A PROFICIENT MODEL FOR HIGH END SECURITY IN CLOUD COMPUTING

R. Bala Chandar[1], M. S. Kavitha[2] and K. Seenivasan[3]
[1]*Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India*
E-mail: balachandar.raju@gmail.com
[2]*Department of Computer Science and Engineering, Sri Eshwar College of Engineering and Technology, India*
E-mail: kaviktg@gmail.com
[3]*Department of Computer Science and Engineering, Renganayagi Varatharaj College of Engineering, India*
E-mail: srinivassvks@gmail.com

Abstract
Cloud computing is an inspiring technology due to its abilities like ensuring scalable services, reducing the anxiety of local hardware and software management associated with computing while increasing flexibility and scalability. A key trait of the cloud services is remotely processing of data. Even though this technology had offered a lot of services, there are a few concerns such as misbehavior of server side stored data , out of control of data owner's data and cloud computing does not control the access of outsourced data desired by the data owner. To handle these issues, we propose a new model to ensure the data correctness for assurance of stored data, distributed accountability for authentication and efficient access control of outsourced data for authorization. This model strengthens the correctness of data and helps to achieve the cloud data integrity, supports data owner to have control on their own data through tracking and improves the access control of outsourced data.

Keywords:
Data Security, Data Correctness, Data Control, Access Control, Data Integrity

1. INTRODUCTION

Cloud computing is the recent technology that enables cloud customers to enjoy the on demand high quality applications and services from a shared pool of configurable computing resources through storing their data remotely in the cloud. Technology enabled services can be consumed over the Internet on an as needed basis through cloud. It refers to both the applications delivered as a service over the Internet and the hardware and software in the data centers that provide those services.

Cloud computing is a development of virtualization, parallel, distributed, utility computing and SOA. Different service oriented cloud computing models have been proposed which includes Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). Generally cloud computing has several customers such as ordinary users and enterprises who have different motivations to move to cloud [13][14][16][17].

In cloud, there are a set of important policy issues, which includes some non functional requirements such as privacy, security, reliability and liability. But the most important is security and how cloud providers assure it. One of the prominent security concerns is data security and privacy in cloud computing due to its Internet based data storage and management.

While enjoying the provision of huge amount of storage space and customizable computing resources, it also eliminates the responsibility of local machines for data maintenance at the same time. Cloud provides more powerful and reliable features but there are threats which still exist for data integrity such as outages and data loss incidents.

In the mean while users also start worrying about losing control of their own data. Sometimes the data processed on cloud becomes a significant barrier to the wide adoption of cloud services. Data represents an extremely important asset for any organization, and it will be in trouble if its confidential data is disclosed to their business competitors or the public. Thus cloud users want to make sure that their data are kept confidential to outsiders.

In order to address the above problems, a model is proposed which helps to ensure the data integrity and information accountability in cloud with flexible access control of the outsourced data.

2. RELATED WORK

Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance [1]. The main issue is how to frequently, efficiently and securely verify that a storage server is faithfully storing the outsourced data. The storage server is assumed to be untrusted in terms of both security and reliability [2]. A growing number of online services, such as Google, Yahoo!, and Amazon, are starting to charge users for their storage. Customers often use these services to store valuable data such as email, family photos and videos, and disk backups. Today, a customer must entirely trust such external services to maintain the integrity of hosted data and return it intact. Unfortunately, no service is infallible. To make storage services accountable for data loss, we present protocols that allow a third party auditor to periodically verify the data stored by a service and assist in returning the data intact to the customer. In particular, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud [3]. Thus, enabling public audit ability for cloud storage is of critical importance so that users can resort to a third party auditor (TPA) to check the integrity of outsourced data and be worry-free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data privacy, and introduce no additional online burden to user [4]. It describes three key problems for trust management in federated systems and presents a layered architecture for addressing them. The three problems we address include how to express and verify trust in a flexible and scalable manner, how to monitor the use of trust relationships over time, and how to manage and re-evaluate trust relationships based on historical traces of past behavior. Trust management

and are especially relevant in the context of federated systems where remote resources can be acquired across multiple administrative domains [5]. They propose a language that allows agents to distribute data with usage policies in a decentralized architecture. They design a logic that allows audited agents to prove their actions, and to prove their authorization to posses' particular data. Accountability is defined in several flavors, including agent accountability and data accountability [6].

Finally, they show the soundness of the logic. Here it analyses the how accountability is transferred by the delegator when he/she is transferred some of their right to the delegate [7]. It introduces a model for provable data possession (PDP). It stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The client maintains a constant amount of metadata to verify the proof. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage systems. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data [8]. They describe an operational model of accountability-based distributed systems. We describe analyses which support both the design of accountability systems and the validation of auditors for finitely accountability systems. Our study provides the power of the auditor, the efficiency of the audit protocol, the requirements placed on the agents, and the requirements placed on the communication infrastructure [9]. In this Paper they present a system for realizing complex access control on encrypted data that we call Ciphertext-Policy Attribute-Based Encryption. By using this technique encrypted data can be kept confidential even if the storage server is untrusted; moreover, this method is secure against collusion attacks. Here the attributes are used to describe a user's credentials, and a party encrypting data determines a policy for who can decrypt. Thus, this method is conceptually closer to traditional access control methods such as Role-Based Access Control (RBAC) [10].

This approach is used to keep sensitive user data confidential against untrusted servers; existing solutions usually apply cryptographic methods by disclosing data decryption keys only to authorized users. Here they defining and enforcing access policies based on data attributes, and allowing the data owner to delegate most of the computation tasks involved in fine-grained data access control to untrusted cloud servers without disclosing the underlying data contents by combining techniques of attribute-based encryption (ABE), proxy re-encryption, and lazy re-encryption [10]. They develop a new cryptosystem for fine-grained sharing of encrypted data that we call Key-Policy Attribute-Based Encryption (KP-ABE). It constructs a model to supports delegation of private keys which subsumes Hierarchical Identity-Based Encryption (HIBE) [11]. Attribute-Based Encryption (ABE) is a new paradigm where the policies are specified and cryptographically enforced in the encryption algorithm itself. In this work they focused on improving the flexibility of representing user attributes in keys. Specifically, they proposed Ciphertext Policy Attribute Set Based Encryption (CP-ASBE) [12].

3. PROBLEM STATEMENT

3.1 COMPONENTS

The following are the components used in this scheme:

3.1.1 Data User:

It can be an enterprise or individual customer who consumes the data stored in the cloud.

3.1.2 Data Owner:

It plays a major role where it deploys the data into the cloud. It can be an enterprise or individual customer where it depends on cloud for data storage and computation.

3.1.3 Cloud Server:

It is a cloud service provider which has a pool of computation resources and storage space.

3.1.4 Assessor:

It is an auditing entity which completely depends on the users demand.

3.1.5 Logger:

It is responsible for storing the access details of the data items automatically.

3.1.6 Log Assessor:

It is an entity which is responsible for auditing the logged information.

3.1.7 Trusted Authority:

It is a powerful entity used to manage the lower level users.

3.2 SYSTEM ARCHITECTURE

The System architecture is illustrated in Fig.1. The first tier deals with the data at rest to ensure the assurance. The data owner encrypts the original data file. This encrypted file is then tokenized and stored in the cloud servers. The second tire deals with the data under no control to ensure authentication. The encrypted logger is added with the received file with its originality. Now the combined file forms the JAR File. The third tier deal with the data during access to ensure the authorization. Here the JAR File is accessed only by the trusted authority and the authorization for the different level of users is delegated by the trusted authority.

3.3 DESIGN GOALS

To ensure the security, we aim to design efficient mechanisms and achieve the following goals:

3.3.1 To ensure the data correctness for Assurance of stored data:

It mainly focuses on the storage correctness, Data error localization and dynamic data support.

3.3.2 To ensure the distributed accountability for Authentication:

It provides distributed and automated logging in order to leverage the full control of the data under the data owner.

3.3.3 To ensure the efficient access control of outsourced data for Authorization:

It supports the fine grained access to achieve flexibility and scalability of access control.

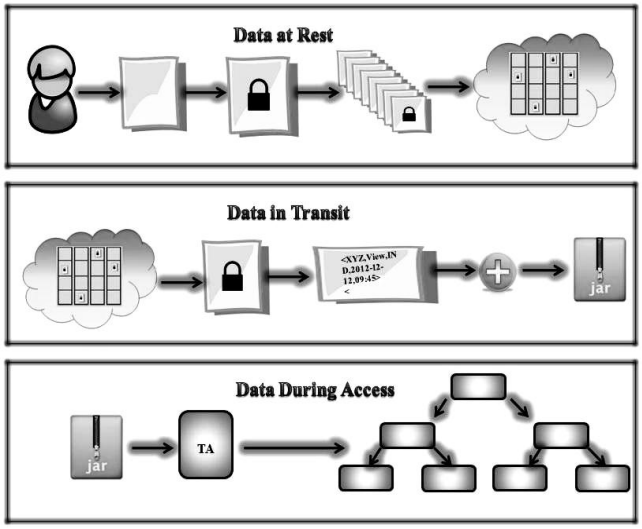

Fig.1. System Model

4. PROFICIENT MODEL

In this section, we design an overview of the enhanced model and discuss how this model meets the design requirements discussed in the previous section.

The model proposed in this work conducts various automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time within any cloud service provider. Then, we create an effective storage verification scheme for dynamic data support to ensure the storage correctness of data with fine grained access control of outsourced data in the cloud with various levels of users in an organization.

4.1 MECHANISMS

4.1.1 Erasure Correcting Code:

Erasure correcting code is used to distribute the file with replication. The communication and storage overhead is reduced as compared with the conventional replication-based file distribution techniques.

4.1.2 Homomorphic Tokenization:

By utilizing the homomorphic tokenization the original file is divided into number of tokens.

4.1.3 Verification of tokens:

The TTS achieves the storage correctness and data error localization by integrating both the erasure correcting code and homomorphic tokenization. The compromised servers are identified during the detection of data corruption.

4.1.4 Logging (logger):

The access details are continuously added into the log file as per the number of data file access. It is automatically downloaded along with the data file whenever the data is copied.

4.1.5 Auditing:

This is processed with the help of the logged files provided by the logger. The following are the two modes of operations:

Shove – In: The log files are shoved back to the data owner periodically in an automated fashion

Fling – Out: The log files are obtained by the data owner as on demand. It is also dependable for handling log file corruption

4.1.6 Attribute based encryption:

Attribute based encryption scheme will be used for encrypting both the cipher text and users' decryption keys that are associated with a set of attributes or policy. A user is able to decrypt a cipher text only if there is a match between his decryption key and the cipher text.

4.2 DATA FLOW

The overall model combines the data items, users, cloud storage, loggers, logger auditor, tokenization process and how ABE scheme is applied in the organization (with various levels of users) is sketched in Fig.2. At the beginning, the client login into the cloud server through their user name and password based on the Identity Based Encryption (IBE) scheme. The uploading file is obtained from the data owner only after successful authorized access verification. Then the original data file is tokenized into equal size streams of tokens and stored into the same size of blocks at various cloud servers randomly with some access control which is desired by the data owner.

When the data is accessed from the cloud server, the streams of tokens are merged to form the original file. Before that, the originality of the file content will be verified through the digital signature that is generated for each stream of tokenized files which are in the cloud servers. If any intruder tries to modify the tokens, the compromised server will be easily identified since the digital signatures are created for every stream of tokens.

After the rearranging process, the original file is added with the logger to form the JAR file with some access policies. This logger file contains the details about the data items which are accessed by the stakeholder or organization. As for the logging, whenever there is a data access a log record is generated automatically, and it is encrypted using the public key which is distributed by the data owner, and it is stored with the data file. The log file encryption prevents the unauthorized changes created by the attackers to the log file. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. If any of the unauthorized users or any cloud service provider is trying to misuse the data items, it will be easily identified by giving an alert to the data owner. Finally, the trusted domain of this mode; will read the JAR and generate a master key for all the domain authorities those who are in need to access the outsourced data. After the data is outsourced from the cloud, the privileged domain authorities will generate the keys for every attribute at each level of users' in the organization.

The domain authority/authorities are under the control of the trusted authority, so it can generate the key for the users in the next level. The management and generation of keys will be reduced as well as the scalability and flexibility via efficient attribute set management will be improved. Each level of users has their separate key for each of their attributes, so fine grained access control of the outsourced data is possible with this scheme.

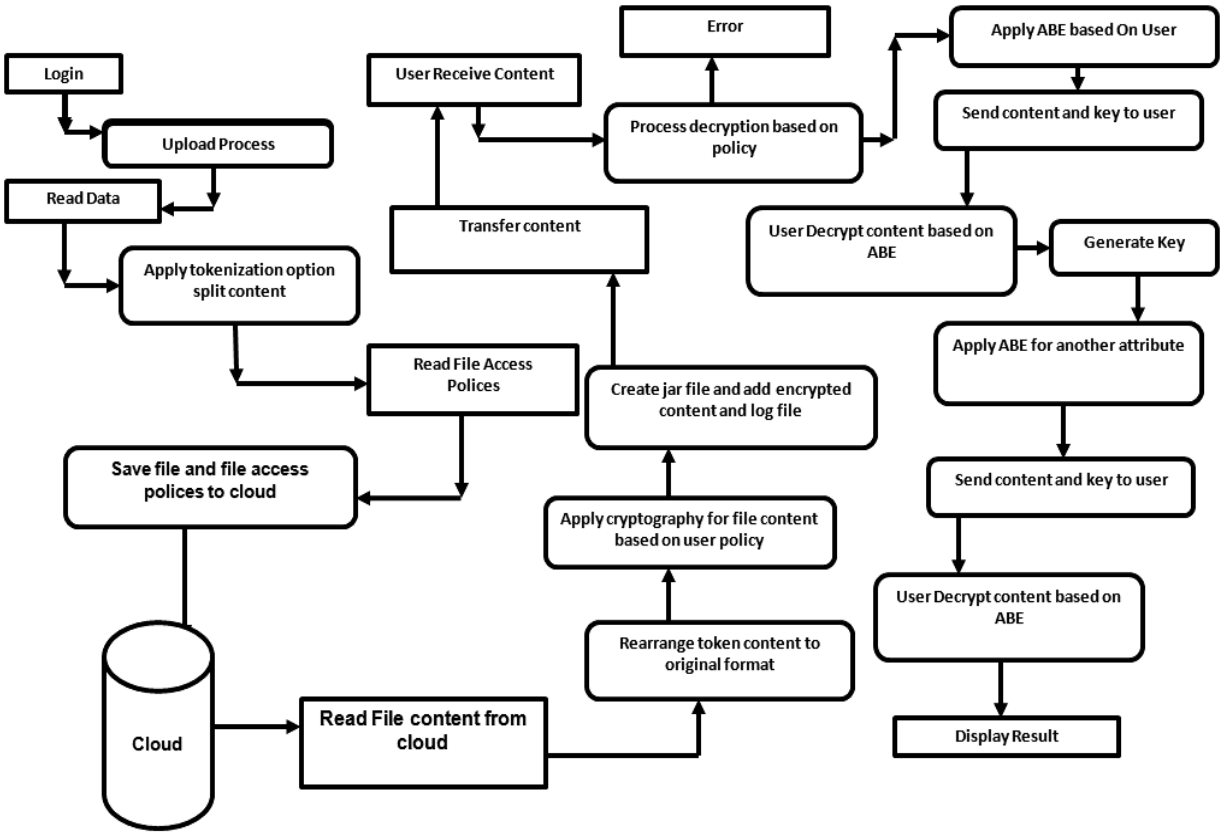

Fig.2. Proficient System Model - Data Flow

5. IMPLEMENTATION

We start this section by considering an illustrative example which serves as the basis of our implementation and demonstrate the main features of our system.

Example: Consider ABC as a jewellery shop with some levels of users as shown in Fig.3, plans to store their jewel reports into the cloud and to access the same with the following requirements:

1. The jewel reports are downloaded only by the authorized users who are in that shop.
2. Users in this shop should be equipped with security means so that they can make continuous correctness assurance of their jewel report even without the existence of local copies.
3. The online burden of data integrity checking tasks must be reduced in an efficient manner.
4. The cloud provider or stake holder should not misuse the data for their own purpose without the knowledge of data owner.
5. The outsourced data from the cloud should be accessed only by the authorized domain authorities (ABC Jewel shop).
6. Each user at various levels in the shop must have limited access to the outsourced data, as desired by the data owner or higher level entities in the organization.

In order to achieve the above requirements, we have created a private cloud environment. According to Fig.3, only the admin or data owner has the authority to store the jewel report into the cloud. Before storing the jewel report into the cloud, the report will be tokenized using homomorphic tokenization mechanism. The number of tokens depends on the cloud servers. Digital Signature is created for all the tokens using RSA algorithm.

Then the digitally signatured tokens are maintained locally in order to assure the data integrity. All the signatured tokens are placed randomly across the servers. Here the trusted authority behaves as an intermediator between the server (cloud service provider) and the client (admin or data owner). When the trusted authority tries to access the jewel report, initially all the signatured tokens in the cloud servers are verified with the locally maintained digital signature tokens.

After the verification, all the tokens are integrated together to form the original jewel report. The verification process helps to find the compromised servers in case if their corresponding tokens are modified.

Through the above procedure we can achieve the second requirement. To the original jewel report, a logger file with some access policies is added in order to form the jar file. Now the jar file behaves as a container which consists the logger file in an encrypted format and the access policies which are enforced by the data owner.

Number of levels in the Organization	Attributes / Users	Jewel Name	Dealer Name	Buying Date	Total Grams	Sales Rate	Carat Type	Buying Date	Original Data File	
									Jewel Image and Description	
Level 1	Administrator or Data Owner	YES	YES	YES	YES	YES	YES	YES	YES	YES
Level 2	Manager	YES	YES	NO	YES	YES	YES	YES	YES	YES
Level 3	Supervisor	YES	NO	NO	YES	YES	YES	YES	YES	YES
Level 4	Worker	YES	NO	NO	YES	YES	YES	NO	YES	YES
YES – Permission granted for access the attributes					NO – Permission not granted for access the attributes					

Fig.4. Access control of entity levels in an organization

The log file contains the information about access of the jewel report such as name, time, location, type of access of the user. The logger file automatically saves the details when the jewel report is accessed by the user. Periodic auditing is processed by the logger auditor to identify the unauthorized access. The periodic auditing process has two separate modes such as shove-in and fling-out.

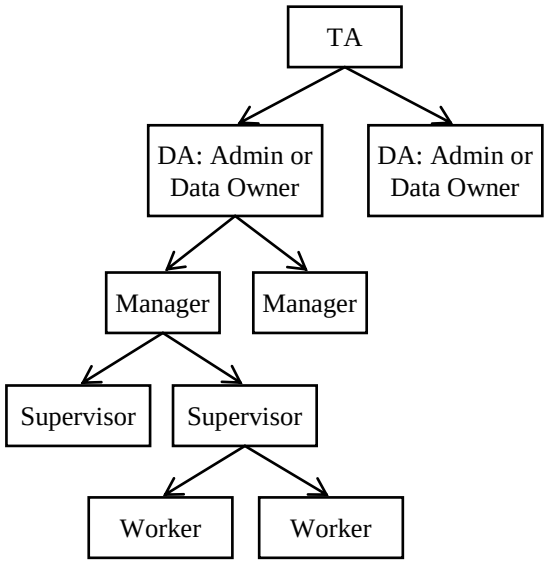

Fig.3. Entity levels in an organization

During the shove – in mode, the log files are automatically shoved back to the data owner periodically. Fling-out is an on demand approach, where the log files are obtained by the data owner whenever it is required. Complete information accountability is achieved along with the identification of the misusage of original jewel report. This helps to satisfy the first and fourth requirements. We implement the third party auditing to conduct the data integrity checking process to reduce the online burden of the user. Using third party auditor, we can achieve the third requirement.

To satisfy the fifth and sixth requirements, the entities at the higher level are responsible for generating the keys for the lower level entities in the jewellery shop using Attribute Based Encryption scheme. According to the access policies defined by the data owner, some details of the jewel are hidden to the lower level entities by the higher level entities as shown in Fig.4. This scheme improves the scalability, flexibility and fine grained access control of the jewel report in the cloud.

6. PERFORMANCE EVALUATION

The following graphs which are shown in the Fig.5, Fig.6 and Fig.7 evaluate the result analysis of the model using various metrics. The metrics such as assessing time, log merging time and key generation time are considered in order to evaluate the performance.

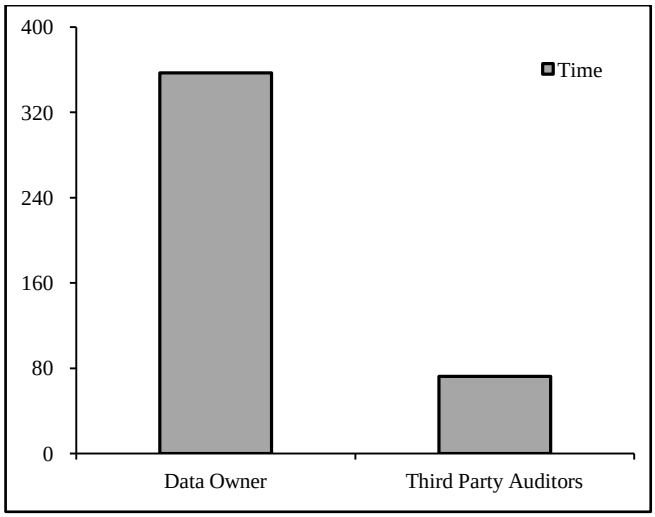

Fig.5. Assessing Time

7. CONCLUSION

In this paper we introduced a proficient model in order to address the problems such as data integrity, data loss and secure data access. It is designed in such a way to provide end – to – end security in the cloud. It helps to assure the data correctness and also helps to simultaneously identify the misbehaving servers in the cloud system. It enables the data owner to have full control of his own data by monitoring the access logs of data file through distributed auditing mechanism. To add more security at the access levels, the data has been converted in a more flexible and scalable form with fine grained access control. Finally the data has been secured at all the levels such as at rest, during transit and access in order to provide a complete end to end security.

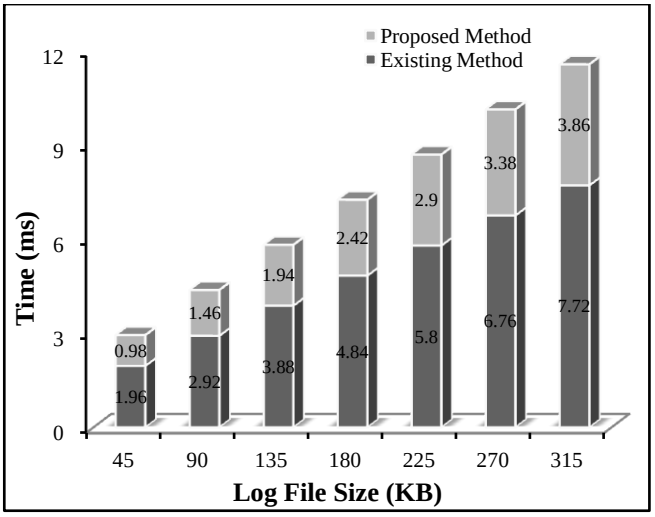

Fig.6. Log Merging Time

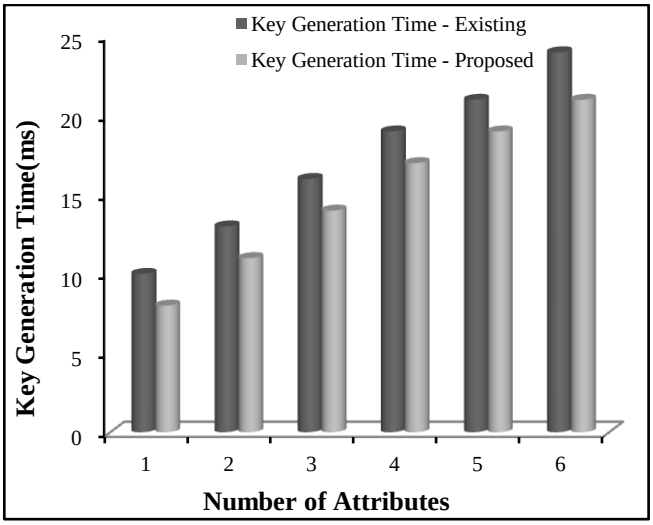

Fig.7. Key generation Time

REFERENCES

[1] C. Wang, Q. Wang, K. Ren and W. Lou, "Privacy Preserving Public Auditing for Data Storage ecurity in Cloud Computing", *Proceedinjgs of IEEE 29th International Conference on Computer Communications*, pp. 525 – 533, 2010.

[2] G. Ateniese, R.D. Pietro, L.V. Mancini and G. Tsudik, "Scalable and Efficient Provable Data Possession", *Proceedings of Fourth International Conference on Security and Privacy in Communication Netoworks*, pp. 1 – 10, 2008.

[3] M. A. Shah, R. Swaminathan and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents", IACR *Cryptology ePrint Archive*, 2008.

[4] Q. Wang, C. Wang, J. Li, K. Ren and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing", *Proceedings of 14th European Conference on Research in Computer Security*, pp. 355 – 370, 2009.

[5] B. N. Chun and A. Bavier, "Decentralized Trust Management and Accountability in Federated Systems", *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 2004.

[6] R. Corin, S. Etalle, J. I. Den Hartog, G. Lenzini and I. Staicu, "A Logic for Auditing Accountability in Decentralized Systems", *Proceedings of International Federation for Information Processing TC1 WG1.7 Workshop on Formal Aspects in Security and Trust, World Computer Congress*, pp. 187 – 201, 2004.

[7] B. Crispo and G. Ruffo, "Reasoning about Accountability within Delegation" *Proceedings of Third International Conference on Information and Communication Security*, pp. 251 – 260, 2001.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable Data Possession at Untrusted Stores", *Proceedings of Association for Computing Machinery Conference on Computer and Communication Security*, pp. 598 – 609, 2007.

[9] R. Jagadeesan, A. Jeffrey, C. Pitcher and J. Riely, "Towards a Theory of Accountability and Audit", *Proceedings of 14th European Conference on Research in Computer Security*, pp. 152 – 167, 2009.

[10] J. Bethencourt, A. Sahai and B. Waters, "Ciphertext-policy attributebased encryption", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 321 – 334, 2007.

[11] S. Yu, C. Wang, K. Ren and W. Lou, "Achieving secure, scalable and fine-grained data access control in cloud computing", *Proceedings of 29th International Conference on Computer Communications*, pp. 534–542, 2010.

[12] V. Goyal, O. Pandey, A. Sahai and B.Waters, "Attibute-based encryption for fine-grained access control of encrypted data", *Proceedings of Association for Computing Machinery Conference on Computer and Communication Security*, pp. 89 – 98, 2006.

[13] R. Bobba, H. Khurana and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption", *Proceedings of European Symposium on Research in Computer Security*, pp. 587 – 604, 2009.

[14] B. Barbara, "Salesforce.com: Raising the level of networking", *Information Today*, Vol. 27, pp. 45–45, 2010.

[15] K. Barlow and J. Lane, "Like technology from an advanced alien culture: Google apps for education at ASU", *Proceedings of the 35th Annual Association for Computing Machinery Special Interest Group on University and College Computing Services Conference*, pp. 8-10, 2007.

[16] Amazon Web Services (AWS). Available: https://s3.amazonaws.com/

[17] Google app Engine. Available: http://code.google.com/appengine/

[18] http://code.google.com/appengine/.