An approach for security and prevent Duplication in Datacenter

Vasanthkumar s, sivakumaran s
Vasanthkumar041@gmail.com

Abstract—Data deduplication is one of important data compression techniques for eliminating duplicate copies of repeating data, and has been widely used in datacenters to reduce the amount of storage space and save bandwidth. In most organizations, the storage systems contain duplicate copies of many pieces of data. For example, the same file may be saved in several different places by different users, or two or more files that aren't identical may still include much of the same data. Deduplication eliminates these extra copies by saving just one copy of the data and replacing the other copies with pointers that lead back to the original copy. To protect the confidentiality of sensitive data while supporting deduplication, the convergent encryption technique has been proposed to encrypt the data before outsourcing.

The Hashing technique has been proposed to find the duplication and preventing the duplication. For protecting the confidentiality of the data the Data encryption standard Algorithm has been used to encrypt the data We show that our proposed authorized duplicate check scheme incurs minimal overhead compared to normal operations.

Index Terms—Deduplication, authorized duplicate check, confidentiality, hybrid cloud

1 INTRODUCTION

Cloud computing provides seemingly unlimited "virtualized" resources to users as services across the whole Internet, while hiding platform and implementation details. Today's cloud service providers offer both highly available storage and massively parallel computing resources at relatively low costs. As cloud computing becomes prevalent, an increasing amount of data is being stored in the cloud and shared by users with specified *privileges*, which define the access rights of the stored data. One critical challenge of cloud storage services is the management of the ever-increasing volume of data.

To make data management scalable in cloud computing, deduplication [17] has been a well-known technique and has attracted more and more attention recently. Data deduplication is a specialized data compression technique for eliminating duplicate copies of repeating data in storage. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Instead of keeping multiple data copies with the same content, deduplication eliminates redundant data by keeping only one physical copy and referring other redundant data to that copy. Deduplication can take

place at either the file level or the block level. For filelevel deduplication, it eliminates duplicate copies of the same file. Deduplication can also take place at the block level, which eliminates duplicate blocks of data that occur in non-identical files.

Although data deduplication brings a lot of benefits, security and privacy concerns arise as users' sensitive data are susceptible to both insider and outsider attacks. Traditional encryption, while providing data confidentiality, is incompatible with data deduplication. Specifically, traditional encryption requires different users to encrypt their data with their own keys. Thus, identical data copies of different users will lead to different ciphertexts, making deduplication impossible. Convergent encryption [8] has been proposed to enforce data confidentiality while making deduplication feasible. It encrypts/decrypts a data copy with a *convergent key*, which is obtained by computing the cryptographic hash value of the content of the data copy. After key generation and data encryption, users retain the keys and send the ciphertext to the cloud. Since the encryption operation is deterministic and is derived from the data content, identical data copies will generate the same convergent key and hence the same ciphertext. To prevent unauthorized access, a secure proof of ownership protocol [11] is also needed to provide the proof that the user indeed owns the same file when a duplicate is found. After the proof, subsequent users with the same file will be provided a pointer from the server without needing to upload the same file. A user can download the encrypted file with the pointer from the server, which can only be decrypted by the corresponding data owners with their convergent keys. Thus, convergent encryption allows the cloud to perform deduplication on the ciphertexts and the proof of ownership prevents the unauthorized user to access the file.

However, previous deduplication systems cannot support *differential authorization duplicate check*, which is important in many applications. In such an authorized deduplication system, each user is issued a set of privileges during system initialization (in Section 3, we elaborate the definition of a privilege with examples). Each file uploaded to the cloud is also bounded by a set of privileges to specify which kind of users is allowed to perform the duplicate check and access the files. Before submitting his duplicate check request for some file, the user needs to take this file and his own privileges as inputs. The user is able to find a duplicate for this file if and only if there is a copy of this file and a matched privilege stored in cloud. For example, in a company, many different privileges will be assigned to employees. In order to save cost and efficiently management, the data will be moved to the storage server provider (SCSP) in the public cloud with specified privileges and the deduplication technique will be applied to store only one copy of the same file. Because of privacy consideration, some files will be encrypted and allowed the duplicate check by employees with specified privileges to realize the access control. Traditional deduplication systems based on convergent encryption, although providing confidentiality to some extent, do not support the duplicate check with differential privileges. In other words, no differential privileges have been considered in the deduplication based on convergent encryption technique. It seems to be contradicted if we want to realize both deduplication and differential authorization duplicate check at the same time.

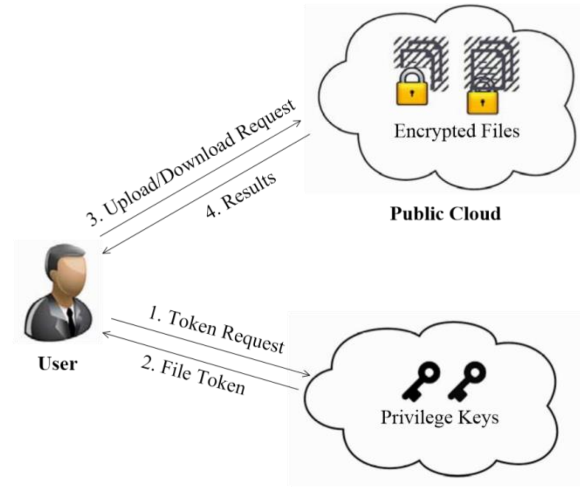
2 SYSTEM MODEL

Hybrid Architecture for Secure Deduplication

At a high level, our setting of interest is an enterprise network, consisting of a group of affiliated clients (for example, employees of a company) who will use the S-CSP and store data with deduplication technique. In this setting, deduplication can be frequently used in these settings for data backup and disaster recovery applications while greatly reducing storage space. Such systems are widespread and are often more suitable to user file backup and synchronization applications than richer storage abstractions. There are three entities defined in our system, that is, *users*, *private cloud* and S-CSP in *public cloud* as shown in Fig. 1. The S-CSP performs deduplication by checking if the contents of two files are the same and stores only one of them.

The access right to a file is defined based on a set of *privileges*. The exact definition of a privilege varies across applications. For example, we may define a *rolebased* privilege [9], [19] according to job positions (e.g., Director, Project Lead, and Engineer), or we may define a *time-based* privilege that specifies a valid time period (e.g., 2014-01-01 to 2014-01-31) within which a file can be accessed. A user, say Alice, may be assigned two privileges "Director" and "access right valid on 201401-01", so that she can access any file whose access role is "Director" and accessible time period covers 2014-0101. Each privilege is represented in the form of a short message called *token*. Each file is associated with some *file tokens*, which denote the tag with specified privileges (see the definition of a tag in Section 2). A user computes and sends *duplicate-check tokens* to the public cloud for authorized duplicate check.

Users have access to the private cloud server, a semitrusted third party which will aid in performing deduplicable encryption by generating file tokens for the



requesting users. We will explain further the role of the private cloud server below. Users are also provisioned with per-user encryption keys and credentials (e.g., user certificates). In this paper, we will only consider the filelevel deduplication for simplicity. In another word, we refer a data copy to be a whole file and file-level deduplication which eliminates the storage of any redundant files. Actually, block-level deduplication can be easily deduced from file-level deduplication, which is similar to [12]. Specifically, to upload a file, a user first performs the file-level duplicate check. If the file is a duplicate, then all its blocks must be duplicates as well; otherwise, the user further performs the block-level duplicate check and identifies the unique blocks to be uploaded. Each data copy (i.e., a file or a block) is associated with a token for the duplicate check.

- *S-CSP*. This is an entity that provides a data storage service in public cloud. The S-CSP provides the data outsourcing service and stores data on behalf of the users. To reduce the storage cost, the S-CSP eliminates the storage of redundant data via deduplication and keeps only unique data. In this paper, we assume that S-CSP is always online and has abundant storage capacity and computation power. • *Data Users.* A user is an entity that wants to outsource data storage to the S-CSP and access the data later. In a storage system supporting deduplication, the user only uploads unique data but does not upload any duplicate data to save the upload bandwidth, which may be owned by the same user or different users. In the authorized deduplication system, each user is issued a set of privileges in the setup of the system. Each file is protected with the convergent encryption key and privilege keys to realize the authorized deduplication with differential privileges.
- *Private Cloud.* Compared with the traditional deduplication architecture in cloud computing, this is a new entity introduced for facilitating user's secure usage of cloud service. Specifically, since the computing resources at data user/owner side are restricted and the public cloud is not fully trusted in practice, private cloud is able to provide data user/owner with an execution environment and infrastructure working as an interface between user and the public cloud. The private keys for the privileges are managed by the private cloud, who answers the file token requests from the

users. The interface offered by the private cloud allows user to submit files and queries to be securely stored and computed respectively.

Notice that this is a novel architecture for data deduplication in cloud computing, which consists of a twin clouds (i.e., the public cloud and the private cloud). Actually, this hybrid cloud setting has attracted more and more attention recently.

3 Proposed System Description

To solve the problems of the construction in Section 4.1, we propose another advanced deduplication system supporting authorized duplicate check. In this new deduplication system, a hybrid cloud architecture is introduced to solve the problem. The private keys for privileges will not be issued to users directly, which will be kept and managed by the private cloud server instead. In this way, the users cannot share these private keys of privileges in this proposed construction, which means that it can prevent the privilege key sharing among users in the above straightforward construction. To get a file token, the user needs to send a request to the private cloud server. The intuition of this construction can be described as follows. To perform the duplicate check for some file, the user needs to get the file token from the private cloud server. The private cloud server will also check the user's identity before issuing the corresponding file token to the user. The authorized duplicate check for this file can be performed by the user with the public cloud before uploading this file. Based on the results of duplicate check, the user either uploads this file or runs PoW.

Before giving our construction of the deduplication system, we define a binary relation R = $\{((p,p')\}$ as follows. Given two privileges p and p', we say that p matches p' if and only if $R(p,p') = 1$. This kind of a generic binary relation definition could be instantiated based on the background of applications, such as the common hierarchical relation. More precisely, in a hierarchical relation, p matches p' if p is a higher-level privilege. For example, in an enterprise management system, three hierarchical privilege levels are defined as Director, Project lead, and Engineer, where Director is at the top level and Engineer is at the bottom level. Obviously, in this simple example, the privilege of Director matches the privileges of Project lead and Engineer. We provide the proposed deduplication system as follows.

System Setup. The privilege universe P is defined as in Section 4.1. A symmetric key k_{p_i} for each $p_i \in$ P will be selected and the set of keys $\{k_{p_i}\}_{p_i \in P}$ will be sent to the private cloud. An identification protocol Π = (Proof, Verify) is also defined, where Proof and Verify are the proof and verification algorithm respectively. Furthermore, each user U is assumed to have a secret key sk_U to perform the identification with servers. Assume that user U has the privilege set P_U. It also initializes a PoW protocol POW for the file ownership proof. The private cloud server will maintain a table which stores each user's public information pk_U and its corresponding privilege set P_U. The file storage system for the storage server is set to be ⊥.

File Uploading. Suppose that a data owner wants to upload and share a file F with users whose privilege belongs to the set $P_F = \{p_j\}$. The data owner needs interact with the private cloud before performing duplicate check with the S-CSP. More precisely, the data owner performs an identification to prove its identity with private key sk_U. If it is passed, the private cloud server will find the corresponding privileges P_U of the user from its stored table list. The user computes and sends the file tag $\phi_F = \mathrm{TagGen}(F)$ to the private cloud server, who will return $\{\phi'_{F,p_\tau} = \mathrm{TagGen}(\phi_F, k_{p_\tau})\}$ If a file duplicate is found, the user needs to run the PoW protocol POW with the S-CSP to prove the file ownership. If the proof is passed, the user will be provided a pointer for the file. Furthermore, a proof from the S-CSP will be returned, which could be a signature on $\{\phi'_{F,p_\tau}\}$, pk_U and a time stamp.

The user sends the privilege set $P_F = \{p_j\}$ for the file F as well as the proof to the private cloud server. Upon receiving the request, the private cloud server first verifies the proof from the S-CSP. If it is passed, the private cloud server computes $\{\phi'_{F,p_\tau} = \mathrm{TagGen}(\phi_F, k_{p_\tau})\}$ for all p_τ satisfying $R(p,p_\tau) = 1$ for each $p \in P_F\text{-}P_U$, which will be returned to the user. The user also uploads these tokens of the file F to the private cloud server. Then, the privilege set of the file is set to be the union of P_F and the privilege sets defined by the other data owners.

- Otherwise, if no duplicate is found, a proof from the S-CSP will be returned, which is also a signature on $\{\phi'_{F,p_\tau}\}$, pk_U and a time stamp. The user sends the privilege set $P_F = \{p_j\}$ for the file F as well as the proof to the private cloud server. Upon receiving the request, the private cloud server first verifies the proof from the S-CSP. If it is passed, the private cloud server computes $\{\phi'_{F,p_\tau} = \mathrm{TagGen}(\phi_F, k_{p_\tau})\}$ for all p_τ satisfying $R(p,p_\tau) = 1$ and $p \in P_F$. Finally, the user computes the encrypted file $C_F = \mathrm{Enc}_{\mathrm{CE}}(k_F, F)$ with the convergent key $k_F = \mathrm{KeyGen}_{\mathrm{CE}}(F)$ and uploads $\{C_F, \{\phi'_{F,p_\tau}\}\}$ with privilege P_F.

File Retrieving. The user downloads his files in the same way as the deduplication system in Section 4.1. That is, the user can recover the original file with the convergent key k_F after receiving the encrypted data from the S-CSP.

$k_{F,p}$. In this way, both the private cloud server and S-CSP cannot decrypt the ciphertext. Furthermore, it is semantically secure to the S-CSP based on the security of symmetric encryption. For S-CSP, if the file is unpredicatable, then it is semantically secure too. The details of the scheme, which has been instantiated with hash functions for simplicity, are described below.

System Setup. The privilege universe P and the symmetric key k_{p_i} for each $p_i \in$ P will be selected for the private cloud as above. An identification protocol Π = (Proof, Verify) is also defined. The proof of ownership POW is instantiated by hash functions H, H_0, H_1 and H_2, which will be shown as follows. The private cloud server maintains a table which stores each user's identity and its corresponding privilege.

File Uploading. Suppose that a data owner with privilege p wants to upload and share a file F with users whose privilege belongs to the set $P = \{p_j\}$. The data owner performs the identification and sends $H(F)$ to the private cloud server. Two file tag sets $\{\phi_{F,p_\tau} = H_0(H(F), k_{p_\tau})\}$ and $\{\phi'_{F,p_\tau} = H_1(H(F), k_{p_\tau})\}$ for all p_τ satisfying $R(p, p_\tau) = 1$ and $p \in P_U$ will be sent back to the user if the identification passes. After receiving the tag $\{\phi_{F,p_\tau}\}$, and $\{\phi'_{F,p_\tau}\}$, the user will interact and send these two tag sets to the S-CSP. If a file duplicate is found, the user needs to run the PoW protocol POW with the S-CSP to prove the file ownership. If the proof is also passed, the user will be provided a pointer for the file. Otherwise, if no duplicate is found, a proof from the S-CSP will be returned, which could be a signature. The user sends the privilege set $P = \{p_j\}$ as well as the proof to the private cloud server for file upload request. Upon receiving the request, the private cloud server verifies the signature first. If it is passed, the private cloud server will compute $\phi_{F,pj} = H_0(H(F), k_{pj})$ and $\phi'_{F,pj} = H_1(H(F), k_{pj})$ for each p_j satisfying $R(p, p_j) = 1$ and $p \in P_F$, which will be returned to the user. Finally, the user computes the encryption $C_F = \text{Enc}_{\text{SE}}(k, F)$, where k is random key, which will be encrypted into ciphertext $C_{k,pj}$ with each key in $\{k_{F,pj} = \phi_{F,pj} \oplus H_2(F)\}$ using a symmetric encryption algorithm. Finally, the user uploads $\{\phi'_{F,pj}, C_F, C_{k,pj}\}$.

File Retrieving. The procedure of file retrieving is similar to the construction in Section 4.2. Suppose a user wants to download a file F. The user first uses his key $k_{F,pj}$ to decrypt $C_{k,pj}$ and obtain k. Then the user uses k to recover the original file F.

4 Security Analysis

Our system is designed to solve the differential privilege problem in secure deduplication. The security will be analyzed in terms of two aspects, that is, the authorization of duplicate check and the confidentiality of data. Some basic tools have been used to construct the secure deduplication, which are assumed to be secure. These basic tools include the convergent encryption scheme, symmetric encryption scheme, and the PoW scheme. Based on this assumption, we show that systems are secure with respect to the following security analysis.

Confidentiality of Data

The data will be encrypted in our deduplication system before outsourcing to the S-CSP. Furthermore, two kinds of different encryption methods have been applied in our two constructions. Thus, we will analyze them respectively. In the scheme in Section 4.2, the data is encrypted with the traditional encryption scheme. The data encrypted with such encryption method cannot achieve semantic security as it is inherently subject to bruteforce attacks that can recover files falling into a known set. Thus, several new security notations of privacy against chosen-distribution attacks have been defined for unpredictable message. In another word, the adapted security definition guarantees that the encryptions of two unpredictable messages should be indistinguishable. Thus, the security of data in our first construction could be guaranteed under this security notion.

We discuss the confidentiality of data in our further enhanced construction in Section 4.3. The security analysis for external adversaries and internal adversaries is almost identical, except the internal adversaries are provided with some convergent encryption keys additionally. However, these convergent encryption keys have no security impact on the data confidentiality because these convergent encryption keys are computed with different privileges. Recall that the data are encrypted with the symmetric key encryption technique, instead of the convergent encryption method. Though the symmetric key k is randomly chosen, it is encrypted by another convergent encryption key $k_{F,p}$. Thus, we still need analyze the confidentiality of data by considering the convergent encryption. Different from the previous one, the convergent key in our construction is not deterministic in terms of the file, which still depends on the privilege secret key stored by the private cloud server and unknown to the adversary. Therefore, if the adversary does not collude with the private cloud server, the confidentiality of our second construction is semantically secure for both predictable and unpredictable file. Otherwise, if they collude, then the confidentiality of file will be reduced to convergent encryption because the encryption key is deterministic.

5 Implementation

We implement a prototype of the proposed authorized deduplication system, in which we model three entities as separate C++ programs. A *Client* program is used to model the data users to carry out the file upload process. A *Private Server* program is used to model the private cloud which manages the private keys and handles the file token computation. A *Storage Server* program is used to model the S-CSP which stores and deduplicates files. We implement cryptographic operations of hashing and encryption with the OpenSSL library [1]. We also implement the communication between the entities based on HTTP, using GNU Libmicrohttpd [10] and libcurl [13]. Thus, users can issue HTTP Post requests to the servers.

Our implementation of the **Client** provides the following function calls to support token generation and deduplication along the file upload process.

- FileTag(File) - It computes SHA-1 hash of the File as File Tag;
- TokenReq(Tag, UserID) - It requests the Private Server for File Token generation with the File Tag and User ID;

- DupCheckReq(Token) - It requests the Storage Server for Duplicate Check of the File by sending the file token received from private server;
- ShareTokenReq(Tag, {Priv.}) - It requests the Private Server to generate the Share File Token with the File Tag and Target Sharing Privilege Set;
- FileEncrypt(File) - It encrypts the File with Convergent Encryption using 256-bit AES algorithm in cipher block chaining (CBC) mode, where the convergent key is from SHA-256 Hashing of the file; and
- FileUploadReq(FileID, File, Token) - It uploads the File Data to the Storage Server if the file is Unique and updates the File Token stored.

Our implementation of the **Private Server** includes corresponding request handlers for the token generation and maintains a key storage with Hash Map.

- TokenGen(Tag, UserID) - It loads the associated privilege keys of the user and generate the token with HMAC-SHA-1 algorithm; and

ShareTokenGen(Tag, {Priv.}) - It generates the share token with the corresponding privilege keys of the sharing privilege set with HMAC-SHA-1 algorithm.

Our implementation of the **Storage Server** provides deduplication and data storage with following handlers and maintains a map between existing files and associated token with Hash Map.

- DupCheck(Token) - It searches the File to Token Map for Duplicate; and
- FileStore(FileID, File, Token) - It stores the File on Disk and updates the Mapping.

6 Evaluation

We conduct testbed evaluation on our prototype. Our evaluation focuses on comparing the overhead induced by authorization steps, including file token generation and share token generation, against the convergent encryption and file upload steps. We evaluate the overhead by varying different factors, including 1) File Size 2) Number of Stored Files 3) Deduplication Ratio 4) Privilege Set Size . We also evaluate the prototype with a real-world workload based on VM images.

We conduct the experiments with three machines equipped with an Intel Core-2-Quad 2.66GHz Quad Core CPU, 4GB RAM and installed with Ubuntu 12.04 32Bit Operation System. The machines are connected with 1Gbps Ethernet network.

We break down the upload process into 6 steps, 1) Tagging 2) Token Generation 3) Duplicate Check 4) Share Token Generation 5) Encryption 6) Transfer . For each step, we record the start and end time of it and therefore obtain the breakdown of the total time spent. We present the average time taken in each data set in the figures.

any deduplication opportunity) of particular file size and record the time break down. Using the unique files enables us to evaluate the worst-case scenario where we have to upload all file data. The average time of the steps from test sets of different file size are plotted in Figure 2. The time spent on tagging, encryption, upload increases linearly with the file size, since these operations involve the actual file data and incur file I/O with the whole file. In contrast, other steps such as token generation and duplicate check only use the file metadata for computation and therefore the time spent remains constant. With the file size increasing from 10 MB to 400MB, the overhead of the proposed authorization steps decreases from 14.9% to 0.483 %.

Number of Stored Files

To evaluate the effect of number of stored files in the system, we upload 10000 10MB unique files to the system and record the breakdown for every file upload. From Figure 3, every step remains constant along the time. Token checking is done with a hash table and a linear search would be carried out in case of collision. Despite of the possibility of a linear search, the time taken in duplicate check remains stable due to the low collision probability.Deduplication Ratio To evaluate the effect of the deduplication ratio, we prepare two unique data sets, each of which consists of 50 100MB files. We first upload the first set as an initial upload. For the second upload, we pick a portion of 50 files, according to the given deduplication ratio, from the initial set as duplicate files and remaining files from the second set as unique files. The average time of uploading the second set is presented in Figure 4. As uploading and encryption would be skipped in case of duplicate files, the time spent on both of them decreases with increasing

deduplication ratio. The time spent on duplicate check also decreases as the searching would be ended when duplicate is found. Total time spent on uploading the file with deduplication ratio at 100% is only 33.5% with unique files.

Privilege Set Size

To evaluate the effect of privilege set size, we upload 100 10MB unique files with different size of the data owner and target share privilege set size. In Figure 5, it shows the time taken in token generation increases linearly as more keys are associated with the file and also the duplicate check time. While the number of keys increases 100 times from 1000 to 100000, the total time spent only increases to 3.81 times and it is noted that the file size of the experiment is set at a small level (10MB), the effect would become less significant in case of larger files.

image snapshots collected over a 12-week span in a university programming course, while the same dataset is also used in the prior work [14]. We perform blocklevel deduplication with a fixed block size of 4KB. The initial data size of an image is 3.2GB (excluding all zero blocks). After 12 weeks, the average data size of an image increases to 4GB and the average deduplication ratio is 97.9%. For privacy, we only collected cryptographic hashes on 4KB fixed-size blocks; in other words, the tagging phase is done beforehand. Here, we randomly pick 10 VM image series to form the dataset. Figure 6 shows that the time taken in token generation and duplicate checking increases linearly as the VM image grows in data size. The time taken in encryption and data transfer is low because of the high deduplication ratio. Time taken for the first week is the highest as the initial upload contains more unique data. Overall, the results are consistent with the prior experiments that use synthetic workloads.

7 Conclusion

In this paper, the notion of authorized data deduplication was proposed to protect the data security by including differential privileges of users in the duplicate check. We also presented several new deduplication constructions supporting authorized duplicate check in hybrid cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model. As a proof of concept, we implemented a prototype of our proposed authorized duplicate check scheme and conduct testbed experiments on our prototype. We showed that our authorized duplicate check scheme incurs minimal overhead compared to convergent encryption and network transfer.

References

[1] OpenSSL Project. http://www.openssl.org/.
[2] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In *Proc. of USENIX LISA*, 2010.
[3] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Serveraided encryption for deduplicated storage. In *USENIX Security Symposium*, 2013.
[4] M. Bellare, S. Keelveedhi, and T. Ristenpart.
[5] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. *J. Cryptology*, 22(1):1–61, 2009.
[6] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, pages 162–177, 2002.
[7] S. Bugiel, S. Nurnberger, A. Sadeghi, and T. Schneider. Twin clouds: An architecture for secure cloud computing. In *Workshop on Cryptography and Security in Clouds (WCSC 2011)*, 2011.
[8] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS*, pages 617–624, 2002.
[9] D. Ferraiolo and R. Kuhn. Role-based access controls. In *15 th NIST-NCSC National Computer Security Conf.*, 1992.
[10] GNU Libmicrohttpd. http://www.gnu.org/software/libmicrohttpd/.
[11] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 491–500. ACM, 2011.
[12] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou. Secure deduplication with efficient and reliable convergent key management. In *IEEE Transactions on Parallel and Distributed Systems*, 2013.
[13] A.Suresh (2016), “Speech Stress Analysis based on Lie Detector for Loyalty Test”, in International Journal of Printing, Packaging & Allied Sciences,(IJPPAS) ISSN: 2320- 4387, *Vol. 04, No.01, December 2016*, pp.*631* – 638.