Asian Journal of Information Technology 15 (20): 3973-3979, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Cryptographic Tree and Log File Based Secret Sharing Key Management Scheme for Securing Outsourced Data in Cloud

[1]S. Ahamed Ali and [2]M. Ramakrishnan
[1]Department of Information Technology, Velammal Engineering College, 66 Chennai, India
[2]School of Information Technology, Madurai Kamaraj University, Madurai, India

Abstract: Cloud computing from the last few years has grown from a promising business concept to one of the fastest growing segments of IT Enterprise. This unique paradigm brings about many new security challenges. This research studies the problem of ensuring the storage security to the data outsourced in the cloud which is accessed and controlled by multiple legitimate parties. For this we propose a very practical approach of providing security to the data outsourced in the cloud environment based on both Cry ptographic Tree and Log File (CTLF).The Cryptographic tree and log file is used to authenticate a set a data items and preserve block level integrity to the outsourced data. This approach generates a master key for encryption and decryption using the primary key k_p which is provided by the cloud client and the secondary key k_s, the primary key k_p is secret shared among "n" share holders nodes. This scheme is more reliable, decentralized light weight key management scheme which provides better security and performance than existing ones. This scheme also allows multiple legitimate parties to access the data outsourced in the cloud without making compromise on security.

Key words: Cloud computing, data outsourcing, cryptographic tree, log file, master key, secret sharing

INTRODUCTION

Cloud computing is a cutting edge technology which is widely used in all applications where efficient management of resources is considered as a prime factor. It provides computation, software, data access and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. Cloud computing is transforming the very nature of how businesses use information technology. One fundamental aspect of this paradigm shifting is that data is being centralized or outsourced into the Cloud. From users perspective, including both individuals and IT enterprises, storing data remotely into the cloud in a flexible on-demand manner brings appealing benefits: relief of the burden for storage management, universal data access with independent geographical locations and avoidance of capital expenditure on hardware, software and personnel maintenances, etc.

Cloud computing offers us an efficient and cost effective schemes without any compromise in regard to performance, control or flexibility which is expected in near future. Cloud computing is a technology which is preferred by many organizations to relocate their resources and maintain them outside their enterprise, regardless of the location of the cloud server. Due to the nature and demand of emerging cloud technologies, there is a certain degree of inexperience when dealing with cloud security.

The data or the information that is stored in a cloud server is referred as data outsourcing and generally they are managed by a third party. So when the data or application is being outsourced by an enterprise, their privacy becomes a very challenging task because cloud computing clients have to trust third party cloud providers on many fronts, especially on the availability of cloud service as well as data security. Therefore the Service Level Agreements (SLAs) forms an integral part of a client's first line of defense. Further security defenses are hosted along with the other user data on the cloud and are controlled by the service-providers. Thus other than the SLAs user can't hold a complete control over data security over his own critical data.

The situation becomes even more complicated when the outsourced data is accessed by multiple users having different access rights. So the present situation becomes even more complicated when a large number of organizations outsource the data or application, so issues to be given prime importance are security and privacy.

Corresponding Author: S. Ahamed Ali, Department of Information Technology, Velammal Engineering College, 66 Chennai, India

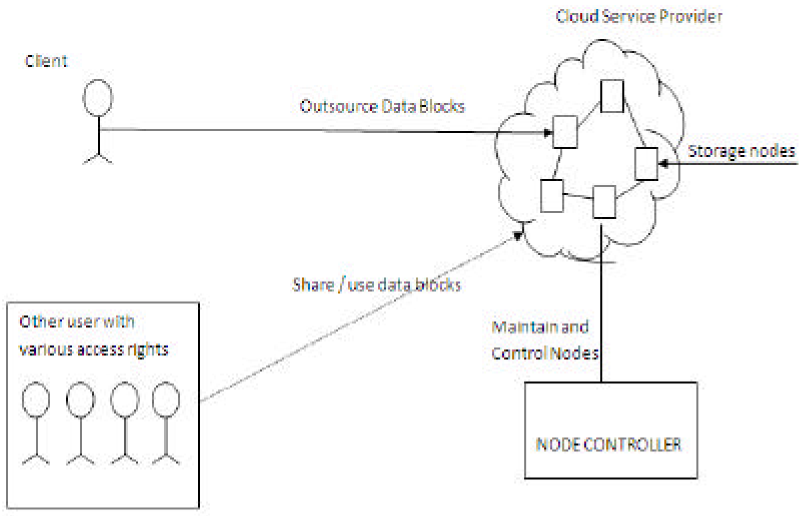


Fig.1: Data outsourcing model

Literature review: Tree-based cryptographic key management scheme was proposed by (Zhou *et al.,* 2012) to provide security to the outsourced data, where a single root node holds the master key that can be used to derive other node keys. Each node key can be used to derive the keys of its children in the hierarchy. With that scheme, a data block stored in the cloud can be updated by a party who holds either the specific decryption key or a node key corresponding to one of its parents (Atallah *et al*., 2005, 2009) proposed "Dynamic and efficient key management for access hierarchies". The hierarchy is modeled as a set of partially ordered classes (represented as a directed graph) and a user who obtains access (i.e., a key) to a certain class can also obtain access to all descendant classes of her class through key derivation. The major problem is that the security of their scheme relies only on the use of pseudo-random functions shared with other authorized parties. Tree based cryptographic key management scheme for outsourced data in cloud was also proposed by Wang *et al*. (2009, 2010) in their scheme the outsourcing server is authorized to manage a node other than the root node, thus the outsourced party is granted the node key which can be used to derive all sub-keys for its child nodes. The other schemes such as (Blundo *et al*., 2009, Damiani *et al*., 2005, 2007; DiVimercati *et al*., 2007) are used to minimize the number of cryptographic keys being stored but have many limitations. In all these schemes once a key for a parent node in the tree is given and then all the child node keys can be derived. This is a common problem that exists in many tree-based key management schemes.

Considering these problems, we propose a very practical approach of providing security to the data outsourced in the cloud environment which is referred to as "cryptographic tree and log file based secret sharing key management scheme for securing outsourced data in cloud**"**, wherein each node in the tree consists of a encrypted data which can be managed or shared with another trusted party along with a secured log file without making compromise on security.

System model: In this study we present the cryptographic model used for the scenario in Fig. 1. The system consists of the following communication parties.

The cloud storage server (S): A third party who provides storage services to the users. The Storage server partitions the data outsourced by the C into fixed length blocks and perform encryption

The cloud Client (C): Who outsources the data in the cloud and generate / provide the primary key (K_p). The cloud client will also provide the secret pin for generating the pseudo shares.

The node controller: This server control and update the log file and computes the partial log file key value(L_f).

The Key management server (K): It is a server that splits the encryption keys into different shares and stores the splitted keys in different share holder nodes. The key management server generates the master key for encryption and decryption process.

The User (U): who has limited access rights to share or use the data block in a particular node in the tree.

MATERIALS AND METHODS

Cryptographic tree structured storage: In this study we review the construction of our cryptographic tree (CT) which is derived from (Buchmann *et al.,* 2007) The data outsourced in the cloud by cloud client is divided into equal sized block B1,B2 ...Bn ,by the cloud storage server 'S' and each block is encrypted individually with the encryption key of the ?le and they are stored in the storage nodes managed by the 'S'. The storage nodes are maintained in a hierarchical tree based structure called cryptographic tree. The CT is used for authenticating a set of data items and provide integrity to the files. The CT for data items D1,...,Dn, denoted CT (D1, ..., Dn), is a binary tree that has D1, ..., Dn as leaves.

We define the cryptographic tree (CT) for a fille F with n blocks B1, ... , Bn to be the binary tree (which can also be extended to a general tree) CTF = CT ($h(L_{f1}||B1),...,h(L_{fn}||Bn)$). The Log fille value L_f(described in the next section) is concatenated with the block index and hashed together. This is done to preserve the integrity and also to prevent block swapping attacks. A CT with a given set of leaves can be constructed in multiple ways. Here we choose to append a new block in the tree as a right-most child, so that the tree has the property that all the left subtrees are complete. We give an example in Fig. 2 for a cryptographic tree that has four blocks.

We define the following notations and algorithms for ensuring block level integrity to cryptographic tree.For a tree CT, CT.root is the root of the tree, CT.leaf[i] is the i-th leaf in the tree counting from left to right. For a node 'e' in the tree, e.hash is the hash value stored at the node and for a non-leaf node e, e.left and e.right are pointers to the left and right children of e, respectively. e.sibling and e.parent denote the sibling and parent of node e, respectively. We also define two algorithms namely CT. Evaluate Root Path (R, e) where R is the root and e is a node and CT. Evaluate C Tree (R, I, hval) as follows

In the CT.EvaluateRootPath(R,e) algorithm for tree CT, the hashed value stored in the nodes on the path from node e to the root of the tree are computed, by reading all the hashed values stored in the siblings of those nodes. Finally, the hash of the root of CT is checked to match the parameter R. If the hash stored in the root matches the parameter R, then the client is assured that all the siblings read from the tree are authentic.

```
CT.EvaluateRootPath(R,e)
 e.hash = e.hash || Gen_L_f()
 hval← e.hash
while v ≠ CT.root
 s← e.sibling
if e.parent.left = e
 hval ← h(hval||s.hash)
else
 hval ← h(s.hash||hval)
e← e.parent
 if R = hval
 return true
else
 return false
```

The CT.EvaluateCTree (R, I, hval) algorithm for tree CT checks that the hash stored at the i-th leaf matches hval. Gen_L_f() algorithm computes Log file value L_f. Using algorithm CT.EvaluateRootPath(R,e) all the hashed value stored at the nodes on the path from the i-th leaf to the root are computed and the root of CT is

```
CT.EvaluateCTree(R,i,hval)
if CT.leaf[i].hash ≠ hval
 return false
return CT. EvaluateRootPath (R,CT.leaf[i])
```

Generation of log file key value (l_f): A separate log file is being maintained to enhance the security to the outsourced data in the cloud. The log file is maintained by the node controller (third party who provides services in the cloud). The node controller periodically updates the log file, when a user views or shares the data block. This log file is invisible to the users and this file is used to derive the partial log file key value). The information updated in the log file are as follows

- Date
- Time
- User name (U)
- Block Id
- Mode of access (Reading (R) or Writing (W))
- A random number ()

Based on the information stored in the log file, we calculate the partial log file key value (L_f) (Fig. 3).

Derivation of: The following parameters are used for computing L_f

$$U_{lf} = U_1+U_2+U_3+....+U_n \tag{1}$$

For Data block-m and User-n,

$$U_n = \sum_{i=1}^{j}\left((c_i * H(d_i) + R_f\right) \tag{2}$$

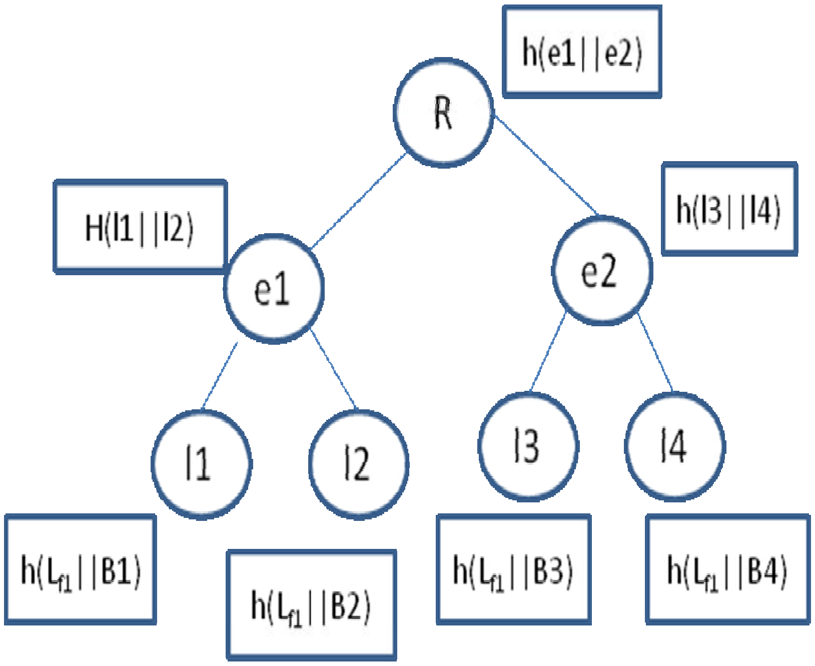
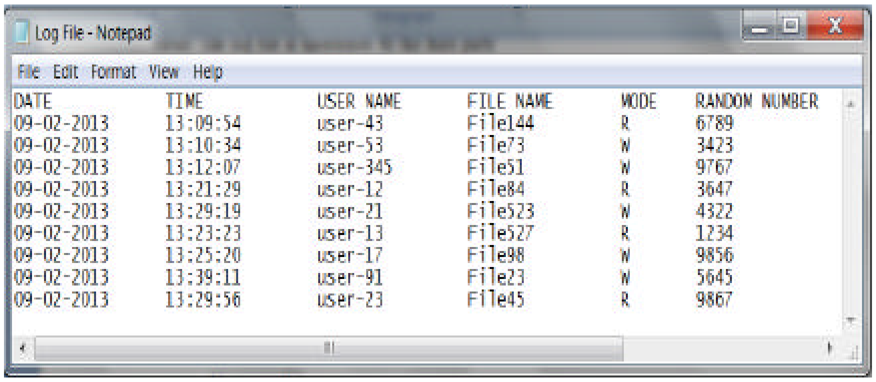


Fig. 2: Structure of cryptographic tree



```
DATE        TIME       USER NAME   FILE NAME   MODE   RANDOM NUMBER
09-02-2013  13:09:54   user-43     File144     R      6789
09-02-2013  13:10:34   user-53     File73      W      3423
09-02-2013  13:12:07   user-345    File51      W      9767
09-02-2013  13:21:29   user-12     File84      R      3647
09-02-2013  13:29:19   user-21     File523     W      4322
09-02-2013  13:23:23   user-13     File527     R      1234
09-02-2013  13:25:20   user-17     File98      W      9856
09-02-2013  13:39:11   user-91     File23      W      5645
09-02-2013  13:29:56   user-23     File45      R      9867
```

Fig. 3: Sample log file

$$TR_n = \sum_{i=1}^{j} R_f \qquad (3)$$

Where:

m	=	The number the data blocks in the cloud server at any time t
n	=	Denotes the number of users sharing the cloud server
j	=	Total number of times the data block is being read or modified since the data block is outsourced in the cloud
C_i	=	Number of times the data block is being read or modified by the user-m during a specific time period
d_i	=	Denotes the value of the timestamp during which the data block is being read or modified by the user-m
$H(d_i)$	=	Denotes the hashed value of the timestamp during which the data block is being read or modified by the user-m
R_f	=	Random number
Tr_n	=	Sum of random numbers(R_f)

Calculation of U_{lf}: Similarly, we calculate for all n users in the cloud for a particular file data block using the recursive Equation:

$$U_{lf} = \sum_{i=1}^{n} U_n \qquad (4)$$

$$TR_{1f} = \sum_{i=1}^{n} TR_n \quad (5)$$

L_f for a data block can be calculated using eqn.6

$$L_f = U_{1f} * TR_{1f} \quad (6)$$

The calculated L_f is used in key generation.

RESULTS AND DISCSUSSION

Share generation scheme: The cloud client which outsources the data will provide the primary key k_p and it will be combined with secondary key k_s to generate the master key K which is used for encryption and decryption process. Initially, the primary key k_p is encoded using the pyramid code technique (Huang and Li, 2007). The encoded primary key k_p value is given as input to an one way collision resistant hash function.

The hashed primary key is divided into n secret shares by using a random polynomial of degree n, n denotes the number of share holder nodes in which the secret is distributed. The usage of random polynomial enhances the security. For example, if the number of share holder nodes on which the secret is shared 4 (i.e when n=4), then the random polynomial generated will be of the form:

$$f(x) = a_1x^3+a_2x^2+a_3x+a_4 \quad (7)$$

where, a_1,a_2,a_3,a_4 are random coefficients which are given as input by the user, by using the random polynomial, n shares can be obtained as follows

$$S_i = f(i), i = 1\text{-}n \quad (8)$$

If the secret pin given by the user is 5639 then using the polynomial equation and we get $5x^3+6x^2+3x+9$, where 5, 6, 3, 9 are the values for $a_{1,}$ $a_{2,}$ $a_{3,}$ a_4 respectively. The secret shares generated are as follows:

$$S_1 = 23;\ S_2=79;\ S_3 = 207;\ S_4 = 437$$

Once the shares S_1, S_2, …,S_n are obtained the next step is to compute the pseudo share for each of the sub shares. The pseudo share is computed using the cube hash method by cyclic concatenation of the subsequent sub shares and performing XOR with the share S_i. Once all the pseudo shares are generated, they are stored in n share holder nodes. The hashed values are also stored in the share holder servers in order to obtain the n shares for the reconstruction of original data. Thus the shares that are generated using the secret pin 5369, will be as follows:

$$S_1 = H(X_1) = H(S_1||S_2) = H(2379)$$

$$S_2 = H(X_2) = H(S_2||S_3) = H(79207)$$

$$S_3 = H(X_3) = H(S_3||S_4) = H(207437)$$

$$S_4 = H(X_4) = H(S_4||S_1) = H(43723)$$

Once all the hash values are obtained, the shares are XORed with the corresponding hash value to obtain the pseudo shares. The pseudo shares for the above generated shares and hash values are as follows:

$$S_{11} = S_1 \text{ XOR } H(X_1)$$

$$S_{22} = S_2 \text{ XOR } H(X_2)$$

$$S_{33} = S_3 \text{ XOR } H(X_3)$$

$$S_{44} = S_4 \text{ XOR } H(X_4)$$

These n pseudo shares are stored in n share holder nodes.

Generation of master key using complexity function F: The Key management server uses the complexity function F to derive the master key for encryption and decryption. Every time when the user request to outsource the data in the cloud the Key management server generates a unique secondary key called "k_s". The Key management server generates the master key K for encryption/decryption process by using k_s and K_p. Equation 9 represents the complexity function F which is used to generate the master key 'K'. The function F takes k_s and the regenerated k_p as input and produces K as output. The output is created by splitting k_s into two halves (k×1 and k×2) concatenated with the K_p and performing one way hash function on each half. The two results are XORed together and the result is hashed using HMAC algorithm to produce the master key K.

$$K = HMAC(H(k\times 1||k_s) \oplus H(k\times 2||k_s)) \quad (9)$$

Encryption and decryption: We use write counters that will automatically update every time when a write operation is performed on the storage block. The updated counter value is used to generate initial vector which is used to construct stateful encryption and decryption scheme. Let (Gen_IV,E,D) be an encryption scheme constructed from a block cipher in counter mode. To encrypt a n-block message in the counter encryption mode, a random initialization vector is chosen. The ciphertext consists of n + 1 blocks with the frst being the initialization vector. We denote by $E_k(B,iv)$ the output of the encryption of B using the master key k and initialization vector iv and similarly by $D_k(C,iv)$ the decryption of C using key k and initialization vector iv.

We replace the random initialization vector for encrypting a block in the file using counter mode with a function Gen_IV() that generates a hashed value of the write counter for the block concatenated with the index of the storage node for the block in the cryptographic tree. This is very secure because different initialization vectors are used for different encryptions of the same block.

The Gen_IV,E,D algorithms for stateful encryption scheme is described below here B_i denotes the current block, K_{PRF} denotes a pseudo random function that generates a session key $k_{session}$ from the master key K, UpdCtr(bid) is a function that returns write counter for the storage block B_i

```
Gen_IV()
$k_{session}$- $K_{PRF}$
iv- $E_{ksession}$ (H(index||UpdCtr(bid))
return<iv>

$E_k$ (bid,$B_i$)
Update(bid)
Gen_IV()
C = $E_k$(B, iv)
return C

$D_k$ (bid,C)
Gen_IV()
B = $D_k$(C, iv)
return B
```

Performance analysis: In this study we report on performance measurements of our proposed CTLF based approach. We initially consider the time required to split and recover the key for encryption and decryption purpose. Figure 4 shows the split and recovery time for the secret key based on the number of shares. It is observed that the time increases linearly with increase in the number of shares. Figure 5 shows the split and recovery time for the secret key based on the size of the key. It is observed that the time increases linearly with increase in the size of the key. Our secret sharing scheme has good computational security as the generated share has high degree of randomness as the attacker cannot generate secret by compromising any 'n' key storing nodes.

We present an evaluation for the text and compressed image file sets. We plot the total time to write and read the set of text and compressed image files,

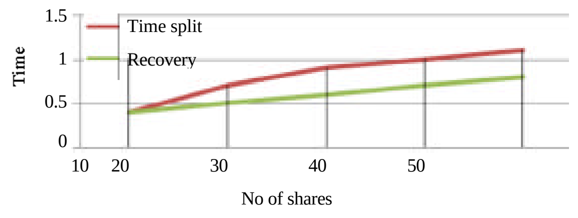


Fig. 4: Split and recovery time based on no. Of shares

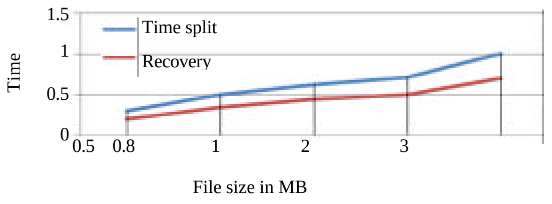


Fig. 5: Split and recovery time based on size of the file

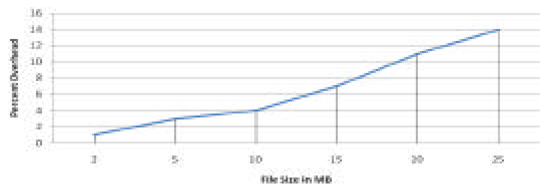


Fig. 6: Percent overhead based on file size

respectively. The block size was typically set as 2 KB for both the type of files. The write /read time for a set of ?les includes the time to divide the file in equal sized blocks , computation of the log file value L_f , build the integrity information for each file block, recover the shares from the share holder nodes, generate the master key and encrypt all the data blocks in the set, create new files write the encrypted contents in the new files and performing cryptographic tree operations such as given a leaf index, Find its index in inorder traversal, Find the inorder index of its sibling and parent; the time to update and check the root of the tree, we call these set of operations as our framework components. To show the effect of a large range of data sizes, we measured the overhead when transferring input data from 2-25 MB, The X axis shows the amount of input data (including the text and compressed image files) for the computation and the Y axis shows the percentage overhead for each of the components (Fig. 6).

CONCLUSION

In this study we have proposed a mechanism for securing outsourced data in cloud which is managed and controlled by multiple legitimate parties. In all the tree based approaches the entire child node keys can be derived from root node key. Further in all these approaches the data block stored in the cloud can be updated by a party who holds either the specific decryption key or a node key corresponding to one of its parents. The proposed CTLF based approach overcomes these issues by secret sharing the master key among the 'n' share holder nodes. The CTLF approach also is used to authenticate data items and it will preserve block level integrity. Further Log file created in this approach is made visible only to the node controller and this ensures more privacy to the encrypted data blocks. We believe that our approach "Cryptographic Tree and Log File based Secret Sharing Key Management Scheme for Securing Outsourced Data in Cloud" provides enhanced privacy and secure key management for outsourced data in the cloud.

REFERENCES

Atallah, M.J., K.B. Frikken, M. Blanton, 2005. Dynamic and efficient key management for access hierarchies. Proceedings of the 12th ACM Conference on Computer and Communications Security, November 07-10, 2005, ACM, New York, USA., pp: 190-202.

Atallah, M.J., M. Blanton, N. Fazio and K.B. Frikken, 2009. Dynamic and efficient key management for access hierarchies. ACM. Trans. Inf. Syst. Secur. TISSEC., 12: 1-14.

Blundo, C., C. Stelvio, C.D.V. De, Sabrina, A. De Santis and S. Foresti *et al.*, 2009. Efficient Key Management for Enforcing Access Control in Outsourced Scenarios. In: Emerging Challenges for Security, Privacy and Trust, IFIP Advances in Information and Communication Technology. Gritzalis, D. and J. Lopez (Eds.). Springer, Boston, Massachusetts, ISBN: 978-3-642-01243-3, pp: 364-375.

Buchmann, J., E. Dahmen, E. Klintsevich, K. Okeya and C. Vuillaume, 2007. Merkle Signatures with Virtually Unlimited Signature Capacity. In: Applied Cryptography and Network Security. Katz, J. and M. Yung (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-540-72737-8, pp: 31-45.

Damiani, E., S. DeVimercati, S. Foresti, S. Jajodia and S. Paraboschi *et al.*, 2007. An Experimental Evaluation of Multi-Key Strategies for Data Outsourcing. In: IFIP International Information Security Conference. Venter, H., M. Eloff, L. Labuschagne, J. Eloff and R.V. Solms (Eds.). Springer, New York, USA., ISBN: 978-0-387-72366-2, pp: 385-396.

Damiani, E., S. DiVimercati, S. Foresti, S. Jajodia and S. Paraboschi *et al.*, 2005. Key management for multi-user encrypted databases. Proceedings of the 2005 ACM Workshop on Storage Security and Survivability, November 7-10, 2005, ACM, Alexandria, Virginia, ISBN: 1-59593-233-X, pp: 74-83.

DiVimercati, S.D.C., S. Foresti, S. Jajodia, S. Paraboschi and P. Samarati, 2007. Over-encryption: Management of access control evolution on outsourced data. Proceedings of the 33rd International Conference on Very large Data Bases, September 23-28, 2007, VLDB Endowment Inc., Austria, ISBN: 978-1-59593-649-3, pp: 123-134.

Huang, M.C. and J. Li, 2007. Pyramid codes flexible schemes to trade space for access efficiency in reliable data storage systems. Proceedings of the 6th IEEE International Symposium on Network Computing and Applications (NCA 2007), July 12-14, 2007, IEEE, Cambridge, Massachusetts, ISBN: 0-7695-2922-4, pp: 79-86.

Vimercati, S.D.C.D., S. Foresti, S. Jajodia, S. Paraboschi and G. Pelosi *et al.*, 2008. Preserving confidentiality of security policies in data outsourcing. Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society, October 27-31, 2008, ACM, New York, USA., ISBN: 978-1-60558-289-4, pp: 75-84.

Wang, C., Q. Wang, K. Ren and W. Lou, 2010. Privacy-preserving public auditing for data storage security in cloud computing. Proceedings of the IEEE INFOCOM, March 14-19, 2010, San Diego, CA., USA., pp: 1-9.

Wang, W., Z. Li, R. Owens and B. Bhargava, 2009. Secure and efficient access to outsourced data. Proceedings of the 2009 ACM Workshop on Cloud Computing Security, November 9-13, 2009, ACM, Chicago, Illinois, ISBN: 978-1-60558-784-4, pp: 55-66.

Zhou, M., Y. Mu, W. Susilo, J. Yan and L. Dong, 2012. Privacy enhanced data outsourcing in the cloud. J. Network Comput. Appl., 35: 1367-1373.