International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 14 Issue 1 –APRIL 2015.

# EFFECTIVE ACCESS CONTROL AND SECURED LOG MANAGEMENT TECHNIQUES FOR CLOUD STORAGE

KARTHIKEYAN.K, EUGENE BERNA.I
*PG Student, Department of Computer Science &Engineering,St. Joseph College of Engineering, Chennai, India.*
*Assistant Professor, Department of Computer Science &Engineering,St. Joseph College of Engineering, Chennai, India.*
karthikapil19@gmail.com
eugeneberna31@gmail.com

**ABSRACT—Log Management is an important service in Cloud Computing. In any business, maintaining the log records securely over a particular period of time is absolutely necessary for various reasons such as auditing, forensic analysis, evidence etc. In this paper, we introduce a secured log management technique for cloud storage using cryptography method. Here, Integrity and confidentiality of the log records are maintained at every stage of Log Management namely the Log Generation phase, Transmission phase and Storage phase. Moreover, Secured Log Management techniques are also implemented to provide security to maintain transaction history in cloud. In addition to this, security to log management is provided by encrypting the log data before they are stored in the cloud storage. In this system single key which includes the power of all the aggregated keys-KAC (key aggregate cryptosystem). One can aggregate any set of secret keys and make them as compact as a single key, but encompassing the power of all the keys being aggregated. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage.The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes-AGGREGATE KEY. The extracted key can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys. The decryption power for any subset of cipher text classes is made possible with the single constant size aggregate key. The sizes of ciphertext, public-key, and master-secret key and aggregate key in our KAC schemes are all of constant size. They are also stored in batches for easy retrieval.**

**Index Terms**—Cloud computing, Data Logging, Aggregate key, Data Integrity, Log Management.

## 1. INTRODUCTION

Cloud logs are machine or transaction data generated by any sort of application ortheinfrastructure used to run that application. Logs in cloud environment are composed of log entries in which each entry contains information related to a specific event that has occurred within a system or network. Many logs generated by an application developed for and used computer security pertaining to the organization.

These computer security logs are mostly generated by many sources, including security software, such as antivirus software, firewalls and intrusion detection and prevention systems. They are also generated by operating systems on servers, workstations, and networking equipment and applications. In the field of databases, a transaction log is a history of actions executed by a database management system to guarantee Atomicity, consistency, Isolation and Durability (ACID) properties over crashes or hardware failures. Physically, a log is a file of updates carried out in the database, which are stored in stable storage. World Wide Web Consortium (W3C) has provided with a standard log file format for storing web logs. The format which also goes by the name Common Log file format is given as follows: "RemoteHostRFC931 user date request status bytes". They refer to the host name, the remote log name of the user, username, date and time, request line from the client, the HTTP status code and the content length respectively.

Log management (LM) comprises of an approach to deal with large volumes of computer-generated log messages. LM covers log collection, centralized aggregation, long-term retention, log analysis and reporting. A log management infrastructure consists of the hardware, software, networks, and media used to generate, transmit, store, analyze and dispose of log data. Since cloud service providers (CSP) are separate administrative entities, data outsourcing is actually relinquishing user's ultimate control over the fate of their data. As a result, the correctness of the data in the cloud is being put at risk due to the following reasons.

First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they are still facing the broad range of both internal and external threats for data integrity.

In this paper, we introduced a secured log management technique for cloud storage using cryptography. Storage and maintenance of large volume of log data locally involves huge investment and significant hardware resources. Cloud storage provides lower startup costs, increased data security, continuous availability, reduced costs, faster deployment and a highly scalable and customizable interface. This paper is organized as follows: Section 2 provides the literature survey. Section 3 explains the overall system architecture. Section 4 explain in detail about the proposed work. Section 5 depicts the result and discussion. Section 6 gives the conclusion.

## 2.RELATED WORK

Many works have been done in this direction in the past by various researchers. Among them, Bellareand Yee [3]described the importance of forward integrity and its application in the field of intrusion detection and accountability. The notion of forward integrity is achieved by the use of message authentication codes. In their scheme, even if the attacker gains control of the system, he could only erase the entries. He could never modify or create new entries in the log file.

Schneier and Kelsey [5]proposed a logging scheme that ensures that the attacker will gain little or no information from the stored log files and limit the ability to corrupt the log records. They also proposed a method for protecting all log entries generated prior to the logging machine's compromise impossible for the attacker to read and also impossible to undetectably modify or destroy. Their scheme leverages on the fact that the untrusted machine creating the log file initially shares a secret key with a trusted verification machine. In their model, the log's authentication key is hashed, using a one-way hash function, immediately after a log entry is written. Each log entry's encryption key is derived, using a one-way process, from that entry's authentication key. In their work, they have presented a general scheme that allows keeping an audit log on an insecure machine, so that log entries are protected even if the machine is compromised. The primary limitation of this work is that an attacker can seize control of an insecure machine and simply continue creating log entries, without trying to delete or change any previous log entries.

Holt [9] provided a strong cryptographic assurances that the data stored by a logging facility before a system compromise, cannot be modified after the compromise, without detection. He describes how the process of log creation can be separated from log verification process. The major improvements include a method of securing multiple, concurrently maintained logs using a single initial value, and a method of aggregating multiple log entries to reduce latency and computational load.The most significant aspect of their scheme is the ability to use public key cryptography. Using the symmetric techniques described any user who wishes to verify a log must possess the secret used to create the MACs. This secret gives the entity the ability to falsify log entries as well, which could be a major drawback in many applications. He also showed how to allow forward secrecy as well as tamper evidence in the simple system and then added a public key variant allowing verifiability without the ability to forge entries.

Dolev and Yao [4] proposed a system or a model that could detect the vulnerabilities of an improperly designed protocol. They argued that an improperly designed protocol could be vulnerable to an active saboteur who may impersonate another user or alter the message during transit. They proposed several models as well as characterizations and algorithms to determine the protocol security. Their scheme encodes the name of the sender together with the plaintext in the encrypted text. They used two sets of protocols namely the Cascade protocols and Name-Stamp protocols. In Cascade protocols, users can apply the public key encryption-decryption operations to form messages. In Name-stamp, users are allowed to append, delete, and check names encrypted together with the plaintext.Dolev–Yao attacker model describes the system which allows the attacker the following capabilities.The attacker can try to impersonate legitimate hosts. The attacker can attempt to read, delete, and modify data stored in the storage.The attacker can intercept any message that is sent over the network.The attacker can duplicate, replicate, and replay messages in his control.

Rayand Belyaev [1] described the algorithms for log file preparation, upload tag generation and the algorithms necessary for uploading the log data to the cloud and also the algorithms for rotating and deleting log records. They described a novel scheme for transforming and transmitting the log records to the cloud storage. The log file preparation algorithm deals with the preparation of the log file to be uploaded to the logging cloud. The raw log records received from the log generators are gathered and then converted into batches whose size is determined at random to increase the security against attackers. Three keys are used to provide integrity, confidentiality and privacy. They also proposed a scheme in which they create a sealed log entry which consists of three parts namely the log initialization part to denote the starting of the batch, the normal log entries and a log close part to indicate the ending of the batch. After that they are associated with an upload-tag, which is used to identify the log batch uniquely in the cloud and a delete-tag. After that they are uploaded to cloud storage. The delete and rotating the log records require authorization. The deletion and rotation of log records can be delegated to a third party.

## 3. SYSTEM ARCHITECTURE

The overall architecture of the proposed system is shown in figure 3.1. It comprises the system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system.The proposed system architecture consists of various components includes the log generators, log monitor, logging client and logging cloud. A log generator connects with logging client through wire or wireless the admin should decrypt the log files and download.

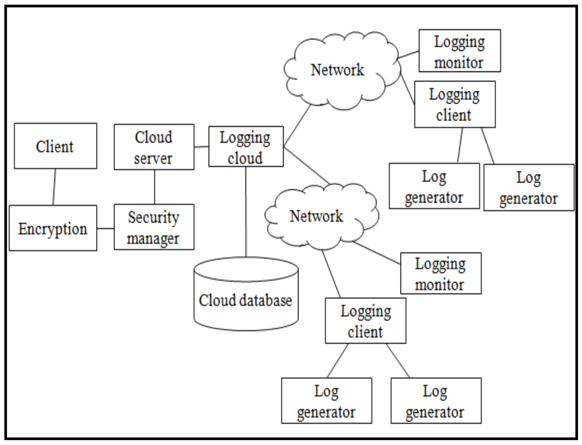

Figure 3.1 System Architecture

In this architecture, the network and logging client, log monitor connects directly through wireless or wired network. The client connects to the cloud storage through an anonymous channel. Here the Logging information of the clients are generated by the Log generators. Logging monitor is monitoring the whole activities of the clients. After that these logging client's Logging details are encrypted and stored in the Cloud Database through the Logging cloud. If system admin wants to use those stored logging files then the admin should decrypt the log files and download.

## 4. PROPOSED ALGORITHM

RSA (Rivest-Shamir-Adleman) is the public key algorithm that generates public and private key pair. RSA is one of the first practicable public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret.The mathematical details of the algorithm used in obtaining the public and private keys are available at the RSA Web site. Briefly, the algorithm involves multiplying two large prime numbers (a prime number is a number divisible only by that number and 1) and through additional operations deriving a set of two numbers that constitutes the public key and another set that is the private key. Once the keys have been developed, the original prime numbers are no longer important and can be discarded. Both the public and the private keys are needed for encryption /decryption but only the owner of a private key ever needs to know it. Using the RSA system, the private key never needs to be sent across the Internet.The private key is used to decrypt text that has been encrypted with the public key. Thus, if I send you a message, I can find out your public key (but not your private key) from a central administrator and encrypt a message to you using your public key. When you receive it, you decrypt it with your private key. In addition to encrypting messages (which ensures privacy), you can authenticate yourself to me (so I know that it is really you who sent the message) by using your private key to encrypt a digital certificate. When I receive it, I can use your public key to decrypt it. A table might help us remember this.

Let $k_p(e, n)$ be the public key and $k_r(d, n)$ be the private key. We have used intelligent agents to do the following steps:

**Step 1**: Agent chosen two large prime numbers p and q usually it is of the order 10 power 100.

**Step 2:** Agent finds the product of these two prime numbers

$$n=p*q$$

**Step 3:** Agent choose a variable z such that

$$z= (p-1)*(q-1), \text{ z-co prime to n.}$$

**Step 4:**Agent choose a variable 'e' such that 1<e<z

e should be a prime number and co-prime to n

**Step 5:**Agent choose a variable‘d’according to that (d*e) mod z=1,d-prime number

**Encrypt:** $c=m^e \bmod n$

**Decrypt:**$c^d \bmod n=m$

We have used the existing RSA algorithm for encryption and decryption with the introduction of intelligent agent. The intelligent agent can perform well over the process of encryption and decryption.

## 5. RESULTS AND DISCUSSION

In this secure data storage work, new techniques for log security in cloud are designed and implemented. For this purpose, the log data are aggregated to form batches and are encrypted and stored. The main advantage of encrypting log files is that the user behavior analysis cannot be carried out by unauthorized users.In order to perform effective encryption, AES algorithm is used for in log management leading to effective recovery of information.In addition, MD5 algorithm is used to compress and secure the log data. In addition, MAC is used to perform authentication so that log records can be created and maintained only by authorized users. The main contribution of this project work is the provision of security.

For evaluating the proposed system, we have to answer some questions such as how does encryption of log records affect the overall performance of our solution?, and how does anonymous communication affect the performance of our solution?. For the responsible of answer the first question is to be designed a set of simulations that allows us to analyze the effectiveness of different encryption settings. The test scenarios also created to understand the time (overhead) that is also introduced by the log record preparation process. We have conducted various experiments with different batch sizes of 10, 20, 30, 40 and 50 encrypted log records. For each and every log batch is measuredand three different log preparation settings.

First, we have measured the time that it takes to fill up the log batch with log records without any security related preparation and it produces the best achievable performance. Second, we measured an overhead for filling up log batch with partial security related log preparation. This is also has unencrypted log records with calculated MAC and aggregated MAC functions. Third, we have calculated an overhead for security preparation that includes encrypted log records with corresponding MAC and aggregated MAC functions (correctness, tamper resistance, verifiability and confidentiality assurances).

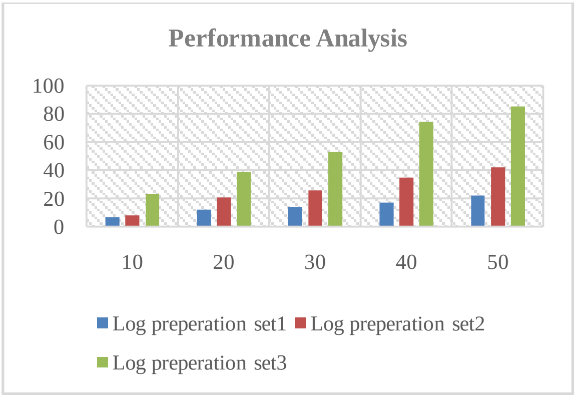

Figure 5.1 Performance Analysis (Batch Creation Overhead)

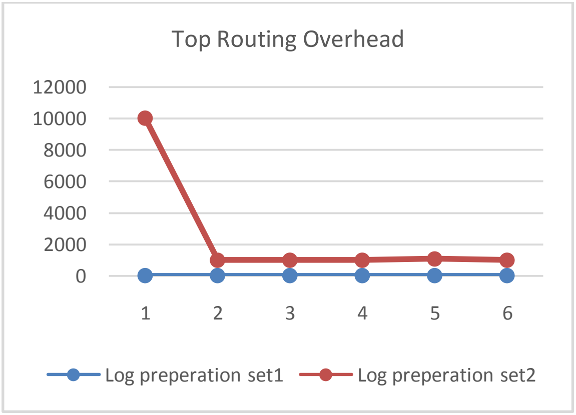

Figure 5.2 The Routing Overhead

The experiments involved two identical machines with a2.13 GHz Intel Xeon E7 CPU and 128GB memory. Both machines were running Fedora 15 64-bit Linux with 2.6.42.7 kernel and were stationed on the same LAN. Figure 5.1 shows the performance analysis of routing. Figure 5.2 shows results from our experiments. From the experimental results, it can be observed that the performance of the proposed system is better than the existing system due to the introduction of a secured log management technique and intelligent agents for cloud storage. The intelligent agent can be performed well over the decision making and improve the processing speed of the proposed system.

## 6. CONCLUSION

A secured log management technique is proposed and implemented in this paper for cloud storage using cryptography. We have used intelligent agents for improving the performance of the system. Log data now represents over 30% of all data generated by enterprises, creating a real need for log management. Storage and maintenance of large volume of log data locally involves huge investment and significant hardware resources. Cloud storage provides lower startup costs, increased data security, continuous availability, reduced costs, faster deployment and a highly scalable and customizable interface. In this project, we developed a system and used suitable techniques for securely storing log records to data storage. We have studied the existing log management protocols and identified the vulnerabilities. We also used a scheme that addresses not only security and integrity issues but also ensures the security during other stages in the log management process, including log collection, transmission, storage and retrieval. Finally, we have implemented the secure logging techniques that sends the generated log records securely to the data storage.

## REFERENCE

[1] Ray I, Belyaev K, Strizhov M, Mulamba D, Rajaram M, "Secure Logging as a Service—Delegating Log Management to the Cloud," Systems Journal, IEEE , vol.7, no.2, pp.323-334, June 2013.

[2] Ma D, Tsudik G, "A New Approach to Secure Logging," ACM Transactions Storage, vol. 5, no. 1, pp. 2:1–2:21, March 2009.

[3] Bellare M, Yee B S, "Forward Integrity for Secure Audit Logs," Department of Computer Science, University of California, San Diego, Technical Reports, November 2011.

[4] Dolev D, Yao Y, "On The Security of Public Key Protocols," IEEE Transactions on Information Theory, Vol. It-29, No. 2, March 1983.

[5] Schneier B, Kelsey J, "Security Audit Logs to Support Computer Forensics," ACM Transactions on Information System Security, vol. 2, no. 2, pp. 159–176, May 2011.

[6] Shamir A, "How to Share a Secret," ACM journal, vol. 22, no. 11, pp. 612–613, November 2009.

[7] Blakley G.R, "Safeguarding Cryptographic Keys," in Proceedings of the National Computer Conference, p. 313-314, June 2008.

[8] Ik Rae Jeong, Jeong Ok Kwon, and Dong Hoon Lee, "Strong Diffie-Hellman-DSA Key Exchange" IEEE Communiacations Letters, vol. 11, no. 5, pp. 432-433, May 2007.

[9] HoltJ. E., "Logcrypt: Forward security and public verification for secure audit logs," In Proceedings of the 4th Australasian Information Security Workshop, pp. 203–211, 2010.

[10] BalaBit IT Security. Syslog-ng—Multiplatform Syslog Server and Logging Daemon[Online]. Available: http://www.balabit.com/network-security/syslog-ng, September 2011.