International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Key Share Management to Protect Data Over Cloud

Prof. Priyanka Ambatkar[1], Prof. Reena Gadge[2]

[1]Dept. Computer Science & Engineering, DMIETR, Sawangi(M) Wardha, Maharashtra, India
[2]Dept. Computer Technology, YCCE, Hingna Road, Wanadongri, Nagpur, Maharashtra, India

Abstract: ***Cloud computing is not a new technology it is a new technique of computing. Data which is store over cloud is of sensitive nature that must be protect over cloud from being read and modify by intruder, User data may be stored in a cloud to take advantage of its scalability, accessibility, and economics. This triggered a lot of research activities, resulting in a quantity of proposals targeting the various cloud security threats. According to propose method works on key management and encrypt the data while sending that to the server and automatically get vanished based on passage of time or user activity. The process does not require additional coordination by the data owner, which is of advantage to a very large population of resource-constrained mobile users. By putting the access limit for users over the data can achieve good data confidentiality and integrity. The rate of expiration may be controlled through the initial allocation of shares and the heuristics for removal. A simulation of the scheme and also its implementation on commercial mobile and cloud platforms demonstrate its practical performance.***

Keywords: Distributed system, Mobile computing, Security, Cryptography

1. Introduction

As the number of cloud computing users has increased to infinity the various problems raised regarding storage. The data of sensitive nature must be protected over the cloud. Also by putting the access limit for users over the data can achieve good data confidentiality and integrity. According to propose method works on key management and encrypt the data while sending that to the server. Cloud computing systems offer nearly unbounded storage and computation for clients. In many applications, however, the provider of cloud services cannot be deemed to be sufficiently trustworthy to permit storing and processing of data in the clear. Given that contemporary cloud applications are accessed by potentially thousands of mobile device users, an encrypted cloud storage solution requires scalable key management. In addition, because many users will be operating resource-constrained devices, any security protocol employed must minimize the amount of communication sessions required. Current key management practices typically focus on key generation and distribution among a large population of users. The primary concern is that as authorized users join and leave a system, current keys must be re-generated and re-distributed to valid users, which is an unrealistic cost for mobile device users. Some approaches suggest performing computationally-intensive key re-generation operations within the cloud to take advantage of its scalability, but these computations may prove too expensive in certain applications where processing overhead is undesirable.

2. Existing System

Various access control techniques have been proposed for encrypted file storage in the cloud. The cloud provider typically controls key management activities, or the data owner or a trusted proxy does so if the provider is untrusted, requiring additional network communication and components [2]. In some mechanisms where control rests within the domain of the client, such as cloud-based data re-encryption, the ability of the provider to scale for computation has been exploited by performing intensive cryptographic computation in the cloud [3]. The cloud's potential for scalable storage, however, has apparently been under-utilized for key management operations. NIST (National Institute of Standards and Technology), in its Electronic Authentication Guideline [4], recommends secret sharing as a technique to be used to protect long-term credentials in its level 3 security definition for a CSP (Cloud Service Provider). Secret key sharing allows a secret such as key information to be divided into multiple shares [5]; these Shares may be distributed among key generators using the concept of threshold decryption [6], or portions of a private key are distributed among users [7]. The challenge is that the client must assemble a key from multiple sources, potentially resulting in expensive communication overhead. Rather than key shares being distributed on demand by some authority, it has been proposed that they be distributed across a network of nodes whose accessibility is subject to degradation over time. The Vanish system [8] distributes shares onto a DHT (Distributed Hash Table) that underlies a peer-to-peer file sharing network. It suggests the concept of "self-destructing data," where copies of data become unreadable over time due to the effect of user churn on the index. The problem with adapting the scheme to a cloud-based context is that it relies upon the availability of the shares among the nodes, which cannot be guaranteed. It requires that each user obtain key shares from multiple other nodes that form the index, which is an expensive proposition if the user is operating a mobile device. In the DEPSKY [9] storage system, shares are necessarily distributed across multiple clouds to form distributed trust and to restrict access. Each cloud provider has access to a single share and thus cannot decode the stored data; this requires support for a cloud-of-clouds. Also, because the data shares are unencrypted, each cloud must be independent and collusion assumed to be impossible. A straightforward approach employing PGP encryption [10] would encounter challenges with scalability; for instance, the symmetric key used for encryption of user data may need to be encoded with the public key of each recipient. Rather, it is preferable for a data owner to perform a one-time encryption. If the same private key is shared by all users, then revocation would require some form of authentication

to prevent access; the enforcement of it would require trust in the provider.

3. Proposed Methodology

A. Main Technique:

Key generation and encryption: Consider a technique based on Shamir's secret sharing [4]. U is the set of users accessing the cloud, and an access structure Γ_U is a list of subsets of U such that each subset is trusted. Any trusted subset U_{tr} of parties, where U_{tr}, Γ_U, can recover the secret from the set KS of shares stored in the cloud. Any untrusted subset, however, cannot obtain information about the secret. The access control structure can be defined such that any (t+1) or more parties in U can recover the secret, while any t or less cannot do so; this secret sharing scheme is threshold-based.

In the ENCRYPT operation, Alice, or user A, proceeds to generate key shares and encrypt a message m to be stored in the cloud and identified with a unique identifier mid. User A generates a symmetric key K and divides it into multiple shares KS[1] to KS[n], where n is the current total number of shares, and a minimum of t + 1 shares are required for decryption, where $t + 1 \leq n$. Parameter t may be decreased or increased in value for a corresponding adjustment in the level of security, while parameter n determines the number of users supported and the storage requirements for the shares. Each share KS[i] is encrypted as EKS[i], using a symmetric encryption key AK[i] belonging to user A, known as an access key; it is also possible for the same access key AK[i] to protect multiple shares, instead, to conserve associated storage and communication costs. The encrypted shares are stored in a key database in the cloud and cannot be read in plaintext form by the provider, although they remain accessible for download by users. Alice requests sufficient storage in the cloud to hold n shares, which represents an upper bound; as described later, a replacement strategy will replace older shares with newer ones while utilizing the same total capacity. The plaintext user data m requiring protection is assigned a unique record identifier of mid and encrypted by A as cipher text c using K, is uploaded to the provider, and is stored in the cloud. Since the cloud provider cannot unlock any share stored in the key database, it is unable to decode c. To the cipher text of the user data is appended a description key L identifying the set of key shares eligible to decrypt the data, of which only the threshold amount is required by any user.

Metadata: In order to detect malicious modification of a share by the cloud provider, A will also create a signed metadata header EK Shdr for each share and upload it with the encrypted share payload. The metadata will consist of the following fields: the record identifier mid, the key share identifier i {0..n}, the key share version v, and a digest KS[i]dig of the key share content, such as a cryptographic hash of it. The metadata is signed with A's private key SKA. This verifiability is an optional feature of the algorithm, and is not strictly required if the cloud provider is regarded as being honest-but-curious since it is not expected to modify c.

Decryption: Bob, or user B, wishes to access c, and so he executes the DECRYPT operation. Suppose that B is an authorized member of Utr. B obtains symmetric access keys AK[x] to AK[y] from A, where the range of keys is of at least size t + 1, the required threshold; this assumes that each key AK[i] permits decryption of the key share KS[i] stored in the cloud, where i is in the range 1 to n, such that all shares are in L. Again, it is also possible to have one access key unlock multiple shares, instead. Regardless, every user is given a random set of access keys in the initial allocation; each set satisfies the threshold at a minimum. Optionally, to ensure that a share downloaded from the cloud is the correct one and it has not been modified by the cloud provider, B may inspect the metadata associated with the cipher text, and decode it using A's known public key P KA. The digest in the metadata can then be compared against the one computed from the downloaded share to detect tampering. For instance, in one sample configuration, suppose that the total number of shares n for a particular data record is 100, and that the number of shares required for decryption, t + 1, is 3. Refer to Figure 1. Data owner A will provide access key AK[1] to B, which provides access to 5 shares stored in the cloud; only 3 are required. B may download the cipher text, as well as all three encrypted key shares, directly from the cloud. B will then be able to decrypt the required user data. The entire user population may be significantly greater in number than the total number of shares n; thus, the same key shares may be randomly assigned to multiple users. For instance, Charlie, user C, is also an authorized member of Utr and is allowed to access the cipher text c. Owner A may issue the same access key to C, to unlock the same key shares in common with user B. Note that if an access key unlocks only a single key share, instead, then finer-grained control is attained; in that case, user B can be given access keys AK[1] to AK[5] to unlock shares EKS[1] to EKS[5], and user C can be given access keys AK[3] to AK[7] to unlock overlapping shares EKS[3] to EKS[7]. However, this flexibility is at the cost of additional storage for each user.

Key share deletion: Over time, individual key shares in the cloud are independently deleted by the cloud provider. This process can occur at regularly scheduled time intervals, such that a random share is deleted every day, for instance. If user B had locally cached all decrypted key shares, then B will continue to be able to decrypt data from the cloud until his cache needs to be refreshed, or the entire key store expires. Suppose that key share KS [1] is erased and that B needs to re-fetch key shares from the cloud. B will find that EKS[1] (the encrypted version of KS[1]) is no longer available, and so another key share must be randomly chosen from the available set. B will be required to use an appropriate access key in the set AK[x] to AK[y] to access another available key share outside of the initial three, such as KS[4]. Any users that hold the same deleted share may also need to do the same. If three key shares of the initial set of five are deleted, then user B cannot satisfy the threshold, and will be unable to decrypt the user data. If B holds no other access keys, he may optionally obtain AK[z] from A, where z is an access key that was not previously held, and which unlocks a valid remaining key share from the cloud. B may obtain this key

from A or from another user in U_{tr}. Otherwise, B must wait for the key store to expire and new valid key shares to become available.

Keys may be deleted according to different schedules, such as based on regular time intervals, or based on the number of accesses of the user data, or the number of joins and leaves of users in the user set. The size of the valid remaining key store in the cloud will decrease from the initial maximum until the store is re-generated. Access keys must only be re-generated for shares that belonged to users whose access rights were revoked from the last time that shares were generated. In other words, if no user left the authorized user membership Utr in the last round, then key shares will require replacement after deletion, but the access keys held by users need not be updated. To effect control over access key replacement, an expiration flag may be set by the data owner for each access key, if the key unlocks a share that was assigned to a user whose access rights have been revoked; the tradeoff made is between additional record-keeping and the computation of new keys.

Key share replacement: Once the number of outstanding valid shares subject to random replacement decreases to t, it becomes impossible for users to download sufficient valid shares to replace those that are deleted, even if additional access keys are obtained from the data owner or other users. The key store in the cloud then expires; this event can also occur at a prearranged point before the threshold is reached. The content owner A can then proceed to replace the deleted shares in the cloud with newly-generated valid shares of a new version of the symmetric key K. A new access key will also be generated for each share, or set of shares, to protect them. Thus, for instance, key AK [1] will protect the new key shares KS [1] to KS [5], from 1 to 5, and so on; a total of n key shares are again stored in the cloud with headers reflecting the new version. In this case, the user data that is stored in the cloud and encrypted with the older key K must be replaced with a version that is encrypted with the new key K ; this may be done by user A. To avoid a key consistency issue, the cipher text may be appended with information on the key version required to decrypt it.

Revocation: Suppose that user A decides that B should no longer have access to the encrypted user data stored in the cloud. B will be unable to obtain additional access keys from A to obtain more shares, will be unable to obtain updated access keys once the key store expires, and thus will be unable to decrypt shares of the new key K from the cloud. Note that B will not immediately lose access to the originally given access keys. B will eventually lose access rights when sufficient key shares are deleted non-deterministically by the cloud provider and B's key cache cannot be refreshed.

Symbol	Description
U_{tr}	Authorized user set.
A, B, C	Users Alice, Bob, and Charlie in U_{tr}.
$GenKey()$	Function to generate a random key of some predetermined length.
K	Symmetric data key.
$KS[i]$	Share i of key K.
$v[i]$	Version associated with a key share $KS[i]$.
$EKS[i]$	Encrypted key share i.
$KS[i]_{hdr}$	Metadata header for key share $KS[i]$.
$KS[i]_{dig}$	Digest of key share $KS[i]$.
$EKS[i]_{hdr}$	Signed metadata header.
$Partition(K, i, n)$	Function to generate share i of key K, where n is the total number of shares.
$Encrypt_y(x)$	Function to encrypt data x using key y.
$Decrypt_y(x)$	Function to decrypt data x using key y.
$Hash(x)$	Compute the digest of message x.
$Reconstruct([z])$	Function to reconstruct secret key from shares in array $z[]$.
$Compare(x, y)$	Function to perform a bitwise comparison of values x and y.
$AK[i]$	Access key to unlock share $KS[i]$.
m	Plaintext of user data.
m_{id}	Unique plaintext record identifier.
c	Ciphertext of user data.
t	The threshold number of key shares KS, above which (at $t+1$ or greater), there is a sufficient number to compute K.
n	The total number of key shares KS generated for a particular data record.
L	A description key identifying the set of key shares eligible to decrypt c.
PK_X	Public key of user X.
SK_X	Private (secret) key of user X.

Table I: Legend for symbolic notation

4. Implementation

It is browser based application and can easily be deployed on any open source or paid cloud. The application includes modules like user registration, login, file sending and receiving to and from cloud. The send files module includes sending particular file to any user of the same application. You can send files to multiple users at the same time. No files will be available to unauthorized user. File sending means using cloud to store files on cloud storage using access control method, so that only particular user can access the data. During transmission of files over cloud, files has been saved using secure way so that even untrusted cloud cannot tamper with users original data. First we encrypt the file using AES encryption method. This step also generates the key and key share for each user to access this file on cloud.

When the other user access the file on cloud it actually uses one key share and then download the file from cloud. After all key shares used, the user may no access that file. Key shares are also encrypted while sending to CSP. Only authorized user can decrypt the key shares. CSP does not contain key to decrypt the key shares, so not able to use actual files.

Here AES algorithm is used to develop encryption key, AES is based on a design principle known as a substitution-permutation network, combination of both substitution and permutation, and is fast in both software and hardware.[8]Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which

has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per se is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits. AES operates on a 4×4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field. The key size used for an AES cipher specifies the number of repetitions of transformation rounds that convert the input, called the plaintext, into the final output, called the cipher text.

5. Result

If the key is of 12 bit there are 3 shares each of 4 bit.

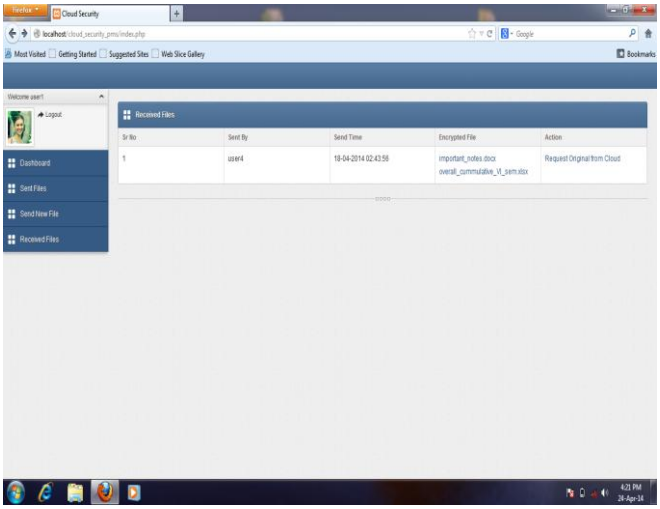

Figure 1: Received file by another user

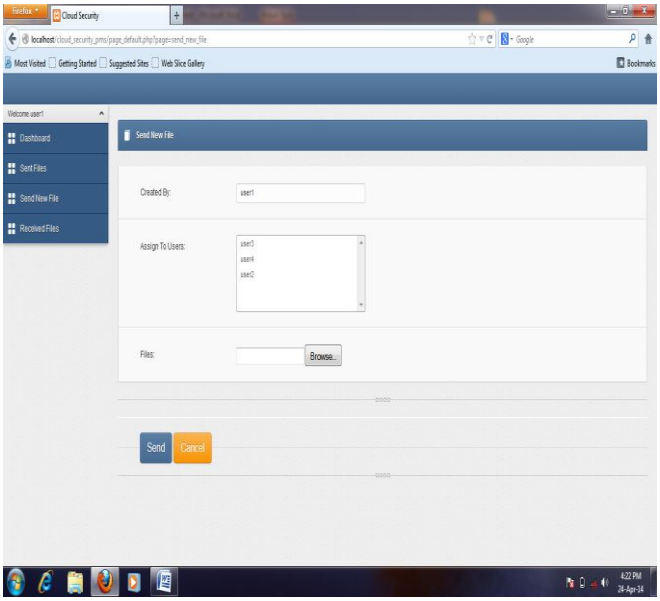

Figure 2: If sender wants to send New File

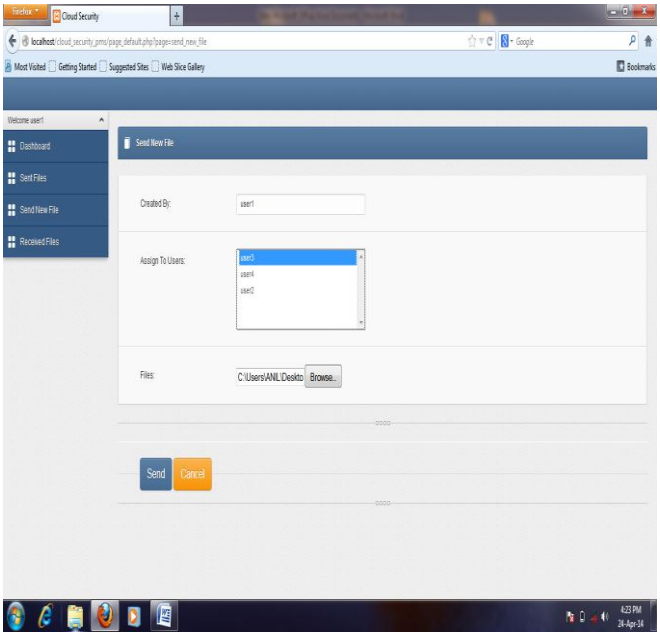

Figure 3: Sender selected file to be send

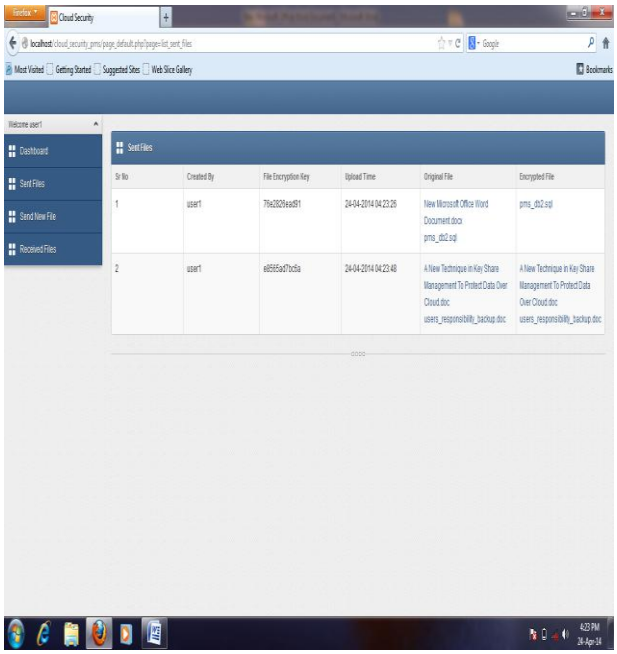

Figure 4: How many files one user sending file to another user

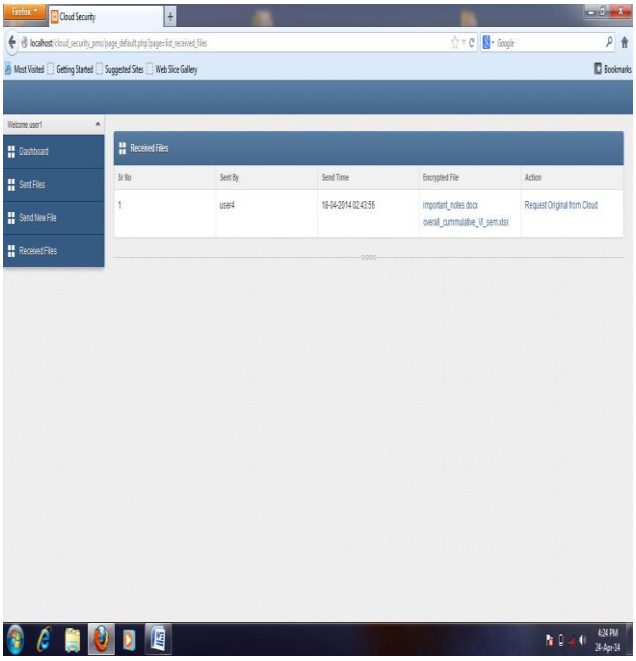

Figure 5: At receiver side it shows how many files received

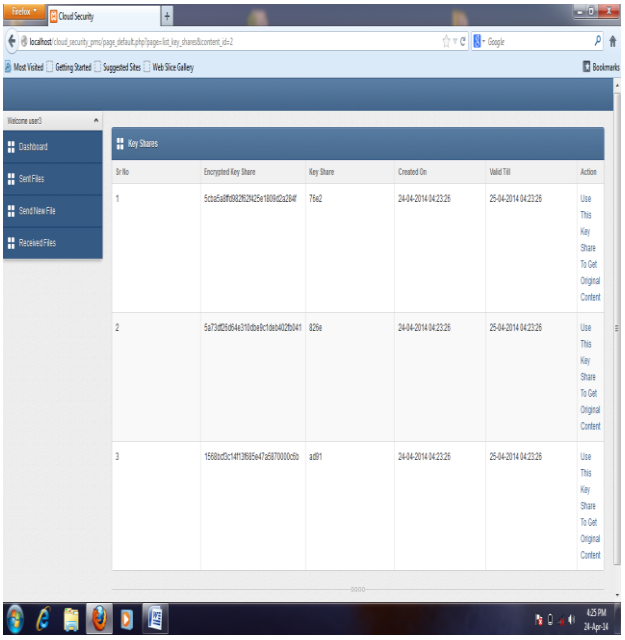

Figure 7: It will show 3 Key shares

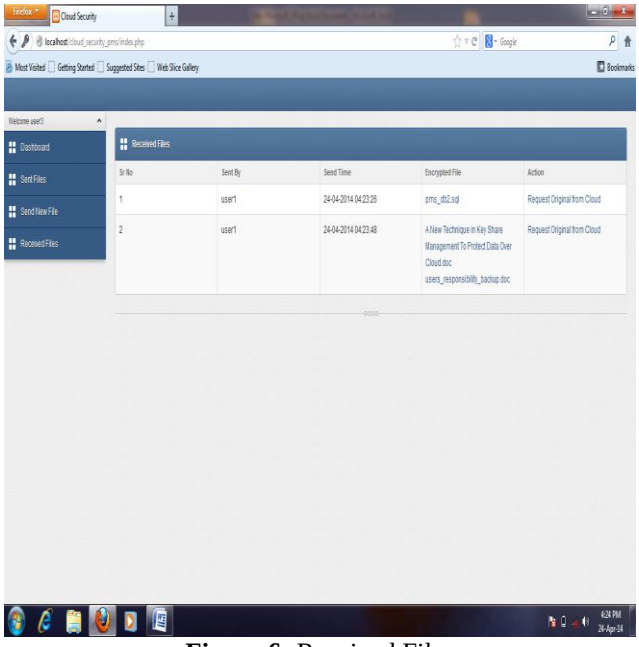

Figure 6: Received Files

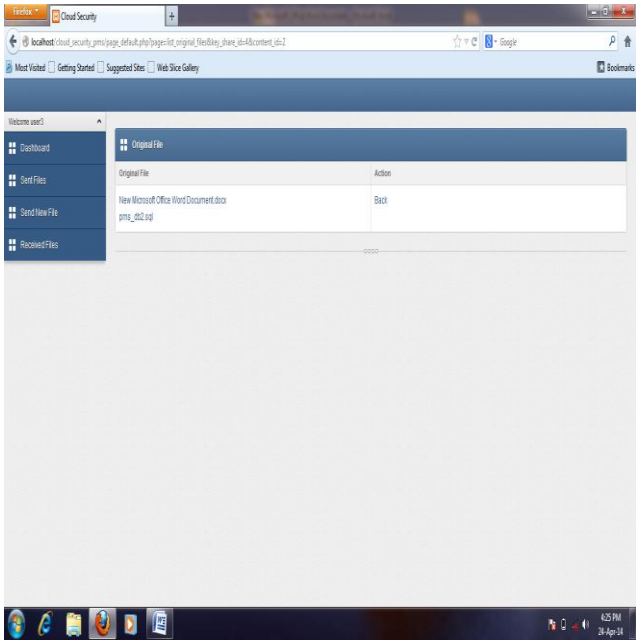

Figure 9: Using key share original file can access

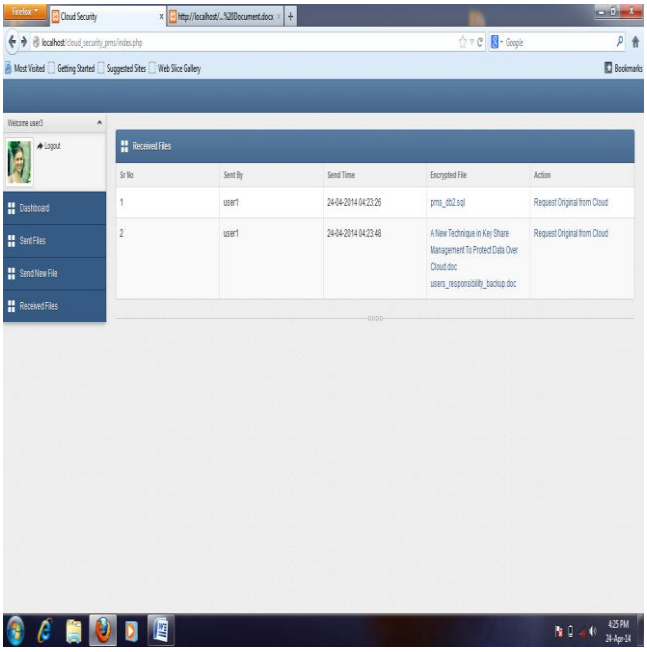

Figure 8: Two key share remaining

6. Conclusion

It has been demonstrated that scalable key management may be attained by leveraging the inexpensive storage capacity and high accessibility offered by a cloud provider. One of the benefits of using centralized and reliable cloud storage for key shares is that there is full control over share management; it is not subject to outside factors such as user churn. Thus, various additional heuristics for key share deletion may be explored. For instance, high-priority or trustworthy users could retain key shares for longer or be assigned a greater number.

The use of multiple cloud providers for gaining security and privacy benefits is nontrivial. As the approaches investigated in this paper clearly show, there is no single optimal approach to foster both security and legal compliance in an omni applicable manner. Moreover, the approaches that are favorable from a technical perspective appear less appealing from a regulatory point of view, and vice versa. The few approaches that score sufficiently in both these dimensions lack versatility and ease of use, hence can be used in very rare circumstances only. As can be seen from the discussions of the four major multi-cloud approaches, each of them has its pitfalls and weak spots, either in terms of security guarantees, in terms of compliance to legal obligations, or in terms of feasibility. Given that every type of multi-cloud approach falls into one of these four categories, this implies a state of the art that is somewhat dissatisfying.

References

[1] P. Tysowski and M. A. Hasan, "Towards Secure Communication for Highly Scalable Mobile Applications in Cloud Computing Systems," Centre for Applied Cryptographic Research, University of Waterloo,Tech. Rep. CACR 2011-33, 2011.
[2] S. Jahid, P. Mittal, and N. Borisov, "EASiER: encryption-based access control in social networks with efficient revocation," in Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ser. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 411–415.
[3] W. E. Burr, D. F. Dodson, E. M. Newton, R. A. Perlner, W. T. Polk, S. Gupta, and E. A. Nabbus, "Electronic Authentication Guideline," National Institute of Standards and Technology (NIST), Tech. Rep. Special Publication 800-63-1, December 2011.
[4] A. Shamir, "How to share a secret," Commun. ACM, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[5] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in Advances in Cryptology — CRYPTO 2001, ser. Lecture Notes
[6] in Computer Science, J. Kilian, Ed. Springer Berlin / Heidelberg, 2001, vol. 2139, pp. 213–229.
[7] J. Baek and Y. Zheng, "Identity-Based Threshold Decryption," in PublicKey Cryptography – PKC 2004, ser. Lecture Notes in Computer Science, F. Bao, R. Deng, and J. Zhou, Eds. Springer Berlin / Heidelberg, 2004, vol. 2947, pp. 262–276.
[8] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy, "Vanish: Increasing data privacy with self-destructing data," in Proc. of the 18th USENIX Security Symposium, 2009.
[9] A. Bessani, M. Correia, B. Quaresma, F. Andr´e, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," in Proceedings of the sixth conference on Computer systems, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 31–46.
[10] N. Leavitt, "Is Cloud Computing Really Ready for Prime Time?" Computer, vol. 42, pp. 15{20, January 2009