Dynamic Auditing Services for Integrity Verification of Outsourced Storages on Cloud

Project Report submitted in partial fulfilment of the requirement

for the degree of Bachelor of Technology

in

Computer Science & Engineering

under the Supervision of

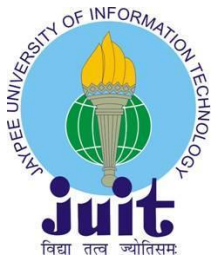
Mr. Punit Gupta

By

Nitika Thakur

111274

to



Jaypee University of Information and Technology

Waknaghat, Distt. Solan -173234 (H.P)

CERTIFICATE

This is to certify that project report entitled "**Dynamic Auditing Services for Integrity Verification of Outsourced Storages in Cloud**", submitted by **Nitika Thakur** in partial fulfilment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: **Supervisor's Name: Mr. Punit Gupta**

ACKNOWLEDGEMENT

I would like to use this opportunity to express my gratitude to everyone who supported me throughout the course of this B.Tech project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and inspiring views on a number of issues related to the project.

I am especially grateful to **Mr. Punit Gupta** , Project Supervisor , for his valuable suggestions, support and constant encouragement during the course of the project.

Date: **Nitika Thakur**

(111274)

TABLE OF CONTENTS

S.NO.	LIST OF TABLES	PAGE NO.

S.NO.	LIST OF FIGURES	PAGE NO.

ABSTRACT

The cloud storage service (CSS) relieves the burden for storage management and maintenance. However, if such an important service is vulnerable to attacks or failures, it would bring irretrievable losses to the clients because their data or archives are stored in an uncertain storage pool outside the enterprises. These security risks come from the following reasons: First, the cloud infrastructures are much more powerful and reliable than personal computing devices, but they are still susceptible to internal threats (e.g., via virtual machine) and external threats (e.g., via system holes) that can damage data integrity ; second, for the benefits of possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users ; furthermore, disputes occasionally suffer from the lack of trust on CSP because the data change may not be timely known by the cloud users, even if these disputes may result from the users' own improper operations . Therefore, it is necessary for CSP to offer an efficient audit service to check the integrity and availability of stored data . Security audit is an important solution enabling traceback and analysis of any activities including data accesses, security breaches, application activities, and so on. Furthermore, compared to the common audit, the audit services for cloud storages should provide clients with a more efficient proof for verifying the integrity of stored data. Unfortunately, the traditional cryptographic technologies, based on hash functions and signature schemes, cannot support for data integrity verification without a local copy of data. In addition, it is evidently impractical for audit services to download the whole data for checking data validation due to the communication cost, especially for large-size files. So, In this paper, we introduce a dynamic audit service for integrity verification of untrusted and outsourced storages.

CHAPTER – 1

1. INTRODUCTION

1.1 Introduction To Cloud Computing

Cloud computing is a subscription-based service where you can obtain networked storage space and computer resources. Cloud computing is is typically defined as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications.In cloud computing, the word cloud (also phrased as "the cloud") is used as a metaphor for "*the Internet*," so the phrase cloud computing means "a type of Internet-based computing," where different services — such as servers, storage and applications —are delivered to an organization's computers and devices through the Internet.

For example one way to think of cloud computing is to consider your experience with email. Your email client, if it is Yahoo!, Gmail, Hotmail, and so on, takes care of housing all of the hardware and software necessary to support your personal email account. When you want to access your email you open your web browser, go to the email client, and log in. The most important part of the equation is having internet access. Your email is not housed on your physical computer; you access it through an internet connection, and you can access it anywhere

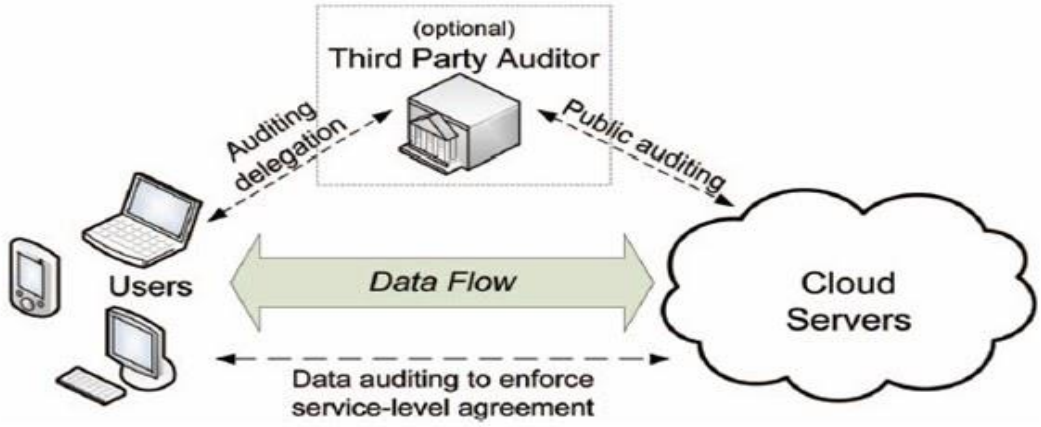


Fig. 1 CLOUD CLOMPUTING

1.2 Types of cloud:

There are different types of clouds that you can subscribe to depending on your needs. As a home user or small business owner, you will most likely use public cloud services.

1. **Public Cloud** - A public cloud can be accessed by any subscriber with an internet connection and access to the cloud space.
2. **Private Cloud** - A private cloud is established for a specific group or organization and limits access to just that group.
3. **Community Cloud** - A community cloud is shared among two or more organizations that have similar cloud requirements.
4. **Hybrid Cloud** - A hybrid cloud is essentially a combination of at least two clouds, where the clouds included are a mixture of public, private, or community.

1.3 Choosing a cloud provider

Each provider serves a specific function, giving users more or less control over their cloud depending on the type. When you choose a provider, compare your needs to the cloud services available. Your cloud needs will vary depending on how you intend to use the space and resources associated with the cloud. If it will be for personal home use, you will need a different cloud type and provider than if you will be using the cloud for business. Keep in mind that your cloud provider will be pay-as-you-go, meaning that if your technological needs change at any point you can purchase more storage space (or less for that matter) from your cloud provider.

There are three types of cloud providers that you can subscribe to: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These three types differ in the amount of control that you have over your information, and conversely, how much you can expect your provider to do for you.

Briefly, here is what you can expect from each type.

1. **Software as a Service** - A SaaS provider gives subscribers access to both resources and applications. SaaS makes it unnecessary for you to have a

physical copy of software to install on your devices. SaaS also makes it easier to have the same software on all of your devices at once by accessing it on the cloud. In a SaaS agreement, you have the least control over the cloud.

2. **Platform as a Service** - A PaaS system goes a level above the Software as a Service setup. A PaaS provider gives subscribers access to the components that they require to develop and operate applications over the internet.

3. **Infrastructure as a Service** - An IaaS agreement, as the name states, deals primarily with computational infrastructure. In an IaaS agreement, the subscriber completely outsources the storage and resources, such as hardware and software, that they need.

As you go down the list from number one to number three, the subscriber gains more control over what they can do within the space of the cloud. The cloud provider has less control in an IaaS system than with an SaaS agreement.

1.4 Benefits of Cloud Computing

- **Scalable**—Cloud vendors have excess capacity to serve needs of individuals or organizations. There are numerous examples of companies using the cloud to expand capacity with little or no notice. For example, when Indianapolis Motor Speedway, home of the Indianapolis 500, streams IndyCar races live online, it causes a huge spike in web site traffic. The Speedway worked with a cloud service provider to mirror its web site and scale up as needed during events.[5] This allows it to use only the servers it needs and save costs by monitoring servers remotely. There are other examples of cloud-hosted web site usage numbers supporting spikes of 600 percent during pivotal points in the 2008 US presidential campaign.[6]
- **Speed of execution**—A cloud computing service could be up and running in a few hours. For example, when the *Washington Post* newspaper[7] wanted to analyze 17,481 pages of data stored as image files, it turned to a cloud service provider and launched 200 instances to process the image files to its specifications within nine hours. The total cost was US $144.62.

- **Cost transparency**—It is easier to allocate cloud computing cost since most cloud service providers bill for each instance. This leads to better accountability.
- **Outsourcing competencies that are not core to the business**—Companies are not required to attract and retain human resources with critical IT infrastructure management skills.

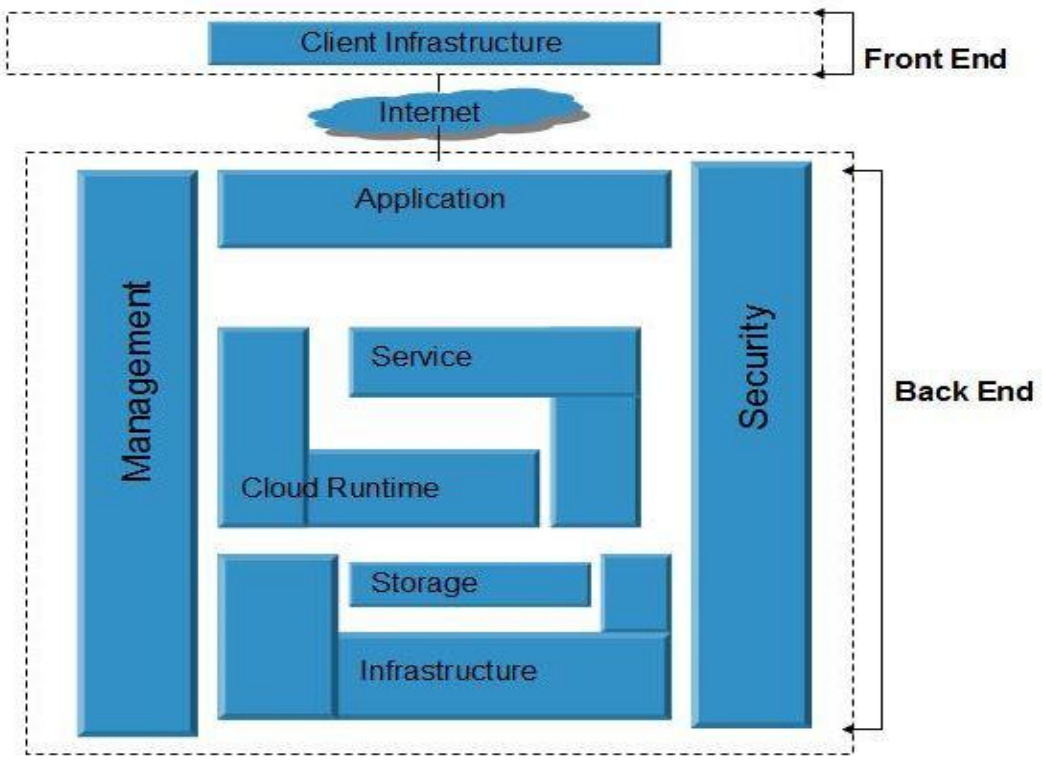
1.5 Cloud Computing Architecture

Cloud Computing architecture comprises of many cloud components, which are loosely coupled. We can broadly divide the cloud architecture into two parts:

- Front End
- Back End

Each of the ends is connected through a network, usually Internet. The following diagram shows the graphical view of cloud computing architecture:

Fig. 2 Cloud Architecture



Front End

The **front end** refers to the client part of cloud computing system. It consists of interfaces and applications that are required to access the cloud computing platforms, Example - Web Browser.

Back End

The **back End** refers to the cloud itself. It consists of all the resources required to provide cloud computing services. It comprises of huge data storage, virtual machines, security mechanism, services, deployment models, servers, etc.

1.6 CloudSim : A framework of modelling and simulation of cloud computing infrastructures and services

Recently, cloud computing emerged as the leading technology for delivering reliable, secure, fault-tolerant, sustainable, and scalable computational services, which are presented as Software, Infrastructure, or Platform as services (SaaS, IaaS, PaaS). Moreover, these services may be offered in private data centers (private clouds), may be commercially offered for clients (public clouds), or yet it is possible that both public and private clouds are combined in hybrid clouds.

These already wide ecosystem of cloud architectures, along with the increasing demand for energy-efficient IT technologies, demand timely, repeatable, and controllable methodologies for evaluation of algorithms, applications, and policies before actual development of cloud products.

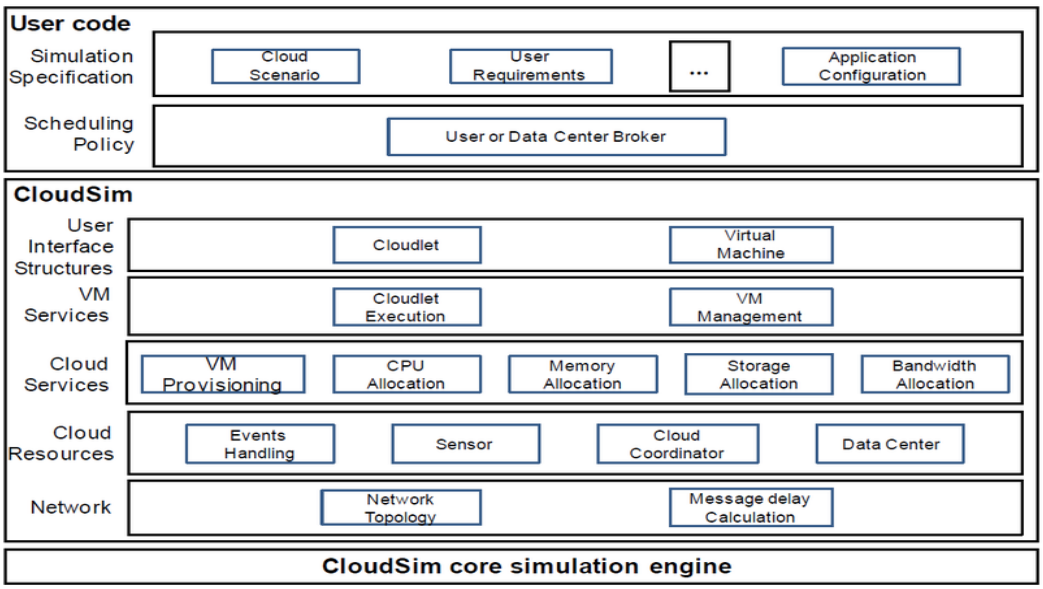
A suitable alternative is the utilization of simulations tools, which open the possibility of evaluating the hypothesis prior to software development in an environment where one can reproduce tests. Specifically in the case of Cloud computing, where access to the infrastructure incurs payments in real currency, simulation-based approaches offer significant benefits, as it allows Cloud customers to test their services in repeatable and controllable environment free of cost, and to tune the performance bottlenecks before deploying on real Clouds. CloudSim provides a generalised and extensible simulation framework that enables seamless modelling and simulation of app

performance. By using CloudSim, developers can focus on specific systems design issues that they want to investigate, without getting concerned about details related to cloud-based infrastructures and services. By using CloudSim, researchers and industry-based developers can focus on specific system design issues that they want to investigate, without getting concerned about the low level details related to Cloud-based infrastructures and services.

Main Features :

- support for modelling and simulation of large scale Cloud computing data centers
- support for modelling and simulation of virtualized server hosts, with customizable policies for provisioning host resources to virtual machines
- support for modelling and simulation of energy-aware computational resources
- support for modelling and simulation of data center network topologies and message-passing applications and for modelling and simulation of federated clouds
- support for dynamic insertion of simulation elements, stop and resume of simulation and for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines

Fig. 3 CloudSim layered Architecture



CHAPTER – 2

2. INTRODUCTION TO AUDITING

2.1 Basics of Auditing

Auditing is the ability for cloud customers to verify the presence and functioning of their provider's security measures. The process of collecting and evaluating evidence to determine whether a computer system safeguards asset, maintain data integrity, achieves organisational goals effectively and consumes resources efficiently. Important security measures are divided into those needed to support data and infrastructure security. Users choose cloud storage because of its convenient service provision. During the service process, user focus on the problem whether the data stored in the cloud is safe or not. But for the service provider, the main concern is the profits while providing convenient services. For both parties that focus on different aspects, the TPA operating as an independent and credible entity plays well in guaranteeing the trust relationship between the two parties. The TPA has professional authenticate knowledge and audit skill.

Traditionally, owners can check the data integrity based on two-party storage auditing protocols.In cloud storage system, however, it is inappropriate to let either side of cloud service providers or owners conduct such auditing, because none of them could be guaranteed to provide unbiased auditing result. In this situation, third-party auditing is a natural choice for the storage auditing in cloud computing. A thirdparty auditor (auditor) that has expertise and capabilities can do a more efficient work and convince both cloud service providers and owners. To avoid or to overcome such security issues we are going for auditing scheme. There are two types of audit namely

1. **Public audit:** Here the trusted person or the TPA is used for the verification of data i.e., to ensure the data integrity. The TPA will be between user and Service Provider(SP). The user need not to be worried about their data security. The public auditing system of data storage security in Cloud Computing is motivated, i.e., our scheme supports an external auditor to audit user’s outsourced data in the cloud without learning knowledge on the data

content. The notion of public auditability has been proposed in the context of ensuring remotely stored data integrity under different systems and security models. Public auditability allows an external party, in addition to the user himself, to verify the correctness of remotely stored data

2. **Private audit:** In the case of private auditing the user is fully responsible for verifying the integrity of their data. Here there is no involvement of the TPA. It generally uses Asymmetric encryption techniques for performing private auditing. For eg AES , DES are some of the examples.

Auditing was hard but now:

- What typically used to be static is not anymore.
- Audit analysis- Data storm problem.
- Auditing is becoming a service.

2.2 Need of Auditing

There are various risks associated with a cloud computing platform which makes auditing necessary to insure customer confidence.

1. **Data Storage:** When customers use a cloud computing platform they are essentially storing their data at external servers which are not under their control. The audits insure that the cloud provider complies with SLAs and has required security measures to protect the data. Customers may also have some legal requirements that make auditing of the cloud computing platform necessary. For example, Massachusetts has regulation 201 CMR 17.00 which mandates that certain aspects of the state resident's personal information comply with a WISP.
2. **Segregation of data:** The customers who store and process their enterprise data with external cloud providers may require that it may be stored separately from other customers as part of security measure. This can be more prevalent in SaaS applications where the customers only interact with a user interface.

3. **Identity management:** A Regular auditing process will also insure that user access and their roles are carefully monitored. If access to physical cloud infrastructure is compromised then all the hosted clients can face problems.
4. **Logging and monitoring:** Auditing will ensure that the cloud provider will have security mechanisms in place in case there is a server crash or a security incident. Cloud providers must be able to track down each security issue and through the relevant logs be able to diagnose it.
5. **Availability:** Availability would mean that the cloud providers are able to provide service 24*7 without interruptions. In case of security attack they would have enough security measures to recover from the incidents. Auditing will ensure that the measures to ensure availability are in place and customers' needs not worry about business disruptions.

2.3 Audit Considerations

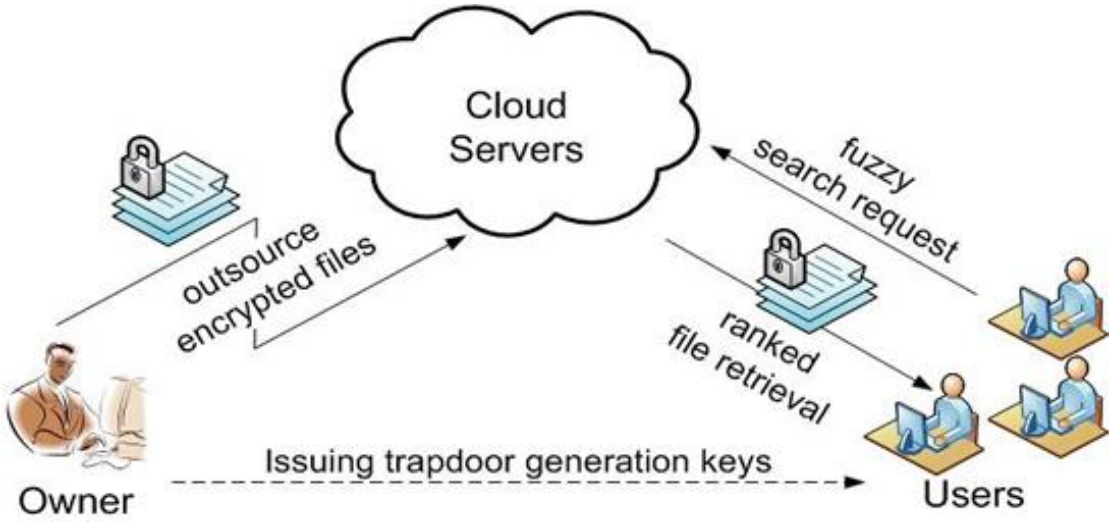
Auditors need to be involved with their organization's cloud computing plans right from the idea conception stage to help ensure identification and mitigation of risks. A number of aspects should be considered by auditors when reviewing a cloud computing project:

- Criticality of the application being sent to the cloud. While it is less risky to start with, sending noncritical applications to the cloud (for example, budgeting and expense tracking tools), significant applications such as a business-to-business (B2B) or business-to-consumer (B2C) web site should be moved to the cloud only after careful consideration.
- Country/regional regulations that affect the organization's business and require specific safeguards. Industry regulations such as the Gramm-Leach-Bliley Act (GLBA) in the US require safeguards to protect a client's nonpublic personal information, depending on how the organization collects, stores and uses the information. Under the US model of privacy, consumers have the choice to opt out of the information being shared with affiliated parties; in the European Union, Canada and some other countries, privacy laws are stringent and require specific opt-in by consumers.
- Auditors examining the cloud vendor's policy on vulnerability management and reporting (beyond basic "contact us" web site links), commitment to

following up on potential security incidents, and ability to respond promptly to reports

- Cloud users' experience with service level agreements (SLAs) and vendor management
- Auditors gaining independent assurance about controls at the cloud service provider—whether through an independent auditor's report or through audit rights in the agreement. The independent auditor's report could be a Statement on Auditing Standards (SAS) No. 70 or Trust Services report, depending on the type of application and processes outsourced. See **figure 2**for details of the key differences.

Fig. 4 Auditing



CHAPTER – 3

3. LITERATURE SURVEY

3.1 An Efficient And Secure Dynamic Auditing Protocol For Data Storage In Cloud Computing

3.1.1 Summary

We design an auditing framework for cloud storage systems and propose a privacy-preserving and efficient storage auditing protocol. Our auditing protocol ensures the data privacy by using cryptography method and the Bilinearity property of the bilinear pairing, instead of using the mask technique. Our auditing protocol incurs less communication cost between the auditor and the server. It also reduces the computing loads of the auditor by moving it to the server. The main challenge in the design of data storage auditing protocol is the data privacy problem (i.e., the auditing protocol should protect the data privacy against the auditor.).

This is because:

1. For public data, the auditor may obtain the data information by recovering the data blocks from the data proof.
2. For encrypted data, the auditor may obtain content keys somehow through any special channels and could be able to decrypt the data.

To solve the data privacy problem, our method is to generate an encrypted proof with the challenge stamp by using the bilinearity property of the bilinear pairing, such that the auditor cannot decrypt it, but the auditor can verify the correctness of the proof without decrypting it.

To improve the performance of an auditing system, we apply the data fragment technique and homomorphic verifiable tags in our method. The data fragment technique can reduce number of data tags, such that it can reduce the storage overhead and improve the system performance. By using the homomorphic verifiable tags, no matter how many data blocks are challenged, the server only responses the sum of data blocks and the product of tags to the auditor, whose size is constant and equal to only one data block. Thus, it reduces the communication cost.

Due to the large number of data owners, the auditor may receive many auditing requests from multiple data owners. In this situation, it would greatly improve the system performance, if the auditor could combine these auditing requests together and only conduct the batch auditing for multiple owners simultaneously. On the other hand, some data owners may store their data on more than one cloud servers. To ensure the owner's data integrity in all the clouds, the auditor will send the auditing challenges to each cloud server that hosts the owner's data and verify all the proofs from them.

To reduce the computation cost of the auditor, it is desirable to combine all these responses together and do the batch verification. Thus, our multi-cloud batch auditing protocol does not have any commitment phase, such that our method does not require any additional trusted organizer.

3.1.2 Conclusion

In this paper, we proposed an efficient and inherently secure dynamic auditing protocol. It protects the data privacy against the auditor by combining the cryptography method with the bilinearity property of bilinear paring, rather than using the mask technique. Thus, our multi-cloud batch auditing protocol does not require any additional organizer. Our batch auditing protocol can also support the batch auditing for multiple owners. Furthermore, our auditing scheme incurs less communication cost and less computation cost of the auditor by moving the computing loads of auditing from the auditor to the server, which greatly improves the auditing performance and can be applied to large-scale cloud storage systems.

3.2 Taking Account Of Privacy While Designing Cloud Computing Services

3.2.1 Summary

In this paper the privacy challenges that software engineersface when targeting the cloud as their production environment to offer services are assessed, and key design principles to address these are suggested.

Following three different scenarios have been considered :

Sales data analysis.

A cloud service for storage and analysis of a large database to analyse sales data and answer queries for a business. The privacy threat is the theft of sales data from the service provider's system, and its possible resale to business competitors or identity thieves.

Mining multiple databases with different owners.

A cloud service could be offered by the owner of some retail data which would identify the strongest patterns in the combination of their own data and data submitted by customers of the service, who would typically be retail businesses in the same segment. The service provider and customers are both likely to wish to minimize disclosure of data during this process.

Customized end-user services.

Information may be automatically gathered about end-user context and user data in the cloud assessed, in order to provide targeted end user services. For example, in a non-enterprise scenario, people could be notified which of their friends are near their current location.

The main threats in these types of scenarios are:

1. Personal information about a user being collected, used, stored and/or propagated in a
   way that would not be in accordance with the wishes of this user.

2. People getting inappropriate or unauthorized access to personal data in the cloud by taking advantage of certain vulnerabilities, such as lack of access control enforcement, security holes, data being exposed ‘in clear’, policies being changeable by unauthorized entities, or uncontrolled and/or unprotected copies of data being spread within the cloud.

3. Legal non-compliance. In particular, transborder data flow legislation may apply, also some of the data may count as sensitive data in a legal sense, dependant upon the jurisdiction, and more restrictive legislation about its treatment apply as a result.

3.2.2 Conclusion

We have argued that it is very important to take privacy into account when designing cloud services, if these involve the collection, processing or sharing of personal data. Privacy should be built into every stage of the product development process: it is not adequate to try to bolt on privacy at a late stage in the design process. Furthermore, we have suggested a variety of guidelines and techniques that may be used by software engineers in order to achieve this, in particular to ensure that the risks to privacy are mitigated and that data is not excessive, inaccurate or out of date, or used in unacceptable or unexpected ways beyond the control of data subjects.

3.3 Privacy Preserving Data Auditing For Data Storage Security In Cloud Computing

3.3.1 Summary

1. Guarantee that TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process.

2. Utilize the homomorphic authenticator and random masking.

3. Extend our privacy-preserving public auditing protocol into a multi-user setting, where TPA can perform the multiple auditing tasks in a batch manner.

To securely introduce an effective third party auditor (TPA), the following two fundamental requirements have to be met:

1. TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user;
2. The third party auditing process should bring in no new vulnerabilities towards user data privacy.

The system model used:

We consider a cloud data storage service involving three different entities, as illustrated in Fig. 1: the **cloud user (U)**, who has large amount of data files to be stored in the cloud; the **cloud server (CS)**, which is managed by **cloud service provider (CSP)** to provide data storage service and has significant storage space and computation resources (we will not differentiate CS and CSP hereafter.); the **third party auditor (TPA)**, who has expertise and capabilities that cloud users do not have and is trusted to assess the cloud storage service security on behalf of the user upon request.

A public auditing scheme consists of four algorithms (KeyGen, SigGen, GenProof, VerifyProof). **KeyGen** is a key generation algorithm that is run by the user to setup the scheme. **SigGen** is used by the user to generate verification metadata, which may

consist of MAC, signatures, or other related information that will be used for auditing. **GenProof** is run by the cloud server to generate a proof of data storage correctness, while **VerifyProof** is run by the TPA to audit the proof from the cloud server.

Scheme 1:

The cloud user pre-computes MACs of each block *mi* , sends both the data file *F* and the MACs onto the cloud server, and releases the secret key *sk* to TPA. During the Audit phase, the TPA requests from the cloud server a number of randomly selected blocks and their corresponding MACs to verify the correctness of the data file. The insight behind this approach is that auditing most of the file is much easier than the whole of it.

Scheme 2:

Before data outsourcing, the cloud user chooses *s* random message authentication code keys, pre-computes *s* MACs, for the whole data file *F*, and publishes these verification metadata to TPA. The TPA can each time reveal a secret key $sk_\tau$to the cloud server and ask for a fresh
keyed MAC for comparison, thus achieving privacy-preserving auditing.

In this paper, we utilize and uniquely combine the public key based homomorphic authenticator with random masking to achieve the privacy-preserving public cloud data auditing system, which meets all above requirements. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient.

Security analysis and performance evaluation:

1. Storage correctness guaranteed
2. Privacy preserving guaranteed
3. Security guarantee for branch auditing

3.3.2Conclusion:

In this paper, we propose a privacy-preserving public auditing system for data storage security in Cloud Computing. We utilize the homomorphic authenticator and random masking to guarantee that TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process, which not only eliminates the burden of cloud user from the tedious and possibly expensive auditing task, but also alleviates the users' fear of their outsourced data leakage. Considering TPA may concurrently handle multiple audit sessions from different users for their outsourced data files, we further extend our privacy-preserving public auditing protocol into a multi-user setting, where TPA can perform the multiple auditing tasks in a batch manner, i.e., simultaneously.

3.4 Data Storage Auditing Service In Cloud Computing

3.4.1 Summary

Cloud computing is a promising computing model that enables convenient and on-demand network access to a shared pool of configurable computing resources. The first offered cloud service is moving data into the cloud: data owners let cloud service providers host their data on cloud servers and data consumers can access the data from the cloud servers. This new paradigm of data storage service also introduces new security challenges, because data owners and data servers have different identities and different business interests. Therefore, an independent auditing service is required to make sure that the data is correctly hosted in the Cloud. In this paper, we investigate this kind of problem and give an extensive survey of storage auditing methods in the literature. First, we give a set of requirements of the auditing protocol for data storage in cloud computing. Then, we introduce some existing auditing schemes and analyze them in terms of security and performance. Finally, some challenging issues are introduced in the design of efficient auditing protocol for data storage in cloud computing. The existing data storage auditing methods can be classified into three categories: Message Authentication Code (MAC)-based methods, RSA-based Homomorphic methods and Boneh–Lynn–Shacham signatur (BLS)-based Homomorphic methods. For simplification, we use the Owner, the Server and the Auditor to denote the data owner, the cloud service provider (or cloud server) and the third party auditor respectively.

1. **MAC-based methods**

   The message authentication code (MAC) is a kind of hash function which has been used for checking the data integrity for a long time.

2. **RSA-based homomorphic methods**

A homomorphism is a mapping f :P $\rightarrow$Q between two groups, which has the property of $f(g1 \oplus g2) = f(g1) \otimes f(g2)$ for all $g1, g2 \in$P, where $\oplus$and $\otimes$denote the operations in P and Q respectively. The homomorphism has been used to define the homomorphic hash value or homomorphic tag which have two main types: One is based on RSA, such as the homomorphic hash value in and homomorphic tag .

3. **BLS-based homomorphic methods**

Let G1, G2 and GT be three multiplicative groups with the same prime order p. A

bilinear mapping is a mapping e : G1 ×G2 →GT with the following properties:

– **Bilinearity**: $e(ua, vb) = e(u, v)ab$ for all u ∈G1, v ∈G2 and a, b ∈Zp.

– **Non-degeneracy**: There exist u ∈G1, v ∈G2 such that $e(u, v)$ _= I, where I is the identity element of GT.

– **Computability**: e can be efficiently computed.

Such a bilinear mapping is called a bilinear pairing. Especially, if $g1$ and $g2$ are the generators ofG1 andG2 respectively, $e(g1, g2)$ is the generator ofGT. The BLS-based homomorphic tag is generated in the scenario of bilinear pairing.

Dynamic auditing: As we can see, most of the previous auditing protocols (also denoted as Proof Of Retrievability (POR) or Provable Data Possession (PDP)) are designed for the static archive storage system, e.g., libraries and scientific datasets. However, in cloud computing, the dynamic scalability is a significant issue for various applications which means that the data stored on the cloud server can be dynamically updated by data owners such as: block modification, deletion and insertion. Therefore, an efficient dynamic auditing protocol is essential in practical cloud storage systems.
There are some challenging issues for dynamic data storage auditing:
(1) Computation complexity for updates
(2) Communication cost for updates
(3) Storage overhead for updates
(4) Security requirement for updates

3.4.2 Conclusion

This paper investigates the auditing problem for data storage in cloud computing and proposes a set of requirements of designing the *Third Party Auditing* protocols. It also describes and analyses the existing auditing methods in the literature. Finally, some challenging issues in the design of efficient auditing protocols for data storage in cloud computing are discussed.

3.5 Improved Verifiability Scheme For Data Storage In Cloud Computing

3.5.1 Summary

In Cloud computing, data and service requests are responded by remote processes calls on huge data server clusters that are not totally trusted. The new computing pattern may cause
many potential security threats. This paper explores how to ensure the integrity and correctness of data storage in cloud computing with user's key pair. In this paper, we aim mainly at constructing of a quick data chunk verifying scheme to maintain data in data center by implementing a balance strategy of cloud computing costs, removing the heavy computing load of clients, and applying an automatic data integrity maintenance method. In our scheme,
third party auditor (TPA) is kept in the scheme, for the sake of the client, to periodically check the integrity of data blocks stored in datacenter. Our scheme supports quick public data integrity verification and chunk redundancy strategy. Compared with the existing scheme, it takes the advantage of ocean data support and high performance.

The contribution of this paper is summarized as follows:

1. A formal PoR scheme and its detailed protocol are presented with quick public verifiability for ocean file storage center.
2. We present the new cloud data center verification protocols to balance system efficiency and data availability.
3. The formal security proof of our protocols is presented.

Light PoR scheme:

Bilinear Map: This map can be denoted as e:$G \square G \rightarrow GT$, in which G is a Diffie-Hellman group, and group T G is another multiplicative group with the prime order p . The map holds the following properties:

(i) Computable: An efficiently computable algorithm can be found for computing e;

(ii) Bilinear: for all the 1 2 h ,$h \in G$ and two randomly chosen elements , , p a $b \in Z$ 1 2 1 2 $e(ha, hb) \square\square e(h, h)^{ab}$ holds;
(iii) Non-degeneration: $e(g,g) \neq \square 1$, in which g is one of the generators of G.

Merkle Hash Tree

The Merkle Hash Tree (note as MHT) is commonly used as one of the authentication structures presented in Ref. [12]. It can conveniently and securely verify that whether an element set is integral or corrupted. The binary MHT leaves are the hash results of the elements. In Ref. [10], the author fully took the advantage of MHT to construct a verification scheme supporting blockless data updating including data block insert. However, the client or TPA has to compute the hash root. Considering the huge capacity of a CCS, the
scheme is not as efficient as the author depicted when multi blocks are checked.

3.5.2 Conclusion

In this paper, we construct a new quick data verification scheme (Light PoR), with considering not only the secure public data verification but also the communication and computing costs. Our design can meet the goals ofunforgeablility and public verification. Because of the simple interactions of the scheme, our protocol can be executed fast and quietly. Besides, in this paper, the detail of data recovery is also shown in our scheme. Once
the verification fails, our scheme can automatically maintain the data availability of CCS clusters at a relatively high level.

3.6 Privacy Preserving Audit Of Secure Data Storage Services In Cloud Computing

3.6.1 Summary

Cloud computing is environment which enables convenient, efficient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud is kind of centralized database where many organizations/clients store their data, retrieve data and possibly modify data. Cloud is a model where user is provided services by CSP(Cloud Service Provider) on pay per use base. Means here Client has to pay for what he is using or being served. Data stored and retrieved in such a way may not be fully trustworthy so here concept of TPA(Third Party Auditor) is used. TPA makes task of client easy by verifying integrity of data stored on behalf of client. In cloud, there is support for data dynamics means clients can insert, delete or can update data so there should be security mechanism which ensure integrity for the same. Here TPA can not only see the data but he can access data or can modify also so there should be some security mechanism against this.

3.6.2 Conclusion

This paper explained different existing paper techniques and their merits and demerits. It discussed their methods of data security and privacy etc. In all those papers some haven't described proper data security mechanisms, some were lack in supporting dynamic data operations, some were lack in ensuring data integrity, while some were lacking by high resource and computation cost. Hence this paper gives overall clue of all existing techniques for cloud data security and methods proposed for ensuring data authentication using TPA.

3.7 Dynamic Audit: Dynamic Audit Services For Achieving Data Integrity In Cloud

3.7.1 Summary

Cloud computing is a forthcoming revolution in information technology (IT) industry because of its performance, accessibility i.e., cloud storage enables users to access their data anywhere and at any time, pay per use service. Cloud computing is a way to increase the capacity or add capabilities dynamically without investing in new infrastructures, training new personnel or licensing new software. Cloud enables users to remotely store their data and enjoy on-demand high quality cloud applications without the burden of local storage and maintenance. Eventhough, the benefits are higher while storing data in cloud there may be chances for security risks such as missing or corruption of data. To ensure the data integrity and availability of the data we are using the auditing scheme. Thus, enabling public auditability for cloud storage is of critical importance so that the users can resort to the third party auditor (TPA) to check the integrity of outsourced data and can be worry-free. For this in our proposed scheme we are using the provable data possession (PDP) which is the cryptographic technique for verifying the integrity of data without retrieving it at an untrusted server which achieve zero knowledge property and the communication cost is also reduced here.

Phases of third party auditing:

There are three phases of third party auditing
namely

1) Audit planning phase:
2) Execute audit phase
3) Post audit phase

In this section the author introduces an audit system architecture for outsourced data in clouds in Figure1. It consists of four main entities:

1) **Data owner(DO):** who has data files to be stored in the cloud and relies on the cloud for data maintenance, can be an individual customer or an organization.

2) **Cloud Storage Service Provider(CSP):** who provides data storage service and has enough storage space to maintain clients data.

3) **Third Party Auditor(TPA):** a trusted perso who manage or monitor outsourced data under request of the data owner.

4) **Authorized Application(AA):** who have the right to access and manipulate stored data.

3.7.2 Conclusion

There are various existing auditing schemes available to ensure integrity. But in most of the scheme there is a leakage of information from the verifier side because verification is done with local copy of data, or it lead to extra burden to the data owner in the case of private auditing. In our proposed scheme the zero knowledge property is achieved such that the third party auditor who is responsible to verify the users' data will not be having any knowledge

about data. So, the proposed scheme can be more secured when compared to the existing conventional schemes. Next, for verification only signatures are sent instead of the whole

data such that communication cost can also be reduced in the scheme. According to the paper, all these advantages of the proposed schemes will shed light on economies of scale of Cloud Computing.

3.8 Dynamic audit service outsourcing for data integrity in clouds

3.8.1 Summary

Cloud-based outsourced storage relieves the client's load for storage management and preservation by providing an equivalently scalable, low-cost, location-independent platform. Clients no longer have physical control of data indicates that they are facing a potentially frightening risk for missing or corrupted data. To keep away from the security risks, audit services are significant to make sure that the integrity and availability of outsourced data. Provable data possession (PDP), which is a cryptographic technique for verifying the integrity of data without retrieving it at anun trusted server, can be used to recognize audit services. In this we introduced the construction of an well-organized audit service for data integrity in clouds. Profiting from the typical interactive verification system, we projected an interactive audit procedure to implement the audit service based on a third party auditor. In this audit examination, the third party auditor can concern a periodic authentication to check the change of outsourced data by providing an optimized to-do list. To understand the audit model, we only need to preserve the security of the third party auditor and organize an insubstantial daemon to execute the verification protocol. We present a capable method for selecting an optimal parameter value to minimize computational expenditure of cloud audit services. Our results demonstrate the effectiveness of our approach.

3.8.2 Conclusion

Cloud Computing releases the world of computing to a wider range of uses and increases the ease of usage by giving access through any kind of internet connection. Though with these increased ease of usage also come drawbacks. Privacy security is a key issue for cloud storage and is to be considered very important. To ensure that the risks of privacy have been mitigated a variety of techniques that may be used in order to achieve privacy. This paper has addressed some privacy approaches for overcoming the issues in privacy on un trusted data stores in cloud computing. Categories the methodologies in the literature as encryption based methods, access

control based mechanisms, query integrity/ keyword search schemes, and audit ability schemes. The work is giving an efficient privacy-preserving storage compared to other works. Even though there are many approaches in the literature for mitigating the concerns in privacy, no approach is fully sophisticated to give a privacy-preserving storage that overcomes all the other privacy concerns. Thus to deal with the concerns of privacy, we need to develop privacy–preserving framework that overcomes the worries in privacy security and encourage users to adopt cloud storage services more confidently.

CHAPTER – 4

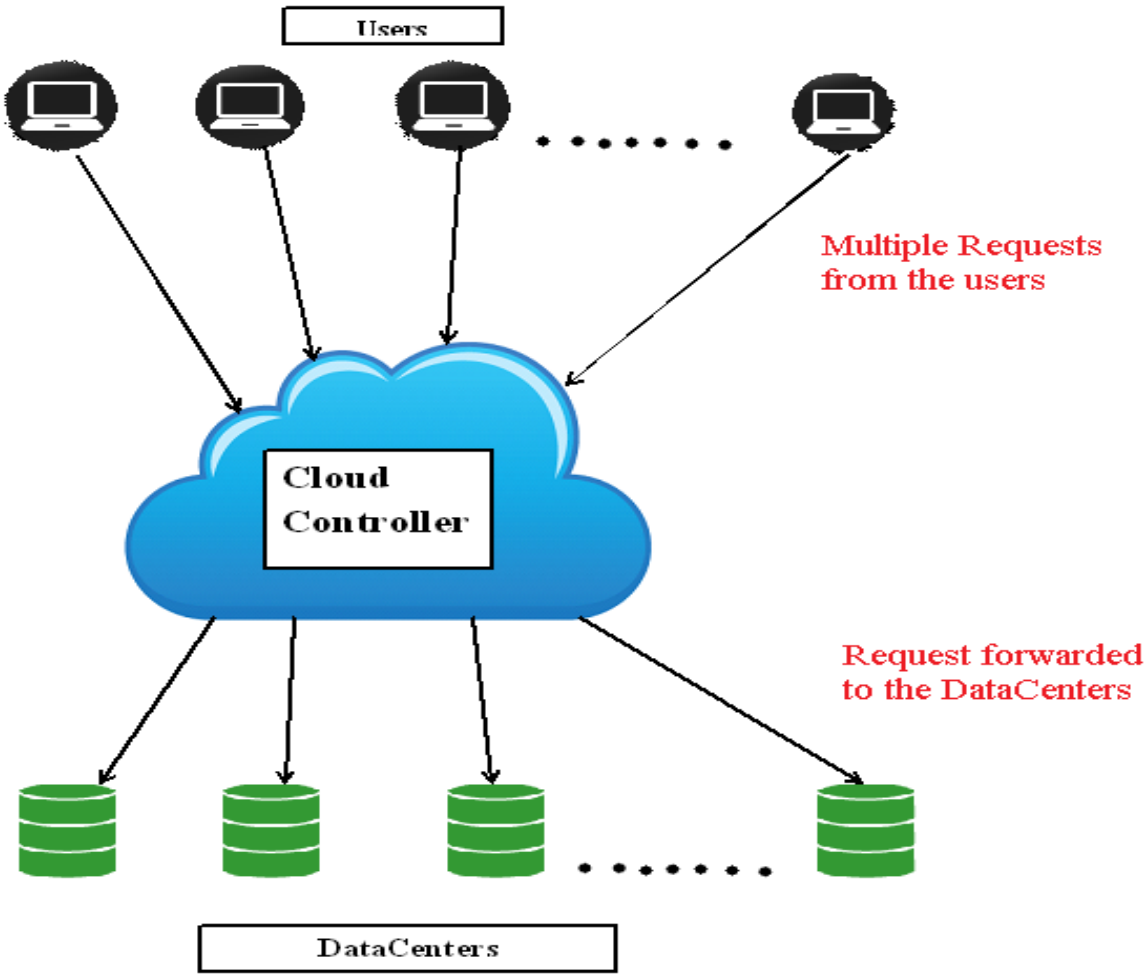
4. PROPOSED MODEL

4.1 Problem Statement

To devise such dynamic audit services that efficiently and accurately verifies the integrity of outsourced storages on cloud for improved security in cloud computing.

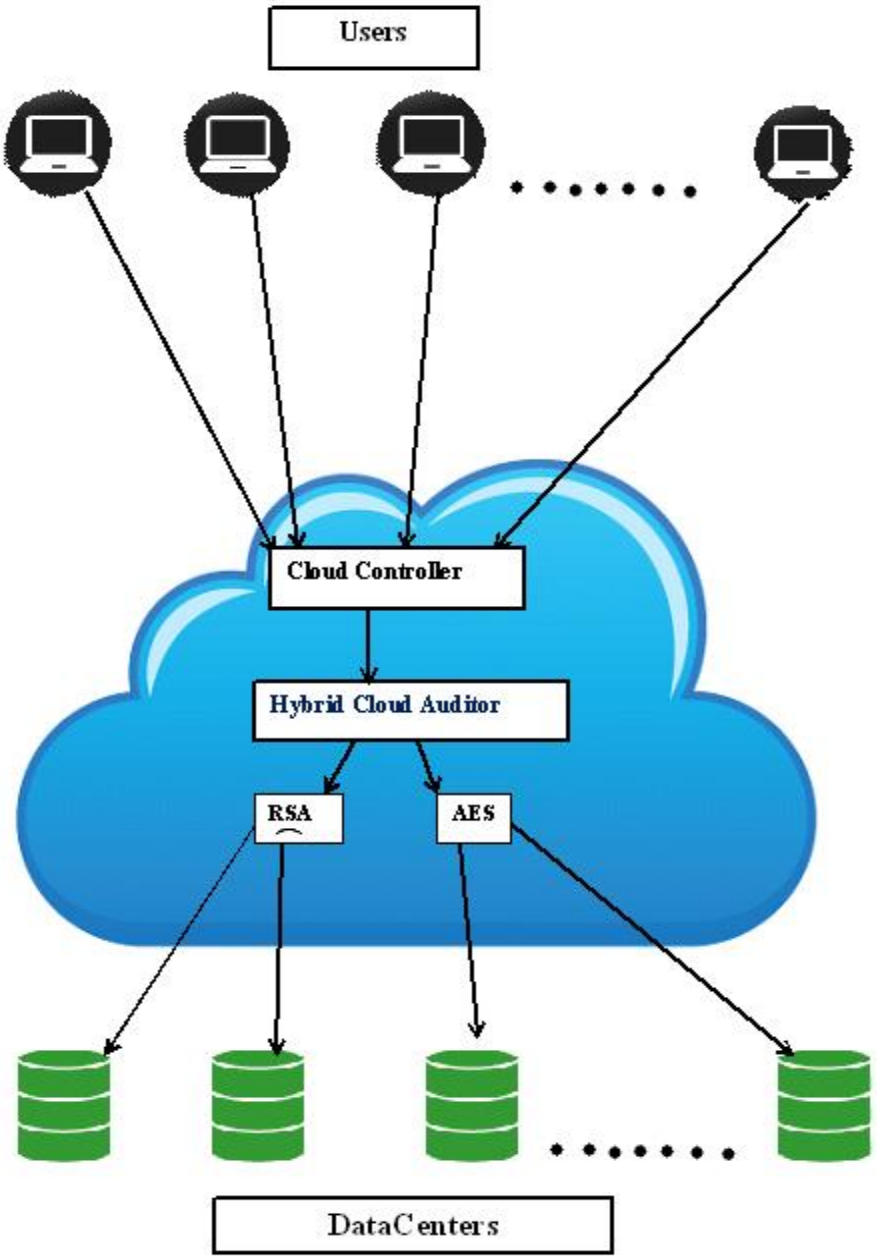
4.2 Proposed Model

Fig. 5 TRADITIONAL MODEL



The traditional model used by cloud is simply a cloud controller which uses a single level of security and takes multiple requests from the users and then forwards any one of them to the data center as shown in the diagram below.

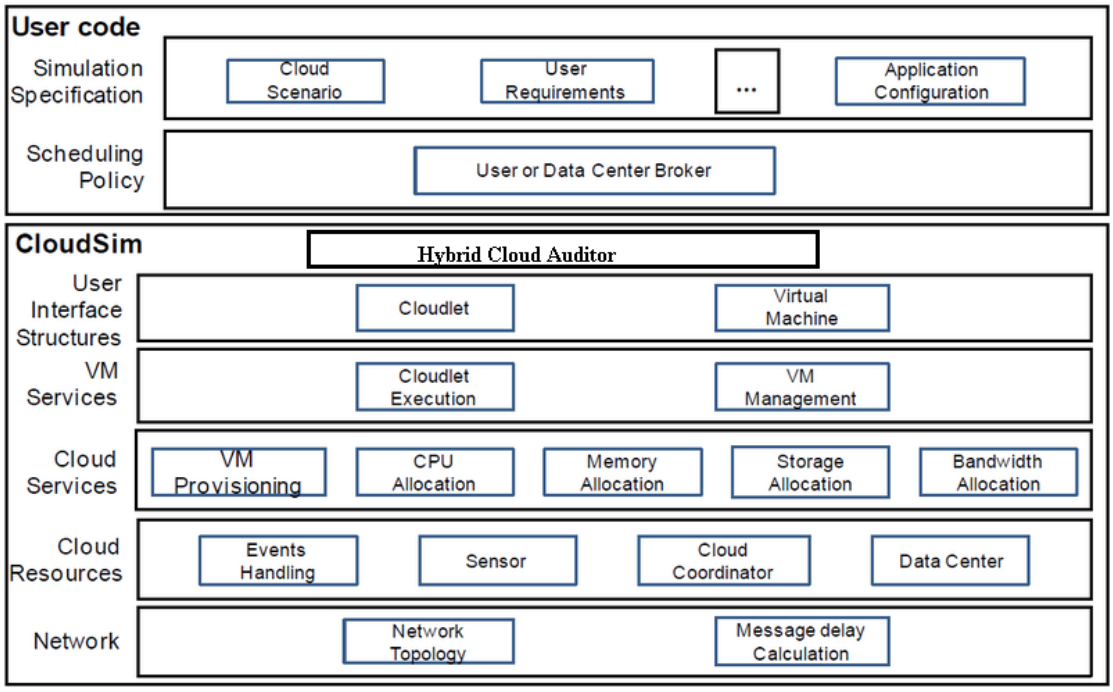
Fig. 6 PROPOSED MODEL USING HYBRID CLOUD AUDITOR



In this paper the model proposed for improving the security in the cloud computing uses Software as a Service (SaaS) which is a type of services offered by cloud where consumers are able to access software applications over the internet. The applications

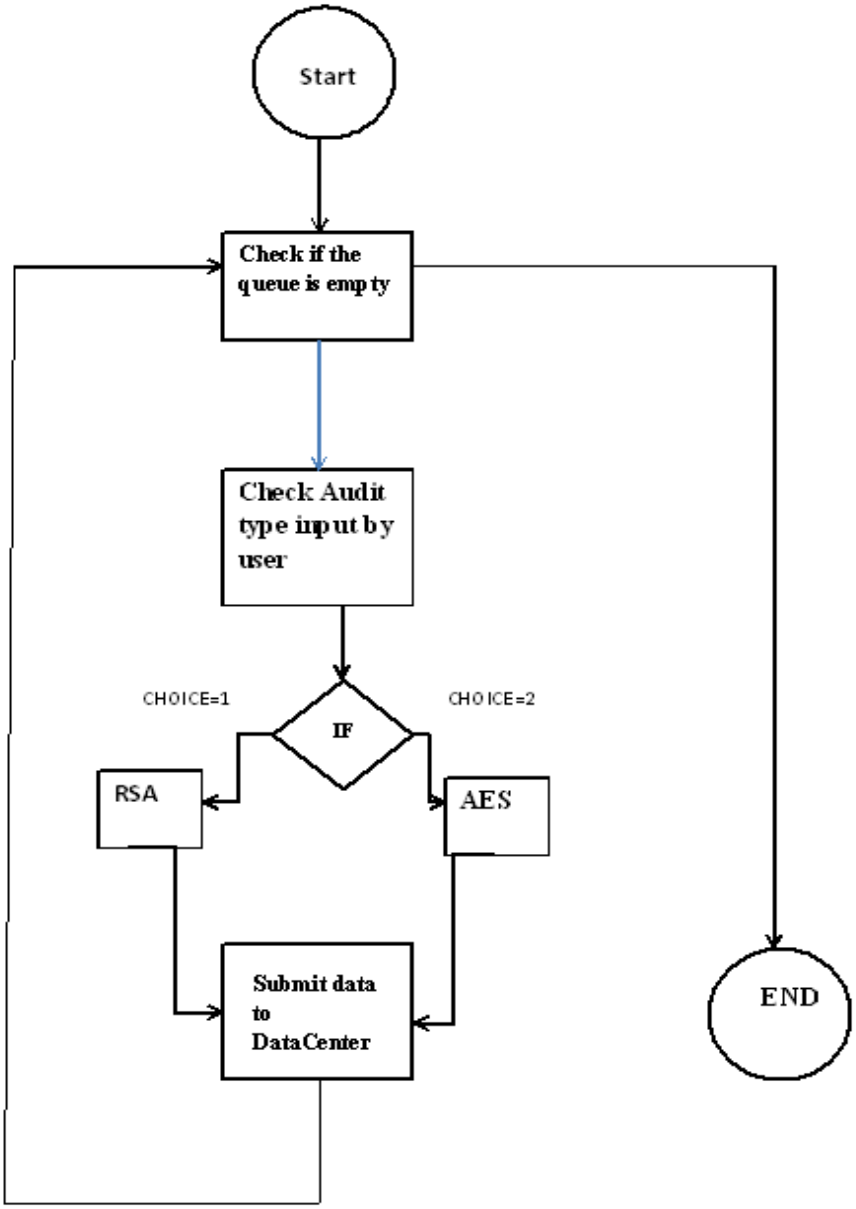
are hosted in “the cloud” and can be used for a wide range of tasks for both individuals and organisations. This proposed model revolves around the concept of auditing and therefore the concept of “Hybrid Cloud Auditor” has been introduced which uses two levels of security i.e. takes both private and public auditing requests, the reason it is called hybrid, and then forwards one of the multiple requests to the cloud controller which further sends the request to the data center. The inclusion of an auditor with two levels of security using two highly efficient cryptosystems RSA and AES makes the system more secure and reliable as well as makes it easier to run the model for multiple requests from the user. The use of two levels of security

Fig. 7 CloudSim Proposed model



The traditional CloudSim architecture has been modified with the addition of hybrid Cloud Auditor in between the user code and CloudSim. This hybrid auditor verifies the data sent by the user before submitting it to datacenters and therefore improves security.

Fig. 8 Flow Chart of the proposed model



4.3 Algorithm for the proposed model

Step 1: Start

Step 2: Check if the queue is empty. If full then stop else move to next step

Step 3: Check the Audit type input by the user

Step 4: If Choice = 1

Perform RSA encryption-decryption

Else for Choice = 2

Perform AES encryption-decryption

Step 5: Submit the data to DataCenter

Step 6: Goto Step2

Step 7: Stop

4.4 Tools used:

- NetBeans IDE 8.0.1
- CloudSim – The CLOUDS Lab

CHAPTER -5

5. SIMULATION AND RESULTS

5.1 CloudSimExample1.java

```
public class CloudSimExample1 {

	/** The cloudlet list. */
	private static List<Cloudlet> cloudletList;

	/** The vmlist. */
	private static List<Vm> vmlist;

	/**
	 * Creates main() to run this example.
	 *
	 * @param args the args
	 */
	public static void main(String[] args) {

		Log.printLine("Starting CloudSimExample1...");
	CloudSimExample1 obj=new CloudSimExample1();


		try {
			// First step: Initialize the CloudSim package. It should be called
			// before creating any entities.
			int num_user = 1; // number of cloud users
			Calendar calendar = Calendar.getInstance();
			boolean trace_flag = false; // mean trace events

			// Initialize the CloudSim library
			CloudSim.init(num_user, calendar, trace_flag);

			// Second step: Create Datacenters
			// Datacenters are the resource providers in CloudSim. We need at
			// list one of them to run a CloudSim simulation
			Datacenter datacenter0 = createDatacenter("Datacenter_0");

			// Third step: Create Broker
			DatacenterBroker broker = createBroker();
			int brokerId = broker.getId();

			// Fourth step: Create one virtual machine
			vmlist = new ArrayList<Vm>();

			// VM description
			int vmid = 0;
			int mips = 1000;
```

```
                        long size = 10000; // image size (MB)
                        int ram = 512; // vm memory (MB)
                        long bw = 1000;
                        int pesNumber = 1; // number of cpus
                        String vmm = "Xen"; // VMM name

                        // create VM
                        Vm vm = new Vm(vmid, brokerId, mips, pesNumber, ram, bw, size,
vmm, new CloudletSchedulerTimeShared());

                        // add the VM to the vmList
                        vmlist.add(vm);

                        // submit vm list to the broker
                        broker.submitVmList(vmlist);

                        // Fifth step: Create one Cloudlet
                        cloudletList = new ArrayList<Cloudlet>();

                        // Cloudlet properties
                        int id = 0;
                        long length = 400000;
                        long fileSize = 300;
                        long outputSize = 300;
                int type=1;
               UtilizationModel utilizationModel = new UtilizationModelFull();
                int audittype=0;
                Scanner ina=new Scanner(System.in);
                System.out.println("enter the type of auditing(audittype)");
                System.out.println("1 : Public  2: Private");
                audittype=ina.nextInt();
                //audittype=1;
                String text="abc";
                        Cloudlet cloudlet = new Cloudlet(id, length, pesNumber, fileSize,
outputSize, utilizationModel, utilizationModel, utilizationModel,audittype,text);
                        DataInputStream in=new DataInputStream(System.in);
                String teststring="aaa" ;
      cloudlet.setUserId(brokerId);
                        cloudlet.setVmId(vmid);

                        // add the cloudlet to the list
                        cloudletList.add(cloudlet);

                        // submit cloudlet list to the broker
                        broker.submitCloudletList(cloudletList);

                        // Sixth step: Starts the simulation
                        CloudSim.startSimulation();

                        CloudSim.stopSimulation();
```

```
                //Final step: Print results when simulation is over
                List<Cloudlet> newList = broker.getCloudletReceivedList();
                printCloudletList(newList);

                // Print the debt of each user to each datacenter
                datacenter0.printDebts();

                Log.printLine("CloudSimExample1 finished!");
            } catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
            }
        }


        /**
         * Creates the datacenter.
         *
         * @param name the name
         *
         * @return the datacenter
         */
        private static Datacenter createDatacenter(String name) {

            // Here are the steps needed to create a PowerDatacenter:
            // 1. We need to create a list to store
            // our machine
            List<Host> hostList = new ArrayList<Host>();

            // 2. A Machine contains one or more PEs or CPUs/Cores.
            // In this example, it will have only one core.
            List<Pe> peList = new ArrayList<Pe>();

            int mips = 1000;

            // 3. Create PEs and add these into a list.
            peList.add(new Pe(0, new PeProvisionerSimple(mips))); // need to store Pe
id and MIPS Rating

            // 4. Create Host with its id and list of PEs and add them to the list
            // of machines
            int hostId = 0;
            int ram = 2048; // host memory (MB)
            long storage = 1000000; // host storage
            int bw = 10000;

            hostList.add(
                new Host(
                    hostId,
                    new RamProvisionerSimple(ram),
                    new BwProvisionerSimple(bw),
```

```
                                storage,
                                peList,
                                new VmSchedulerTimeShared(peList)
                        )
                ); // This is our machine

                // 5. Create a DatacenterCharacteristics object that stores the
                // properties of a data center: architecture, OS, list of
                // Machines, allocation policy: time- or space-shared, time zone
                // and its price (G$/Pe time unit).
                String arch = "x86"; // system architecture
                String os = "Linux"; // operating system
                String vmm = "Xen";
                double time_zone = 10.0; // time zone this resource located
                double cost = 3.0; // the cost of using processing in this resource
                double costPerMem = 0.05; // the cost of using memory in this resource
                double costPerStorage = 0.001; // the cost of using storage in this
                                                                                // resource
                double costPerBw = 0.0; // the cost of using bw in this resource
                LinkedList<Storage> storageList = new LinkedList<Storage>(); // we are not
adding SAN

                // devices by now

                DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                                arch, os, vmm, hostList, time_zone, cost, costPerMem,
                                costPerStorage, costPerBw);

                // 6. Finally, we need to create a PowerDatacenter object.
                Datacenter datacenter = null;
                try {
                        datacenter = new Datacenter(name, characteristics, new
VmAllocationPolicySimple(hostList), storageList, 0);
                } catch (Exception e) {
                        e.printStackTrace();
                }

                return datacenter;
        }

        // We strongly encourage users to develop their own broker policies, to
        // submit vms and cloudlets according
        // to the specific rules of the simulated scenario
        /**
         * Creates the broker.
         *
         * @return the datacenter broker
         */
        private static DatacenterBroker createBroker() {
                DatacenterBroker broker = null;
                try {
```

```
                broker = new DatacenterBroker("Broker");
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
            return broker;
        }

        /**
         * Prints the Cloudlet objects.
         *
         * @param list list of Cloudlets
         */
        private static void printCloudletList(List<Cloudlet> list) {
            int size = list.size();
            Cloudlet cloudlet;

            String indent = "    ";
            Log.printLine();
            Log.printLine("========== OUTPUT ==========");
            Log.printLine("Cloudlet ID" + indent + "STATUS" + indent
                    + "Data center ID" + indent + "VM ID" + indent + "Time" +
indent
                    + "Start Time" + indent + "Finish Time"+ indent
+"Audittype"+ indent+ "TestString");

            DecimalFormat dft = new DecimalFormat("###.##");
            for (int i = 0; i < size; i++) {
                cloudlet = list.get(i);
                Log.print(indent + cloudlet.getCloudletId() + indent + indent);

                if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                    Log.print("SUCCESS");

                    Log.printLine(indent + indent + cloudlet.getResourceId()
                            + indent + indent + indent +indent+
cloudlet.getVmId()
                            + indent + indent
                            + dft.format(cloudlet.getActualCPUTime()) +
indent
                            + indent +
dft.format(cloudlet.getExecStartTime())
                            + indent + indent+indent
                            +
dft.format(cloudlet.getFinishTime())+indent+indent+indent
                            + dft.format(cloudlet.audittype));
                  //Log.print("XXX");
                    }
            }
        }}
```

5.2 OUTPUTS AND RESULTS

Comparative analysis of RSA and AES algorithm for different string lengths

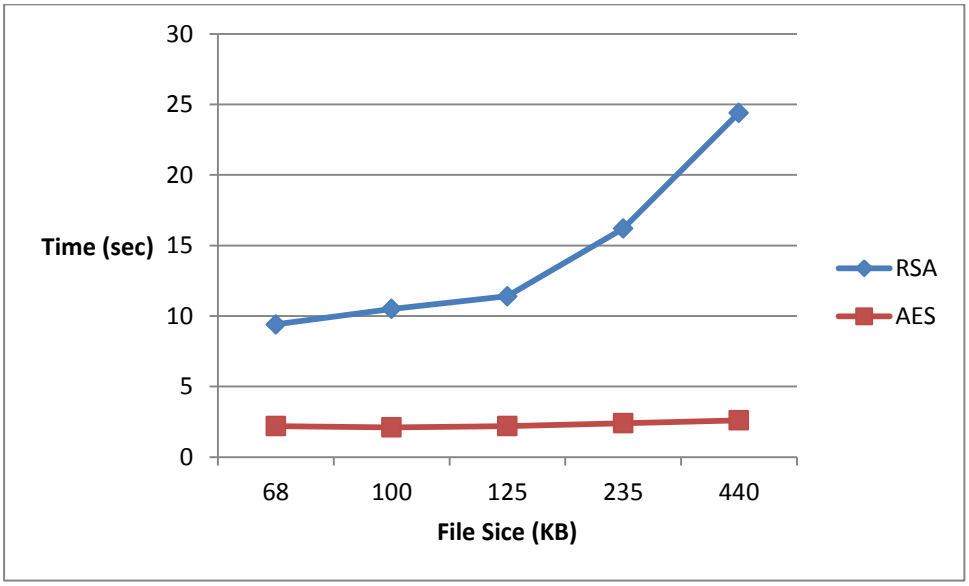
RSA is an asymmetric cipher. It is ideal for secure exchange of messages across an untrusted network, because the public key can be known by everyone - a message encrypted with the public key can only be decrypted by the private key. As such, if two parties know each other's public keys, they can exchange messages securely. This means that no secret information has to be transmitted - as long as *authenticity* and *integrity* are maintained you're safe. Thankfully, RSA provides a method of generating *signatures* on data, which help prove that it is authentic. Given a message signed by a private key, it is possible to verify that signature using the corresponding public key.

As a rule of thumb, you can only encrypt data as large as the RSA key length. So, if you've got a 4096-bit RSA key, you can only encrypt messages up to 4096 bits long. Not only that, but it's incredibly slow. RSA isn't designed as a full-speed data transport cipher.

AES is a symmetric block cipher, and is *incredibly* fast. The plaintext is split into chunks called blocks, and each block is encrypted in a chain. There are different ways of doing this, but a common one is called Cipher Block Chaining or CBC for short. This allows for theoretically infinite message sizes. However, symmetric ciphers like AES require a secret key to be exchanged first. Unlike RSA, the shared key must remain unknown to attackers, so you have to provide authenticity, integrity, and secrecy. That's difficult to do directly.

In the analysis done, RSA and AES algorithms were applied on files of different sizes one by one and the execution time was calculated. Based on the time calculated the graph was plotted and it was found out that AES was much more efficient and faster than RSA and the difference can be clearly seen in the graph where the execution time for a file pf same size was much less when AES was applied on it as compared to RSA.

Fig. 9 RSA vs AES EXECUTION TIME FOR DIFFERENT FILE SIZES



File Size (KB)	68	100	125	235	440
RSA Execution Time (sec)	9.4	10.5	11.4	16.2	24.4
AES Execution Time (sec)	2.2	2.1	2.2	2.4	2.6

Output for CloudSim class intially

Output for 1 datacentre and single request from the user for a particular audittype (public or private) i.e using two levels of security.

```
Run:
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time    Audittype
TestString
   0      SUCCESS      2          0      400      0          400          1
*****PowerDatacenter: Datacenter_0*****
User id          Debt
3                35.6
**********************************
CloudSimExample1 finished!
```

RSA encryption-decryption Output

One of the cryptosystem algorithm used is RSA for public-key cryptosystems. Below is the output for a string entered for encryption and then generated back by decryption using RSA.

Run:

Entered plain text: hello

Encrypting String: hello

String in Bytes: 104101108108111

Encrypted String in Bytes: 0-12795-98100-9183-52-2177-97-9036-2294-78681810170104-100-55-24-123-12-40-8637-698412-50-42-4587-816-31-64-509-27-108-92-77742872-924695961-764-912-6873-80123-1079311223-15-1-166487-89-65-12166-5140499280126214755 32-126-650-6-61-121-2935-108-124-271141962817991-76620-101-10028-116892-51-859-443910564-898512-6821-53-17122100115-54-50-112-7699938236-53-4762-29-24-761038937-2894-102120-38-819274780-1128-4125-21-44-63101-78-48-1028868-107231061203611748-31-61106-3965-66-57-3913133856-5364-100-611259085-10-54-2542-51-8-85120-215012511012460-11279754993-125-12011290125-32-64861146-12679-87-9-16-96115-142-29-3679-92-11687-7105-83111-9-11687-19-55102-2969-15-82-65114-41

Decrypted String in Bytes: 104101108108111

Decrypted String: hello

AES encryption-decryption output

Another cryptosystem algorithm used is AES for private-key cryptosystems. Below is the output for a string entered for encryption and then generated back by decryption using AES.

Run:

AES running

plain: hello how r you?

cipher: -70 27 70 -93 69 38 33 109 -66 92 -102 -80 108 -91 -61 39

decrypt: hello how r you?

Public Auditing

The notion of public auditability has been proposed in the context of ensuring remotely stored data integrity under different systems and security models. Public auditability allows an external party, in addition to the user himself, to verify the correctness of remotely stored data. Here the user is entering the choice for public or private auditing and based on choice 1 public cryptography is applied using RSA algorithm

========== OUTPUT ==========
Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time Audittype TestString
0 SUCCESS 2 0 400 0 400 1
*****PowerDatacenter: Datacenter_0*****
User id Debt
3 35.6
****
CloudSimExample1 finished!

Private Auditing

In the case of private auditing the user is fully responsible for verifying the integrity of their data. Here there is no involvement of the TPA. Here the user is entering the choice for public or private auditing and based on choice 2 private cryptography is applied using AES algorithm.

========== OUTPUT ==========

Cloudlet ID STATUS Data center ID VM ID Time Start Time Finish Time Audittype TestString

0	SUCCESS	2	0	400	0	400	2

*****PowerDatacenter: Datacenter_0*****

User id Debt

3 35.6

****

CloudSimExample1 finished!

Multiple requests to CloudSim

Implementing the auditing for multiple requests from the user involving public/private encryption-decryption in the multiple requests made by the user.

1. **For Public encryption-decryption**
   (a) **For 20 requests from the users**

========== OUTPUT ==========

Cloudlet ID	STATUS	Data center ID	VM ID	Time	Start Time	Finish Time	Audittype
0	SUCCESS	2	0	32768.08	0	32768.08	1
1	SUCCESS	2	0	32768.08	0	32768.08	1
2	SUCCESS	2	0	32768.08	0	32768.08	1
3	SUCCESS	2	0	32768.08	0	32768.08	1
4	SUCCESS	2	0	32768.08	0	32768.08	1
5	SUCCESS	2	0	32768.08	0	32768.08	1
6	SUCCESS	2	0	32768.08	0	32768.08	1
7	SUCCESS	2	0	32768.08	0	32768.08	1
8	SUCCESS	2	0	32768.08	0	32768.08	1
9	SUCCESS	2	0	32768.08	0	32768.08	1
10	SUCCESS	2	0	32768.08	0	32768.08	1
11	SUCCESS	2	0	32768.08	0	32768.08	1
12	SUCCESS	2	0	32768.08	0	32768.08	1
13	SUCCESS	2	0	32768.08	0	32768.08	1
14	SUCCESS	2	0	32768.08	0	32768.08	1
15	SUCCESS	2	0	32768.08	0	32768.08	1
16	SUCCESS	2	0	32768.08	0	32768.08	1
17	SUCCESS	2	0	32768.08	0	32768.08	1
18	SUCCESS	2	0	32768.08	0	32768.08	1
19	SUCCESS	2	0	32768.08	0	32768.08	1

*****PowerDatacenter: Datacenter_0*****

User id Debt

4 1025.6

****

*****PowerDatacenter: Datacenter_1*****
User id Debt
****
CloudSimExample6 finished!
Execution Time : 3 min 14 sec

2. **For 50 Requests from the users**

   No. of Requests : 50

   Execution Time : 6 min 22 sec

3. **For 80 Requests from the users**

   No. of Requests : 80

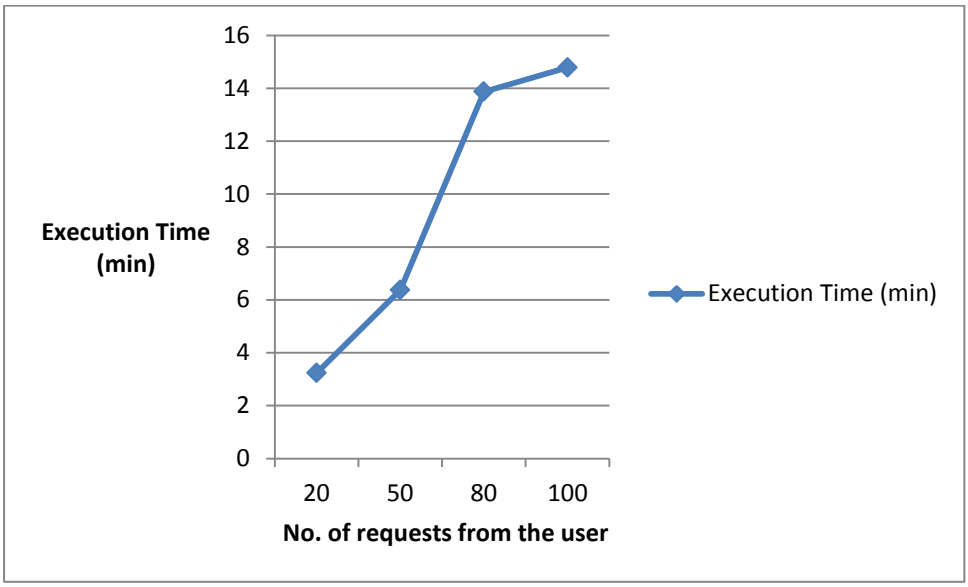
   Execution Time : 13 min 52 sec

4. **For 100 Requests from the users**

   No. of Requests : 100

   Execution Time : 14 min 47 sec

Fig. 10 Graph for public encryption-decryption for multiple requests by the user



No. of Requests	20	50	80	100
Execution Time (min)	3.23	6.37	13.87	14.78

5. For Private encryption-decryption

(a) For 20 requests from the users

```
========== OUTPUT ==========
Cloudlet ID  STATUS  Data center ID  VM ID  Time    Start Time    Finish Time    Audittype
   0       SUCCESS      2          0        32768.08      0          32768.08          2
   1       SUCCESS      2          0        32768.08      0          32768.08          2
   2       SUCCESS      2          0        32768.08      0          32768.08          2
   3       SUCCESS      2          0        32768.08      0          32768.08          2
   4       SUCCESS      2          0        32768.08      0          32768.08          2
   5       SUCCESS      2          0        32768.08      0          32768.08          2
   6       SUCCESS      2          0        32768.08      0          32768.08          2
   7       SUCCESS      2          0        32768.08      0          32768.08          2
   8       SUCCESS      2          0        32768.08      0          32768.08          2
   9       SUCCESS      2          0        32768.08      0          32768.08          2
   10      SUCCESS      2          0        32768.08      0          32768.08          2
   11      SUCCESS      2          0        32768.08      0          32768.08          2
   12      SUCCESS      2          0        32768.08      0          32768.08          2
   13      SUCCESS      2          0        32768.08      0          32768.08          2
   14      SUCCESS      2          0        32768.08      0          32768.08          2
   15      SUCCESS      2          0        32768.08      0          32768.08          2
   16      SUCCESS      2          0        32768.08      0          32768.08          2
   17      SUCCESS      2          0        32768.08      0          32768.08          2
   18      SUCCESS      2          0        32768.08      0          32768.08          2
   19      SUCCESS      2          0        32768.08      0          32768.08          2
*****PowerDatacenter: Datacenter_0*****
User id          Debt
4                1025.6
**********************************
*****PowerDatacenter: Datacenter_1*****
User id          Debt
**********************************
CloudSimExample6 finished!
Execution Time : 19 sec
```

(b) For 50 requests from the users

No. of Requests : 50

Execution Time : 25 sec

(c) For 80 requests from the users

No. of Requests : 80

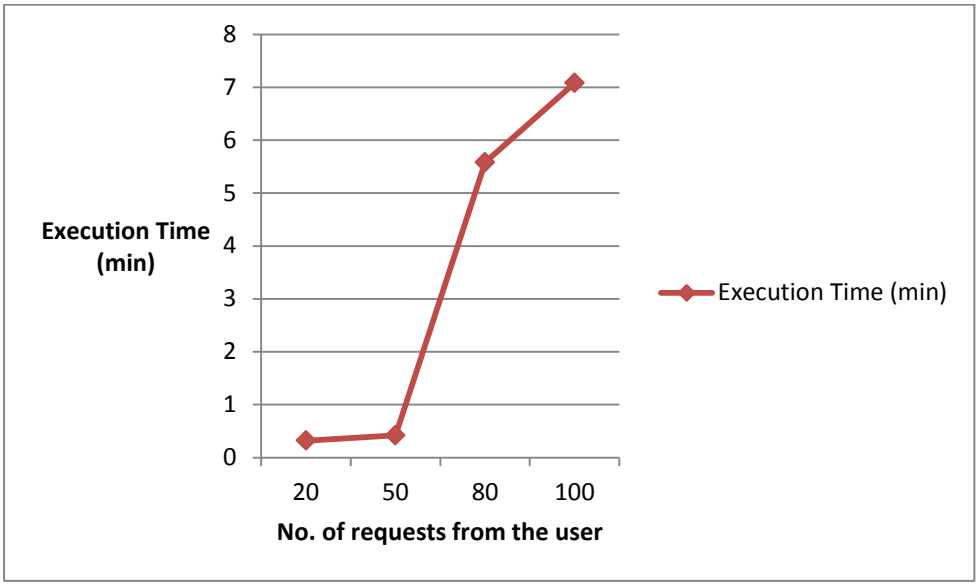
Execution Time : 5 min 35 sec

(d) For 100 requests from the users

No. of Requests : 100

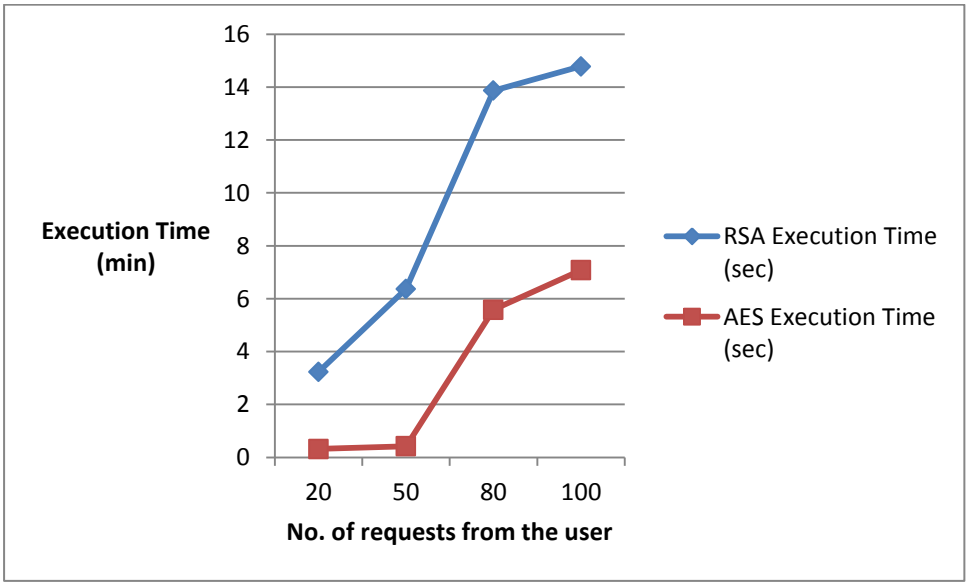
Execution Time : 7 min 5 sec

Fig. 11 Graph for private encryption-decryption for multiple requests by the user



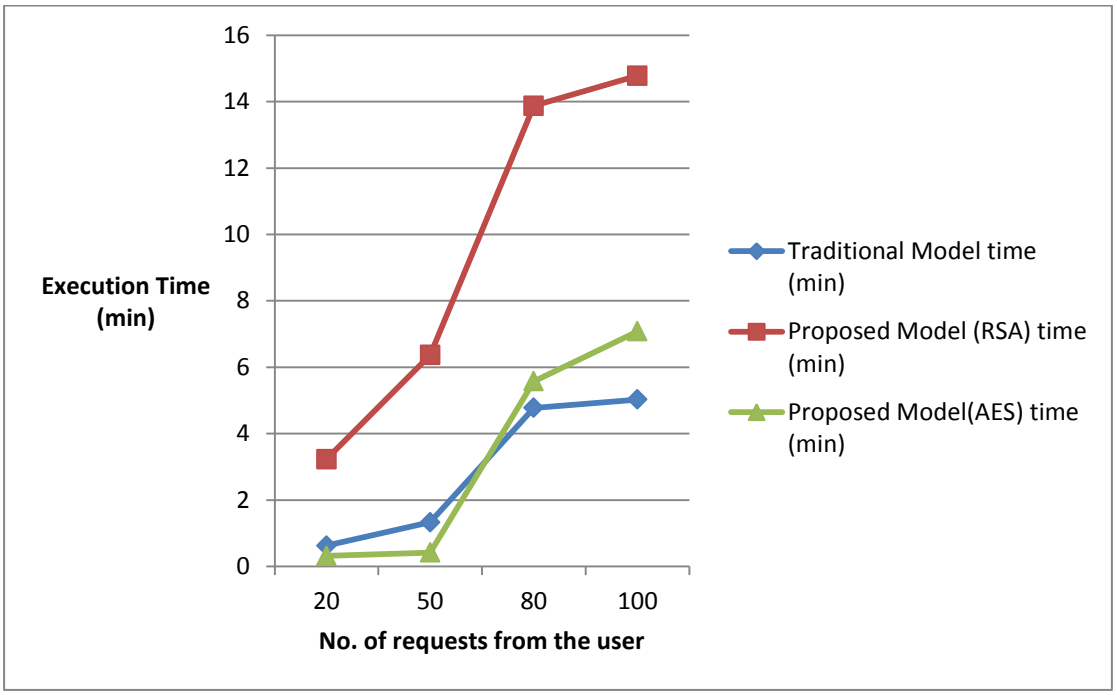
No. of requests	20	50	80	100
Execution Time (min)	0.32	0.42	5.58	7.08

Fig. 12 Combined graph for RSA & AES cryptosystems for multiple requests by the user



No. of requests	20	50	80	100
RSA Execution Time (sec)	3.23	6.37	13.87	14.78
AES Execution Time (sec)	0.32	0.42	5.58	7.08

Fig. 13 Graph on comparison of multiple requests in traditional model to that in the proposed using two levels of security



No. of requests	20	50	80	100
Traditional Model time (min)	0.63	1.33	4.78	5.03
Proposed Model (RSA) time (min)	3.23	6.37	13.87	14.78
Proposed Model(AES) time (min)	0.32	0.42	5.58	7.08

5.3 Conclusion and Future Work

Based on the results of the analysis we concluded that using two levels of security between the user and the auditor proves to be very useful for integrity verification in Cloud Computing. In our analysis and experimentation in the cloud environment using CloudSim, using a hybrid auditor in between the users and the Datacenters made the usage of cloud more reliable and secure and also led to an efficient working of the system. The use of two cryptography algorithms AES and RSA for public and private encryption-decryption and their analysis led to the conclusion that AES is much more efficient than RSA. Finally to test the system for more critical situations and real-time environment we analysed it for multiple requests from the user and for different type of auditing and based on the choice of user that particular auditing was applied and the system analysis was done for multiple requests and based on the execution time taken by the algorithms for multiple requests by the user graphs were plotted. In future, more parameters can be considered in the hybrid auditing and also multiple levels of security can be implemented.

CHAPTER – 6

6. REFERENCES

[1] Pearson, Siani. "Taking account of privacy when designing cloud computing services." *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*. IEEE Computer Society, 2009.

[2] Wang, Cong, et al. "Privacy-preserving public auditing for data storage security in cloud computing." *INFOCOM, 2010 Proceedings IEEE*.Ieee, 2010.

[3] Yang, Kan, and XiaohuaJia. "Data storage auditing service in cloud computing: challenges, methods and opportunities." *World Wide Web* 15.4 (2012): 409-428.

[4] Srinivas, J., K. VenkataSubba Reddy, and A. MOIZ Qyser. "Cloud Computing Basics." *International Journal of Advanced Research in Computer and Communication Engineering, 1 (5)* (2012).

[5] Yang, Kan, and XiaohuaJia. "An efficient and secure dynamic auditing protocol for data storage in cloud computing." *Parallel and Distributed Systems, IEEE Transactions on* 24.9 (2013): 1717-1726.

[6] Wang, Cong, et al. "Toward secure and dependable storage services in cloud computing." *Services Computing, IEEE Transactions on* 5.2 (2012): 220-232.

[7] Amazon.com, &ldquo,Amazon Web Services (AWS),&rdquo, http://aws. amazon.com, 2009.

[8] Marshal, ShingareVidya. "Secure Audit Service by Using TPA for Data Integrity in Cloud System." *International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN*: 2278-3075.

[9] Google website, http://www.google.com.

[10] http://cybersecurity.mit.edu/2012/11/auditing-the-cloud-platform/

[11] http://www.webopedia.com/TERM/C/cloud_computing.html

[12] http://cloudsim-setup.blogspot.in/2012/01/cloudsim-installation-with-netbeans_31.html