RESEARCH ARTICLE OPEN ACCESS

Service Delegating Log Management-For Secure Logging In Cloud Environment.

B.Sudha(M.Tech)[1], Mr.S.ArunKumar (Assistant Professor)[2]
SRM,Ramapuram.

----------------------------**********************------------------------

Abstract:
Securely maintaining log details over extended periods of time is very significant for the proper operations of any organization. The information about the log files often contains trustworthy information, secrecy and privacy of log records are constantly imperative. However, building a secure logging infrastructure involves more capital expenses that many organizations may find exhausted. Providing log management to the cloud appears to be a great cost saving measure.
In this paper, we identify the goals for a secured cloud-based log management service and propose an infrastructure for doing the same.

***Keywords* — Log Management, Cloud Computing,Encryption,Decryption,Authentication,vIntegrity.**

----------------------------**********************------------------------

I. INTRODUCTION

Log management (LM) is an approach to deal with big measures of events taking place within the organization's system or net. LM covers log collection, centralized aggregation, long-term retention, log rotation, log analysis (in real-time and in bulk after storage), log search and reporting. In current era - concerns about security, system and network operations drives Log management.

A large part of software development involves - monitoring, troubleshooting and debugging. Logging makes this, a much more comfortable and smoother operation. Application performance monitoring (APM) tools are great to access some of the core performance metrics. Nevertheless, it is common for traditional APM solutions, which have taken in the past to give 100%

end-to-end visibility for on-premise systems, to just give a fraction of that visibility for cloud-based arrangements. It can be difficult to instrument the cloud and thus, alternative approaches like logging, are asked to give visibility into cloud-based components which otherwise can become black boxes from a security, performance or a system monitoring perspective.

Log records may often need to be made available to outside auditors who are not connected to the system. Deploying a secure logging infrastructure to take on all these challenges entails significant infrastructural support and capital expenses that many organizations may find overwhelming.

The emerging paradigm of cloud computing promises a low cost opportunity for organizations to store and manage log records in a proper way. Organizations can outsource the long-term storage requirements of log files to the cloud. Cloud computing is typically specified as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to manage applications.

Log data can provide valuable security and operations insight into enterprises and IT operations. Many companies with limited IT staffing will find that outsourcing log management can bring them more value from their log information than they could accomplish on their own—without all the expenditures in hardware, staffing and merchandise management. If in-cloud providers can deliver prompt, safe, dependable service, cloud-based log management could be a development sector over the adjacent few years, especially for the SMB marketplace. One of the great questions in creating a decision between internal and in-cloud log management is how much time can be allocated to monitoring and upgrading the system internally to fit the commercial enterprise demands of the governing body. If an organization's primary goal is regulatory compliance or to minimize IT staff requirements, then outsourcing log management to cloud application providers will probably be desirable. Organizations that decide to outsource their log management should be careful to select flexible services that grant for expanded correlation and use of the log data for organizational benefit. This is likewise true of internally-developed log management systems, which today is experiencing interoperability issues that make data normalization and correlation difficult for organizations of all sizes.

Our main objective of this paper is to prove secure framework for data sharing in a cloud computing environment. This task is based on protection issues involved in log management for good logging.

II. LOG MANAGEMENT IN CLOUD

Logging is often not the most exciting function of an application, but definitely an important one. Logging and managing logs become even more important in a multi node clustered environments where logs are spread across nodes. It is not a very productive proposition to log on to each node to control what is bumping on a node or to follow the transaction which might be open across multiple clients. Moreover, ability to filter and parse logs on a node is limited by native instruments such as grep on *nix systems. However, log management or centralized logging, demonstrated at a high stage in the below image provides a safer resolution for this:

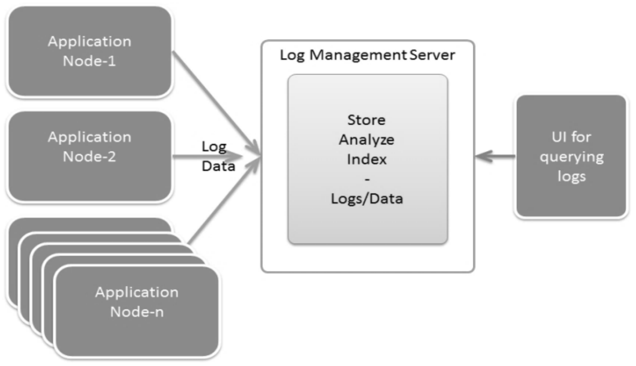

Fig.1 Log Management in Cloud

This might sound complicated to setup, but it has immense benefits and also easy to pose up.

A. Production Troubleshooting & Application Debugging

While troubleshooting a live application, logs will be the lifeline which will salvage our day. On development environments, in that location are enough tools at hand which make debugging application much more easygoing. But for an application which is live in production, logs provide key data about what is running on in the application. On the other hand too verbose logging might hamper performance of an application.

"Debugging statements stay with the program; debugging sessions are transient"

For applications deployed in the private / public cloud, or even deployed within enterprise in a clustered fashion, it will be much easier to troubleshoot and debug application if all our logs can be searched and parsed all in one place. Inefficiencies of scrolling over thousands of lines of raw data through multiple nodes and trying to connect dots would be resolved by a log management system which offers a unified survey of all logs and a much richer query interface.

B. Information Analysis

A working application has multiple user interactions and transactions running and being capable to collect real time insights from these would be extremely valuable. Applications can be planned to log desired information about transactions and this information from logs can be analysed and monitored in real time. Just imagine if the CEO of a retail company could see in real time statistics of how many shopping carts are converted into actual buyers. While analytics is a huge field in itself, we have the idea of the possibilities.

C. Monitoring and alarming

Being able to report on errors, exceptions and crashes in real time can lead to natural processes which reduce downtime and much proactive response to issues in a live application. Alarms can be set up based on severity level and actions can be chosen accordingly.

III. CHARACTERISTICS OF LOG MANAGEMENT

Logstorage allows searching our log data through a simple browser interface. We can even combine multiple search conditions to create complex searches. Key features include:

- Shape-based search (8 types of condition)
    1. From/to dates (taken)
    2. Free keyword selector tool
    3. Log host address
    4. Diligence
    5. Ticket
    6. Installation
    7. Precedence
    8. Operation ID
- Combine search conditions using AND, OR
- Restrict search using regular expressions, NOT, "Greater than", "Less than", "Starts with"
- Define 'column sets' to show only the data we need

A. Drill down within search results ('tracking' feature)

When we click on any data within a search result, the system will automatically re-run the search, this time restricting the search domain to show only results containing that data. We can also add highlighting to search results, permitting one to easily spot important log entries.

B. Log statistics (optional module)

Using the statistics module, we can compile log statistics, even for logs of many different formats. For example, one can view statistics on the number of accesses per user, the number of accesses to a particular file, error log statistics, etc.. This would be incredibly flexible and powerful module allowing users to make a spacious sort of statistics. It's possible to use raw forms, maxima, minima, averages, totals and standard deviations in calculations. The terminations can be presented as a table, bar graph, line graph or pie chart.

C. Sensors & alert (optional module)

It is potential to sense incoming logs. For instance, an unauthorized access occurs on our network or one of the applications has a fatal mistake, the system will immediately post an alarm. Alerts can take the shape of an SNMP trap, an email and/or a user-defined command.

D. Coverage (optional module)

Using this module, we can output the results of searches and statistics as a report. Reports can be done immediately or scheduled to be executed at regular intervals. Report formats can be text, CSV, PDF and HTML. It is as well possible to make custom reports in any desired format.

E. Log format definitions

We can put up whatever number of log format definitions, which distinguish the system what each component of a given log message means. Using tags, meaning can be bound to a log message, letting us to confine searches to a particular application, create statistics comparing applications, etc.

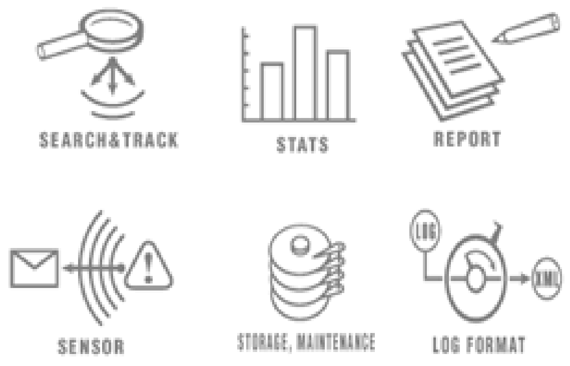

Fig.2Features of Log Management

F. Log verification

By enabling anti-tampering protection, it is potential to pick out any alterations to the logs and to verify that logs are all over.

G. Chronicle entries access control

By defining users and groups, we can have fine Grained control over who can view logs and who can perform which actions within Logstorage.

H. Scalability

With funding for broadcast performance and redundant servers, it is potential to manage logs even for very large scale organizations.

I. Great collection

Log files such as those output by Apache and Oracle are supported by Log storage, as are Windows event logs and many other log formats. In fact, Logstorage can collect and manage textual logs in absolutely any format.

Below are the main log collection methods:

a) Syslog collection

Logs transmitted over the network using the SYSLOG protocol can be collected.

b) Realtime log collection using Agent

Specially designed Logstorage Agent (available in Java and .NET editions) can retrieve logs which are salted away as local files on terminals and transport them in real-time to the Logstorage system.

c) Periodic log collection by FTP

Logs stored as local files on terminals can be periodically posted to the Logstorage system using any standard FTP client.

d) Periodic log collection usingSecureBatchTransfer

This instrument can be applied to easily send log files, from our servers to Logstorage.

e) Periodic log collection and analysis using EventLogCollector

ELC Connects to our Windows client machines to download their event logs and transmits them to Logstorage. Automated log analysis turns raw Windows event logs into more meaningful information.

In all the above proposed Log collection and management, security of logs is a central component.

IV. PRIMARY NEEDS AND THREATS OF SECURITY LOGGING

Log files need to be guaranteed by the establishment. But, security of an establishment is an overhead. The desirable properties for secure logging are analyzed in the below section. Besides the threats are discussed in the subdivision.

A. Desirable Properties

The desirable properties for secure logging are listed under:

- Correctness: The log data stored in log files should be right. It stands for the data should exactly the same as it was fathered.
- Tamper Resistance: The log files should not permit anyone other than the creator to enter valid log entries. And if any manipulation is performed, it should be observed.
- Verifiability: Each log entry should carry adequate data to verify its authenticity to make sure that the log entries are not modified.
- Confidentiality: As attacker can gather raw information from log books, log records should be stored so that they should not be accessible or searchable to anyone in the mesh.
- Privacy: Log records should not be followed by the attacker during its transportation system and its memory.

B. Major Functional Components

The major factors required to produce the cloud based framework for log security they are as follows:

- Log Generator: These are the computing devices in the organization generating log data by executing several actions.
- Logging Client: The logging client acts as a collector which collects log data of several log generators together.

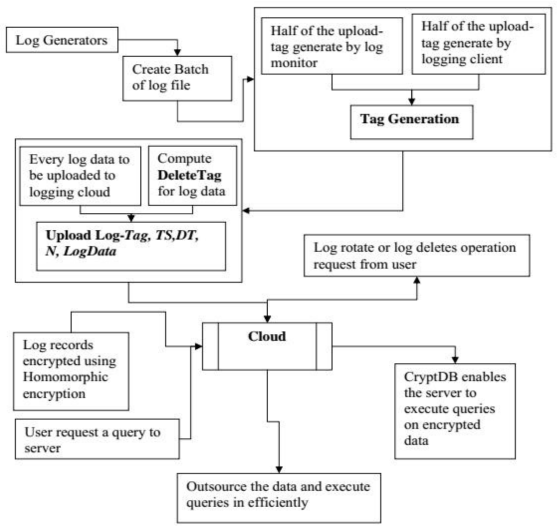
Log Generators
Create Batch of log file
Half of the upload-tag generate by log monitor
Half of the upload-tag generate by logging client
Tag Generation
Every log data to be uploaded to logging cloud
Compute DeleteTag for log data
Upload Log-Tag, TS,DT, N, LogData
Log rotate or log deletes operation request from user
Cloud
Log records encrypted using Homomorphic encryption
CryptDB enables the server to execute queries on encrypted data
User request a query to server
Outsource the data and execute queries in efficiently

Fig.3System Architecture for cloud-based secure logging

- Logging Cloud: The long term warehousing and maintenance of log data are the services provided by logging cloud to several systems.
- Log Monitor: Log monitor can generate multiple queries to retrieve information from the swarm.

C. Threat Model

This part identifies the threats present in the current position while securing log records. Dissimilar types of attacks that we need to protect against are:

- Integrity of Log Records during transit: The attacker can obtain unauthorized access to the communication medium. And he can not just have access to the data, but also he can modify the data which is transposed to the log server for secured storage.
- Authenticity of Log Record Generator: The attacker may pretend to be valid network user and lead off to send log records from someone else's identity.
- Privacy of Log Records: The attacker may try to correlate the log records over network traffic to find the data about sensitive transactions of an establishment.

Module Description

(1) Module 1: Log File Preparation for Secure Storage
- Log Aggregation and Encryption Module
- Compute MAC and aggregated MAC Module

(2) Module 2: Secret Sharing Module

(3) Module 3: Upload, Retrieval and Deletion of Log Data
- Upload Tag generation and storage Module
- Delete Tag generation Module
- Log Retrieval Module

Module 1: Log File Preparation for Secure Storage: In this module, Log files are aggregated from different log generators. It can instantly upload to cloud, but here there is problem of cloud providers. Cloud provider can be fair, but if curios then major security problem will come. For security from a cloud provider, log records will upload to local cloud in an extremely secured manner. Log files from log generators are combined in some sights and then encryption is performed by using blowfish algorithm. After encryption, MAC will be engendered from a batch of encrypted log files. After MAC generation, aggregated MAC will generate for log term storage at local cloud server or logging cloud. Log aggregation, encryption, MAC generation and aggregated MAC generation is caused by logging client.

Module 2: Secret Sharing Module: We don't have confidence one trustworthy entity to store and handle keys and all log data. Thus, in this scenario, to give out the keys across several servers, proactive secret sharing algorithm will be applied.
(1) No single entity has got the whole secret S;
(2) Any subgroup of entities of size threshold T will conjointly recreate or recover the secret S;
(3) No subgroup of entities of size Q ¡ T will re-produce or retrieve the secret S.

Module 3: Upload, Retrieval and Deletion of Log Data: Local cloud server or Logging cloud can just accept the request of upload, retrieve and delete from authenticated and authorized guests. To unambiguously identify the log books, log records have a master key. Upload, tag can act as the primary key of log books. A log record is stacked away at local cloud server indexed by upload tag. To retrieve log records, an uploaded log batch of log records will use. To erase the log books, logging cloud throws the challenge for requester proves authorization by presenting a delete tag. The log deletes operation can be requested only by authorized entities. Evidence of possession of the Delete tag proves the necessary sanction. If deleting tag matches, then log records will permanently erase.

V.Proxy Resignature:

A technique of revocation is used so that only existing

Users can view the log events.The proxy signs on the block of messages of the previous user instead of the existing user.

VI. MATHEMATICAL MODEL

When solving problems, we have to determine the difficulty degree of problems. There are three cases of classes provided for that. These are specified as follows:

1) **P Class: Informally the class P is the class of decision problems solvable by some algorithm within a number of steps bounded by some fixed polynomial in the duration of the input.** Turing was not concerned with the efficiency of his cars, but rather his concern was whether they can simulate arbitrary algorithms given sufficient time.

2) **NP-hard Class: A problem is NP-hard if solving it in polynomial time would make it possible to resolve all problems in class NP in polynomial time.** Some NP-hard problems are also in NP (these are called"NP-complete"), some are not. We adopt an honest but a curious threat model for the logging cloud. This signifies that the cloud provider is constantly available and correctly provides the services that are required. Even so, it may try to breach confidentiality or privacy of any information that is stored locally.

(3) NP-Complete Class: A decision problem L is NP-complete if it is in the set of NP problems so that any given solution to the decision problem can be verified in polynomial time, and also in the set of NP-hard problems so that any NP problem can be converted into L by a transmutation of the inputs in polynomial time.

Summary: From all the above looks, this proposed model is of the P Class because:

(1) The problem can be solved in polynomial time.
(2) This system always produces strong effects.

Let S be the set of Inputs, Functions and Outputs S = {I, F, O} where I represent input, i.e. log file and encryption keys which are input to log files, F represents the set of roles that are done on the input. O is the Set of output.

Inputs:
I1= Log File
I2=Encryption Keys

Functions:
F1= Log File Preparation for Secure Storage
F2= Secret Sharing Module
F3= Upload, Retrieval and Deletion of Log Data

Output: O1= Retrieve Secured File
Sets:
I = {I1, I2}
F= {F1, F2, F3}
O= {O1}

Input is mapped to an output which is shown in the following Venn diagram: State Diagram showing Input and output after each Process, where:

F1= Log File Preparation for Secure Storage
F2= Secret Sharing Module
F3= Upload, Retrieval and Deletion of Log Data
F4= Success of Secret Share F5= Secure Storage

As can see from the diagram there are three sets; one of the set is of input, i.e. Log file and Encryption Keys having n tuples. Similarly for number of operations can provide secure storage to log file. So for a normal flow the time complexity of the Encryption algorithm is O (n), while the same complexity of the Hash Function become O (n+m). So time complexity for overall system become O(n) ignoring m.

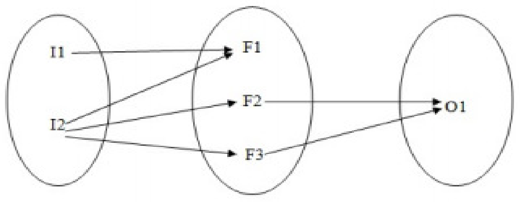

Fig.4Venn diagram

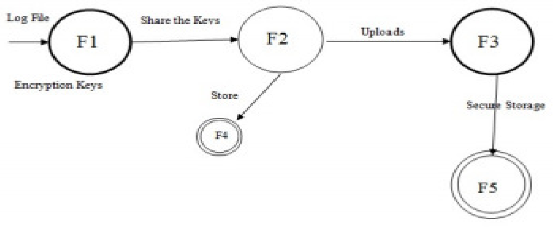

Fig.5 Process State Diagram

V. ANALYSIS:

The major objective of these protocols is to protect the anonymity of the logging client by preventing the linking of some log record stored in the cloud with the logging client that generated it. To get assurance of the anonymity we must ensure that:

1) No single party has enough information to link log records uploaded by the logging client to it, and
2) It is not possible for all parties to collude and get this information.

Assuming that the anonymous authentication protocol used by the logging cloud is sufficiently robust, attackers can- not successfully cover-up as a valid logging client. If an attacker replays messages the logging cloud can detect that because every log upload is associated with a unique timestamp. If the cloud colludes with the attacker and does not flag such messages, there will just be duplicate log data in the cloud.

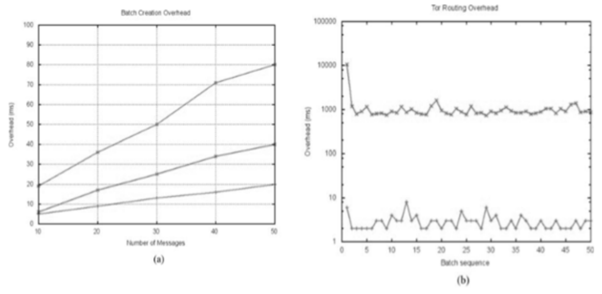

Fig.6 Showing results of experiments. (a) Batch creation overhead. (b) Tor routs overhead.

In short, by a combination of the log preparation protocol and log upload, retrieval and delete protocols, all the desirable security properties for a cloud based log management protocol are satisfied.

VII.CONCLUSION AND FUTURE SCOPE

In this project, a complete system is proposed to firmly delegate log records to a cloud. It examines existing protocols and system with their problems in numerous situations and provides complete extremely secured cloud rendered log management service. It outlines the features of Log management along with primary needs and threats of security logging. Then projected a comprehensive scheme that addresses all security issues like confidentiality, privacy and integrity not simply throughout log generation phase, however conjointly throughout all alternative stages in a logs management process. One of the unique challenges is the problem of log privacy that arises when we outsourced log management to the cloud. Log information in this case should not be casually linkable or traceable to their sources during storage, retrieval and deletion. The current implementation of the logging client is loosely coupled with the operating system based logging.

In the future, we plan to refine the log client implementation so that it is tightly integrated with the OS to replace current log process. In addition, to address privacy concerns current implementation allows access to log records that are indirectly identified by upload-tag values. We plan to investigate practical homomorphic encryption schemes that will allow encryption of log records in such a way that the logging cloud can execute some queries on the encrypted logs without breaching confidentiality or privacy. This will greatly reduce the communication overhead between a large monitor and the logging cloud needed to answer queries on logs.

VI. REFERENCES

[1] I. Teranishi, J. Furukawa, and K. Sako, "*k*-times anonymous authentication (extended abstract)," in *Proc. 10th Int. Conf. Theor. Appl. Cryptology Inform. Security*, LNCS 3329. 2004, pp. 308–322.

[2] D. L. Wells, J. A. Blakeley, and C. W. Thompson, "Architecture of an open object- oriented database management system," *IEEE Comput.*, vol. 25, no. 10, pp. 74–82, Oct. 1992.

[3] K. Nørv°ag, O. Sandst°a, and K. Bratbergsengen, "Concurrency control in distributed object oriented database systems," in *Proc. 1st East-Eur.Symp. Adv. Databases Inform. Syst.*, Sep. 1997, pp. 32–32.

[4] Subedari Mithila, P. Pradeep Kumar, "Data Security through Confidentiality in Cloud Computing Environment", Subedari Mithila et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 , 1836-1840, 2011.

[5] G. Jai Arul Jose, C.Sanjeev, Dr. C.Suyambulingom, "Implementation of Data Security in Cloud Computing", International Journal of P2P Network Trends and Technology, Vol 1, Issue 1, 2011.

[6]. Simarjeet Kaur, "Cryptography and Encryption in Cloud Computing", VSRD International Journal of Computer Science and Information Technology, 2012.

[7]. H. Harney, A. Colgrove, and P. D. McDaniel, "Principles of policy in secure groups," in Proc. NDSS, San Diego, CA, 2001.