rightward implications are only with cryptographic confidence. This will be important later in the security analysis. For permanent storage, we will use an associative array with interface
Store: $\{0,1\}^{l_B} \times \{0,1\}^* \rightarrow \emptyset$
Retrieve: $\{0,1\}^{l_B} \rightarrow \{0,1\}^*$

The convergent encryption store will have the interface
Put: $\{0,1\}^* \rightarrow \{0,1\}^{l_B} \times \{0,1\}^{l_A}$
Get: $\{0,1\}^{l_B} \times \{0,1\}^{l_A} \rightarrow \{0,1\}^*$

The Put operation takes a piece of information and stores it encrypted in the associative array. It is implemented as
Put(X) =:(H,K)
$K = H_A(X)$
$X' = E(K,X)$
$H = H_B(X')$
Store(H,X′)

The Get operation is simply the inverse operation
Get(H,K) =:X
X′= Retrieve(H)
$X = D(K,X')$

The Get operation can optionally verify the integrity of the data by checking $K=H_A(X)$ and/or $H=H_A(X')$. The later has an advantage we will show below.

The implementation given above requires three passes through the data. The passes have data dependencies so they cannot be parallelized. It is possible to develop a two-pass system. The hash H of the output stream X′ can be calculated while the output stream is being produced. This is similar to authenticated encryption but differs in the use of a hash instead of a message authentication code. But it is not possible to construct a one-pass system that deduplicates and has at least the security of convergent encryption. In a one-pass system, the first couple of bytes cannot rely on all the remaining bytes. But this is necessary to have a key with the entropy of the entire file. But this method is vulnerable to brute force attack. This can be solved by two methods:

1) Domain Separation
2) Cryptographic tuning

IV. CONTRIBUTION

A. Domain Separation

Suppose we make H_A a keyed hash function with a key K_A. This will thwart attacks relying on knowledge of H_A, unless the attacker knows the key. This includes all major the attacks mentioned above. The downside is that it also restricts deduplication to data encrypted with the same K_A. The different K_A effectively create different domains. Within such a domain, deduplication happens. But anyone within the domain can also confirm that certain files are stored, or even use the learn the remaining attack to find which of a small set of possible files is stored. From outside the domain, all the encryption keys K depend on the domain key K_A. The domains are thus at least as secure as when they were encrypted using just the domain key.

Where the domain boundaries should be, or when to use different K_A's will be an important design decision. Larger domains will result in more deduplication and a more efficient system, but it will also leak some knowledge about what is stored in the system to a larger group.

B. Cryptographic Tuning

For the system to function properly it relies on:
H_B needs pre-image resistance to prevent an attacker from manipulating data undetectably. Collision resistance would be required if the implementation cannot handle hash collisions. H_A needs to be a good randomness extractor. Its role is to extract sufficient entropy from the plaintext to create a key. A hash that maps the plaintext distribution uniformly to the set of encryption keys will satisfy this.

V. CONCLUSION

The notion of authorized data deduplication was proposed for protecting the data security by using differential privileges of users in the duplicate check. We also presented new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the the private cloud server generates the duplicate-check tokens of files with private keys. We used convergent encryption with modification version to deal with brute force attack using Domain Separation and Cryptographic tuning to make better authorized deduplication technique.

REFERENCES

[1]. K. Jin and E. Miller, "The effectiveness of deduplication on virtual machine disk images" In Proc. SYSTOR 2009: The Israeli Experimental Systems Conference.
[2]. Q. He, Z. Li, and X. Zhang, “Data deduplication techniques,” in International Conference on Future Information Technology and Management Engineering, 2010, pp. 431–432.
[3]. Z. Li, X. Zhang, and Q. He, “Analysis of the key technology on cloud storage,” in International Conference on Future Information Technology and Management Engineering, 2010, pp. 427–428.
[4]. 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 ICDCS, pages 617–624, 2002.
[5]. M. W. Storer, K. M. Greenan, D. D. E. Long, and E. L. Miller. Secure data deduplication. In StorageSS, pages 1-10, 2008.
[6]. D. Harnik, B. Pinkas, and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. IEEE Security & Privacy, 8(6), 2010.
[7]. J. Yuan and S. Yu. Secure and constant cost public cloud storage auditing with deduplication. IACR Cryptology ePrint Archive, 2013:149, 2013.

[8]. B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottolneck in the data domain deduplication file system," in 6th USENIX Conference on File and Storage Technologies (FAST '08), 2008, pp. 269–271.
[9]. Mark Lillibridge, Kave Eshghi, Deepavali Bhagwat, Vinay Deolalikar, Greg Trezise, and Peter Camble. Sparse indexing: Large scale, inline deduplication using sampling and locality. In Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST). USENIX, 2009.
[10]. J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl. A secure data deduplication scheme for cloud storage. In Technical Report, 2013
[11]. 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 ICDCS, pages 617–624, 2002.
[12]. Google App Engine [Online]. Available: http://code.google.com/appengine/
[13]. P. Mell and T. Grance, "The nist definition of cloud computing (draft)," NIST special publication, vol. 800, no. 145, p. 7, 2011
[14]. S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In Proc. USENIX FAST, Jan 2002.
[15]. M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-lockedencryption and secure deduplication. In EUROCRYPT, pages 296– 312, 2013.
[16]. D. Ferraiolo and R. Kuhn. Role-based access controls. In 15th NIST-NCSC National Computer Security Conf., 1992.
[17]. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman.Role-based access control models. IEEE Computer, 29:38–47, Feb 1996.
[18]. P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In Proc. of USENIX LISA, 2010.
[19]. 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 3rd International Workshop on Security in Cloud Computing, 2011.
[20]. M. W. Storer, K. Greenan, D. D. E. Long, and E. L. Miller. Secure data deduplication. In Proc. of StorageSS, 2008.

DOI 10.17148/IJARCCE.2016.5575