International Journal of Computer Applications (0975 – 8887)
Volume 51– No.12, August 2012

A Heuristic Approach for Encryption Policies in Data Outsourcing

B. N. Jagdale
Associate Professor
(Information Technology Dept.)
MITCOE
Pune, INDIA

Vidya S. Kurtadikar
Lecturer
(Computer Engg. Dept.)
MMCOE
Pune, INDIA

J. W. Bakal
Phd, Principal & Professor
Computer Engg.
S. S. Jondale College of Engineering
Mumbai University, INDIA

ABSTRACT

In the era of globalization and dynamic world economies, data outsourcing is inevitable. Security is major concern in data outsourcing environment since data is under the custody of a third party service provider. In present systems DBAs of third party can access and view data even though they are not authorized to do so. This may lead to serious data theft and leakages causing severe business impact to data owner. There are certain many such cases occurred in financial and insurance sector. In this paper we have proposed a novel solution to overcome the problem by combining access control with encryption and digital signature of data. A heuristic approach is presented to convert an authorization policy into an equivalent encryption policy while minimizing the no of keys and tokens to be managed. Different policy enforcement can be applied to different dataset as per security and integrity requirement.

General Terms

Data Security

Keywords

Access Control, Data outsourcing, Encryption Policy, Efficient Key Derivation, and Digital Signature.

1. INTRODUCTION

Data outsourcing is an evolving paradigm in current generations. It allows users and companies to give their sensitive data to external servers that become responsible for their storage, management, and dissemination. Organizations are able to concentrate on their core business activities, and thus outsourcing alleviate them from managing substantial hardware, software, and personnel costs involved in maintaining applications in-house. Although Data outsourcing provides many benefits, it introduces new privacy and security concerns [7]. Organizations outsource sensitive data for sharing on servers, which are not within the same trusted domain as data owners. In data outsourcing scenario there is selective access to data, with different users enjoying different views over the data. When data are outsourced there is therefore the problem of enforcing possible access control restrictions on it. In data outsourcing scenario resources are under the strict custody of a trusted party which monitors each access request to verify if it is compliant with the specified access control policy [1]. This approach requires some additional measures to be considered. There is need for Data Owner (Business Organizations) to manage access to legitimate users. Outsourcing healthcare Insurance services is extremely popular today. However, there are several concerns being voiced about data security and adhering to standard quality norms. The Health Insurance Portability and Accountability Act (HIPAA) are widely acknowledged as the norm for healthcare services and Indian companies are well versed with the Act and other regulatory bodies. Some other standards/acts relevant for data security are:

- The Information Technology Act 2000 (ITA-2000).
- Payment Card Industry Data Security Standard (PCI DSS).
- ISO 27001, ISO27001 Information Security Standard.

At present, few technologies used to protect sensitive data using encryption are

- Encryption during transmission – IPSec VPN tunnels or point to point dedicated connectivity, 3rd party tools (PGP, encrypted zip).
- Encryption during access – Masking of data displayed in front end applications e.g. Credit numbers etc.
- Encryption during storage – Call recording and image encryption.
- End point encryption – Laptop/Desktop encryption.
- Additionally security of sensitive data is also achieved through administrative access controls.
- Data Leak prevention – e-DLP.

In this paper we presented an approach in which data is categorized into three types a) Non-sensitive Data b) Sensitive data and c) Very high sensitive data. Depending upon category we can apply different security policy. Non-sensitive data will be directly accessible to users without any encryption. Sensitive data will be encrypted and lastly very high sensitive data will be encrypted and digitally signed by owner so as to achieve data integrity as well. Legitimate users will be provided corresponding symmetric keys and public keys when data is encrypted and digitally signed respectively to decrypt the data and for verification. Possible authorizations are to be enforced by the owner. In this way, the confidentiality & integrity of information does not rely on an implicit assumption of trust on the server for on the legal protection offered by specific service contracts, but instead relies on the technical guarantees provided by encryption techniques. Moreover in this paper we have reduced number of symmetric keys user has to manage. Only one symmetric key user has to manage rest of the keys will be derived. The rest of the paper is organized as follows: section 2 will introduce about related work done so far. Section 3 will introduce preliminaries, section 4 will give proposed work, section 5 will discuss results. Section 6 will conclude this paper.

2. RELATED WORK

As mentioned earlier some work has been done on this topic. In [1] a novel approach that combines cryptography with authorizations, thus enforcing access control via selective encryption is proposed. They proposed formal model for access control management and illustrates how an authorization policy can be translated into an equivalent encryption policy while minimizing the amount of keys and cryptographic tokens to be managed also introduces a two-layer encryption approach that allows the data owner to outsource, besides the data, the complete management of the authorization policy itself, thus providing efficiency and scalability in dealing with policy updates. In [2] a key derivation method is presented with the following properties: (i) the space complexity of the public information is the same as that of storing the hierarchy; ii) the private information at a class consists of a single key associated with that class; (iii) updates (revocations, additions, etc.) are handled locally in the hierarchy; (v) the scheme is provably secure against collusion; and (vi) each node can derive the key of any of its descendant with a number of symmetric-key operations bounded by the length of the path between the nodes. In [3] authors address problem of simultaneously achieving fine-grainedness, scalability, and data confidentiality of access control issue by, on one hand, defining and enforcing access policies based on data attributes, and, on the other hand, allowing the data owner to delegate most of the computation tasks involved in fine-grained data access control to un-trusted cloud servers without disclosing the underlying data contents. This is done by exploiting and uniquely combining techniques of attribute-based encryption (ABE), proxy re-encryption, and lazy re-encryption. Their proposed scheme also has salient properties of user access privilege confidentiality and user secret key accountability. In [4] a balanced access control system is proposed, where a robust system becomes flexible to meet its users' needs. On one hand, the system administrator sets system wide policies that all users must comply with. Policies are integrated into private keys of users, setting an access structure over attributes (resources) they can access. On the other hand, users are able to set their own access structure over system policies for documents they generate in the system. Users are in control of whom and under what conditions can access their documents. This way, a system administrator can help users set their own access control policies while both users' privacy and system's security are preserved. This system is based on two attribute-based encryption schemes: KP-ABE and CP-ABE. The former puts access policies into decryption keys, and the latter combines access policies with cipher texts. In [5] a new key-assignment approach based on secret sharing is presented. They first give two different key derivation schemes, and then combined them as one scheme. By analyzing the amount of public tokens required by the original over-encryption scheme and their scheme, they show that their scheme can provide the same over-encryption capability more efficiently. In [6] a family of generic key assignment schemes is proposed and their respective advantages and disadvantages are compared.

3. PROPOSED METHOD

In data outsourcing scenario we are assuming following architecture. In this owner will have full access to resources and can define authorization policy and define resource category. Our system will generate required keys and encrypt and digitally sign resources as per resource category defined by owner.

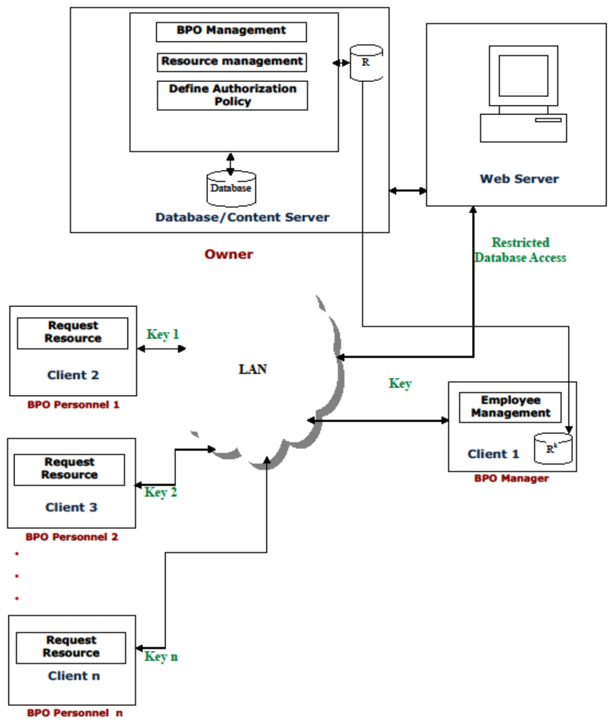


Fig 1: System Architecture Diagram

3.1 Authorization and Encryption Policy for different resource category

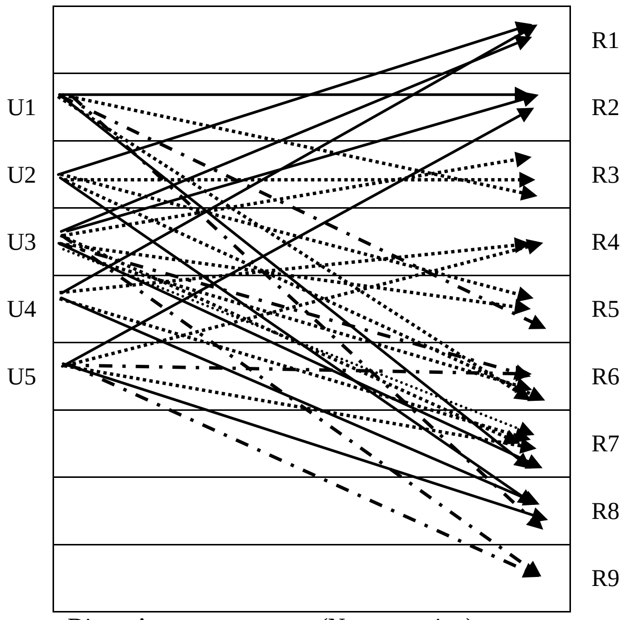
We assume that data owner will define authorization policy on resources. We categorize resources into three types as discussed in section 1 as a) Non-sensitive Data b) Sensitive data and c) Very high sensitive data. So there will be four possible values for data access i) 0: No access ii) 1: Direct access to resource (No encryption) iii) 2: Resource is encrypted iv) 3: Resource is encrypted as well as digitally signed. Sample access control matrix is shown in Fig. 2 and 3 respectively.

	R1	R2	R3	R4	R5	R6	R7	R8	R9
U1	0	1	0	2	0	3	2	1	3
U2	1	0	2	0	2	0	2	1	0
U3	1	1	2	0	2	3	2	1	3
U4	1	0	0	2	0	0	2	1	0
U5	0	1	0	2	0	3	2	1	3

Fig 2: Access Control Matrix

3.1.1 Authorization Policy

Let *U* and *R* be the set of users and resources in the system, respectively. An *authorization policy* over *U* and *R*, denoted *A*, is a triple <*U*, *R*, *P*>, where *P* is a set of permissions of the form <*u*, *r*>, with $u \in U$ and $r \in R$, stating the accesses to be allowed. Here *P* = {0, 1, 2, 3} where 0: No permission, (read permissions for) 1: Nonencrypted resource, 2: Encrypted Resource, 3: Resource is encrypted and digitally signed. So categories for resource is defined 1: raw (no encryption), 2: encrypted and 3: encrypted and digitally signed. We can have authorization policy graph as shown in Fig. 3 for access control matrix in Fig. 2.



Direct Access to resource (No encryption) $\longrightarrow$

Resource is encrypted $\cdots\blacktriangleright$

Resource is encrypted and digitally signed $- \cdot\blacktriangleright$

Fig 3: Authorization Policy Graph

The two methods discussed in [1] to convert authorization policy into encryption policy cannot be used because of their obvious disadvantages. The first straightforward approach has following disadvantages: i) Generates as many keys as no of users and resources in system. ii) Generates as many tokens as no of permission in system. Refer Fig. 3. The second method is grouping users with the same access privileges and by encrypting each resource with the key associated with the set of users that can access it. As a result it produces stratified graph. The advantage is that a key can be possibly used to encrypt more than one resource. The number of keys (no of vertices) and number of token (no of edges) can be derived from following formula:

No of keys (vertices) = $n + \sum_{r=2}^{n} nCr$, where n=no of users in system, r is level of vertex, nC_r is selection of r vertices from n vertices.

No of tokens(edges) = $\sum_{i=2}^{n} nv(i) * i$, where nv(i) = no of vertices/keys at level(i) , and i is level no of vertex.

Empirical Reading for number of vertices and Edges in stratified Graph is shown in following graph. This graph shows exponential growth of vertices and edges with respect to number of users.

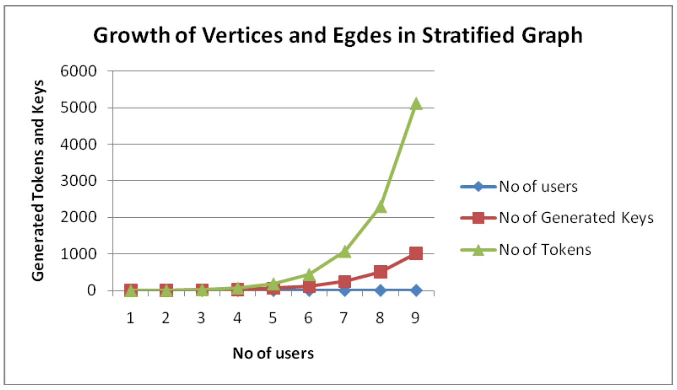


Fig 4: Fast Growth in no of vertices and edges with no of users

So we will convert this authorization policy into equivalent minimum encryption policy as described in [1] with following modifications.

3.1.2 Minimum Encryption Policy

For converting authorization policy into minimum encryption policy we adopted following method.

We have selected vertices that represents i) singleton set of users, whose private keys are required to derive all the other keys used for decrypting resources in the users' capabilities and ii) The acl(Access Control List) of the resources, whose keys are needed for decrypting and verifying such resources. In our paper we have selected only those acls for resources which are falling in category 2 or 3. This ensures that non sensitive resources sensitive will not be included in encryption graph and reduces overhead.

3.2 Key Derivation

If a user requests for certain user then depending upon whether user capability and resource category key derivation will occur. Using token information, path can be found to the vertex in the graph that will contain symmetric key required to decrypt resource as well as public key to verify digital signature of given resource.

3.3 Algorithms

This section describes important algorithms that are used for different purposes. First three algorithms are required to convert authorization policy into encryption policy.

3.3.1 Conversion of Authorization Policy into Encryption Policy

The input for this algorithm is authorization policy defined by data owner and output is equivalent encryption and digital signature policy. Initially ACL will contain all users and acl for resources that fall in category 2 or 3. It works in three parts, first part is initialization that will create one vertex for each entry in ACL and initialize its attributes. In second part, edges will be added and in third part resources will be encrypted and digitally singed as per resource category. And different database tables will be updated.

```
Algorithm 3.3.1 Converting Authorization Policy into Encryption and Digital Signature Policy
Input : Authorization Policy
Output : Equivalent Encryption & Digital Signature Policy
{
// Initialization (ACL = acl for each resource (that has data access value as either 2 or 3 ) & total users in the system)
FOR each entry in ACL DO
    Create a vertex and initialize its attribute (acl, label, key, pub key and counter)
// Phase – 1 vertex cover without redundant edges i.e. Add edges in graph
FOR each level from maxlevel to 2 DO
    FOR each vertex whose level is current level DO
    Find out its vertex cover without redundancy
    //to be deleted
// Phase – II generate encryption policy
Generate Encryption Policy for the graph created in first two phases
}
```

As discussed in [1], this algorithm will work in bottom up approach to find out correct vertex cover. Redundant edges are removed from graph.

```
Algorithm 3.3.2 Finding Vertex Cover
{
Current Level assigned to level of vertex v – 1
// find correct cover for users in tocover
WHILE tocover is not empty DO
    Select a set of vertices $V_l$ from vertex set whose level is current level and whose acl is subset of acl of vertex v
    WHILE (set of vertices selected in previous step is not empty) and (tocover is not empty) DO
        Choose an arbitrary vertex $v_i$ from $V_l$
        IF acl of $v_i \cap$ tocover is not empty THEN
            Remove users present in acl of $v_i$ from tocover
            Draw an edge from $v_i$ to v. Add this edge in edge set
            FOR each user in acl of $v_i$ DO
                increment counter of v by 1
    current level = current level -1
Add edge set to edge set of graph
// Remove redundant edges
FOR each edge in edge set DO
        IF it is redundant THEN
            Remove this edge from edge set
Adjust counter for this user
}
```

This part will generate symmetric keys for encryption, private and public keys for digital signature), labels and tokens. In addition to this algorithm will encrypt and digitally sign resources as applicable. Database tables will be updated accordingly. This will complete conversion of authorization policy into encryption policy.

```
Algorithm 3.3.3 Generate Encryption and Digital Signature Policy
{
FOR each vertex v in graph DO
    Generate private key k, generate sig and pub_key (if applicable) and Assign it to vertex v
Generate label l and Assign it to vertex v
    Add this generated and label, private key and sig_pub_key to private key-set, sig_pub_key set and vertex-set resp.
// Generate tokens
FOR each edge in graph DO
    Compute token using formula $t_{i,j} = v_j$.key xor ($v_i$.key , $v_j$.label)
    Add computed token to token-set
    Upload token $t_{i,j}$ on the server by adding it to table TOKEN
// Define key assignment and encryption schema
FOR each user DO
    Find a vertex v whose acl corresponds to this user
    Assign Encryption schema to this user
FOR each resource DO
    Find an vertex v in graph whose acl corresponds to this resource
    encrypt this resource with key corresponding to vertex v
    and digitally sign this resource(if applicable)
    Upload encrypted resource on server
    Assign Encryption schema to this resource
    Store the hash code of resource in RESOURCE table (if applicable)
    Update table RESOURCE and USER accordingly
}
```

3.3.2 Result of Algorithm Execution

As a result of execution of above algorithms when run on authorization policy as in Fig. 5 we got following encryption and digital signature policy graph.

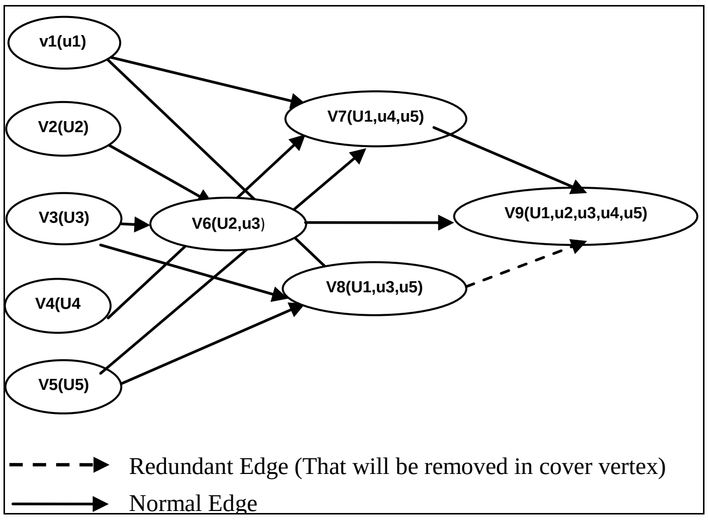


Fig 5: Minimum Encryption Policy Graph for Access Control Matrix shown in Fig. 2

4. INTRODUCTION

We have considered .doc files as resources. These resources are of different sizes ranging from 26KB to 50 MB. We have generated resources using shell program. We have taken reading on user size ranging from 10 to 100 and resource size ranging from 20 to 140. Access control Matrix is generated dynamically in each access control policy. As a result we got following graphs.

4.1 Token Complexity

In Fig. 6, number of generated tokens (i.e. number of edges in graph) is evaluated for number of users in the system. We observed that no of tokens generated are linear for number of users for a particular access control policy.

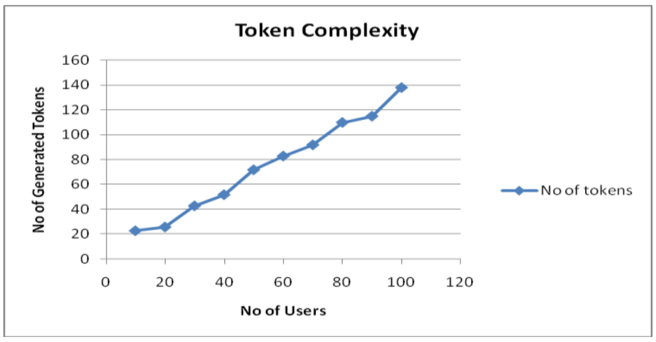


Fig 6: Token Complexity

4.2 Normal or Body Text

In Fig. 7 we analyzed space required for tokens. Each token requires 20 bytes so the space complexity is linear.

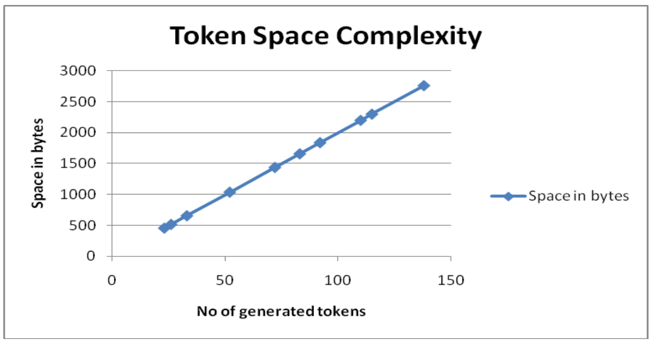


Fig 7: Token Space Complexity

4.3 Token and Key Complexity

Number of keys are nothing but number of vertices in graph. Fig. 8 shows number of vertices and number of edges required to convert access control policy into encryption policy.

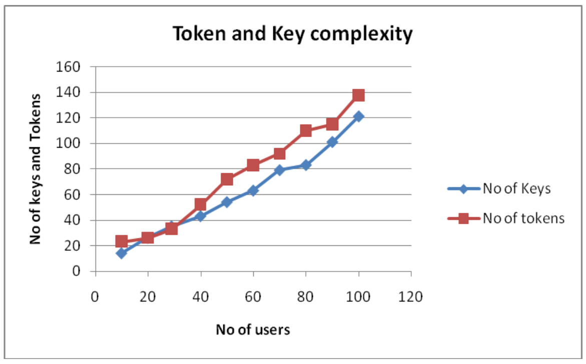


Fig 8: Token and Key complexity

4.4 Integrated Cryptography Performance

As discussed previously, category 2 and category 3 resources are considered while creation of graph. So here we have taken readings for time required for key derivation which is a chained process as discussed in proposed methodology and implementation. This requires searching through tokens table to calculate key required to decrypt requested resource. So we have taken reading for time required for key derivation and decryption of resource for category 2 resources. For category 3 resources in addition to key derivation and decryption time, time required to verify digital signature is also noted. These readings are taken for .doc files of different sizes.

Result Analysis

If we see in Fig. 6 number of tokens generated for a particular access control policy are on an average 1.33 tokens per user. If we compare this with group user approach in proposed methodology with stratified graph, this heuristic approach for minimum encryption policy is quite efficient. Similarly number of keys generated are also very less, thus key management is also efficient. Decrease in number of tokens means key derivation will be fast. Because key derivation requires search through token table to generate intermediate keys required in key derivation process. Similarly small number of tokens means less space is required to store it.

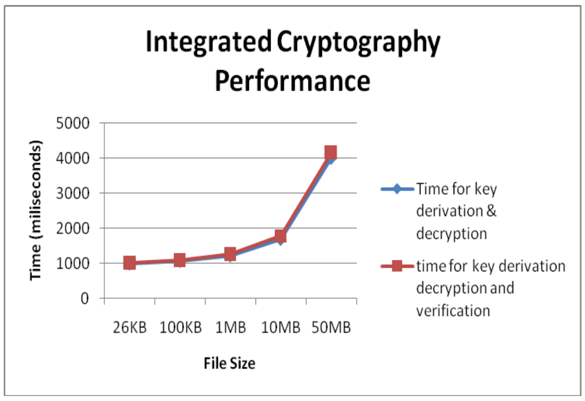


Fig 9: Time for (key derivation, decryption) and (key derivation, decryption and verification)

5. CONCLUSION

In this paper problem of enforcing access control in a scenario where data are outsourced to external servers that, while trusted for data management, are not authorized to read the data content (honest but curious servers) has been addressed. Now instead of encrypting all resources we categorized data into three types and keep some data as it is, encrypt some resources and digitally sign some of the resources. This paper also puts forward a novel approach combining authorizations and encryption. This will avoid overhead on encryption and decryption when data is not sensitive. In addition to security incurred by encryption we have addressed problem of data integrity and authentication of owner by digitally signing data. Similarly we have provided a formal characterization of the problem of translating authorization policies into equivalent encryption policies, while minimizing the overhead in terms of storage and computation needed for the enforcement, and key derivation. This will be useful approach for number of data outsourcing applications. As a future work we can consider other privileges such as write privilege on resource as well as from users point of view, users should be able define access control policy for documents he has generated.

6. REFERENCES

[1] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, Pierangela Samarati, Encryption Policies for Regulating Access to Outsourced Data ACM Transactions on Database Systems, Vol. 35, No. 2, Article 12, Publication date: April 2010.

[2] Mikhail J. Atallah, Marina Blanton, Nelly Fazio, Keith B. Frikken,, Dynamic and efficient key management for access hierarchies, ACM Transactions on Information and System Security, Vol. 12, No. 3, Article 18, Pub. date: January 2009.

[3] Shucheng Yu, Cong Wang, Kui Ren, Wenjing Lou, Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing, INFOCOM, 2010 Proceedings IEEE , Publication Year: 2010.

[4] Malek, B. Miri, A., Combining Attribute-Based and Access Systems , Computational Science and Engineering, 2009. CSE '09. International Conference, Publication Year: 2009.

[5] Shuai Liu ,Wei Li, Lingyu Wang, Towards Efficient Over-Encryption in Outsourced Databases Using Secret Sharing, New Technologies, Mobility and Security, 2008. NTMS '08.

[6] Pierangela Samarati, Sabrina De Capitani di Vimercati, Data protection in outsourcing scenarios: issues and directions, April 2010 ASIACCS '10**:** Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security.

[7] Sabrina De Capitani di Vimercati, Sara Foresti, Stefano Paraboschi, Pierangela Samarati,Privacy of Outsourced Data, 2007 Auerbach Publications (Taylor and Francis Group), Digital Privacy: Theory, Technologies and Practices.

[8] Crampton, J.,Martin, K., And Wild, P. 2006., On key assignment for hierarchical access control, In Proceedings of the IEEE Computer Security Foundations Workshop (CSFW’06). IEEE Computer Society, Washington, 98–111.

[9] S. MacKinnon, P. Taylor, H. Meijer, and S. Akl., An optimal algorithm for assigning cryptographic keys to control access in a hierarchy, IEEE Transactions on Computers, C-34(9):797–802, 1985.

[10] DE SANTIS, A., FERRARA, A.L., AND MASUCCI, B. 2004. Cryptographic key assignment schemes for any access control policy, Inform. Process. Lett. 92, 4, 199–205.

[11] GUDES, E. 1980. The design of a cryptography based secure file system. IEEE Trans. Software. Engineering. 6, 5, 411–420.

[12] HACIG ¨UM¨US, H., IYER, B., AND MEHROTRA, S. 2002a. Providing database as a service. In Proceedings of the International Conference on Data Engineering (ICDE’02). IEEE Computer Society, Washington, 29–39.

[13] HARN, L. AND LIN, H. 1990. A cryptographic key generation scheme for multilevel data security. Computer. Security. 9, 6, 539–546.