EFFICIENT LOG MANAGEMENT OF CLOUD WITH ENCRIPTED QUERY PROCESSING FOR PROTECTION AND CONFIDENTIALITY

Ravishankkar.A.M[1] Ganeshkumar.T[2]

[1]*Associate Professor, Jay Shriram Group of Institutions,Avinashipalayam,Tirupur(India)*

[2]*PG Scholar, Jay Shriram Group of Institutions,Avinashipalayam,Tirupur(India)*

ABSTRACT

To address privacy concerns current implementation allows access to log records that are indirectly identified by upload-tag values. We plan to propose a 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. CryptDB is a system that provides practical and provable confidentiality in the face of these attacks for applications backed by databases. CryptDB can also chain encryption keys to user passwords, so that a data item can be decrypted only by using the password of one of the users with access to that data. It greatly reduces the communication overhead between a log monitor and the logging cloud needed to answer queries on logs.

Keywords: Antimalware Software, Intrusion Detection, Intrusion Prevention Systems, Log Generators, Logging Client, Logging Cloud, Remote Access Software.

I INTRODUCTION

There are many considerations for cloud computing architects to make when moving from a standard enterprise application deployment model to one based on cloud computing. There are public and private clouds that offer complementary benefits, there are three basic service models to consider, and there is the value of open APIs versus proprietary ones. IT organizations can choose to deploy applications on public, private, or hybrid clouds, each of which has its trade-offs.

The terms public, private, and hybrid do not dictate location. While public clouds are typically “out there” on the Internet and private clouds are typically located on premises, a private cloud might be hosted at a collocation facility as well. Companies may make a number of considerations with regard to which cloud computing model they choose to employ, and they might use more than one model to solve different problems. An application needed on a temporary basis might be best suited for deployment in a public cloud because it helps to avoid the need to purchase additional equipment to solve a temporary need.

Likewise, a permanent application or one that has specific requirements on quality of service or location of data might best be deployed in a private or hybrid cloud. Public clouds Public clouds are run by third parties and applications from different customers are likely to be mixed together on the cloud’s servers, storage systems,

and networks. Public clouds are most often hosted away from customer premises, and they provide a way to reduce customer risk and cost by providing a flexible, even temporary extension to enterprise infrastructure.

If a public cloud is implemented with performance, security, and data locality in mind, the existence of other applications running in the cloud should be transparent to both cloud architects and end users. Indeed, one of the benefits of public clouds is that they can be much larger than a company's private cloud might be, offering the ability to scale up and down on demand, and shifting infrastructure risks from the enterprise to the cloud provider, if even just temporarily. Portions of a public cloud can be carved out for the exclusive use of a single client, creating a virtual private datacenter. Rather than being limited to deploying virtual machine images in a public cloud, a virtual private datacenter gives customers greater visibility into its infrastructure. Now customers can manipulate not just virtual machine images, but also servers, storage systems, network devices.

Cloud computing services are often roughly classified into a hierarchy of as a service terms as following:

Infra structure as a Service (IaaS) is providing general on-demand computing resources such as virtualized servers or various forms of storage (block, key/value, database, etc.) as metered resources. This can often be seen as a direct evolution of shared hosting with added on-demand scaling via resource virtualization and use-based billing.

Platform as a Service (PaaS) is providing an existent managed higher-level software infrastructure for building particular classes of applications and services. The platform includes the use of underlying computing resources, typically billed similar to IaaS products, although the infrastructure is abstracted away below the platform.

Software as a Service (SaaS) is providing specific, already-created applications as fully or partially remote services. Sometimes it is in the form of web-based applications and other times it consists of standard non-remote applications with Internet-based storage or other network interactions.

The traditional approach to resource access in grid environments is based on a queuing model that provides best-effort quality of service. In this model jobs are queued until they can be matched with appropriate resources for execution. This approach ensures that access to resources is shared equally and fairly among all users of the system, but can result in long delays when competition between users forces jobs to wait for resources to become available. For applications with only one job, or with a few jobs that can be submitted in parallel, these delays are encountered only once.

The traditional approach to resource access in grid environments is based on a queuing model that provides best-effort quality of service. In this model jobs are queued until they can be matched with appropriate resources for execution. This approach ensures that access to resources is shared equally and fairly among all users of the system, but can result in long delays when competition between users forces jobs to wait for resources to become available. For applications with only one job, or with a few jobs that can be submitted in parallel, these delays are encountered only once.

There are two policies that can be used to guide these decisions. In *static provisioning* the application allocates all resources required for the computation before any jobs are submitted, and releases the resources after all the jobs have finished. This method assumes that the number of resources required is known or can be predicted in advance. In dynamic provisioning resources are allocated by the system at runtime.

This allows the pool of available resources to grow and shrink according to the changing needs of the application. Dynamic provisioning does not require advanced knowledge of resource needs, but it does require policies for acquiring and releasing resources. It also relies on the ability of the provisioning system to acquire

resources on- demand when they are needed, which may not be possible if the resources are shared with other users.

II. CLOUD COMPUTING INFRASTRUCTURE MODELS

2.1. Advanced Reservation

Advance reservation is a resource provisioning mechanism supported by many batch schedulers. Users create advance reservations by requesting slots from the batch scheduler that specify the number of resources to reserve and the start and end times of the reservation. During the reservation period the scheduler only runs jobs that belong to the user on the reserved resources. Although batch schedulers used by many resource providers have advance reservation features, few providers support the use of reservations. Singh, et al. conducted a survey of advance reservation capabilities at several grid sites.

They discovered that 50% of the sites surveyed did not support reservations at all, and that most of the sites that did support reservations required administrator assistance to create them. Only a few sites allowed users to create their own reservations. This lack of support makes using advance reservations time-consuming and cumbersome. Scheduler-based advance reservations also increase resource usage costs. In many grid environments these costs are measured in service units. Users of advance reservations are typically charged a premium for dedicated access to resources. These premiums can be 20% to 100% above normal costs. Furthermore, users are often forced to pay for the entire reservation, even if they are not able to use it all (e.g. if there is a failure that causes the application to abort, or if the actual runtime of the application is shorter than predicted).

An alternative to scheduler-based advance reservations is the use of probabilistic advance reservations. In this method reservations are made based on statistical estimates of queue times. The estimates allow jobs to be submitted with a high probability of starting some time before the desired reservation begins. This allows "virtual reservations" to be created by adjusting the runtime of the job to cover both the time between the submission of the job and the desired reservation start time, and the duration of the reservation itself. Unlike scheduler-based reservations, probabilistic reservations do not require special support from resource providers. However, probabilistic reservations are not guaranteed because the actual queue delay may exceed the predicted delay, and the final cost of a probabilistic reservation is difficult to predict because the actual runtime of the reservation job may exceed the desired reservation time.

2.2. Problem Statement

Since log files contain record of most system events including user activities, they become an important target for malicious attackers. An attacker, breaking into a system, typically would try not to leave traces of his or her activities behind. Consequently, the first thing an attacker often does is to damage log files or interrupt the logging services. Furthermore, the sensitive information contained in log files often directly contributes to confidentiality breaches.

Frequently, log information can be helpful to an attacker in gaining unauthorized access to system. One example of this is the case when a user mistakenly enters her password in the username field while logging into a system. Logging programs will store the password as the user-id to record the information that a user has failed to log in.

Last, but not least, information in log file can also be used to cause privacy breaches for users in the system since the log file contains record of all events in the system.

In this work does not consider the 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 increase the communication overhead between a log monitor and the logging cloud needed to answer queries on logs.

2.3. Guide to Computer Security Log Management

A log is a record of the events occurring within an organization's systems and networks. Logs are composed of log entries; each entry contains information related to a specific event that has occurred within a system or network. Originally, logs were used primarily for troubleshooting problems, but logs now serve many functions within most organizations, such as optimizing system and network performance, recording the actions of users, and providing data useful for investigating malicious activity. Logs have evolved to contain information related to many different types of events occurring within networks and systems. Within an organization, many logs contain records related to computer security; common examples of these computer security logs are audit logs that track user authentication attempts and security device logs that record possible attacks. This guide addresses only those logs that typically contain computer security-related information.

Because of the widespread deployment of networked servers, workstations, and other computing devices, and the ever-increasing number of threats against networks and systems, the number, volume, and variety of computer security logs has increased greatly. This has created the need for computer security log management, which is the process for generating, transmitting, storing, analyzing, and disposing of computer security log data.

Logs can contain a wide variety of information on the events occurring within systems and networks. This section describes the following categories of logs of particular interest:

- Security software logs primarily contain computer security-related information.
- Operating system logs and application logs typically contain a variety of information, including computer security-related data.

Most organizations use several types of network-based and host-based security software to detect malicious activity, protect systems and data, and support incident response efforts. Accordingly, security software is a major source of computer security log data. Common types of network-based and host-based security software include the following

2.3.1. Antimalware Software

The most common form of antimalware software is antivirus software, which typically records all instances of detected malware, file and system disinfection attempts, and file quarantines.[3] Additionally, antivirus software might also record when malware scans were performed and when antivirus signature or software updates occurred. Antispyware software and other types of antimalware software (e.g., rootkit detectors) are also common sources of security information.

2.3.2. Intrusion Detection and Intrusion Prevention Systems

Intrusion detection and intrusion prevention systems record detailed information on suspicious behavior and detected attacks, as well as any actions intrusion prevention systems performed to stop malicious activity in progress. Some intrusion detection systems, such as file integrity checking software, run periodically instead of continuously, so they generate log entries in batches instead of on an ongoing basis.

2.3.3. Remote Access Software

Remote access is often granted and secured through virtual private networking (VPN). VPN systems typically log successful and failed login attempts, as well as the dates and times each user connected and disconnected, and the amount of data sent and received in each user session. VPN systems that support granular access control, such as many Secure Sockets Layer (SSL) VPNs, may log detailed information about the use of resources.

2.3.4. Web Proxies

Web proxies are intermediate hosts through which Web sites are accessed. Web proxies make Web page requests on behalf of users, and they cache copies of retrieved Web pages to make additional accesses to those pages more efficient. Web proxies can also be used to restrict Web access and to add a layer of protection between Web clients and Web servers. Web proxies often keep a record of all URLs accessed through them.

III PROPOSED SYSTEM

3.1 Log File Preparation for Secure Storage

Central to our log file preparation protocol is the creation and management of three sets of keys—Ai and Xi which are keys for ensuring log record integrity, and Ki which is a key for ensuring log record confidentiality. These keys are deriving din sequential manner starting with three master keys $A0$, $X0$, and $K0$. We use a secret key cryptosystem to provide integrity and confidentiality. We do not rely on a single trusted entity to store and manage keys. Instead, we propose using a secret sharing scheme (such as the ones by Shamir [19] or Blakley[20]) to distribute the keys $A0$, $X0$, and $K0$ across several hosts. The idea is that given a secret S, and n and q two nonnegative integers such that $0 < q \leq n$, we would like n entities to share the secret S such that: 1) no single entity holds the complete secret; 2) any subgroup of entities of size $\geq q$ can collectively recreate or recover the secret S; and 3) no subgroup of entities of size $t < q$ can re-create or recover the secret. One of the problems of the secret sharing schemes proposed by Shamir or Blakley is that an attacker can successively compromise each host till it has compromised q hosts. In such a case, the attacker obtains the secret keys. To protect against such a possibility we use a proactive secret-sharing scheme [21], [22]. The idea behind these schemes is that at the end of a fixed period of time, the shares stored at each host change although the original secret stays the same. In this way, the window of opportunity for the attacker to compromise all q shares is significantly reduced.

3.1.1. Log Generators

These are the computing devices that generate log data. Each organization that adopts the cloud-based log management service has a number of log generators. Each of these generators is equipped with logging capability. The log files generated by these hosts are not stored locally except temporarily till such time as they are pushed to the logging client.

3.1.2. Logging Client or Logging Relay

The logging client is a collector that receives groups of log records generated by one or more log generators, and prepares the log data so that it can be pushed to the cloud for long term storage.

The log data is transferred from the generators to the client in batches, either on a schedule, or as and when needed depending on the amount of log data waiting to be transferred. The logging client incorporates security protection on batches of accumulated log data and pushes each batch to the logging cloud. When the logging client pushes log data to the cloud it acts as a logging relay. We use the terms logging client and logging relay interchangeably. The logging client or relay can be implemented as a group of collaborating hosts. For simplicity however, we assume that there is a single logging client.

3.1.3. Logging Cloud

The logging cloud provides long term storage and maintenance service to log data received from different logging clients belonging to different organizations. The logging cloud is maintained by a cloud service provider. Only those organizations that have subscribed to the logging cloud's services can upload data to the cloud. The cloud, on request from an organization can also delete log data and perform log rotation. Before the logging cloud will delete or rotate log data it needs a proof from the requester that the latter is authorized to make such a request. The logging client generates such a proof. However, the proof can be given by the logging client to any entity that it wants to authorize.

3.1.4. Log Monitor

These are hosts that are used to monitor and review log data. They can generate queries to retrieve log data from the cloud. Based on the log data retrieved, these monitors will perform further analysis as needed. They can also ask the log cloud to delete log data permanently, or rotate logs.

IV. FIGURES AND TABLES

4.1. Architecture Diagram

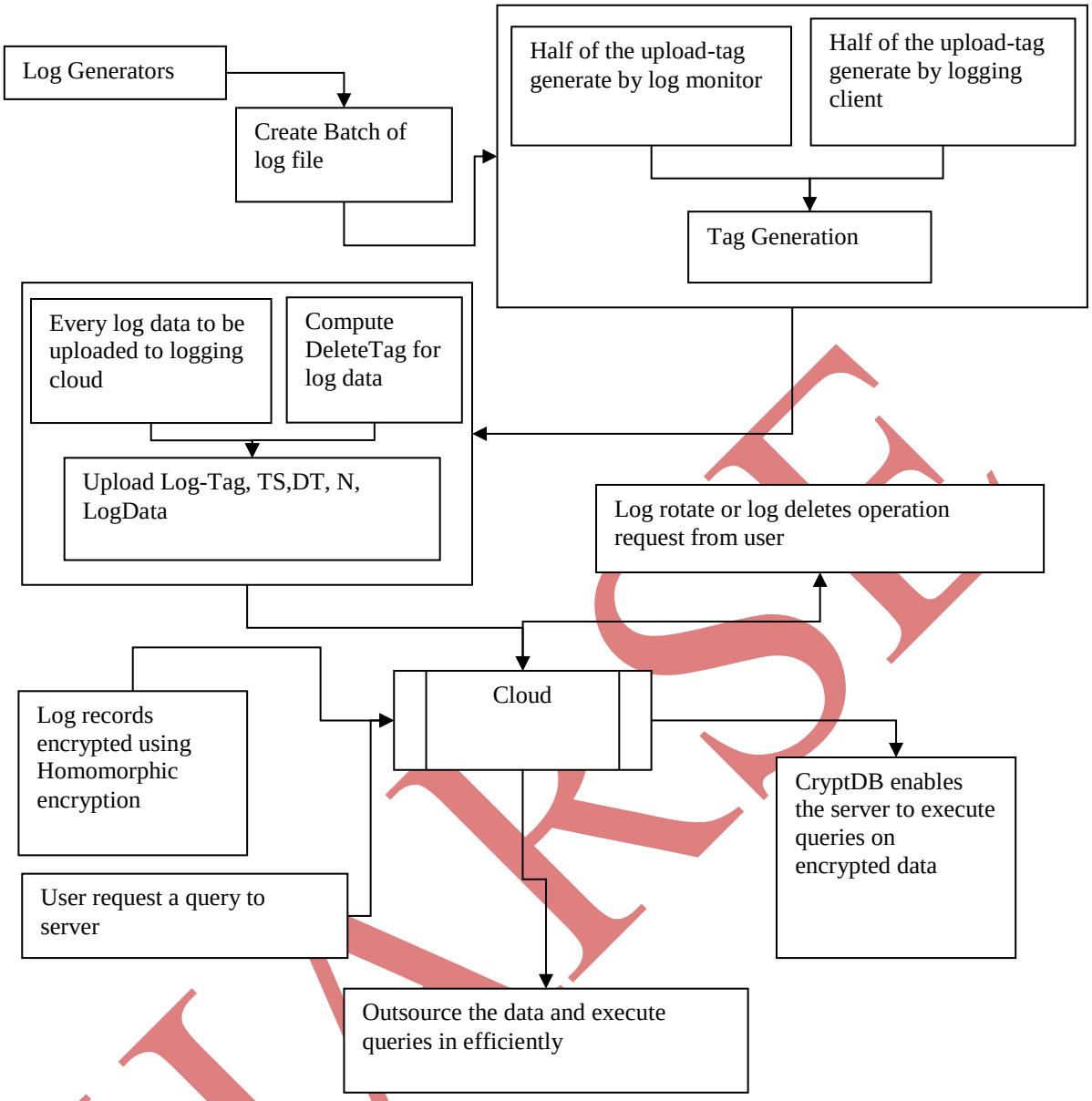

Figure 1: Architecture Diagram

4.2. Data Flow Diagram

4.2.1. Log File Preparation, Integrity and Secrecy for Log Records

4.2.1.1. Level 0

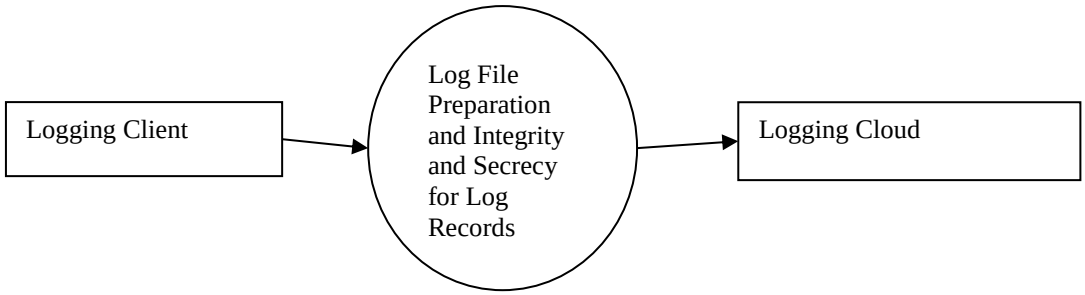

Figure 2: Data Flow Diagram For Level 0

4.2.1.2. Level 1

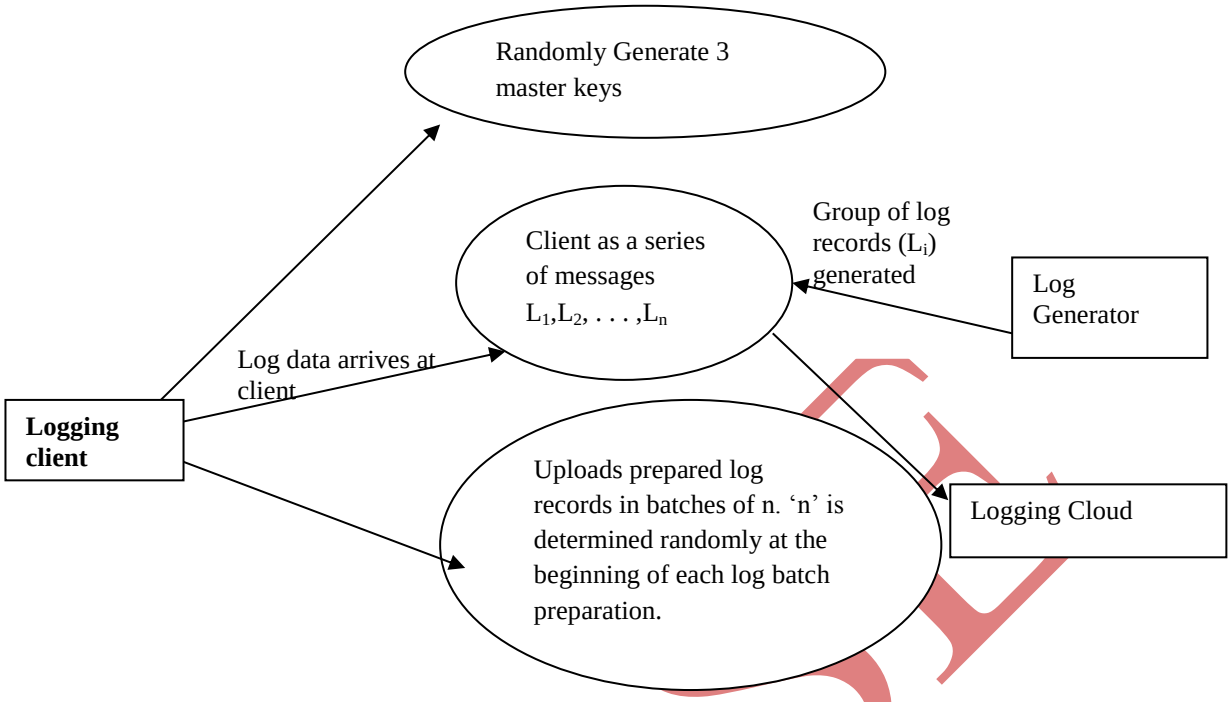

Figure 3: Data Flow Diagram for Level 1

4.2.1.3. Level 2

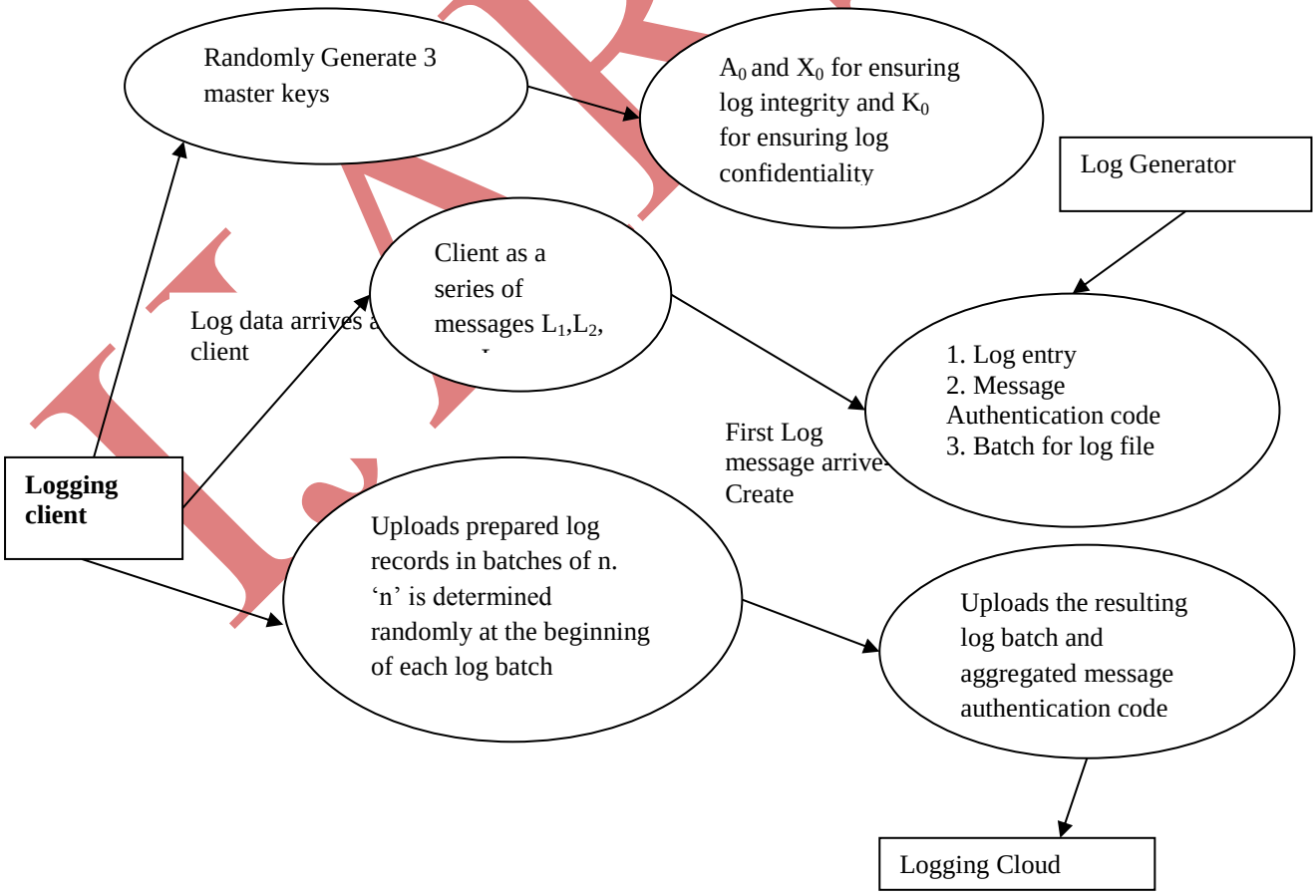

Figure 4: Data Flow Diagram for Level 2

4.3. Result

4.3.1. Screen Shots

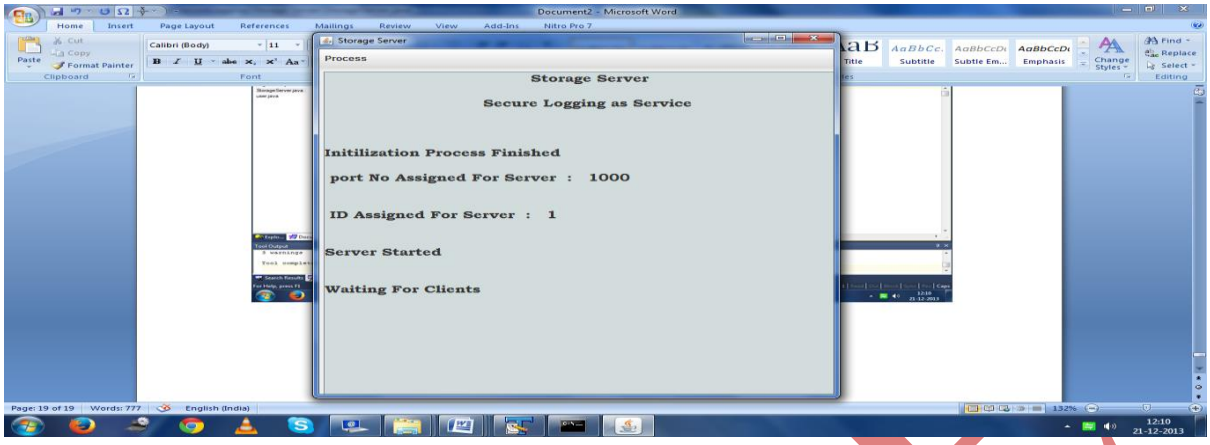

Figure 5: Execution of StorageServer

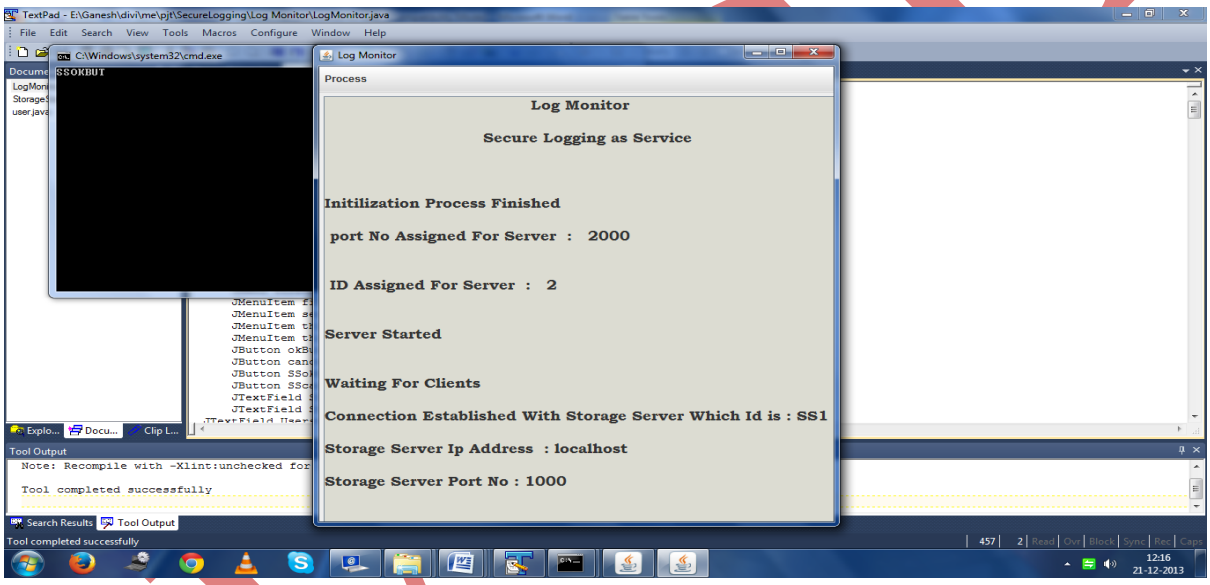

Figure 6: Execution of LogMonitor

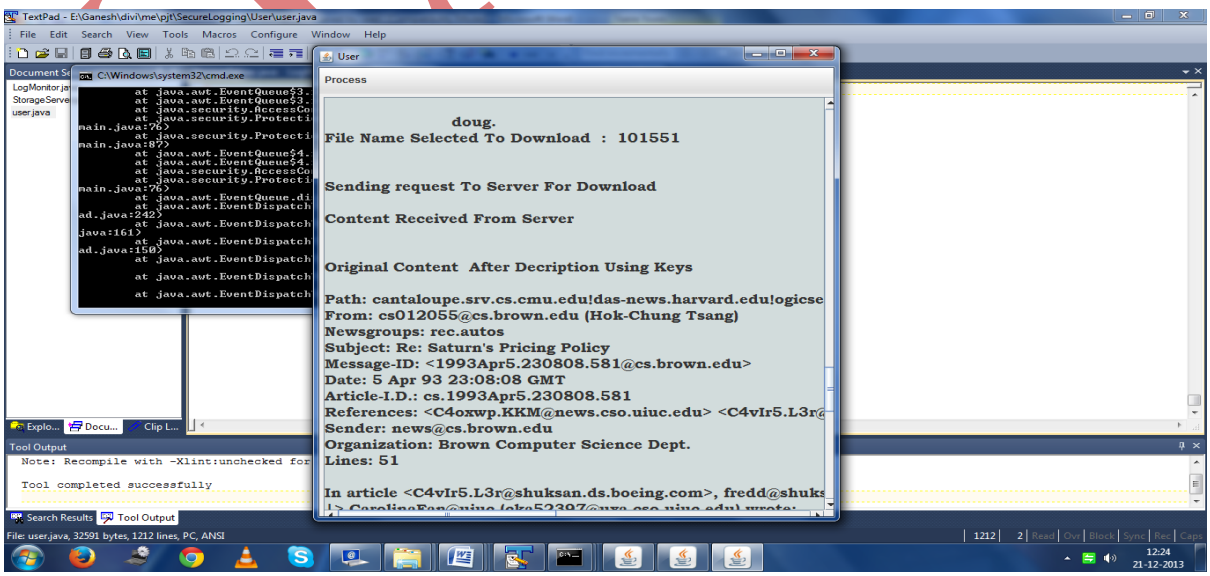

Figure 7: Execution of User

V CONCLUSION

We propose a homomorphic encryption schemes to encrypt the log records. In that the logging cloud can execute some queries on the encrypted logs without breaching confidentiality or privacy. A system that provides a practical and strong level of confidentiality in the face of two significant threats confronting database-backed applications: curious DBAs and arbitrary compromises of the application server and the DBMS. CryptDB meets its goals using three ideas: running queries efficiently over encrypted data using a novel encryption strategy, dynamically adjusting the encryption level using onions of encryption to minimize the information revealed to the untrusted DBMS server, and chaining encryption keys to user passwords in a way that allows only authorized users to gain access to encrypted data.

REFERENCES

[1] Bellare, M. and Palacio, A. 2002. Protecting against key exposure: strongly key-insulated encryption with optimal threshold. In *Cryptology ePrint Archive, Report 2002/64*.

[2] . Bellare, M. and Yee, B. 2003. Forward-security in private-key cryptography. In *Proc. Of CT-RSA'03*.

[3] Bellare, M. and Yee, B. S. 1997. Forward integrity for secure audit logs. Tech. rep. ftp://www.cs.ucsd.edu/pub/bsy/pub/¯.ps.

[4] Burns, R., Gentry, C., Lynn, B., and Shacham, H. 2005. Veri¯able audit trails for a versioning le system. In *StorageSS'05*. 416{432.

[5] Chong, C., Peng, Z., and Hartel, P. 2002. Secure audit logging with tamper resistant hardware.

[6] In *Technical report TR-CTIT-02-29, Centre for Telematics and Information Technology, Univ.Twente, The Netherlands*.

[7] Dodis, Y., Katz, J., Xu, S., and Yung, M. 2002. Key-insulated public key cryptosystems. In *Eurocrypt'02*. 65{82}.

[8] Dodis, Y., Katz, J., Xu, S., and Yung, M. 2003. Strong key-insulated signature schemes. In *Public Key Cryptography (PKC 2003)*. 130{144}.