

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

Cloud computing is a service delivery model whereby shared resources such as hardware, software, platforms, and information are provided to consumers electronically as a utility over an internet (Wang *et al*, 2009). Several trends are opening up the era of cloud computing. These trends include ever cheaper and more powerful processors, together with the mainstream computing architectures such as Software-as-a-Service (SaaS), Platform-as-a Service (PaaS) and Infrastructure-as-a-Service (IaaS). These trends are transforming data centers into pools of computing service on a huge scale.

The increasing network bandwidth and reliable, yet flexible network connections make it even possible that users can now subscribe for high quality services from data and software that reside solely on remote data centers. Moving data into the cloud offers great convenience to users since they do not have to care about the complexities of direct hardware management. Examples of such well known services include Amazon Simple Storage Service (S3), and Amazon Elastic Compute Cloud (EC2).

While these internet-based online services do provide huge amounts of storage space and customizable computing resources, these computing platform shifts, however, are eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are at the mercy of their cloud service providers (CSP) for the availability and integrity of their data (Wang *et al*, 2009).

This research focuses on sensitive data generated from organizations globally such as IT industries, banks, private and corporate organizations, and higher institutions. Academic data generated from a university for example, which are the life wire of such organization and other sensitive records such as students and staff records are being generated from different departments and units to be stored and managed internally. Due to some known risks posed to these data such as internal risks like fire and liquid hazards, conflicts of interest, political intent, and financial baits, this research proposed outsourcing of these data to the cloud where it will be devoid of these risks.

Successful data outsourcing to the cloud requires some facilities and logistics that are to be provided which include: Internet facilities with required network equipments and technologies, acquisition of bandwidth from an Internet Service Provider (ISP) of its choice. Logistics include organizational policies to determine which Cloud Service Provider (CSP) to outsource to. Moreover, there has to be a service level agreement between the CSP and the Organization to be able to determine the quality of service that the CSP will provide for the organization. This service level agreement contains the type of services the provider renders to its client and the amount of money the clients pay for the services with legal backing.

Outsourcing of organizational data to the cloud should not be done without being cautious of both the internal and external security threats to the outsourced data. An example of external security threat considered in this research is man in the middle intercepting the data during transmission. For mitigation of this threat, the research hashed the data into blocks using adapted Merkle Hash Tree (MHT) (Qian et al, 2008), and encrypting the MHT root hash using improved RC6 (IRC6) cryptosystem before outsourcing to the cloud. A broad range of internal threats to data integrity still exist for outsourced data to the cloud. Examples of these internal threats include; services failure or server failure and data loss incidents. These threats occur from time to time and are worth noting. Again, since users may not retain a local copy of outsourced data, there exist various incentives for a CSP to behave unfaithfully toward the cloud users regarding the status of their outsourced data. For example, to increase the profit margin by reducing cost, it is possible for CSP to discard rarely accessed data without being detected in a timely fashion. Similarly, CSP may even attempt to hide data loss incidents so as to maintain a reputation. Therefore, there is need for regular remote auditing of data outsourced to the cloud to ensure data integrity and availability. This serves as a check to internal cloud data storage threats (Cong et al, 2012).

A Merkle Hash Tree (MHT) is a well-studied authentication structure, which is intended to efficiently and securely prove that a set of elements are undamaged and unaltered using homomorphic tokens. It is constructed as a binary tree where the leaves in the MHT are the hashes of authentic data values (Qian et al, 2008).

IRC6 cryptosystem is an improvement on RC6 which was developed in the course of this research to guard against crypto-analytical attack. This is achieved by doubling its security at little or no computational cost. RC6 is an improvement on RC5, and RC5 was an improvement

on RC4. IRC6 is designed to meet the requirements of increased security and better performance. IRC6 makes use of data dependent rotations. Another good feature of IRC6 is the use of four working registers instead of two. RC5 is a fast block cipher, it acts on 128-bit blocks using two 64-bit working registers. IRC6 modified its design to use four 32-bit registers rather than two 64-bit registers. This has the advantage of performing two rotations per round rather than the one found in a half-round of RC5. The improved cryptosystem (IRC6), is used in this research to secure the MHT root hash before data outsourcing to the cloud.

For the purpose of achieving the aim of this study, this research has developed an enhanced hybrid auditing model using MHT and IRC6 cryptosystem that will enable on-demand data correctness verification. The verification of cloud storage correctness is conducted without explicit knowledge of the whole data files on the cloud. The data stored in the cloud may not only be accessed but also be frequently updated by the data owners. The updates include insertion, deletion, modification and appending. These updates are dynamic operations that need to be integrated into the cloud storage correctness assurance (Ren & Wang, 2012).

1.2 Statement of the Problems

From the viewpoint of data security/integrity, which has always been an important aspect of quality of service, cloud computing certainly poses new challenges for a number of reasons:

- i. Many organizations are yet to adopt and enjoy the rich advantages of cloud data storage management capabilities due to fear of losing their data integrity.
- ii. Formal Remote Data Auditing (RDA) models' rootkey which is the main audit parameter, has been left unsecured.
- iii. Direct application of traditional or symmetric cryptosystem for data auditing is not adequate due to the users' loss of control of data outsourced to the cloud. There is need for additional techniques to verify the correctness of data storage in the cloud without explicit knowledge of the whole data.
- iv. Data stored in the cloud may be frequently updated. Hence, the assurance of storage correctness under dynamic data update is of paramount importance. However, this dynamic feature also makes traditional integrity assurance techniques ineffective and this requires new solutions.

- v. Cloud computing deployment is composed of many data centers running in a simultaneous, cooperated and distributed manner. Clients' data are stored in multiple physical locations of these distributed data centers to further reduce the data integrity threats. Therefore, outsourced data should be in blocks to support these distributed protocols for various data centers for effective remote data integrity check.
- vi. The practice of employing the services of the Third Party Auditor (TPA) for periodic remote data integrity check is faced with vulnerabilities which include interception of the original data main auditing tokens, hacking of TPA server or data compromise by TPA itself.
- vii. Aside the vulnerabilities associated with adopting TPA services, hiring and maintaining a TPA is rather costly.

1.3 Aim and Objectives of the Study

The main aim of this research is to develop a hybrid model for dynamic remote data auditing model on cloud computing. The specific objectives of the study are to:

- i. present the design of an Enhanced Model for Dynamic remote Data Auditing;
- ii. develop an enhanced hybrid system to support dynamic remote data auditing and data dynamic operations in the cloud by maintaining data integrity and availability even if users modify, delete, insert or update their data files in the cloud;
- iii. build and apply an adversary data authentication model to evaluate the effectiveness of the system;
- iv. compare performance of the new system with the existing system.

1.4 Significance of the Study

Broader knowledge of cyber attack mitigation and required techniques to checkmate cyber activities against remote data stored on the cloud is established through this research. This model helps to checkmate the wide spread of cyber-attacks in Internet environments tending to attack and compromise data over a cloud network. The work will potentially restore the confidence that organizations, both large and small, have on cloud services they engaged and use as most data stored on the cloud can be critical to the enterprise. It will also lead to improved economy as

organization and individuals use the cloud services to maximize their profits by utilizing more secured cloud services.

Development of an improved RC6 (IRC6) cryptosystem in this research also has great significance to improved security of the main auditing parameter with respect to existing system, as organizations focused more on profits making while using the cloud with less risk. This research also helps organizations to adopt data management principle that is devoid of internal data insecurity such as internal data threats.

The research develops an enhanced hybrid auditing scheme that adopts effective and flexible distributed data preprocessing scheme with explicit dynamic data support such as insertion, modification, deletion and appending; to ensure the correctness of users' file in the cloud. It relies on MHT and IRC6 encryption techniques in the file distribution preparation to guarantee data integrity and dependability. This hybrid technique will drastically reduce the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification blocks of data, this scheme will achieve the storage correctness assurance during RDA, as well as data error localization. It will also eliminate the vulnerabilities that are posed to the outsourced data by bypassing the TPA services.

The developed IRC6 in this research was implemented to double the security of the main auditing key with little or no cost. Encryption module, decryption module and key generation module were properly evaluated against their time lags and it was discovered that their time lags were very negligible which shows a good significant economic and security improvement over the previous models.

It was equally observed that its cipher text is not transparent during crypto-analysis thereby survives any crypto-analytical attack.

Organizations and individuals that outsource data to the cloud and employs this new enhance hybrid dynamic remote data auditing model for periodic data integrity check achieves great economic importance over their business with little or no security challenges.

The main significant of the new auditing model lie on auditing outsourced big-data which is a great task with great economic importance as it take few Auxiliary Authentication Information (AAI) or audit path to achieve effective audit results when compare to other replica base models.

1.5 Scope of the study

This research is based on cloud data security. It uses two techniques which are MHT and IRC6 to remotely audit the outsourced data on cloud. RC6 cryptosystem is improved fundamentally to yield IRC6 cryptosystem which doubles the level of data security. The developed IRC6 and adapted MHT authentication techniques are used to produce a hybrid model which has the capability of carrying out dynamic remote data auditing on cloud. The developed IRC6 was used to secure the security limitation in existing RDA models.

The implementation is done on a Java platform. The research adopts both real and adversary authentication modes as test cases during testing and validation.

1.6 Limitations of the study

The cost (in terms of transportation and access honorarium and time) of acquiring data for analysis from different universities and other organizations during feasibility study was rather high. Implementation of this hybrid remote dynamic data auditing system requires extra expertise experience and steady power availability which was a challenge. In creating Java applets, it has to be written in Java Language (because MHT is built on Java) which is more difficult but must be done for possible integration with IRC6.

1.7 Definition of terms

Amazon Elastic Compute Cloud (EC2): Amazon's EC2 is a cloud computing service that allows users to deploy and run their applications on rented virtual computers. Users can boot what are called Amazon machine images and create an instance, also known as a virtual machine, and pay for the amount of computing power they need by the hour.

Amazon Simple Storage Service (S3): Amazon's S3 is a cloud storage service that provides scalable, unlimited online archiving and backup for Amazon web service users.

Service Provider: This is a company or organization that provides a public or private cloud service.

Byzantine failures: Are defined as arbitrary deviations of a process from its assumed behavior based on the algorithm it is supposed to be running and the inputs it receives. Such failures can occur, e.g., due to a software bug, a (transitional or permanent) hardware malfunction, or a malicious attack.

Cloud computing: This refers to a model of network computing, where a program or application runs on a connected server or servers rather than on a local computing device such as PC, tablet

or smartphone. Like the traditional client-server model or older mainframe computing, a user connects with a server to perform task. The difference with cloud computing is that the computing process may run on one or many connected computers at the same time, utilizing the concept of virtualization.

Cloud Database: This is a database accessible to clients from the cloud and delivered to users on demand via the internet from a cloud database provider's servers. Also referred to as Database-as-a-Service (DDaaS), cloud databases can use cloud computing to achieve optimized scaling, high availability, multi-tenancy and effective resource allocation.

Cloud provider: A company that provides cloud-based platform, infrastructure, application, or storage services to other organizations and/or individuals, usually for a fee.

Cloud Storage: This is a service that allows customers to save data by transferring it over the internet or another network, to an offside storage system maintained by a third party.

Data Center: This is a facility built for the purpose of housing cloud-based data resources such as servers and other service-based equipment. Many cloud-based companies own and operate their own data centres which house the data stored for consumers and ensure the on-going availability of their cloud.

Data Integrity: This refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, and is a critical aspect of the design, implementation and usage of any system which stores, processes and retrieves data.

DO: Data owner that outsources his data to cloud and employs the services of the TPA for constant auditing.

Homomorphic encryption: Is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext.

Infrastructure as a service (IaaS): It is a pay-per-use service where a cloud computing provider offers storage space, software and network equipment as consumable resources. IaaS offerings include Amazon EC2, GoGrid and the Rackspace cloud.

Platform as a service (PaaS): This is a cloud computing model through which a computing platform is delivered to users via the web. PaaS is often used for the development, deployment and hosting of applications. Its offerings include Microsoft Azure, Force.com and google App Engine.

RDA: Remote Data Auditing, this refers to a means of verifying the correctness of data stored in remote cloud server.

Software as a service (SaaS): This is a software distribution model that provides applications to customers via the internet. The most commonly used form of cloud computing, SaaS continues to grow as web service and service-oriented architectures advance. The top sources of SaaS are Netsuite, Adobe and salesforce.com.

Service-level agreements (SLAs): This is a contractual agreement by which a service provider defines the level of service, responsibility, priorities, and guarantees regarding availability, performance, and other aspects of the service.

TPA: Third Party Auditor that constantly audits the remote outsourced data base on some level of agreements with the DO.

CHAPTER TWO

LITERATURE REVIEW

2.1 Theoretical Review

A large number of Information and Communication Technology (ICT) industries are migrating to cloud computing as an emerging area of Information Technology (IT) with the intent of using it to make their businesses more competitive and render more efficient services to their customers.

Before the emergence of cloud computing, there were three options available for acquiring more computational power to meet increasing demands. One option was to order for a more efficient server from one of the big server manufacturers such as HP, IBM, Dell, etc., install it in already established data center for the organization or in rented space reserved for it. The responsibility for acquisition, installation, and maintenance are entirely on the organization (Dave, 2012). The second option was to lease some of the needed equipment from a leasing company. The organization still had the responsibility of installing and configuring it in their data center or a provisioned space meant for it just as if they had purchased them. The third option was to rent a server from a Managed Service Provider (MSP) such as Savvis, Rackspace, Terremark, etc. The MSP would allocate a server from their own internal stock, deploy it for the organization in their data center, and then grant the organization access to operate it. The line between the service provider responsibilities and the organization's responsibilities is flexible. MSPs offered a variety of services as backup and "remote hands" being two likely supports in addition to simple server rental services and supports (Dave, 2012).

Each of these options or models has advantages and disadvantages. If the organization acquired the equipment, they naturally owned it through the full devaluation cycle of four to five years. That might have been the cheapest option, but it is the least flexible. They gained some long-term flexibility when they leased the equipment, but they paid more in trade and are still faced with all the operational costs. Again, to lease an equipment for less than a 3-month time prospect, or with a year span is not ideal, otherwise the overhead cost of receiving, configuring, and then returning the equipment would be too high. Finally, if the organization rents the server, they could get something with determined monthly terms, but will be facing some limitations in

the hardware choices and could not deploy it in their own data center. The advantage is that they will be able to have it up and running in a very good short time (Dave, 2012).

These models form a variety to be considered thus: on one aspect, there is a “buy-and-own-everything model”; on another aspect, there is a “own-nothing model”. In between the two models, there is a wide variety of models where some things are owned and others not. Figure 2.1 shows the conceptual model for cloud computing in which these four models are depicted.

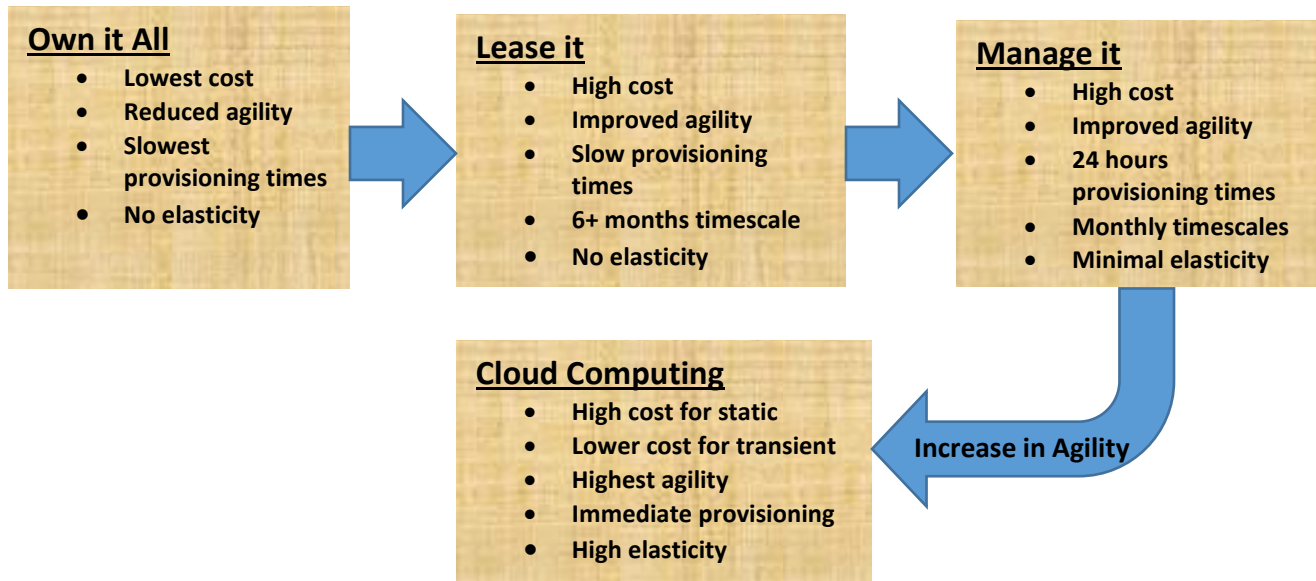


Figure 2.1: Conceptual model for cloud computing (Dave, 2012).

2.1.1 The birth of Cloud Computing

Cloud Computing can be described as a refinement of the own-nothing model, with smaller periods for resource rental, and greater flexibility for the customer as a result.

To put it straight, cloud computing implies delivery of scalable IT resources over the internet, as against the old tradition of hosting and using these resources locally within an individual or organizational network such as a university network. These shared pool of resources provided by cloud computing include computer processing power, data storage facilities and networks as cloud infrastructure, applications and services that will run on the infrastructure. When these cloud IT infrastructures and services are deployed over the network, an organization can purchase these resources over the network on the basis of as and when needed in order to avoid

the capital costs of software and hardware. Cloud computing helps to adjust IT capacity easily and quickly so as to accommodate changes in demand. The models of cloud computing grants access to cloud information and computer resources from anywhere that a network connection is available. Some other features that contribute to the emergence of cloud computing include ubiquitous networks, maturing standards, the rise of hardware and software virtualization, and the push to make IT costs variable and transparent (Educause, 2009).

According to The National Institute of Standards and Technology (NIST), cloud computing is a model for enabling convenient, 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. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models (Peter *et al*, 2011).

2.1.2 The five essential characteristics of cloud computing according to NIST (Peter *et al*, 2011) are:

- i. On-demand self-service: Resources are made available to customers in full automation without the interaction or involvement of the service provider. The ultimate goal is a resource to be available “instantly,” whenever a customer requires it.
- ii. Broad network access: Huge cloud computing resources and services that are available over the network can be accessed by different customers or end-user terminal systems such as laptops and desktop computers as well as other customers on mobile terminal systems such as phones and tablets.
- iii. Resource pooling: The cloud computing service providers are required to design the cloud computing physical infrastructures in order to have all the resources located in one or more common pools. This implies that the customers do not know the exact location of their resources in the midst of cloud computing pool. In practice, most providers offer some high-level location choices, such as a geographical region or data center.
- iv. Rapid elasticity: Users should be able to quickly allocate and release resources as required by applications. Ideally, users could request many resources immediately but

some resources such as new hardware still takes time to be received and configured then leading to scale the services as soon as they are ready. Services can still be scaled or delivered in elastic requests with enough or large common pool of resources serving numerous customers. It is necessary to state at this point that elasticity request is a two-way thing which demands that applications need to be able to allocate new resources as well as release them when they no longer need them.

- v. Measured service: Customers using cloud computing resources should be billed based on a well-defined granularity scheme (hours/days vs. months/years) as they are consumed, using appropriate units for the resource (GHz for CPU, GB for memory, GB/TB for mass storage, and Mbps/Gbps or GB of transfer for network, etc.).

2.1.3 Three categories of cloud computing service models according to NIST:

This brief correlation below stands to give clearer understanding of the three cloud computing models.

Cloud infrastructure as a service is not useful itself unless it is made useful by using it to solve a particular problem. Consider the intercity transportation system in FCT Abuja, despite all the good roads and perfect road networks, they would not be useful if cars and trucks are not on them to move people and goods from one place to another.

In comparison, the infrastructure can be seen as the roads while the means of transportation such as cars and trucks are the platform that sits on top of the infrastructure for the purpose of transportation. Goods and people being transported can be considered as the software and information in the technical form (Ben, 2011). Figure 2.2 shows in some details, the pictorial representation of the three models of cloud computing services.

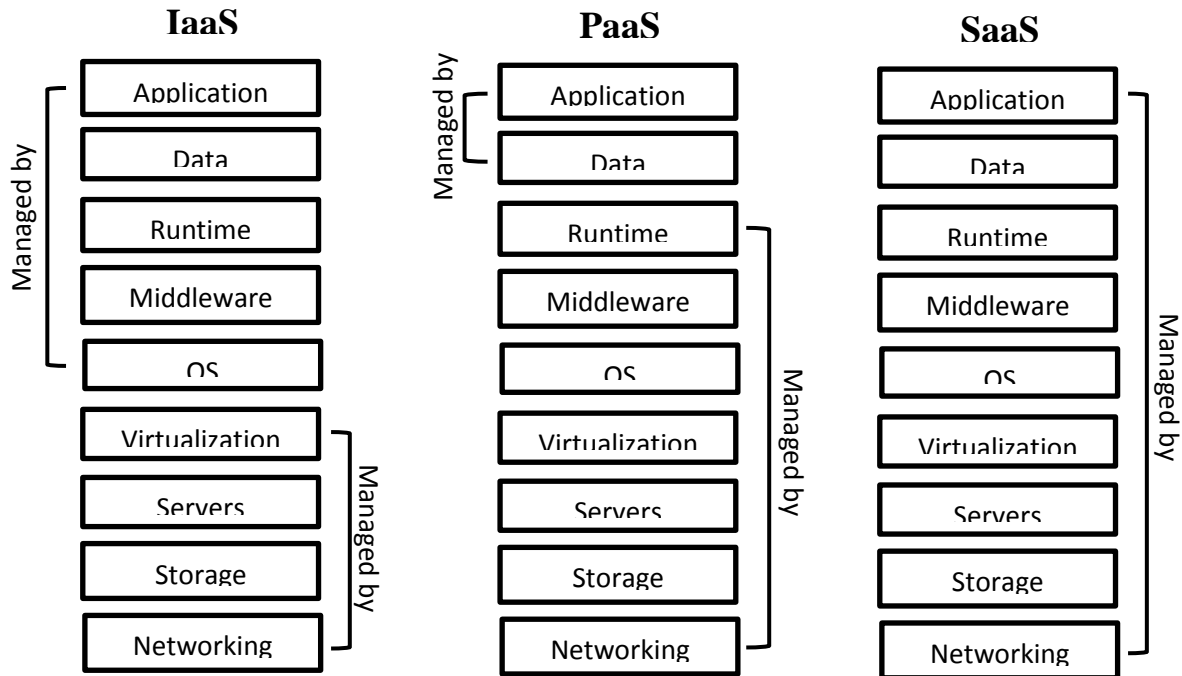


Figure 2.2: Cloud Computing Model (Ben, 2011).

i. Software as a Service

Software as a Service (SaaS): This is software deployed over the internet to support cloud computing customers. Cloud computing providers grant their customers' access to licensed applications as a service resident on their cloud either through subscription, on demand, on instantaneous model known as pay-as-you-go model or free of charge if there are other ways to generate income from cloud users such as advertisement or sales of user list.

Recent reports that envisage the ongoing double digit growth according Ben (2011) have shown that SaaS is a rapid growing market in cloud computing field. This indicates that SaaS is fast gaining ground in every organization which makes it vital to establish at this point the suitability of SaaS to the consumers of cloud computing technology.

Characteristics of SaaS

The uniqueness of SaaS cannot be over emphasized, and it is good to ensure that software sold as SaaS complies with generally known classifications of Cloud Computing. The following are the characteristics of SaaS (Peter *et al*, 2011):

- i. Internet access to commercial software.
- ii. Central management of software from a central location.

- iii. There is “one to many” model for delivery of cloud software model.
- iv. Software upgrades and patches are centrally handled by cloud providers and the users.
- v. Application Programming Interfaces (APIs) permit for incorporation of different pieces of software on cloud computing.

SaaS Suitability

SaaS as a model of Cloud Computing is a fast emergent means of delivering technology. For this fact, there is need to consider the type of application that will be suitable to move to cloud by individual organizations that have decided to move to cloud. The following applications will be good for initial move to SaaS:

- i. “Vanilla” is a good SaaS that provides solutions that are largely homogeneous. A typical example of vanilla is an email services where on different occasions competitors use the same software precisely because this fundamental technology is a requirement for doing business, but does not itself confer a competitive advantage.
- ii. Software that requires important interaction between the organization and its external collaborators or customers. For example, email newsletter campaign software.
- iii. Software that requires important internet or mobile access. An example is mobile sales management software.
- iv. Software that is only required on short term basis for usage. Collaboration software for a specific project is an example.
- v. Software with significant increase in demand, tax or billing software used once a month is an example.

Salesforce Customer Relationship Management (CRM) product introduced SaaS to the business world due to its wide acceptance. As one of the earliest entrants it is not surprising that CRM is the most popular SaaS application area according Peter *et al* (2011), however e-mail, financial management, customer service and expense management have also become good through SaaS.

Unsuitability of SaaS

For the fact that SaaS is a very resourceful tool, does not imply that it does not have its down side, listed below are some examples where SaaS may not be suitable which include:

- i. Software that requires real-time data process at a very high speed of processing.
- ii. Software that is bound by legislative policies or other regulations which permits only internal domicile of data.

- iii. Software where an internal existing data or solution satisfies all the organization's demands.

The best known aspect of Cloud Computing is the SaaS, but Cloud Developers and organizations are bringing in Platform as a Service, such that the simplicity of SaaS with the power of IaaS, helps to drive Cloud Computing to a great extent.

ii. Platform as a Service

Platform as a Service (PaaS) is very advantageous for SaaS in Cloud Computing. PaaS is a computing platform which helps in easily and timely creation of network or internet software without the complexity of buying and maintaining the software and its infrastructure unto which it is built.

PaaS is similar to SaaS, the only difference between them is while SaaS is software delivered over the internet, PaaS is a platform for the creation of software, delivered over the internet (Peter *et al*, 2011).

Characteristics of PaaS

These are basic characteristics of PaaS (Peter *et al*, 2011):

- i. It is the required platform for developing, testing, deployment, hosting and maintaining applications in one integrated development environment. It contains all the different services needed to fulfill the application development process.
- ii. Internet based user interface creation tools help to create, modify, test and deploy different User Interface scenarios.
- iii. It has multi-tenant architecture which helps multiple concurrent users to adapt and utilize the same development application.
- iv. It has built in scalability for software deployment such as load balancing and failover.
- v. It has common standard for integration of databases and internet services.
- vi. It provides support for development team collaboration. Some of the supports for PaaS solutions are communication and project planning tools.
- vii. It has tools to manage billing and subscription.

PaaS is similar to Infrastructure as a Service (IaaS), but it is being differentiated by the addition of value added services which come in two forms:

- 1. It has a collaborative platform that enhances software development which is concentrated on workflow management irrespective of the application's data source. An example of

this approach is Heroku which is a PaaS that utilizes the Ruby on Rails development language (Peter *et al*, 2011).

2. It has a platform that allows for the creation of software utilizing copyright data from an application. This type of PaaS can be seen as a method to create applications with a common data form or type. An example of this type of platform is the Force.com PaaS from Salesforce.com which is used to develop applications that work with the Salesforce.com CRM.

PaaS Suitability

- i. PaaS is suitable for multi-platform development environment which permits multiple developers to develop a project or allow for other external parties to interact with a project development process. It is very resourceful for those that have an existing data source such as sales information from a customer relationship management tool, and wish to develop an application to manage the data.
- ii. PaaS is also suitable where developers wish to automate testing and deployment services. The general acceptance of agile software development, a group of software development methodologies based on iterative and incremental development, will also increase the suitability of PaaS as it reduces the challenges with rapid development and iteration of software. Examples of PaaS include Google App Engine, Microsoft Azure Services, and the Force.com platform according to Peter *et al* (2011).

Unsuitability of PaaS

PaaS is expected to become the widely used means of software development. Its capability to automate application development processes, use of predefined components and building blocks and automatic deployment of software over the internet make it very valuable, but there are certain areas where it may not be suitable which include:

- i. Situation where the portability of an application is highly required in terms of where it is hosted.
- ii. Situation where software development process will be influenced by proprietary languages.
- iii. Situations where there will be hindrance for transfer of service to another provider due to a proprietary language such as vendor lock-in (Peter *et al*, 2011).

- iv. Situations where customization of application's software and hardware is required to improve its performance.

iii. Infrastructure as a Service

Infrastructure as a Service (IaaS) is a means of delivering Cloud Computing infrastructures such as servers, storage, network and operating systems as an on-demand service. Instead of buying all cloud computing infrastructures such as servers, software, datacenter space and other network equipment, organizations or customers purchase these cloud computing infrastructural resources as a completely outsourced service on demand (Ben, 2011).

Characteristics of IaaS

As the case is with SaaS and PaaS, IaaS is equally an emerging and rapidly developing field in IT world. IaaS is generally agreed to conform to the following (Ben, 2011):

- i. Its resources are distributed as a service
- ii. It permits for self-motivated scaling
- iii. It has a flexible cost, utility pricing model
- iv. It usually comprises several users on a single piece of hardware

There are many IaaS providers at present in Cloud Computing world such as Amazon Web Services and Rackspace.

The difference between PaaS and IaaS is becoming more indistinct as cloud computing providers or merchants are introducing tools as part of IaaS that help with cloud deployment such as the ability to deploy multiple kinds of Clouds (Ben, 2011).

IaaS Suitability

The suitability of IaaS is what bring about some benefits of Cloud Computing. Conditions that are predominantly suitable for Cloud infrastructure include:

- i. Situation where demand is very unstable, that is whenever there are substantial increase and decrease in demand for cloud computing infrastructure.
- ii. Situation where there are new organizations without the enough money to acquire required infrastructure such as hardware.
- iii. Situation where there is rapid growth of a client or an organization with increase in hardware demand which may be difficult to achieve.
- iv. Situation where there is pressure on a client to reduce capital spending and to move to operating spending

- v. Situation where there is need for infrastructure to support a particular line of business, trial or temporary need for infrastructure.

Unsuitability IaaS

Scalability and quick availability have been the greatest advantages of IaaS but there are situations limitation poses a challenge. Such situations include (Ben, 2011):

- i. Where there is strict compliance to a regulation that makes it difficult for outsourcing of data storage and processing.
- ii. Where the maximum levels of performance are needed, and internal or dedicated hosted infrastructure has the capability to meet the client's needs

2.1.4 Type of Deployment Models

There are four different types of deployment models outlined in the NIST definition of cloud computing as shown in figure 2.3 which include:

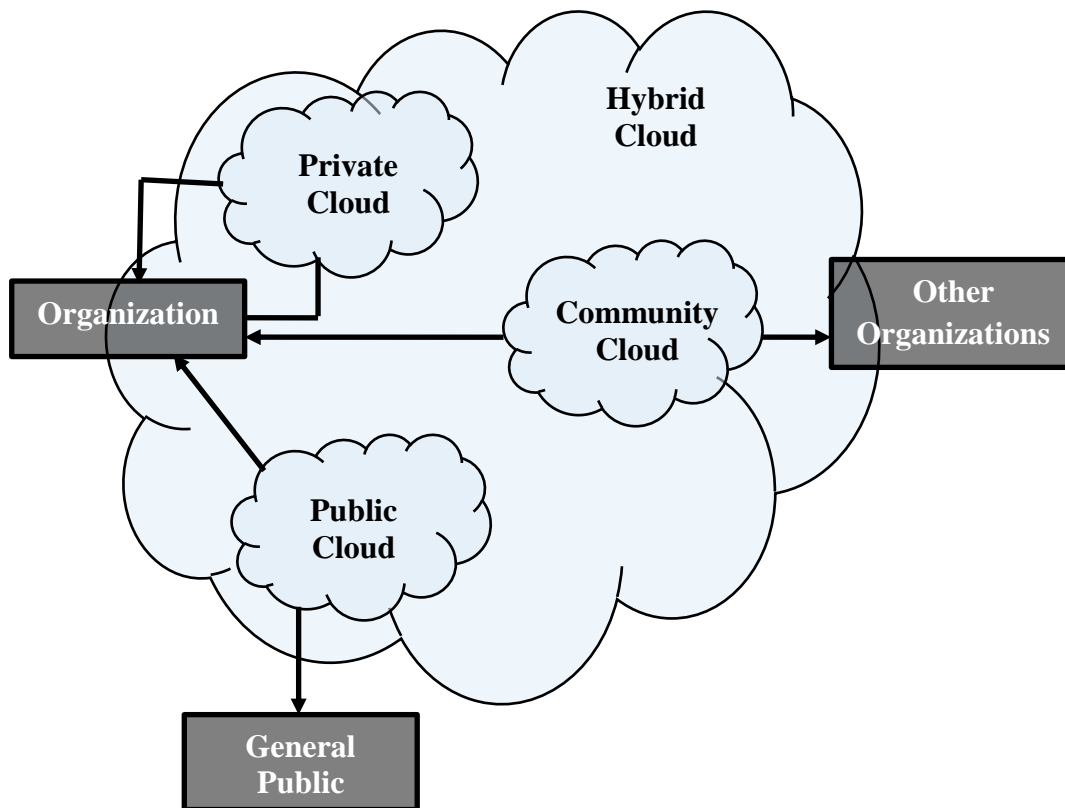


Figure 2.3: Cloud Deployment models (Peter et al, 2011)

2.1.4.1 Private cloud

Private clouds are types of cloud computing deployment model that are created for use by a single organization. Private clouds are created and operated by that organization as well, which did not depict the true value of cloud computing. Examples are private clouds that based on VMware or OpenStack, which are created by individual enterprises.

2.1.4.2 Community cloud

Community clouds are created to be used by a small set of known clients that have certain requirements in common such as performance, security, cost, etc. These clients are selected from same type of business or industry such as financial services, although this did not depict the real value of cloud computing. Examples are the NYSE Capital Markets Community Platform and Amazon's GovCloud.

2.1.4.3 Public cloud

Public clouds are owned and operated by large external providers and deliver service to all clients. Examples are Amazon Web Services EC2 and Google AppEngine.

2.1.4.4 Hybrid cloud

The combination of other cloud computing deployment models gave rise to hybrid deployment model. Hybrid model is built on a cloud environment that spans each of these various models. Each part of a hybrid cloud can be possessed and operated by a different entity with management software providing integration across the environment.

The NIST service and deployment models form a matrix as shown in the table 2.1 below.

Table 2.1: Cloud Computing service and deployment models matrix according NIST (Educause, 2009)

	SaaS	PaaS	IaaS
Private Cloud	Internal enterprise applications, e.g., cooperate emails, payroll, etc.	Apprenda, Stackota	VMware, Hyper-V, OpenStack, CloudStack
Community Cloud	Health regions in countries such as Canada, USA, Japan	NYSE Capital Markets Community Platform	NYSE Capital Markets Community Platform
Public Cloud	Salesforce.com,	Google AppEngine,	Amazon EC2,

	Quickbooks online, Office 360	Microsoft Azure, VMware CloudFoundry.com	Rackspace
Hybrid Cloud		Custom CloudFoundry	Custom, Rackspace

Table 2.1 shows hybrid clouds in the matrix, which are really a mixture of the several public, private, and community cloud types. Thus, it is possible to construct a motivating hybrid cloud environment from a combination of a private IaaS cloud based on VMware, a public IaaS cloud based on Amazon EC2, and a public PaaS cloud based on Google AppEngine (Dave, 2012). Some big organizations create a hybrid cloud environment of some kind, with several public and private clouds, and perhaps a community cloud forms the mixture. Some clouds will provide enhanced security; others will provide enhanced performance; still others will offer optimized pricing. Organizations will thereafter deploy application workloads within a suitable cloud to create an optimized outcome. Management software are equally deployed which helps to match application capabilities to the right clouds given optimization goals, risk and compliance restrictions (Dave, 2012).

2.1.5 Cloud Computing Architecture

Cloud architecture extends to the client where web browsers and/or software applications are used to access cloud applications. The majority of cloud computing infrastructure currently consists of reliable services delivered through data centres that are built on computer and storage virtualization technologies. The services are accessible anywhere in the world, with The Cloud appearing as a single point of access for all the computing needs of consumers. Cloud storage architecture is loosely coupled where metadata operations are centralized enabling the data nodes to scale into the hundreds, each independently delivering data to applications or users. The figure 2.4 below shows cloud computing architecture where clients/users access cloud systems using a web browser regardless of their location or what device they use (e.g. Desktops, Laptops, Servers, Tablets, mobile phones etc.).

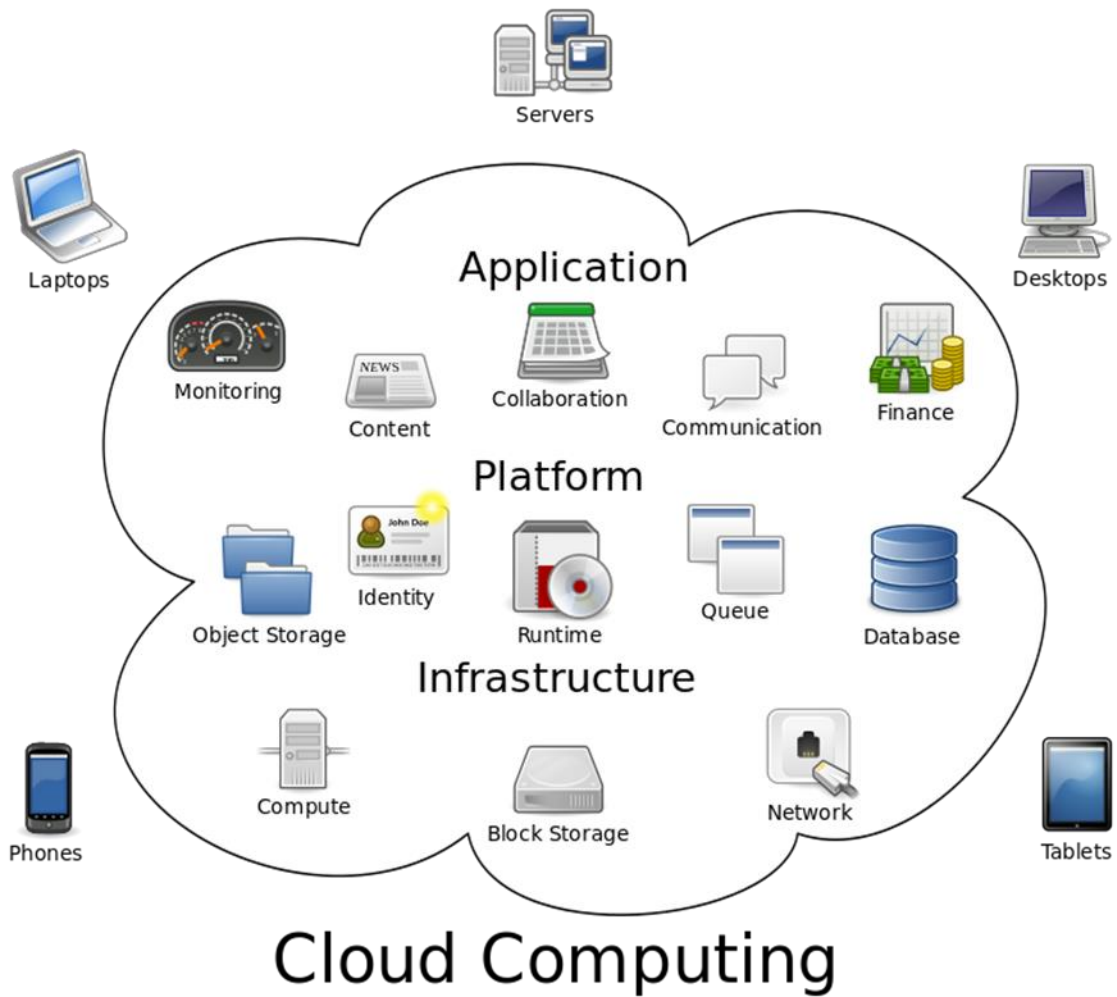


Figure 2.4: Cloud Computing Architecture (Sam, 2009)

2.1.6 Components of Cloud Computing Service

Components of cloud computing services are shown in figure 2.5, explanation on each of the components are given thereafter.

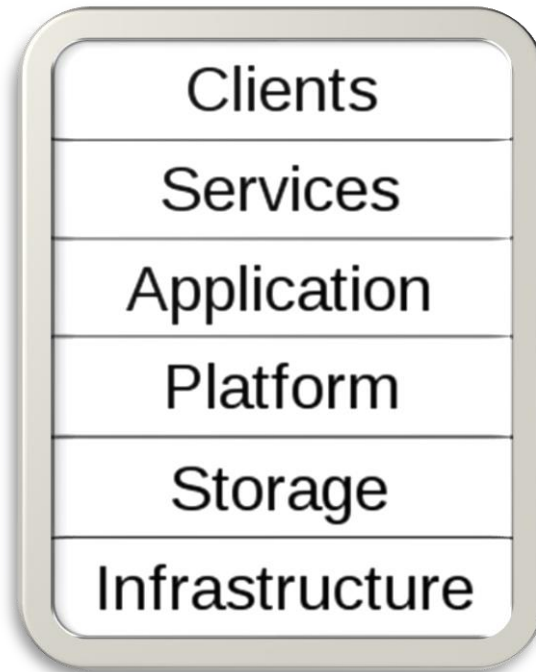


Figure 2.5: Cloud computing Components according to NIST Definition (Peter et al, 2011)

1. **Application:** A cloud application influences the Cloud model of software architecture, often eliminating the need to install and run the application on the customer's own computer, thus reducing software maintenance, ongoing operations, and support. For example:
 - i. Peer-to-peer/volunteer computing (Bittorrent, SETI@home, Skype);
 - ii. Web application (Facebook);
 - iii. Software as a service (Google Apps, Salesforce);
 - iv. Software plus services (Microsoft Online Services).

2. **Client:** A cloud client is computer hardware and/or computer software which relies on the Cloud for application delivery, or which is specifically designed for delivery of cloud services, and which is in either case essentially useless without a Cloud. For example:
 - i. Mobile (Android, iPhone, Windows Mobile);
 - ii. Thin client (CherryPal, Zonbug OS based systems);
 - iii. Thick client/Web browser (Google Chrome, Mozilla Firefox).

3. **Infrastructure:** Cloud infrastructure (e.g. Infrastructure as a service) is the delivery of computer infrastructure (typically a platform virtualization environment) as a service. For example:
 - i. Full virtualization (GoGrid, Skytap);
 - ii. Grid computing (Sun Grid);
 - iii. Management (RightScale);
 - iv. Paravirtualization (Amazon Elastic Compute Cloud).
4. **Platform:** A cloud platform (e.g. Platform as a service) (the delivery of a computing platform and/or solution stack as a service) facilitates deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers. For example:
 - i. Web application frameworks:
 - a. Python Django (Google App Engine);
 - b. Ruby on Rails (Heroku).
 - ii. Web hosting (Mosso);
 - iii. Proprietary (Azure, Force.com).
5. **Service:** A cloud service (e.g. Web Service) is "software system(s) designed to support interoperable machine-to-machine interaction over a network" which may be accessed by other cloud computing components, software (e.g. Software plus services) or end users directly. For example:
 - i. Identity (OAuth, OpenID);
 - ii. Integration (Amazon Simple Queue Service);
 - iii. Mapping (Google Maps, Yahoo! Maps);
 - iv. Payments (Amazon Flexible Payments Service, Google Checkout, PayPal);
 - v. Search (Alexa, Google Custom Search, Yahoo! BOSS);
 - vi. Others (Amazon Mechanical Turk).
6. **Storage:** Cloud storage is the delivery of data storage as a service (including database-like services), often billed on a utility computing basis (e.g. gigabyte per month). For example:
 - i. Database (Amazon SimpleDB, Google App Engine's BigTable datastore);
 - ii. Network attached storage (MobileMe iDisk component, Nirvanix CloudNAS);

- iii. Synchronization (Live Mesh Live Desktop component, MobileMe push functions);
- iv. Web service (Amazon Simple Storage Service, Nirvanix SDN).

2.1.7 Mobile Cloud Computing

The concept of mobile cloud computing implies running an application such as Google's Gmail for mobile version, on a remote resource server such as Google server, while mobile devices work like a client connecting to the remote server through high connection network such as 3G or 4G. Other examples are mobile version of twitter, mobile version of facebook and facebook's location awareness services, mobile weather widgets.

Recently, applications designed for mobile devices are becoming popular and numerous in different categories such as entertainment, health, games, business, social media, travel and news, mobile applications, download centers such as Apple's iTunes and Nokia OVI suite have made this popularity well known.

The major success achieved in this regard is the provision of computing tool by mobile computing devices where and when needed in spite of user movement thereby supporting location independence. This is one of the characteristics of a pervasive computing environment where the user is able to carry out his/her work continuously despite his/her movement.

Though, there are some problems associated with mobility such as low connectivity, finite energy and resource (Satyanarayanan, 1996). These problems of mobility limit execution of many applications that would have being of great help to users to create a good pervasive environment.

A research carried out by Siegele (2008) indicates that future computing will depend on real-time system response either by human or non-human sensor. Real-time mobile applications that depend on high level of responsiveness require computing resources as well (Siegele, 2008).

Location based social networking as a mobile application carries out its processing using various mobile devices sensor data, and extensive use of sensors on mobile devices are very expensive in terms of energy such as obtaining GPS rendering. This limitation hinders many provisions of better location based services to the user through its embedded sensors.

Again, other applications that require extensive processing or computational power such as speech synthesis, natural language processing, image processing for video games and other graphical analysis are difficult to develop and deploy on mobile devices due to the limitations in mobile device architecture and battery.

The limitations in mobile device architecture and battery are fundamental to mobility according to Satyanarayanan (1993) and require to be solved in order to achieve full benefit of mobile computing. These limitations have been addressed through Cloud Computing recently by different researchers. Cloud computing as earlier stated in this work is the aggregation of computing as a utility, where SaaS implies being delivered over the internet as services (Armbrust *et al*, 2009). The basic idea behind Cloud Computing is to offload computation to remote Cloud providers.

The concept of offloading data and extensive computation to the Cloud Computing addresses essential challenges that exist in mobile architecture by utilizing resources provided by cloud computing providers rather than hosting and executing mobile applications and other extensive computational works on mobile devices.

Those infrastructures where data storage and processing could occur outside the mobile devices is called mobile cloud. By exploiting the computing and storage capabilities of the mobile cloud, computer intensive applications can be executed on low resource mobile devices.

Recently, different mobile applications are connected to the cloud which include Apple iCloud, Google's Gmail for mobile and Google Goggles (Sasaki *et al*, 2012).

Current mobile applications are connected to the cloud where most of the extensive computations are carried out. The mobile devices serve as clients that connect to the remote cloud computing server providing extensive service. It has been noted that these mobile devices perform very well with high speed network connections. For the fact that high speed network connections are not always available in most developing countries, this poses another challenge in mobile computing which necessitates the creation of local cloud server for interconnection of local resources. Some other key factors facing mobile computing such as data access fees affordability, good response time and limited energy source are addressed with local interconnection of resources through local cloud server since short range communication consumes less energy, fastens connectivity and better availability (Satyanarayanan *et al*, 2009). An example of local cloud server is called Cloudlet. Mobile devices within this cloud

environment offload their workload to the local cloudlet(s). These cloudlets consist of several multi-core computers with high connecting power to the remote cloud servers. These cloudlets are positioned strategically while other mobile devices connect to them and serve as thin client (Satyanarayanan *et al*, 2009).

With recent technological enhancement in mobile smart phones and other mobile devices which improves their computational power, the future mobile cloud will be hybrid where users of the mobile cloud will serve as cloud resources, with the ability to connect to the remote cloud servers when there is good network connection, active access fees, available strong battery and good response time, else connect to the local cloudlets.

2.1.8 Explanation of the model on Merkle Hash Tree

A Merkle Hash Tree (MHT) is a well-studied authentication structure, which is intended to efficiently and securely prove that a set of elements are undamaged and unaltered. It is constructed as a binary tree where the leaves in the MHT are the hashes of authentic data values (Qian *et al*, 2008).

Merkle developed a Signature Scheme based on a binary tree of hashes in (Merkle, 1980). A typical example of a MHT is illustrated using figure 2.6. Each leaf node holds the hash of a data block. For instance, $H(1)$ holds the hash of the data block 1. Internal nodes hold the hash of the concatenated hashes of their children such as $H(1,2) = H(H(1) | H(2))$ where '|' indicates concatenation.

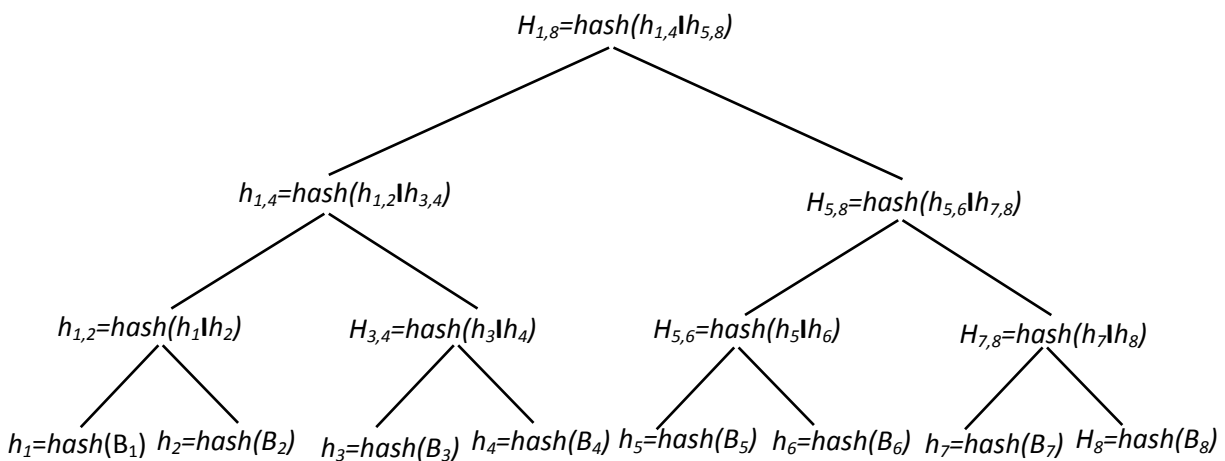


Figure 2.6: Merkle Hash Tree (Qian *et al*, 2008)

The existing scheme is based on the assumption that there is a safe or trusted way to share the root of the tree between the signer which is the data owner and the verifier which is the vulnerable TPA. Our research focused on this point where we encrypt the root hash for security purpose and eliminate the use of TPA which can be hacked. To verify the integrity of any data block, the whole tree of hashes does not need to be transmitted to the verifier.

A signer transmits the hashes of only those nodes which are involved in the authentication path of the data block under consideration. For example, if the receiver needs to verify the integrity of data block 2 then only $H(1)$, $H(3,4)$ and $H(5,8)$ need to be transferred to the receiver. The receiver can calculate the $H(2)$ from data block 2. $H(1,2)$ can then be calculated by using the received $H(1)$ and calculated $H(2)$. In the same way, $H(1,4)$ can be calculated and then $H(1,8)$.

The receiver then can compare the calculated $H(1,8)$ with the already shared $H'(1,8)$ and if both the hashes match, then the integrity of data block 2 is confirmed.

Another example, Suppose a user has obtained the root hash $h(1,8)$ from a trusted server and then retrieved data block 4 from an untrusted host. To verify the integrity of data block 4, the user needs the untrusted host to also send it the hashes $[h(3); h(1,2); h(5,8)]$, so that it can reconstruct the path leading from $H(4)$ to the root hash, as depicted in figure 2.7.

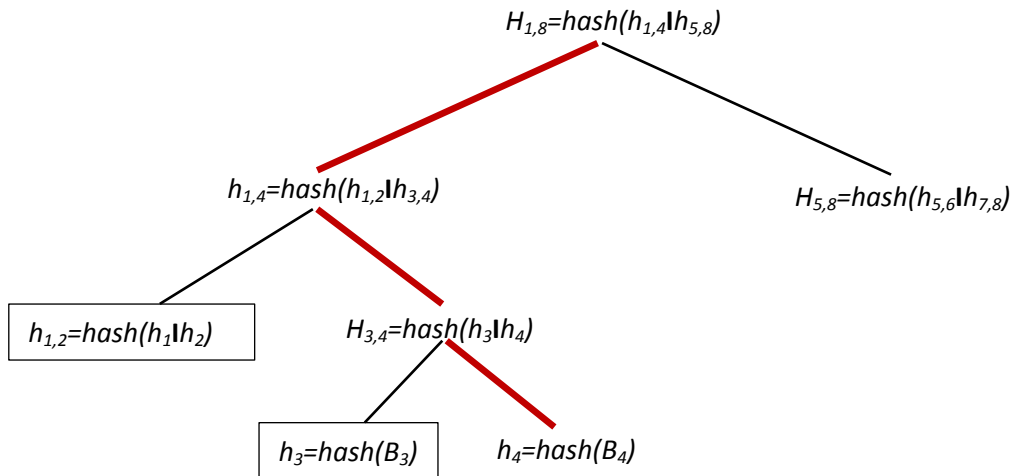


Figure 2.7: Authentication path for Data block 4 (Qian et al, 2008)

Hence, verifying that any given data block is in fact the authentic data block 4 only requires transmitting and computing 3 hashes, not the full tree.

Some important facts regarding Merkle's Signature Scheme are as follows:

- i. Security of this signature scheme depends on the security of the hash function.
- ii. Only one hash needs to be maintained/shared securely.
- iii. To authenticate any data block only $\log^2 n$ hashes need to be transferred, where n denotes total number of data blocks.
- iv. In case of integrity checking of a continuous range of blocks, even less than $\log_2 n$ hashes need to be transferred.

The selection of Merkel hash tree for our model has been mainly due to its data rendering capabilities on large datasets. The hashes are obtained at node level which helps in swift error detection. In case of server failures or data corruption, unlike existing models which recover entire file, only the effected nodes have to be recovered which will facilitate rapid file recovery on every iteration. The following features distinguish MHT from other techniques;

A. Consistency Proofs

Blocks grow with time as new blocks are been uploaded. However, they are only appended to record, that is, blocks may not be modified, deleted, or inserted in the middle. Once a block is uploaded and appended, its presence must be noted.

Suppose a user already has the root hash of the record after n blocks, and now the blocks have been extended to $m > n$ blocks. It can be shown to the user that the new record has only appended blocks to the old record, without the user needing to check the full Merkle trees. Consider the Merkle tree in figure 2.8 with $n = 6$ blocks.

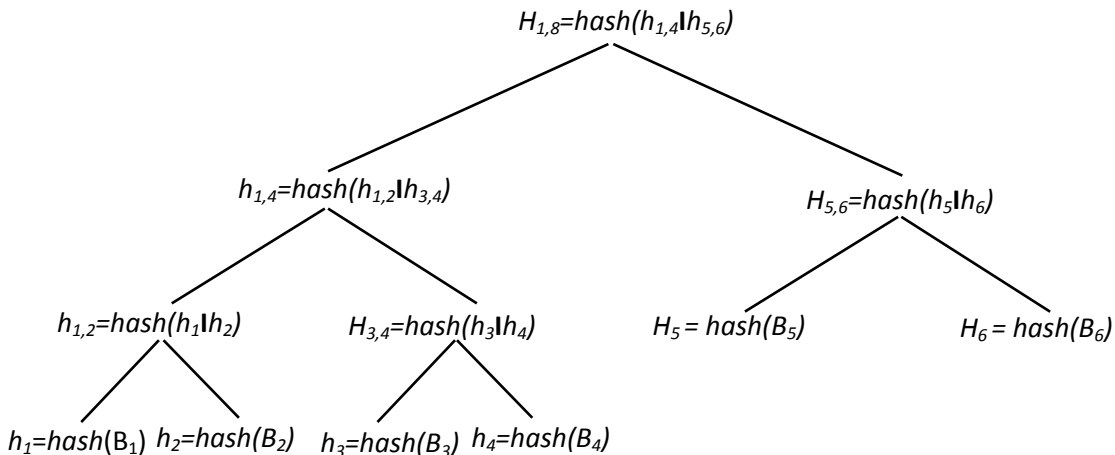


Figure 2.8: Merkle tree with 6 blocks (Qian et al, 2008)

To prove that the root hash $h(1,8)$ for the record with $m = 8$ elements strictly extends the record with $n = 6$ elements, it is enough to provide the user with the three hashes ($h(1,4)$; $h(5,6)$; $h(7,8)$). Using these hashes, the user can reconstruct both root hashes $h(1,6)$ and $h(1,8)$ and be assured that the second corresponds to a tree that extends the first. This sequence of hashes is called a consistency proof.

B. Design for Blockless and Stateless Verification

The simple way of realizing data integrity verification is to make the hashes of the original data blocks as the leaves in MHT, so the data integrity verification can be conducted easily. This method only requires the server to return only the challenged blocks for authentication. To improve on these method, this research on remote data checking adopts a blockless strategy for data integrity verification by asking the server only to return block tag of the requested block for verification instead of the whole blocks. We secure both the block tags and the original data blocks in the verification process. The R metadata is encrypted and stored both on server and at the client side for stateless verification operation.

C. Dynamic Data Operation with Integrity Assurance

It shows how the scheme can explicitly and efficiently handle fully dynamic data operations including data modification (M), data insertion (I) and data deletion (D) for cloud data storage. It assumes that the file and the signature have already been generated and properly stored at server. The root metadata R has been signed by the client and stored at the cloud server, so that anyone who has the client's public key (the root hash) can challenge the correctness of data storage. The following are the descriptions of the data dynamic operations:

1. Data Modification

We start from data modification, which is one of the most frequently used operations in cloud data storage. A basic data modification operation refers to the replacement of specified blocks with new ones.

Suppose the client wants to modify the i^{th} block of m_i to m'_i . Based on the new block m'_i , the client generates the corresponding signature $\sigma'_i = (H(m'_i))$. Then, he constructs an update request message “update = (M, m_i, m'_i, σ'_i)” and sends to the server, where M denotes the modification operation. Upon receiving the request, the server runs $\text{ExecUpdate}(F, \Phi, \text{update})$. Specifically, the server

- (i) replaces the block m_i with m'_i and outputs F' ;
- (ii) replaces the σ_i with σ'_i and outputs Φ' ;
- (iii) replaces $H(m_i)$ with $H(m'_i)$ in the Merkle hash tree construction and generates the new root R' .

Figure 2.9 shows the diagrammatic description of update operation.

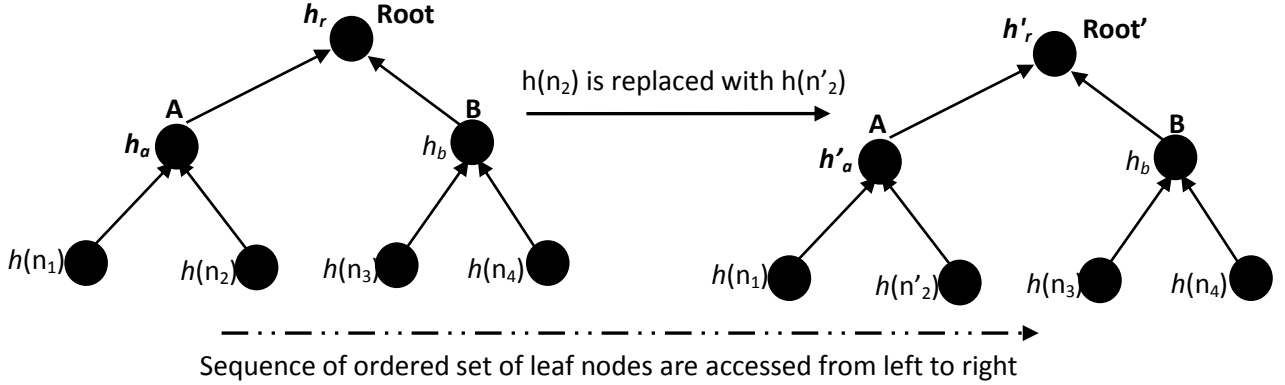


Figure 2.9: Example of MHT update under block modification operation. Here, n_i and n'_i are used to denote $H(m_i)$ and $H(m'_i)$, respectively (Qian et al, 2008).

2. Data Insertion

Compared to data modification, which does not change the logic structure of client's data file, another general form of data operation, data insertion, refers to inserting new blocks after some specified positions in the data file F .

Suppose the client wants to insert block m^* after the i^{th} block of m_i . The protocol procedures are similar to the data modification case but in this case, m'_i can be seen as m^* . Based on m^* the client generates the corresponding signature $\sigma^* = (H(m^*))$. Then, he constructs an update request message "update = (I, m_i, m^*, σ^*)" and sends to the server, where I denotes the insertion operation. Upon receiving the request, the server runs $\text{ExecUpdate}(F, \Phi, \text{update})$. Specifically, the server

- (i) stores m^* and adds a leaf $h(H(m^*))$ "after" leaf $h(H(m^i))$ in the Merkle hash tree and outputs F' ;
- (ii) adds the σ^* into the signature set and outputs Φ' ;
- (iii) generates the new root R' based on the updated Merkle hash tree.

An example of block insertion is illustrated in Figure 2.10, to insert $h(H(m^*))$ after leaf node $h(H(m_2))$, only node $h(H(m^*))$ and an internal node C are added to the original tree, where $h_c = h(h(H(m_2))||h(H(m^*)))$, after receiving the proof for insert operation from server.

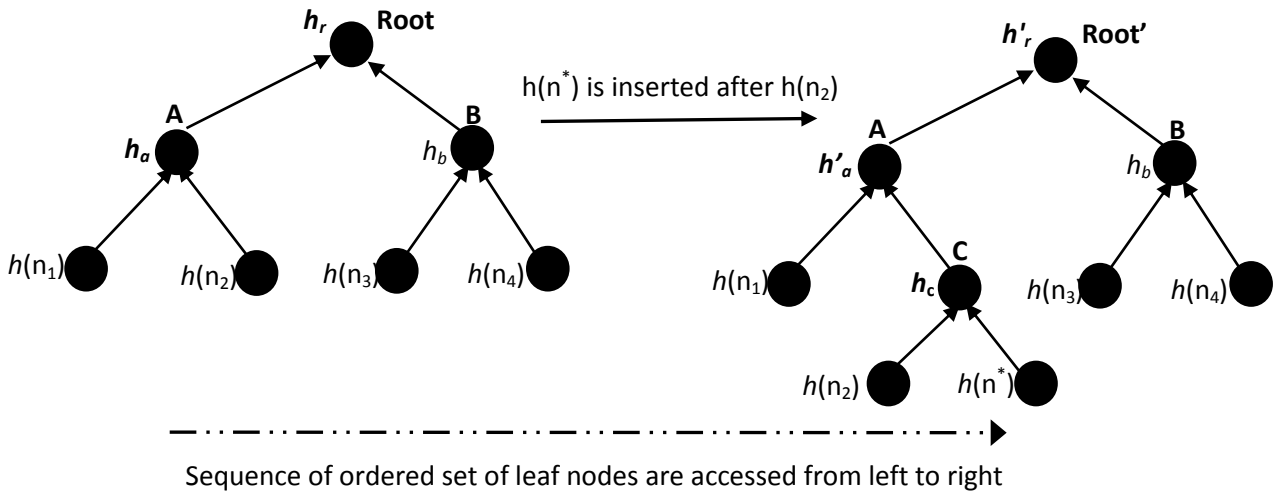


Figure 2.10: Example of MHT update under block insertion operation. Here, n_i and n^* are used to denote $H(m_i)$ and $H(m^*)$, respectively (Qian et al, 2008).

3. Data Deletion

Data deletion is just the opposite operation of data insertion. For single block deletion, it refers to deleting the specified block and moving all the later blocks, one block forward. Supposing that the server receives the update request for deleting block m_i , it will delete m_i from its storage space, delete the leaf node $h(H(m_i))$ in the MHT and generate the new root metadata R' as illustrated in figure 2.11.

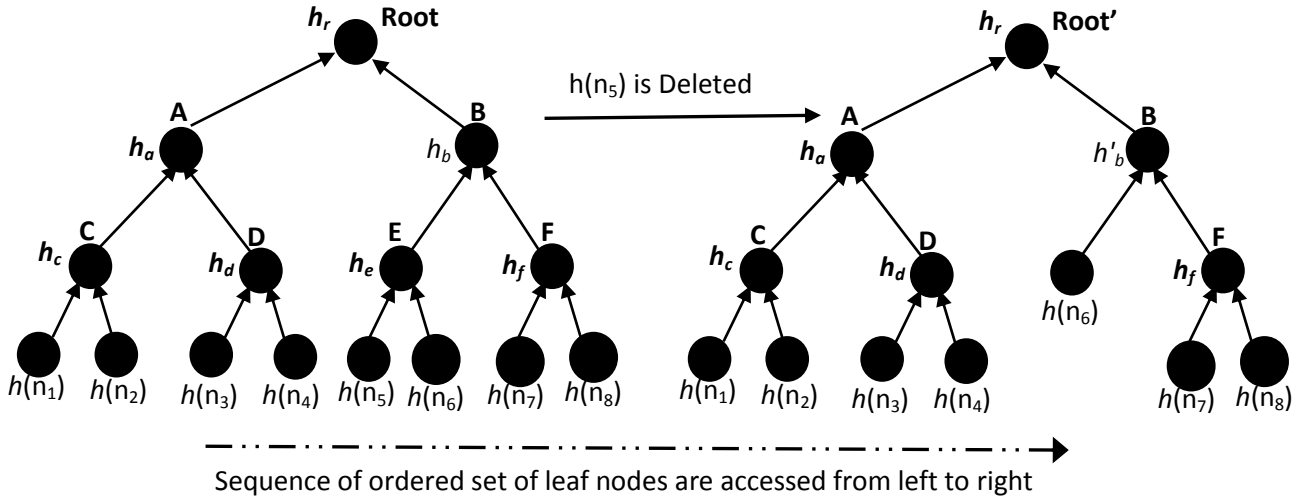


Figure 2.11: Example of MHT update under block deletion operation (Qian et al, 2008).

2.1.9 Further explanation of the model on RC6 Algorithm

RC6 is an improvement over RC5, and RC5 is an improvement over RC4. RC6 was designed to meet the requirements of increased security and better performance. RC6 makes use of data dependent rotations. One new feature of RC6 is the use of four working registers instead of two. While RC5 is a fast block cipher, extending it to act on 128-bit blocks using two 64-bit working registers. RC6 modified its design to use four 32-bit registers rather than two 64-bit registers. This has the advantage that it can be done two rotations per round rather than the one found in a half-round of RC5. The following descriptions illustrate the features of enhanced RC6 techniques;

A. Description of RC6

RC6 is a fully parameterized family of encryption algorithms. A version of RC6 is also specified as RC6- $w/r/b$ where the word size is w bits, encryption consists of a number of rounds r , and b denotes the encryption key length in bytes.

RC6 is targeted at $w = 32$ and $r = 20$, the parameter values specified as RC6- w/r are used as shorthand to refer to such versions. For all variants, RC6- $w/r/b$ operates on four w -bit words using the following six basic operations:

$a + b$: Integer addition modulo $2w$

$a - b$: Integer subtraction modulo $2w$

$a \oplus b$: Bitwise exclusive-OR of w -bit words

$a \times b$: Integer multiplication modulo 2^w

$a \lll b$: Rotate the w -bit word a to the left by the amount given by the least significant $\lg w$ bits of b

$a \ggg b$: Rotate the w -bit word a to the right by the amount given by the least significant $\lg w$ bits of b (where $\lg w$ denotes the base-two logarithm of w).

RC6 exploits data-dependent operations such that 32-bit integer multiplication is efficiently implemented on most processors. Integer multiplication is a very effective diffusion, and is used in RC6 to compute rotation amounts so that these amounts are dependent on all of the bits of another register. As a result, RC6 has much faster diffusion than RC5 and RC4.

B. Key Schedule

The key schedule of RC6- $w/r/b$ is practically identical to that of RC5- $w/r/b$. In fact, the only difference is that in RC6- $w/r/b$, more words are derived from the user-supplied key for use during encryption and decryption.

The user supplies a key of b bytes, where $0 \leq b \leq 255$. Sufficient zero bytes are appended to give a key length equal to a non-zero integral number of words; these key bytes are then loaded into an array of c w -bit words $L[0], L[1], \dots, L[c - 1]$. The number of w -bit words generated for additive round keys is $2r + 4$, and these are stored in the array $S[0, 1, \dots, 2r + 3]$.

The key schedule algorithm is as shown as follows:

Key Schedule for RC6- $w/r/b$

Input: User-supplied b byte key preloaded into the c -word array $L[0, 1, \dots, c - 1]$ Number of rounds, r

Output: w -bit round keys $S[0, 1, \dots, 2r + 3]$

Key expansion:

Definition of the magic constants:

$$P_w = \text{Odd}((e - 2)2^w)$$

$$Q_w = \text{Odd}((\phi - 2)2^w)$$

Where:

$$e = 2.71828182 \dots \text{ (base of natural logarithms)}$$

$$\phi = 1.618033988 \dots \text{ (golden ratio)}$$

Converting the secret key from bytes to words:

for $i = b - 1$ down to 0 do

$$L[i/u] = (L[i/u] \lll 8 + K[i])$$

Initializing the array S

$$S[0] = P_w$$

for $i = 1$ to $2r + 3$ do

$$S[i] = S[i - 1] + Q_w$$

Mixing in the secret key S

$$A = B = i = j = 0$$

$$v = 3 \times \max\{c, 2r + 4\}$$

for $s = 1$ to v do

{

$$A = S[i] = (S[i] + A + B) \lll 3$$

$$B = L[j] = (L[j] + A + B) \lll (A + B)$$

$$i = (i + 1) \bmod (2r + 4)$$

$$j = (j + 1) \bmod c$$

}

C. Encryption

RC6 encryption works with four w -bit registers A , B , C and D which contain the initial input plaintext. The first byte of plaintext is placed in the least significant byte of A . The last byte of plaintext is placed into the most significant byte of D . The arrangement of $(A, B, C, D) = (B, C, D, A)$ is like that of the parallel assignment of values (bytes) on the right to the registers on the left, as shown in Figure 2.12. The RC6 encryption algorithm is shown as follows:

Encryption with RC6- $w/r/b$

Input: Plaintext stored in four w -bit input registers A, B, C, D

Number of rounds, r

w -bit round keys $S[0, 1, \dots, 2r + 3]$

Output: Cipher text stored in A,B,C,D

Procedure:

$$B = B + S[0]$$

$$D = D + S[1]$$

for i = 1 to r do

{

$$t = (B \times (2B + 1)) \lll 1g w$$

$$u = (D \times (2D + 1)) \lll 1g w$$

$$A = ((A \oplus t) \lll u) + S[2i]$$

$$C = ((C \oplus u) \lll t) + S[2i + 1]$$

$$(A, B, C, D) = (B, C, D, A)$$

}

$$A = A + S[2r + 2]$$

$$C = C + S[2r + 3]$$

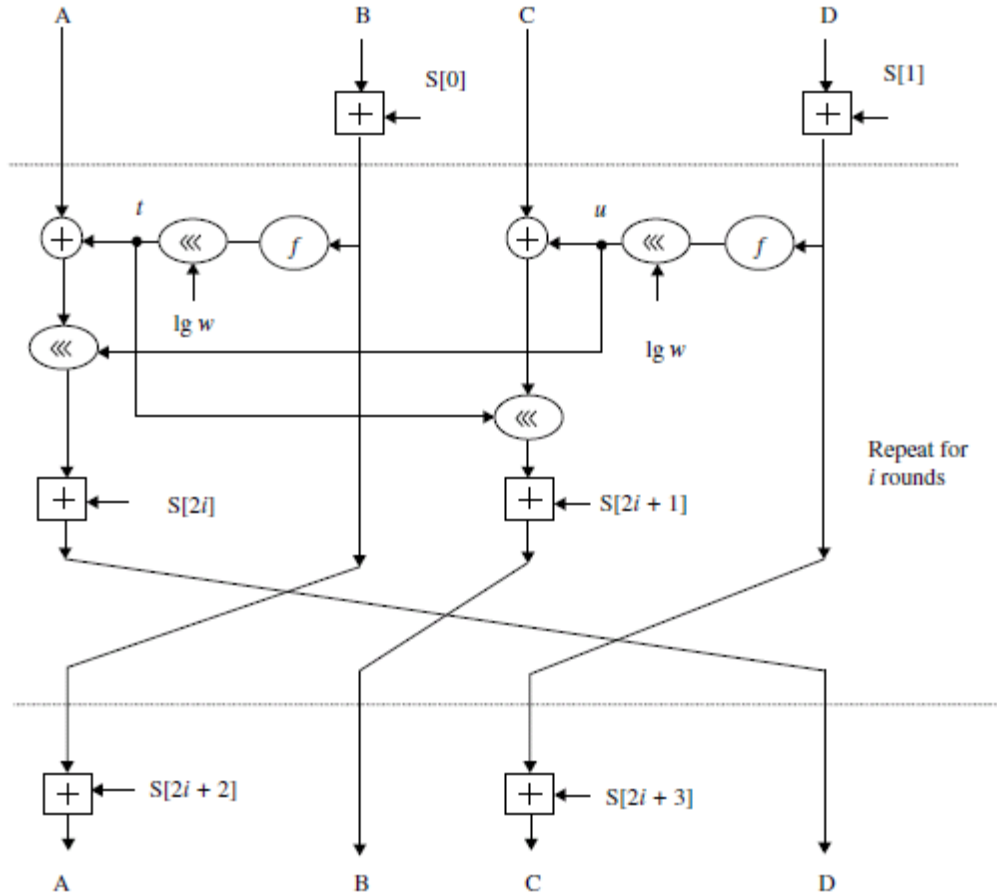


Figure 2.12: RC6- $w/r/b$ encryption scheme (Rhee, 2003).

D. Decryption

RC6 decryption works with four w -bit registers A, B, C, D which contain the initial output ciphertext at the end of encryption. The first byte of ciphertext is placed into the least significant byte of A . The last byte of ciphertext is placed into the most significant byte of D . The RC6 decryption algorithm is illustrated below:

Decryption with RC6- $w/r/b$

Input: Ciphertext stored in four w -bit input registers A, B, C, D

Number of rounds, r

w -bit round keys $S[0, 1, \dots, 2r + 3]$

Output: Plaintext stored in A, B, C, D

Procedure:

$$C = C - S[2r + 3]$$

```

A = A - S[2r + 2]
for i = r down to 1 do
{
(A, B,C,D) = (D,A, B,C)
u = (D × (2D + 1)) <<<< 1g w
t = (B × (2B + 1)) <<<< 1g w
C = ((C - S[2i + 1] >>>> t) ⊕ u
A = ((A - S[2i]) >>>> u) ⊕ t
}
D = D - S[1]
B = B - S[0]

```

The decryption of RC6 is depicted as shown in Figure 2.13

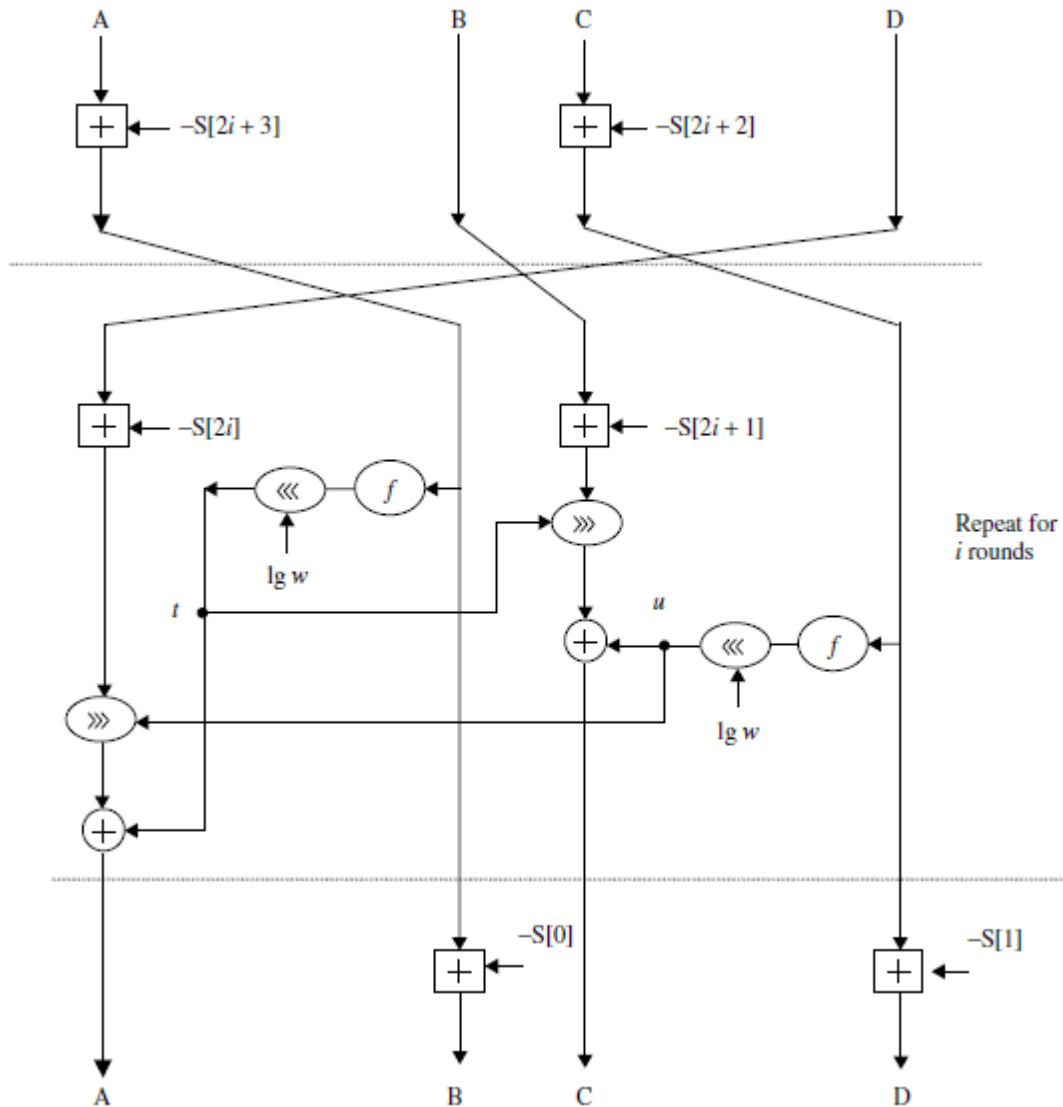


Figure 2.13 RC6-w/t/b decryption scheme (Rhee, 2003).

2.2 Review of Related Works

Juels and Kaliski (2007) describe a formal “proof of retrievability” (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error-correcting code to ensure both possession and retrievability of files on archive service systems. The limitation of this scheme lies on communication overhead and the use of TPAs which are vulnerable.

Shacham and Waters (2008) built on this model and constructed a random linear function based homomorphic authenticator which enables unlimited number of queries and requires less communication overhead.

Bowers, *et al* (2009) developed an improved framework for “proof of retrievability” (POR) protocols that generalizes the works of both Juels & Kaliski and Shacham & Water’s work. Later in their subsequent work, they extended POR model to distributed systems. However, all these schemes are focusing on static data without dynamic data operations. The effectiveness of their schemes rests primarily on the pre-processing steps that the user conducts before outsourcing the data file. Any change to the contents of data file, even few bits, must propagate through the error-correcting code, thus introducing significant computation and communication complexity.

Ateniese *et al.* (2007) define the “provable data possession” (PDP) model for ensuring possession of file on untrusted storages. Their scheme utilizes public key based homomorphic tags for auditing the data file, thus providing public verifiability with services of TPA which can be vulnerable. However, their scheme requires sufficient computation overhead that can be expensive for an entire file. In their subsequent work, they described a PDP scheme that uses only symmetric key cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which will also be supported in this research. However, their scheme focuses on single server scenario and does not address small data corruptions, leaving both the distributed scenario and data error recovery issue unexplored.

Curtmola, *et al* (2008) aimed to ensure data possession of multiple replicas across the distributed storage system. They extended the PDP scheme to cover multiple replicas without encoding each replica separately, providing guarantees that multiple copies of data are actually maintained. It lacks full dynamic data operation capabilities.

In other related work, Lillibridge *et al* (2003) presented a Peer-2-Peer (P2P) backup scheme in which blocks of a data file are dispersed across $m+k$ peers using an $(m+k, m)$ -erasure code. Peers can request random blocks from their backup peers and verify the integrity using separate keyed cryptographic hashes attached on each block. Their scheme can detect data loss from free riding peers, but does not ensure all data is unchanged.

Filho and Barreto (2007) developed a system to verify data integrity using RSA-based hash to demonstrate uncheatable data possession in peer-to-peer file sharing networks. However, their proposal requires exponentiation over the entire data file, which is clearly impractical for the server whenever the file is large.

Shah, *et al* (2007) proposed allowing a TPA to keep online storage honest by first encrypting the data then sending a number of precomputed symmetric-keyed hashes over the encrypted data to the auditor. However, their scheme only works for encrypted files and auditors must maintain long-term state. The TPA services can be vulnerable and the scheme lacks dynamic data operations on cloud.

Schwarz and Miller (2006) developed a scheme to ensure file integrity across multiple distributed servers, using erasure-coding and block-level file integrity checks. However, their scheme only considers static data files and does not explicitly study the problem of data error localization, which is under consideration in this research.

Zion and Kavitha in 2012 developed a new solution to compute signatures instead of Message Authentication Codes (MACs) to obtain public auditability. In their scheme, the data owner precomputes the signature of each block m_i ($i \in [1, n]$) and sends both F and the signatures to the cloud server for storage. To verify the correctness of F , the data owner can adopt a spot-checking approach, i.e., requesting a number of randomly selected blocks and their corresponding signatures to be returned. This basic solution can provide probabilistic assurance of the data correctness and support public auditability. Their new scheme in which the file F is divided into blocks and compute signatures for each block and sends both the signatures and the blocks F to the cloud server for storage using Merkle hash tree suffers a huge set back in terms of the security of their root hash which is not established in their work. Their scheme also adopts the services of TPA which can be hacked.

Gosavi and Umale (2014) built a system to ensure remote data integrity with support of both public audit ability and dynamic data operations using homomorphic authenticator with random mask techniques. The scheme considered concurrently handling of multiple audit sessions from different users for their outsourced data files as well as support for data dynamic operation such as block modification, insertion, deletion, verification and taking log history of client performed operation with the help of TPA. This TPA which is an external cloud operator can be compromised and the entire scheme becomes vulnerable.

Srijanya and Kasiviswanath (2013) presented an auditing model based on Merkle Hash Tree. In their work, they conducted a study on possible auditing mechanisms which can be offered as a service over hybrid or public clouds in which they adopted TPA services. This TPA services can be subscribed by the users to verify the integrity of the data stored in the public clouds. Their

scheme also supports public auditability for storage correctness assurance to allow anyone, not just the clients who originally stored their files on cloud servers, to have the capability to verify the correctness of the stored data on demand. It also supports dynamic data operation which allows the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance and blockless verification in which no challenged file blocks should be retrieved by their TPA which is an external body, during verification process for efficiency concern. The vulnerability of this scheme lies on the fact that they have failed to provide adequate security for their root hash which is the main security component of the scheme and adoption of TPA which can be compromised.

Khaba and Santhanalakshmi (2013) presented an effective and flexible Batch Audit scheme with dynamic data support to reduce the computation overheads using MHT. To ensure the correctness of users' data, the scheme adopts the services of a third party auditor (TPA), to verify the integrity of the data stored in the cloud on behalf of the cloud client, their scheme also uses symmetric encryption for effective utilization of outsourced cloud data. Their scheme also uses data reading protocol and data management algorithm to process integrity of data before and after entering data into the cloud in which the actual size of data is ascertained and maintained even though the user has carried out some dynamic operations on the data. This gives the user full control of data stored in cloud apart from TPA. To avoid server failure and any unexpected error their scheme decides to putting one server restore point in cloud server database for efficient data back up or restore using multi server data comparison method though the process is CSP dependent. The data reading protocol and data management algorithm used in this scheme to ascertain and maintain the actual size of stored data will cause extra computational overhead. There is no provision for adequate security for MHT root hash and services of a TPA can be vulnerable.

Karthikeyan (2015) carried out a research on a particular kind of distributed database called audit logs that are used to log security events in a distributed system, such as granting access to a sensitive resource. The research depicted a scenario in which a hacker can be detected when it breaks into an enterprise network by logging the attacker's activities on machines across the network. For an attacker to escape detection, it may try to erase or tamper with the log, and hence making the integrity of this log to be security-critical. The scheme decides to publish a cryptographically strong hash of the full log to the cloud using Merkle Hash Tree (MHT)

techniques. A user can use this scheme to verify the integrity of a part of a log retrieved from an untrusted server or cloud by downloading and verifying that the hash of the retrieved log matches the hash of the expected value or log retrieved from a trusted TPA server. The scheme emphasizes that all attacks on the logs will be detected if the hash function is collision resistant, but forgets the fact that TPA as an external body with root hash of the MHT can be biased and make the scheme vulnerable as the scheme does not provide any means of securing the root hash from the TPA.

Karthika *et al* (2015) in their research developed a scheme of remote user secret image authentication using the concept of Merkle Hash Tree (MHT). In this scheme, the data owner stores the file in an encrypted form using Advanced Encryption Standard (AES) in the cloud server. The cloud user is expected to register with the owner along with the personal credentials and a generated secret image. The user name, password and root signature will be sent to the user's registered email account. At the client side, the secret image template is split into eight shares using image processing technique Boundary Splitting Algorithm. The split eight shares are given as inputs to MHT in which root hash signature is generated and stored in the cloud server. The user has to submit the root signature for authentication purpose. Though the scheme tries to secure the root hash using symmetric encryption at the TPA called Jelastic server, this Jelastic server can suffer from code manipulation masterminded by the server manager. The scheme also does not support cloud data dynamic operations.

Avinash *et al* (2015) carried out research on cloud data integrity authentication in which large data set are outsourced in encrypted form to the cloud. This encryption hides the connection between the documents and in turn, makes the ciphertext search critical which can suffer from different form of attack at the server side. Thus, a verifiable method should be provided for users to verify the accuracy of the search results. This research developed a hierarchical clustering method in order to get a better clustering result. Their method is based on k-means clustering algorithm. The research adopts the backtracking algorithm as a new search technique on the above clustering method. For the fact that data increases in volume with time, the scheme adopts Merkle hash tree and cryptographic signature as a verification mechanism to ensure the accuracy and perfectness of query results from untrusted cloud. Their scheme reduces the entire search time which only increases sequentially and not exponentially. It equally solves the problem of multi keyword search problems and also showcases any irregularities in the retrieved documents

and raises the search efficiency but did not support cloud data dynamic operations and there is no adequate security measure for the MHT root hash.

Anmol *et al* (2015) and Snehal *et al* (2014) recognized the efficacy of RC6 cryptosystem in their respective researches by adopting it to encrypt and decrypt data in their models. Their models separate Encryption-Decryption services from the actual cloud storage. Because of this separation method, the original data and decryption key will be stored at different storage area so that the attackers will get either encrypted data or decryption key. The model with Encryption-Decryption services only encrypts the data and this encrypted data is sent to cloud storage and then original data is deleted from Encryption-Decryption scheme. Both researches tried to get rid of TPA services due to its vulnerabilities. The limitation of these models exists due to the fact that they do not support dynamic cloud data operations.

2.3 Summary of Literature Review and Knowledge Gap

The Proof of Retrievability (POR) scheme developed by some researchers especially by Juels and Kaliski (2007) as reviewed has a limitation which lies on communication overhead and the use of TPA services which is vulnerable. It also has a limited number of queries it can address during auditing process and it is based on non-distributed system. Bowers, *et al* (2009) and other previous researches as reviewed still focus on static auditing scheme which introduces a significant computation and communication complexity.

Some reviewed works also define the “provable data possession” (PDP) scheme that uses only symmetric key cryptography to achieve lower-overhead but focuses only on a single server scenario and does not address small data corruptions, leaving both the distributed scenario and data error recovery issue unexplored.

Zion and Kayitha (2012), Srijanya and Kasiviswanath (2013) and other researchers as reviewed presented an auditing model based on Merkle Hash Tree. The vulnerability of these schemes lies on the fact that they have failed to provide adequate security for their root hash which is the main security component of the scheme and adoption of TPA which can be compromised.

Anmol *et al* (2015) and Snehal *et al* (2014) recognized the efficacy of RC6 cryptosystem in their respective researches as reviewed by adopting it to encrypt and decrypt data in their new models but the limitation of these models is that they do not support dynamic cloud data operations.

Having observed all these limitations, this research develops an enhanced and secured hybrid cloud data storage auditing model that support both public auditability and dynamic data operation with MHT and IRC6 encryption algorithms. The MHT that is adapted as data authentication structure addresses the full cloud dynamic data operation which in turn addresses computational and communication overhead, while the developed IRC6 cryptosystem addresses the security of MHT root key which is the strength of the auditing scheme. It also bypasses the risk and cost of adopting TPA.

CHAPTER THREE

SYSTEM ANALYSIS AND METHODOLOGY

3.1 System Analysis

3.1.1 Analysis of the Existing System

In order to achieve the assurances of cloud data integrity and availability and then enhance the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed. However, the fact that users no longer have physical possession of data in the cloud prohibits the direct adoption of traditional cryptographic primitives for the purpose of data integrity protection. Hence, the verification of cloud storage correctness must be conducted without explicit knowledge of the whole data files. The data stored in the cloud may not only be accessed but may also be frequently updated including insertion, deletion, modification and appending. Thus, it is also imperative to support the integration of this dynamic feature into the cloud storage correctness assurance.

Hypothesis obtained in this research from pre-analysis of data collected globally through review of existing literature on recent global RDA operations obtained over the internet, structured interview and interactions with staff of different universities and other organizations in Nigeria, showed that many organizations globally have been carrying out RDA activities with one challenge or the other, either insecurity of the root harsh (Avinash *et al*, 2015) or vulnerability of the TPA. While some organizations in Nigeria such as universities, banks, companies and NGOs have been battling to manage their huge data storage internally though with high risks and not cost effective, others have had high level of data storage challenges such as conflict of interest, self-motivated compromise, political motivation, internal risks and fire or liquid hazard (Sadoya *et al*, 2004). It is very important to state here that academic data from higher institutions for example, is the life-wire of every institution and must be properly stored with highest level of security.

3.1.1.1 The Organization and Its Environment

The organization in focus as case study for this research is one of the Federal Universities in Nigeria Known as Federal University Wukari (FUW), Taraba State.

Federal Universities in Nigeria are established by the Federal Government of Nigeria to be student centered and community engaged institutions to provide an enabling environment that enhances intellectual growth, a strong commitment to academic excellence, integrity and entrepreneurship; creating new knowledge and using ICT and other enabling technologies to solve practical problems that benefit humanity; preparing their students as well as professionals in their community for ethical leadership; and promoting service to community and enduring sense of global citizenship.

Currently there are forty (40) Federal Government approved Federal Universities in Nigeria. The majority of which have been in existence for years, carrying out both academic, administrative and community development activities thereby generating tremendous amount of data for storage, of which Federal University Wukari (FUW) as a case study is not an exemption.

FUW was founded in 2011 and the stages of the structure of academic programmes of the university was planned to have five (5) developmental stages with duration of five (5) years for each stage. Therefore, the faculties, colleges, schools, departments and programmes of the university shall be established in stages covering a period of twenty five (25) years. Each stage of academic development shall be characterized by the establishment of faculties, departments, and/or programmes at the undergraduate and postgraduate levels.

Currently, FUW has three (3) faculties namely Faculty of Pure and Applied Sciences, Faculty of Agriculture and Life Sciences and Faculty of Humanities, Management and Social Sciences; with a total of 24 departments. It equally has non-academic departments and units including the administrative arm of the University which will be shown on the organogram.

Each of these departments and units both academic and non-academic generates huge amount of data on a daily basis that require adequate security which necessitates the research. Figure 3.1 below shows the organogram of FUW.

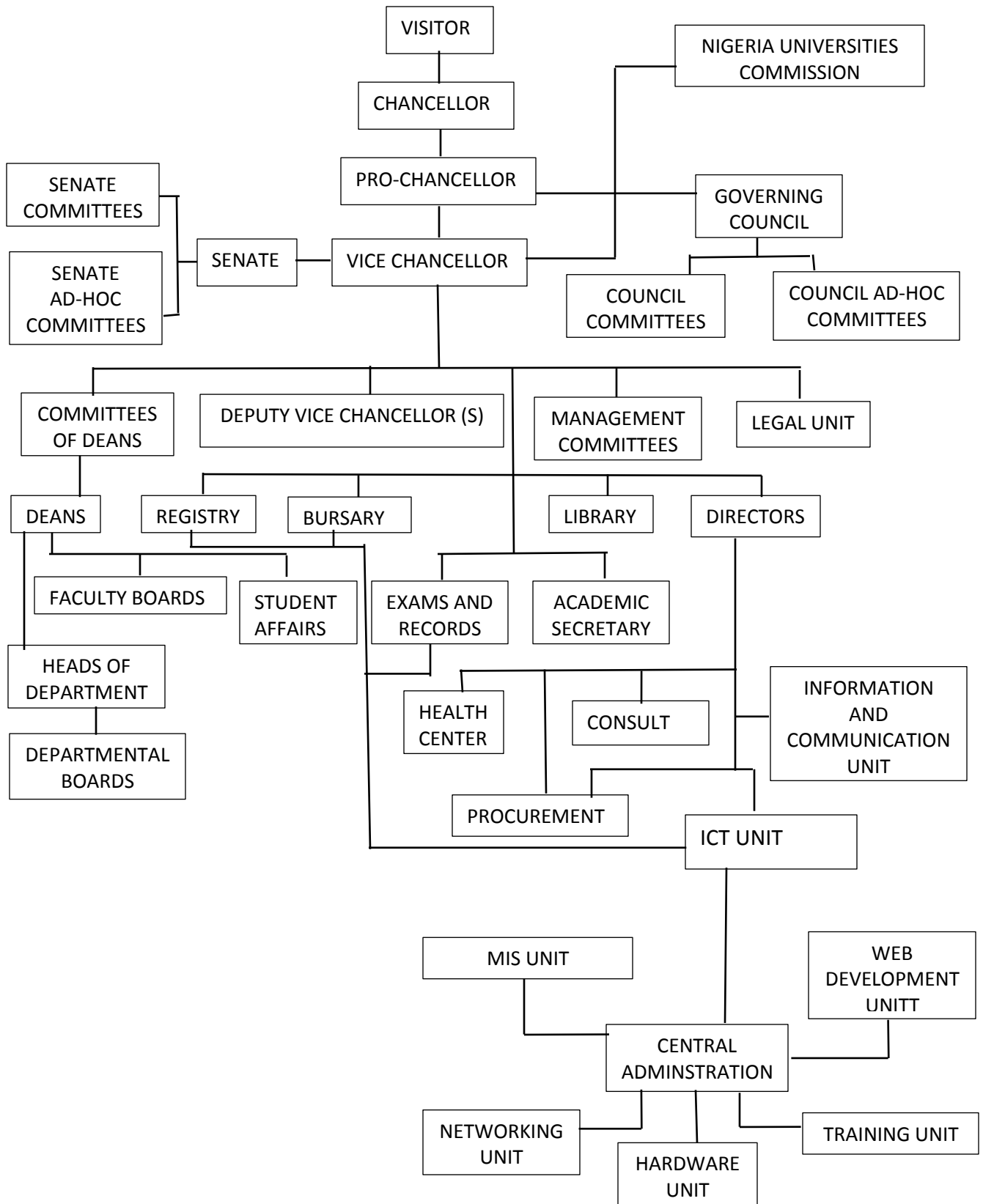


Figure 3.1: FUU Organogram (Marketing & Communication Unit, FUU)

3.1.1.2 Data Flow Diagram of the Existing System

The data flow diagram in figure 3.2 shows the basic operations that exist in various universities for data generation and storage management for example.

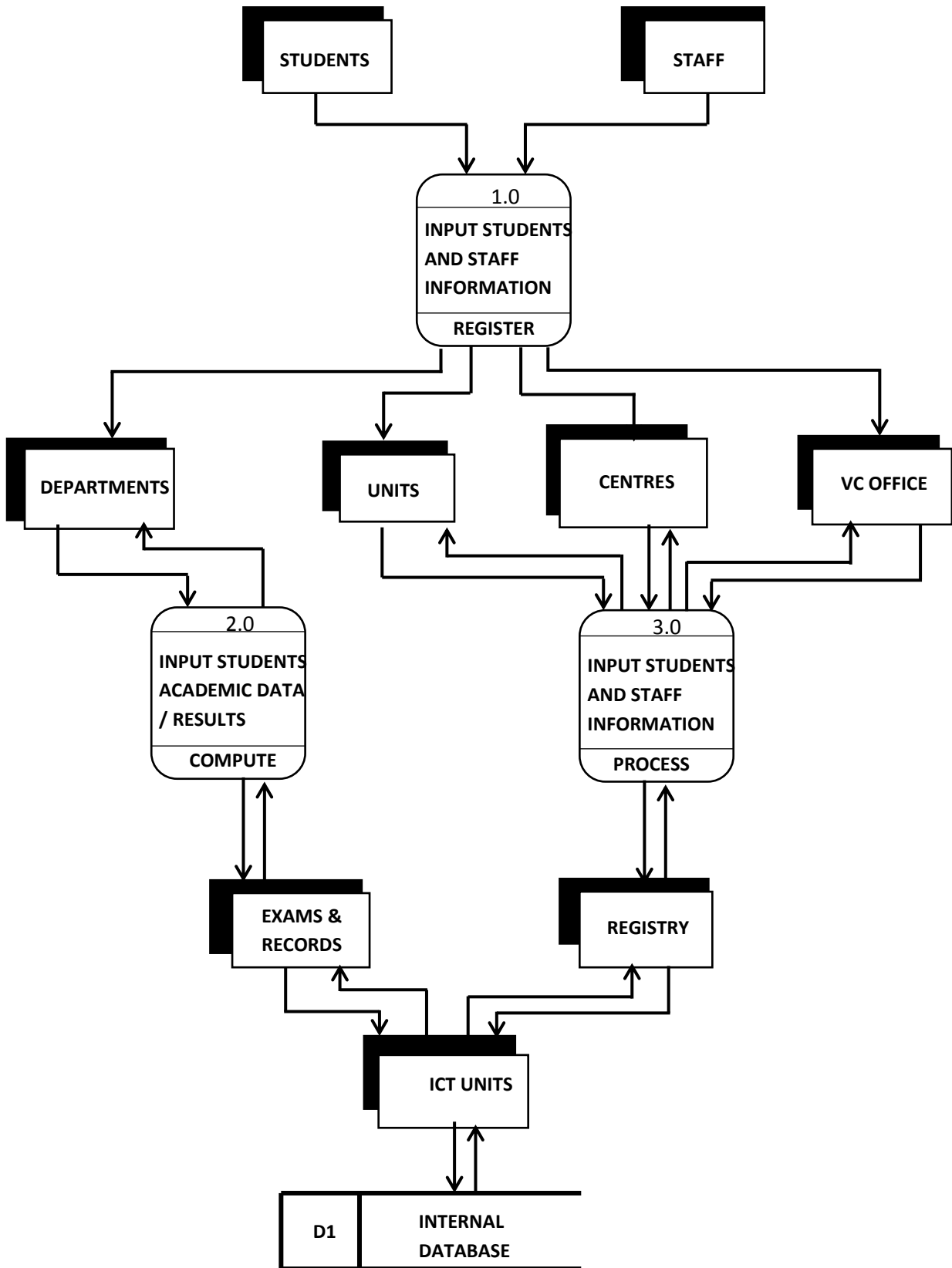


Figure 3.2: Data Flow Diagram showing conventional data generation and storage in Federal Universities

Data are generated from different academic and non-academic departments, units and centres within the university and other organizations, which go through some levels of processing, validations, authentications and necessary approvals from one section of the organization to another until its final storage point at the ICT unit where a local server or central storage device is made available for long time data storage and management.

It was equally observed through the analysis of data collected that some universities and other organizations do not have centralized data storage management system thereby storing data at different sections of the organization as generated, which are managed ineffectively and are prone to errors, compromise and risk of hazards.

Considering all these limitations, this research tends to proffer solution by proposing outsourcing of these sensitive data to the public cloud which provides unlimited storage resources and other technologies for real time data access.

Many existing cloud data storage security techniques have also been studied and their limitations include provision of only binary results about the storage status across the distributed servers; storing data redundantly across multiple physical servers which causes the loss of data integrity and availability threats like Byzantine failure, malicious data modification attack, and even server colluding attack. Dynamic data support is not maintained at the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud. Authentication Handshakes are not available in existing systems.

Also the worst case scenario that can happen on an existing cloud storage system is on the case of Third Party Auditor (TPA) server being compromised as it houses the main secret key entrusted to it by the Data Owner (DO) for subsequent auditing. This will be disastrous as the integrity and confidentiality of the stored data at the cloud can be compromised without the knowledge of both the TPA and DO. Figure 3.3 shows a model of existing cloud architecture with an option to eliminate the services of the TPA.

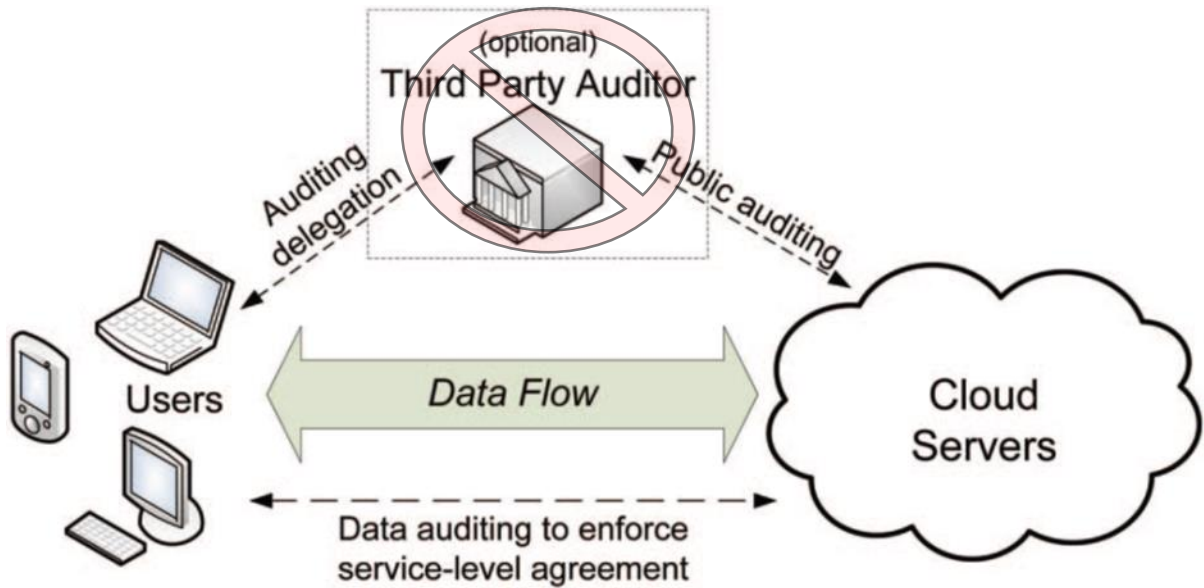


Figure 3.3: Existing Cloud Model Architecture (Gosavi and Umale, 2014)

3.1.1.3 Weaknesses of existing system

- i. The services of the TPA can be a loophole to the original data if hacked or compromised by TPA itself.
- ii. There is no proper security of the MHT root hash which is the backbone of the auditing scheme.
- iii. Computation overhead on replica base auditing scheme as the entire blocks are required before remote integrity check is carried out.
- iv. Provision of only binary results about the storage status across the distributed servers.
- v. Storing data redundantly across multiple physical servers which causes the loss of data integrity and availability threats like Byzantine failure, malicious data modification attack.
- vi. Dynamic data support is not maintained at the same level of storage correctness assurance.
- vii. Authentication Handshakes are not available in existing systems

3.1.2 Analysis of the New System

Having observed all these limitations, this research then developed an enhanced and secured hybrid cloud data storage and auditing model that supports both public auditability and dynamic

data operation with MHT and IRC6 homomorphic encryption algorithms. It also bypasses the risk and cost of adopting TPA.

This developed system supports an effective and flexible distributed storage verification scheme with explicit dynamic data support to ensure the correctness and availability of users' data in the cloud. It relies on homomorphic encryption with Merkle Hash Tree (MHT) and IRC6 encryption techniques in the file distribution preparation to guarantee the data dependability against Byzantine servers, where a storage server may fail in arbitrary ways as contained in the service level agreement with the CSP. The hybrid model is achieved by combining MHT and enhanced RC6 cryptosystem to strengthen the auditing scheme. The MHT is used to achieve remote data auditing on blocks of files and other dynamic operations on files in remote cloud storage. The life wire of MHT depends on the security of the root hash. The security of the root hash is properly secured using IRC6 encryption to solve the problem that exists in the previous work.

This solution drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification of homomorphic encryption techniques, this scheme supports the storage correctness insurance as well as data error localization in case of server failure as also contained in service level agreement with the CSP. Whenever data corruption has been detected during the storage correctness verification, the CSP can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s). In order to further improve the security of system, computation resources, and even the related online burden of users, this system also provides public auditability scheme that avoids the services of TPA which can cause tremendous damage to the organizational data if hacked.

Triangular Handshake Authentication Scheme (THAS) is equally adopted in this research to further strengthen the security of data communication between the DO and other organizational users and between the DO and CSP using simple but highly secured authentication scheme based on encrypted shared secret and a clock value for time stamp.

The developed enhanced hybrid model is depicted in figure 3.4 while the user subsystem of the model is represented in figure 3.5.

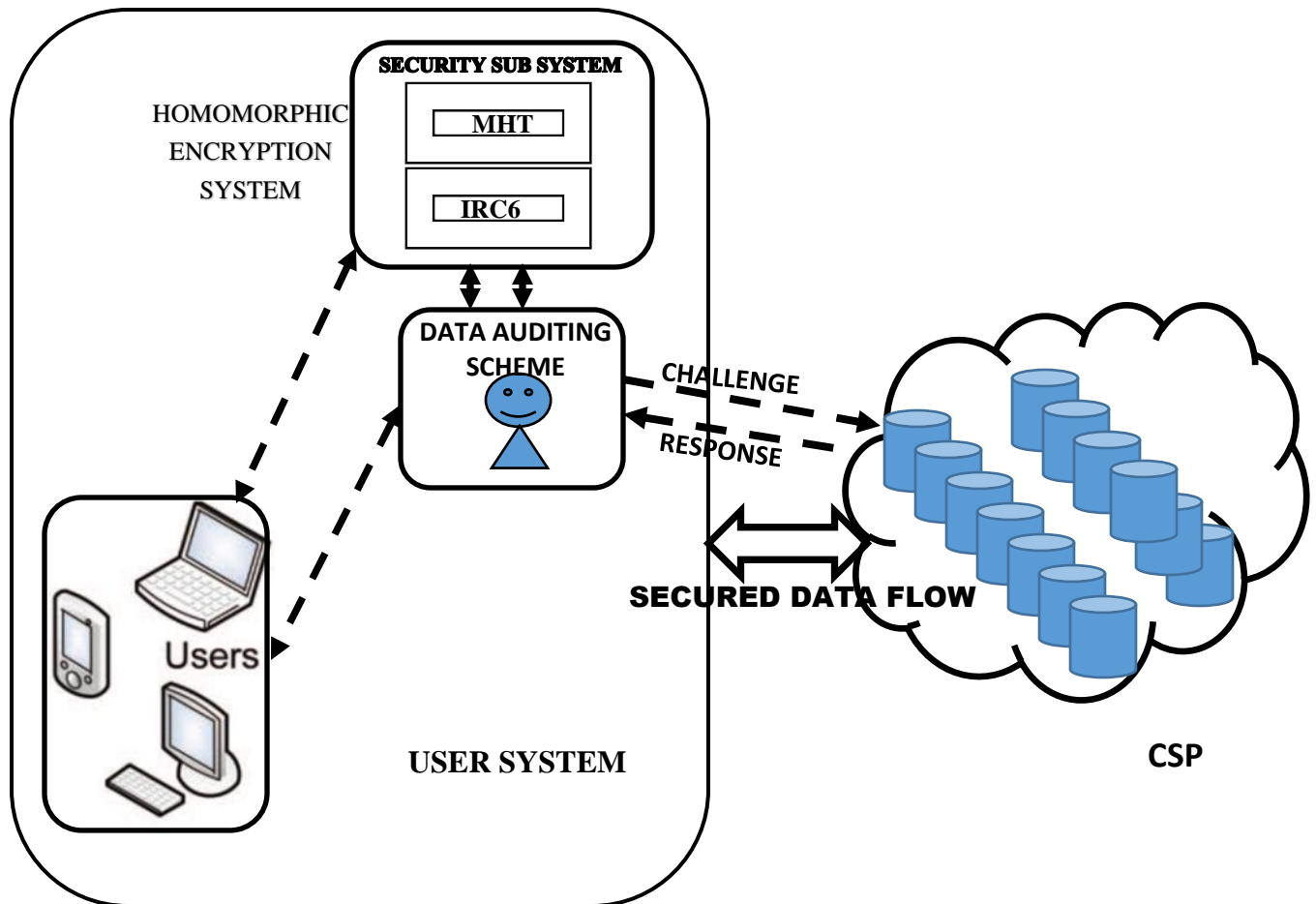


Figure 3.4: New Cloud Storage Service Architecture with Enhanced Hybrid Auditing Scheme

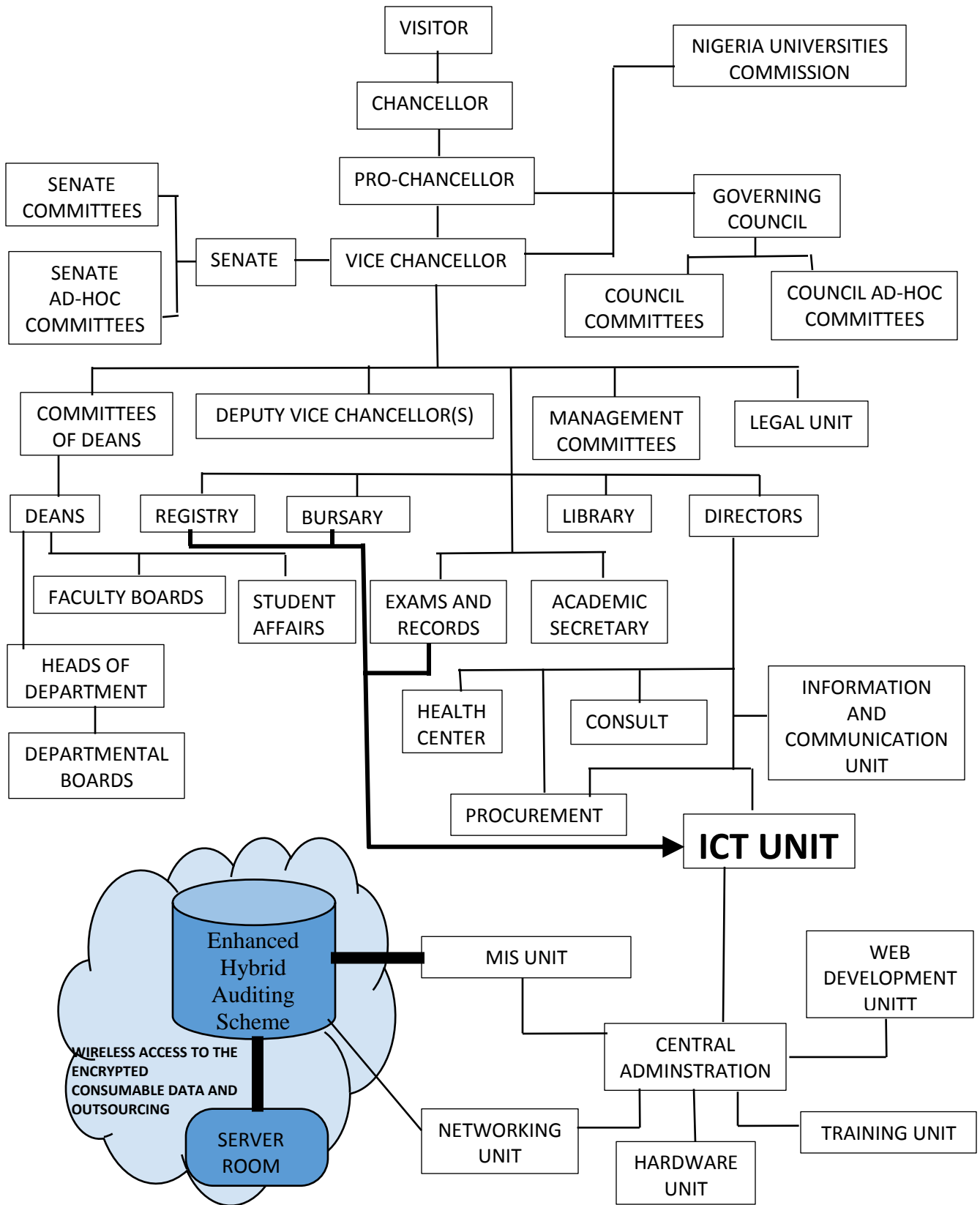


Figure 3.5: Conceptual Model of the New User System with Enhanced Hybrid Auditing Scheme (EHAS)

Representative network of the new cloud storage service architecture with enhanced auditing scheme is illustrated in Figure 3.4 with notable four different network entities while figure 3.5 shows the conceptual model of the user system. The components of these models are explained as follows:

- i. **Users:** The users consist of both individual consumers and organizational Data Owner (DO), who have data to be stored in the cloud and rely on the cloud for data computation.
- ii. **Consumer users:** The user or client sends the query to the server. Based on the query the server sends the corresponding file to the client on read-only format. Before this process, the client authorization step is involved. In the server side, it checks the client's name and its password for security process. If it is satisfied, it will then receive the queries from the client and searches the corresponding files in the database and send to the client. If the server finds the intruder means whose authentication parameters do not match, it set the alternative Path to that intruder. It should also be noted that access to the server has time stamp.
- iii. **Enhanced Hybrid Auditing Scheme:** The developed auditing scheme can be handled by either the DO or any trusted staff of the organization who has the expertise and capabilities other data users may not have, to assess and expose risk of cloud storage services on behalf of the DO upon request by remotely challenging the cloud server without necessarily downloading the audited block using a well structured query. The process is made possible with the MHT auditing architecture.
- iv. **Homomorphic Encryption Subsystem:** The MHT subsystem carries out the job of hashing the organizational data into blocks of cipher text for optimal transmission and effective storage at the cloud server while IRC6 security subsystem provides adequate security for the MHT root hash by encrypting and decrypting the root hash.
- v. **Cloud Service Provider (CSP):** A CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.
- vi. **Cloud Storage Server (CSS):** CSP achieves provision of data storage service to clients by managing Cloud Storage Server which is divided into two components. Management Server, which manages the server and Data Server, which Stores the clients' data.

vii. **Cloud Data Storage:** In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated, and distributed manner. Data redundancy can be employed with a technique of homomorphic encryption to further tolerate faults or server crash as user's data grow in size and importance. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform block level operations on his data. The most general forms of these operations considered in this research are block update, delete, insert, and append.

As users no longer possess their data locally, it is of critical importance to ensure users that their data are being correctly stored and maintained as contained in the services level agreement between the Do and CSP. That is, users should be equipped with security means so that they can make continuous correctness assurance (to enforce cloud storage service-level agreement) of their stored data even without the existence of local copies.

In case those DOs do not necessarily have the time, feasibility or resources to monitor their data online, they can delegate the data auditing tasks to a trusted and experienced staff of the organization rather than TPA which can be hacked.

viii. **New Local Data Centre:** Using university system scenario as depicted in figure 3.5, Data which are generated from different academic and non-academic departments, units and centres within the university and go through some levels of processing, validations, authentications and necessary approvals from one section of the university to another are moved to the new data processing Centre at the ICT unit where they will be preprocessed by the homomorphic encryption subsystem before outsourcing. It is also at this new Centre that the remote auditing of the outsourced data to cloud will be carried out with the developed enhanced auditing scheme.

ix. **Cloud Authentication Server Scheme:** The Authentication Server (AS) functions as any AS would with a few additional behaviors added to the typical client-authentication protocol. The first addition is the sending of the client authentication information to the masquerading router. The AS in this model also functions as a ticketing authority, controlling permissions on the application network. The other optional function that should be supported by the AS is the updating of client lists, causing a reduction in

authentication time or even the removal of the client as a valid client depending upon the request.

3.1.2.1 Overall Object Diagram of the New System

Some aspects of the new system are modeled in this subsection using Unified Modeling Language (UML). Figure 3.6 depict UseCase diagram for data auditing and dynamic data operations for the new system 3.7. shows the UseCase for the new system administration while figure. Figure 3.8 shows the collaboration diagram of the new system for contents/data auditing and dynamic operations.

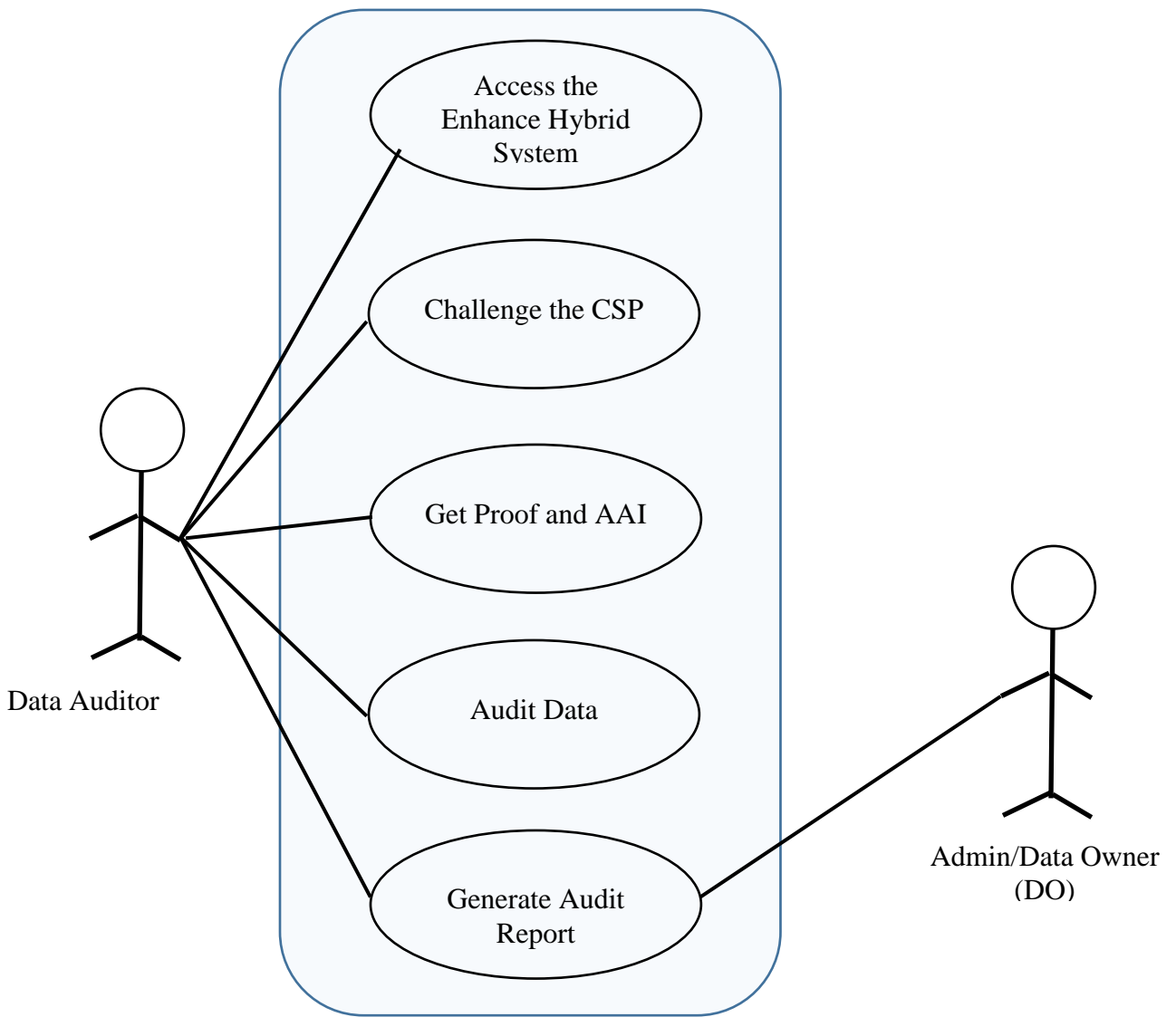


Figure 3.6: UseCase Diagram for Data Auditing

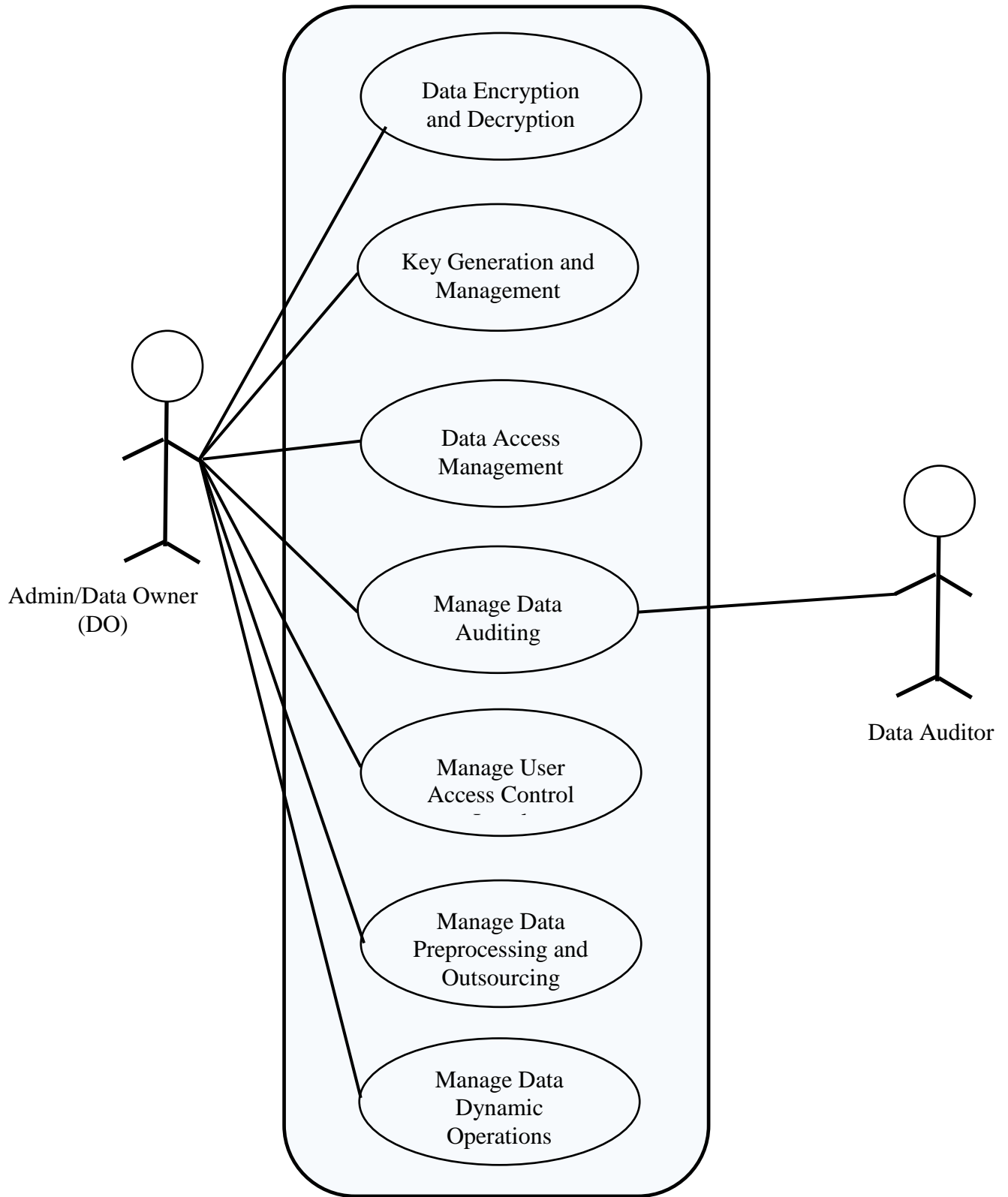


Figure 3.7: UseCase for the New System Administration

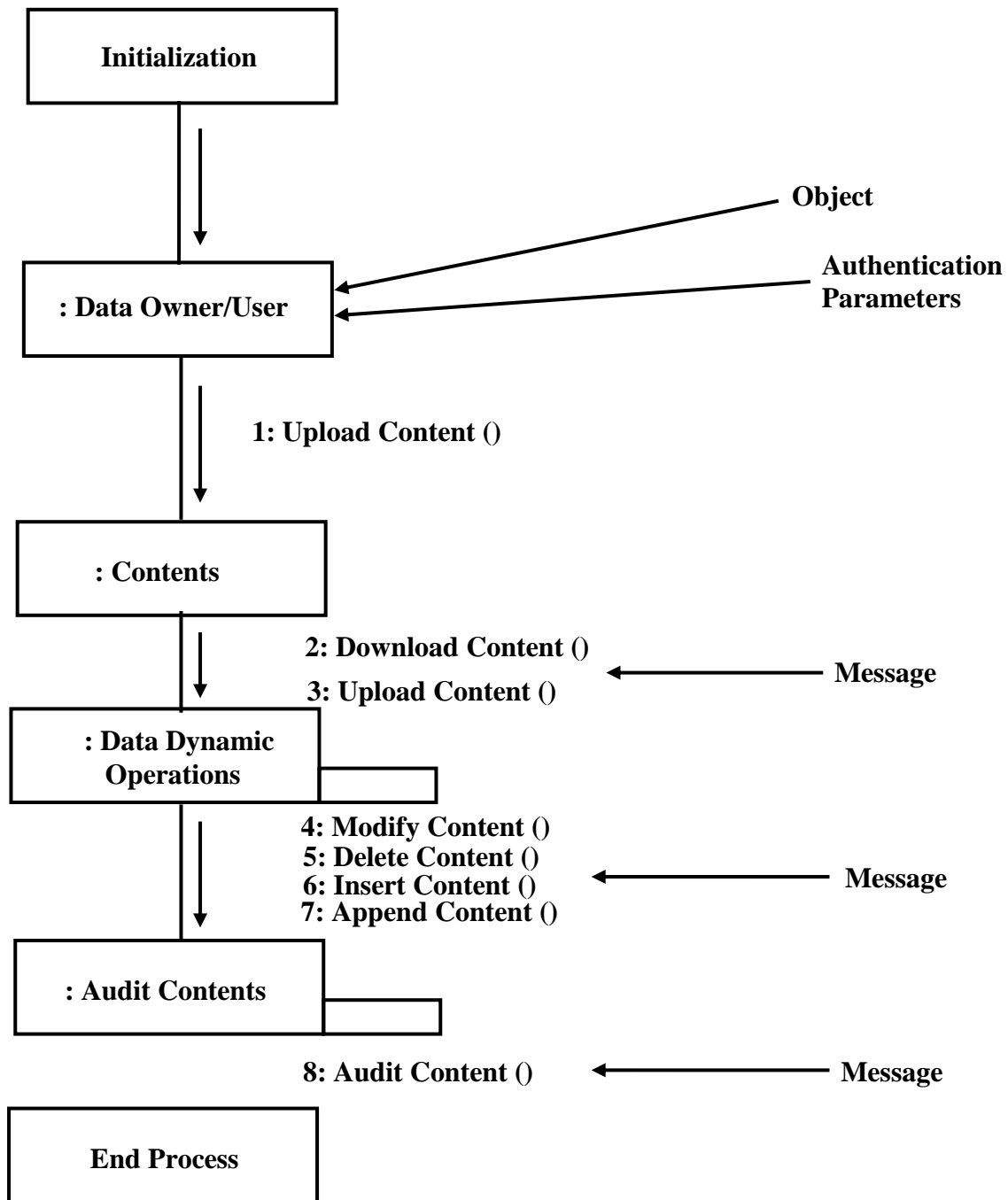


Figure 3.8: Collaboration Diagram of the new System for Data Auditing and Dynamic Operations

3.1.2.2 Advantages of the New System

The developed system provides adequate means of remotely auditing outsourced data to the cloud by using MHT and IRC6 cryptosystem which is an improvement on existing auditing schemes. Auditing based on MHT reduces computation overhead during auditing process while IRC6 cryptosystem secures the MHT root hash which is the main auditing key. The security of the root hash has been problematic in most reviewed researches on RDA. The developed system further strengthens the data integrity assurance to the DOs by eliminating the services of vulnerable TPA and allows the auditing to DOs by themselves.

3.2 Methodology Adopted

Methodology adopted in this Research is Object Oriented Analysis and Design Methodology (OOADM) with MHT and IRC6 techniques. Object-oriented analysis and design methodology (OOADM) was adopted in this research together with Merckle Hash Tree (MHT) authentication technique and Improved Revester Code Version 6 (IRC6) cryptographic technique, to achieve the research goal. IRC6 was developed and used in this research. OOADM stages include system analysis through which the existing system has been analyzed and a new system is developed. Other stages utilized in the cause of this research include system design, object design and implementation.

Using Object Oriented Analysis and Design methodology (OOADM) to develop real-time systems has the potential to produce safer, more reliable and maintainable system or solution (Fredrickson et al, 1998). Instead of using functional decomposition of the system, the OOADM approach focuses on identifying objects and their activities (Wang et al, 1996). This methodology is adopted in this research based on the following advantages:

- i. **User centered advantage:** The analysis phase of OOADM ensures that the user(s) of the system is carried along by taking priority of the system user requirements, analyzing possible data that will help to achieve the aim of the research and development of suitable dynamic interface for accessing the system.
- ii. **Reduced Maintenance:** The primary goal of the adopted methodology is the assurance that the system will enjoy a longer life while having far smaller maintenance costs. Because most of the processes within the system will be encapsulated, the behaviors may be reused and incorporated into new behaviors.

- iii. **Real-World Modeling:** The methodology will help in modeling the new system in a more complete fashion than do traditional methods. Objects which will constitute the components of the new solution are organized into classes of objects, and objects are associated with behaviors. The model will be based on objects, rather than on data and processing.
- iv. **Improved Reliability and Flexibility:** The adopted methodology is more reliable than traditional systems, primarily because new behaviors can be "built" from existing objects. Because objects can be dynamically called and accessed, new objects may be created at any time. The new objects may inherit data attributes from one, or many other objects. Behaviors may be inherited from super-classes, and novel behaviors may be added without effecting existing systems functions.
- v. **High Code Reusability:** When a new component or object of the new solution is created, it will automatically inherit the data attributes and characteristics of the class from which it was reproduced. The new object will also inherit the data and behaviors from all superclasses in which it participates.

3.3 Justification of the New System

The new system provides adequate means of remotely auditing outsourced data to the cloud by using MHT and improved RC6 (IRC6) cryptosystem which is an improvement on existing auditing schemes.

Auditing based on MHT reduces computation overhead during auditing process while IRC6 cryptosystem secures the MHT root hash which is the main auditing key. The security of the root hash has been problematic in most reviewed researches on RDA. The proposed system further strengthens the data integrity assurance to the DOs by eliminating the services of vulnerable TPA and allows the auditing to DOs.

The proposed Triangular Security Handshake (TSH) with timestamp, intrusion dection model through access log and adversary data authentication model will improve access control security.

3.4 High Level Model of the New System

Figure 3.9 shows the high level model of the new system. The model clearly shows the various aspects of the new system, the key actors and their respective activities.

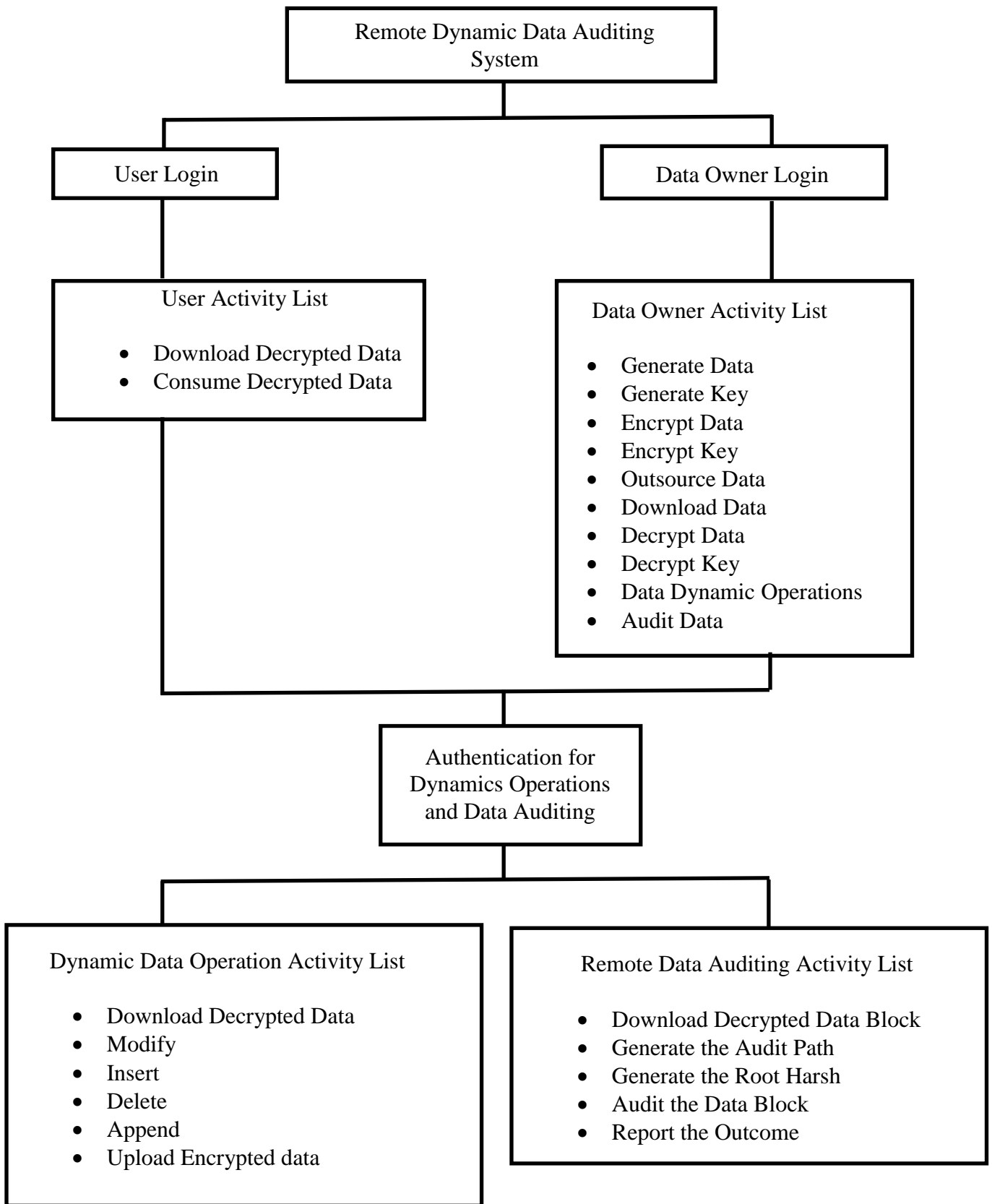


Figure 3.9: High Level Model of the New System

CHAPTER FOUR

SYSTEM DESIGN AND IMPLEMENTATION

4.1 Objectives of the Design

The new system is aimed at improving the security of data outsourced to the public cloud. It assures the DO of the integrity of its data through the process of remote dynamic data auditing and data dynamic operations in the cloud by maintaining the same level of storage correctness assurance even if users modify, delete, insert or append their data files in the cloud.

It will equally eliminate vulnerabilities that are posed to the outsourced data by TPA services, secure the main auditing key (root hash) using IRC6 cryptosystem.

4.2 Control Centre/Main Menu

The main menu of the new system controls two major subsystems namely the user subsystem and Data Owner subsystem. The main menu controls the security access to these subsystems and logs number of failed login attempts to checkmate the intruders into the system. It gives the system users optional opportunity to keep track of their login credentials. Figure 4.1 shows the main menu of the new system.

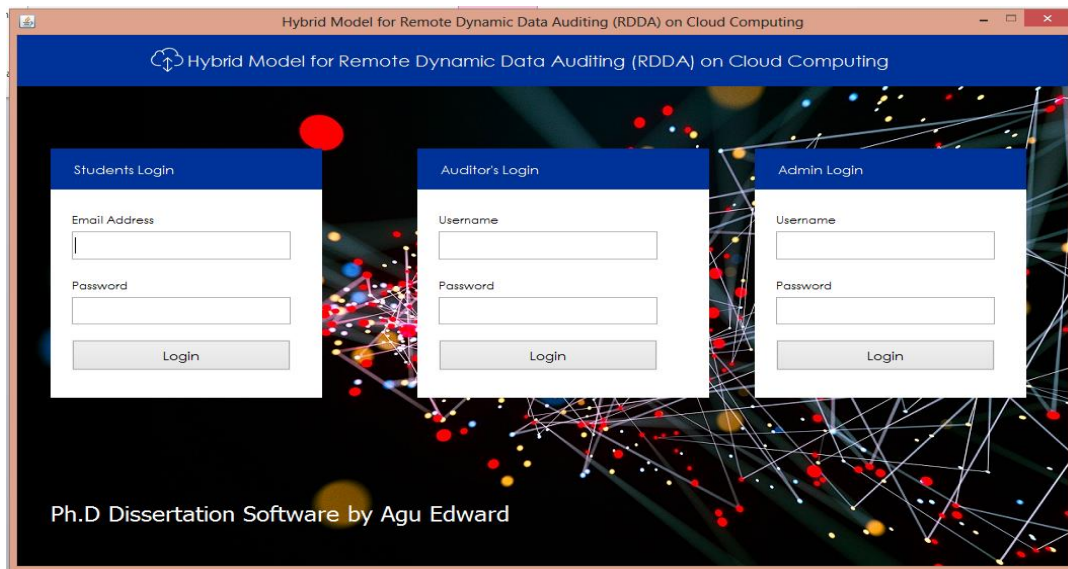


Figure 4.1: Main Menu

4.3 The Submenus/Subsystems

There are three subsystems in the new system namely the user subsystem, audit subsystem and admin subsystem. The user subsystem handles user data consumption in which valid stored data are retrieved and displayed to only authentic users. The audit subsystem handles the authentication of valid auditors, retrieval and integrity proof of the data stored at the cloud. Public auditability function of this system is achieved only when the task attached to this subsystem is assigned to personnel other than the DO within the organization.

The admin subsystem handles the data collection, preprocessing, key generation, data outsourcing to the cloud, user management, access level definition and management.

4.4 System Specifications

4.4.1 Database Development Tools

Computer Assisted Software Engineering (CASE) tool is the database development tool used for this system which provides extensive functionality for database development. Microsoft Visio 2016 is used in designing this new system. Microsoft Office Visio is a Studio.Net Enterprise Architect Edition that is a forward and reverse engineering tool for databases and UML design and development. It supports data dictionary to accompany entity relationship template and also supports name, data type, required, primary and notes properties.

4.4.2 Database Design and Structure

Three database design phases are adopted in this new system database design and structure namely the conceptual data modeling, logical data model and Physical data model for database design. The conceptual data modeling, logical and physical data modeling phases for database design focus on the information content of the database and their relationship in order to achieve efficient implementation of the system.

- i. Conceptual Data Model:- The conceptual data model for this system recognizes the relationship levels between the database entities. Features of this data model include the important entities and the relationship among them. Attribute or relationship keys such as primary or foreign key are not specified in this model. The conceptual data model for this system is shown in Figure 4.2.

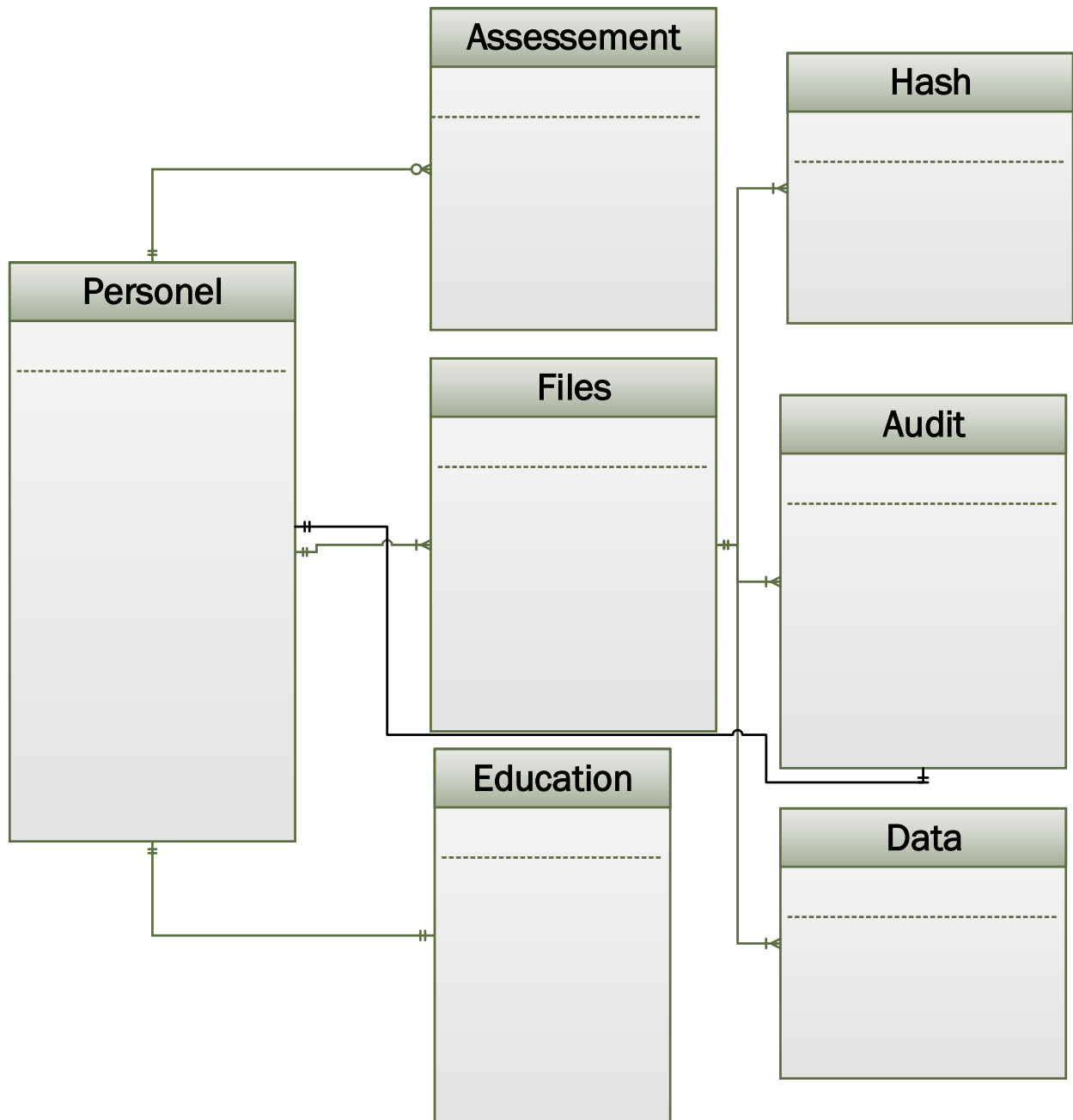


Figure 4.2: Conceptual Data Model showing relationship between entities

The rectangles (Assesement, Personnel, Hash, File, Audit, Education and Data) represent entity types while label lines shows relationship between entities.

- ii. Logical Data Model:- The logical data model for this new system describe the conceptual data model in details without regard to its physical implementation in the database. Identified features in this design include all entities and relationships among them, attributes for each entity are specified, primary and foreign keys are specified. To ensure that there is no redundancy; normalization is carried out in the table design constraints which entail dependencies among columns. Figure 4.3 shows the logical data model for this system.

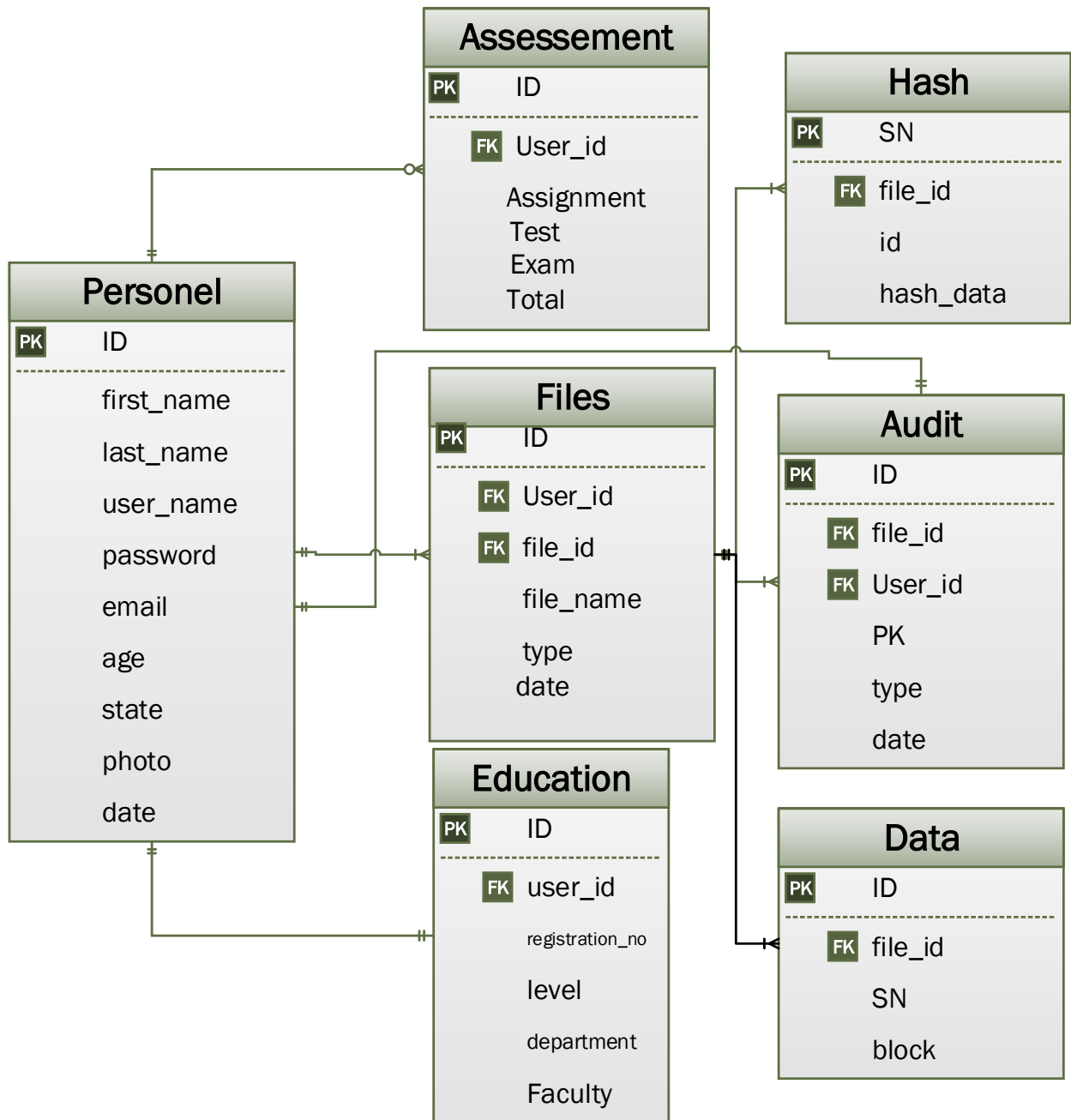


Figure 4.3: Logical data model showing attributes, primary and foreign keys for each entity

Attributes or properties of these entities are itemized inside the rectangle while the primary keys for each entity and foreign key that connects entities are also in the diagram.

iii. Physical Data Model:- The physical implementation of the data model in the database is described using physical data model. It shows table structures with column names and data types used for database implementation. Figure 4.4 shows the physical data model for this system.

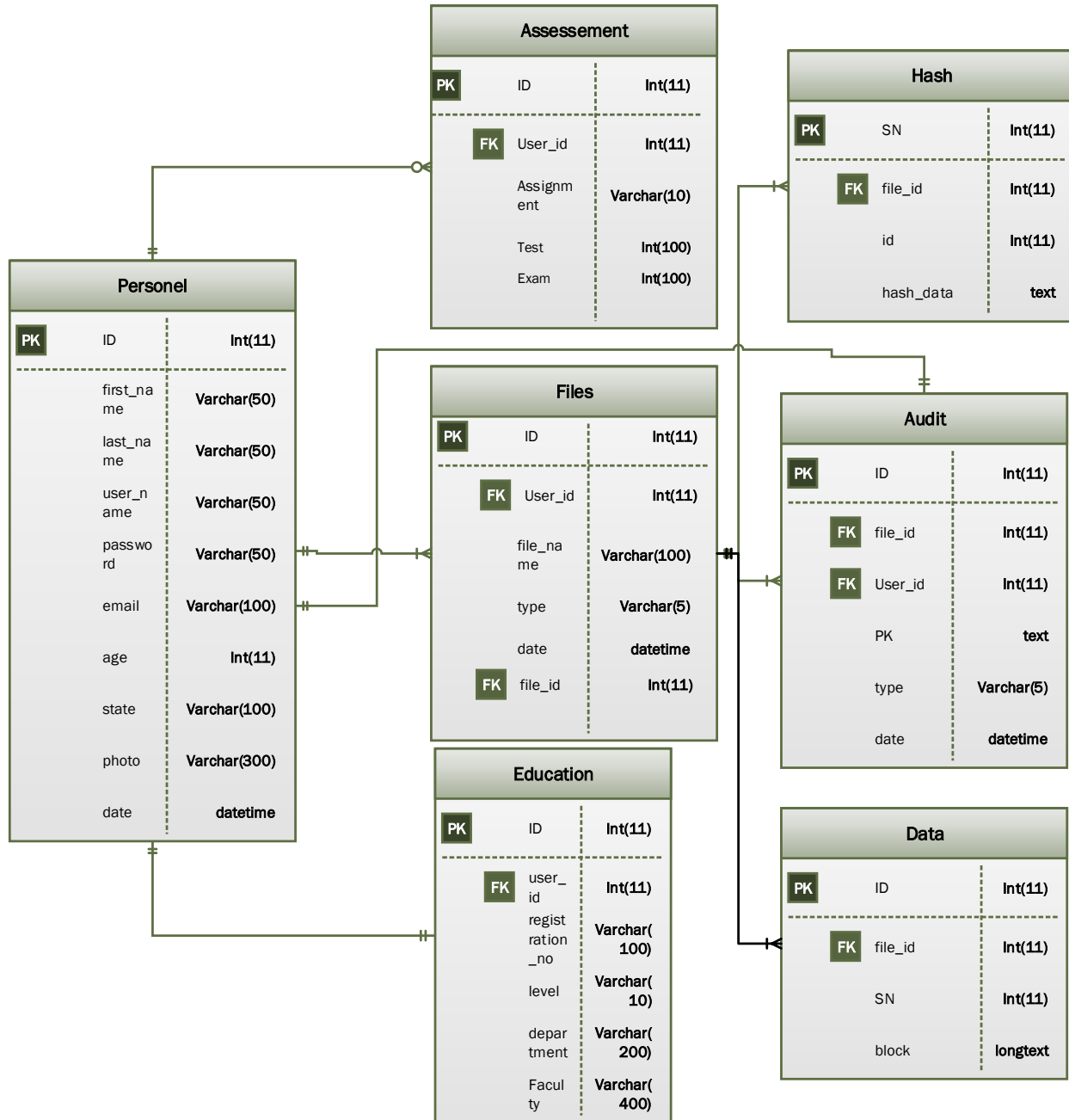


Figure 4.4: Physical data model showing data types

4.4.3 Program Module Specification

This new system basically has three main program modules namely the Admin module, the Auditor module and the Student module.

- a. The Admin Module: this module handles many functions such as:
 - i. Add Students: It handles students' enrollment into the system.
 - ii. Create Students Data/Score: It helps in creation of students' instant results.
 - iii. Upload Data: It helps to upload any existing result (data) to the system.
 - iv. Dynamic Data Auditing: This sub module function handles the audit of any legitimate file modification within the system and reports appropriately.
 - v. Adversary Mode: This function checkmates the authenticity of the audit functions carried out by the auditor module. It achieves its function by creating room for the generated root key to be alterable and checks the results of the altered data in return.
 - vi. View Students: This is a report function that displays the current students' assessments within the system.
 - vii. Change Login: This aspect of the module function gives opportunity for login credentials to be changed.
- b. The Auditor Module: This modules also handles many functions such as:
 - i. Check Data Authenticity: This is the main function of the auditor. This is equally the pivot of this research and when compromised, the essence is lost. This module handles the audition of the data outsourced to the cloud and reports appropriately to the DO. The module enables the auditor to achieve this function using the MHT algorithm with the security of the main root key using IRC6 encryption algorithm.
 - ii. Change Login: This aspect of the module function gives opportunity for login credentials to be changed as well for the auditor.
- c. The Student Module: This module of the program handles selection and display of respective students' activities within the system. Successfully registered individual student's data are viewed through this module.

4.4.4 Input/Output Format

The system is designed to accept several input details efficiently through input forms and user clicks as earlier stated. The data captured through the user keystrokes and clicks are received by specific modules on the system and relayed to the back-end of the system for processing. Input is collected using the following page modules:

- i. Add Students Sub Module;
- ii. Create Students Data/Scores Sub Module;
- iii. Upload Data Sub Module;
- iv. Dynamic Data Auditing Sub Module.

The Upload sub module is used to input data into the system. These data may be existing files in word, excel, text file and will be preprocessed into blocks of data as soon as it is inputted.

The output data format will be in two forms. The encrypted data after preprocessing will be in cypher text especially in auditor sub system due to security reasons while consumable output data will be in plain text form either in e-copy or printed copy, readable to individual users through the output sub module such as:

- i. View Student Sub Module
- ii. Adversary Mode Sub Module
- iii. Dynamic Data Auditing Sub Module
- iv. Check Data Authenticity Sub Module

4.4.5 Algorithm

The algorithm used to develop this new hybrid system is in two folds. First is the improved RC6 algorithm design which is used to secure the main auditing key as specified in the objectives of this research. The second is the new hybrid algorithm designed to achieve the aim of this research.

4.4.5.1 Improved Rivester Code Version 6 (IRC6) Cryptographic Technique Design

// Encryption/Decryption with IRC6-w/r/b

// Input: Plaintext stored in four w-bit input registers A, B, C & D

// r is the number of rounds

// w-bit round keys $S[0, \dots, 2r + 3]$

// Output: Ciphertext stored in A, B, C, D

```
// "Encryption Procedure:"
```

$$B = B + S[0]$$

$$D = D + S[1]$$

```
for k = 1 to r do
```

```
{
```

$$i = \text{hash}(\text{hash}(k))$$

$$t = (B * (2B + 1)) \lll \lg w$$

$$u = (D * (2D + 1)) \lll \lg w$$

$$A = ((A \oplus t) \lll u) + S[2i]$$

$$C = ((C \oplus u) \lll t) + S[2i + 1]$$

$$(A, B, C, D) = (B, C, D, A)$$

```
}
```

$$A = A + S[2r + 2]$$

$$C = C + S[2r + 3]$$

```
// "Decryption Procedure:"
```

$$C = C - S[2r + 3]$$

$$A = A - S[2r + 2]$$

```
for k = r downto 1 do
```

```
{
```

$$i = \text{hash}(\text{hash}(k))$$

$$(A, B, C, D) = (D, A, B, C)$$

$$u = (D * (2D + 1)) \lll \lg w$$

$$t = (B * (2B + 1)) \lll \lg w$$

$$C = ((C - S[2i + 1]) \ggg t) \oplus u$$

$$A = ((A - S[2i]) \ggg u) \oplus t$$

```
}
```

$$D = D - S[1]$$

$$B = B - S[0]$$

The strength of the improved version of the techniques is based on the double cryptographic hashing of the key which is denoted by “**key = hash(i)**” to sustain crypto-analytical attack on cypher text during transmission of data to the cloud.

4.4.5.2 The Enhanced Hybrid Auditing Scheme Using MHT and Improved RC6 (IRC6) Algorithm

The detail of the algorithm used for the new auditing scheme in this research using MHT and IRC6 is given below:

The methodology adopted uses a scheme in which the client file F is divided into blocks and compute signatures for each block of the file. Both the signatures and the blocks F are sent to the cloud server for storage. This is done using Merkle hash tree (MHT) technique.

The client File F is preprocessed using MHT and sent to the public cloud and the copy of the required signature is encrypted using IRC6 and sent to the private cloud for authentication process. During auditing process, the internal auditor who has a copy of the signature asks the public cloud to send required signatures of the blocks of file to be verified and verifies it by comparing the result of the computed signature from the cloud with local copy which it has and then reports the audit result to the data owner.

Algorithm 4.1 describes MHT file preprocessing steps through which files are being processed into blocks of data. The blocks are arranged in left-to-right order and are used to build immediate internal block. The process continues until the root node is obtained.

Algorithm 4.1: File Preprocessing

Open event <MHT Preprocessing>

Construct the binary tree structure of the supplied file in blocks of data

Organize the blocks in a left-to-right order

Hash each block of the leaf node

Concatenate each left and right node to obtain immediate internal node. $imm_int_node = Hash(L||R)$

Continue until the root node is formed

Keep each of the internal node and its position with respect to the root node which forms the AAI for the purpose of auditing.

End event

Algorithm 4.2 describes a computational algorithm called key generator KeyGen (.), is used in generating client's public key used by the internal auditor to access the cloud and private key used in IRC6 to encrypt the MHT root hash.

Algorithm 4.2: Key Generation

Open event < KeyGen(.) >

Generate a random signing key by:

{

Choose $\alpha \leftarrow Z_p$

Compute $v \leftarrow g^\alpha$

Compute $sk = (v\alpha)$ // key used to encrypt the RH

Compute $pk = (v^\alpha)$ // key used by internal auditor to access the cloud

]

Z is a source group of prime numbers of order P to a large chosen digit

g is a generator in a range of (2,..,p-2)

End event

Algorithm 4.3 is another computational algorithm called SigGen (.). By running the algorithm, the data file F is pre-processed, and the homomorphic authenticators together with metadata are produced.

Algorithm 4.3: Signature Generation

Open event < SigGen(.) >

Given $F = (m_1, m_2 \dots, m_n)$

Choose a random element $u \leftarrow G$

[

Where $u \in G$

```

]
Compute  $t = t = \text{filename} \parallel n \parallel u$  //  $t$  is file tag or id
Compute  $\sigma_i \leftarrow (H(m_i)um_i)^a$  //  $\sigma_i$  for each block  $m_i$ 
Compute  $\Phi = \{\sigma_i\}$  //  $\Phi$  is set of all the signatures from individual blocks
where
[
1 ≤ i ≤ n
]

```

End event

Algorithm 4.4 is used to bring in the improved RC6 (IRC6) techniques developed in this research to encrypt the root key of the adapted MHT. The secured RH and internal auditor's access key is sent to the auditor while the preprocessed file blocks and its corresponding hashes and signatures are sent to the cloud.

Algorithm 4.4: Generation and encryption of the Root Hash (RH)

Open event < GenEnc(RH) >

```

Call algorithm 4.1 to generate the RH
Encrypt the RH using IRC6 with  $sk$  as the encryption key
[
Sigs $sk$ (RH) = IRC6(RH)
]
Send {F, t,  $\Phi$ } to the server
Send {sigs $sk$ (RH),  $pk$ } to the internal auditor
Delete { F, t,  $\Phi$ , sigs $sk$ (RH)} from the local storage

```

End event

Algorithm 4.5 is used to handle data integrity request from internal auditor who is the verifier to the server who is the prover. The Internal Auditor verifies the integrity of outsourced data by challenging the server. Before challenging the server, the Internal Auditor first uses spk to verify

signature on t . If verification fails, reject by emitting FALSE else recover u . To generate the message “chal” The message chal specifies the positions of the blocks to be checked in the Merkle hash tree. The verifier sends the message chal $\{(i, c_i)\}$

Algorithm 4.5: Data Integrity Request

Open event < Challenge the server, Chal (.) >

 Verify signature on t using pk

 If (verification of t fails)

 [

 Reject the file by emitting FAILS

]

 Else

 [

 Recover u from t

]

 Pick a random c -element from c_i

 Where

 [

$c_i \in u, u \subseteq Z_p,$

$1 \leq i \leq n$

]

 Send Chal $\{(i, c_i)\}$ to the prover

End event

Algorithm 4.6 is used to generate proof by the prover as requested from the verifier after receiving the challenge chal = $\{(i, c_i)\}$ message. The server finds the corresponding or specified file in the cloud according to the challenge details. It is good to note that both the data blocks and the corresponding signature blocks are combined into single block at this point. The prover will

also send to the verifier the auxiliary authentication information (AAI) required to compute the root hash for the purpose of auditing.

Algorithm 4.6: Server Generates Proof

Open event $\langle \text{GenProof} (F, \Phi, \text{Chal}) \rangle$

 Upon receiving the Chal $\{(i, c_i)\}$

 Find the corresponding or specified file in the cloud according to the chal details

 Combine the data blocks and the corresponding signature blocks into single block as Proof P

 Send P to the Verifier

 Also send $\Omega = \text{AAI}$ required to compute the RH

End event

Algorithm 4.7 handles the verification of the received computed RH. If the verification fails, the Internal Auditor (Verifier) rejects by emitting False.

Algorithm 4.7: Data Authenticity Proof Verification

Open event $\langle \text{VerifyProof} (pk, \text{Chal}, P) \rangle$

 Upon receiving P from the prover

 Generate RH by computing

 {
 $H(m_i, \Omega_i)$ // $\Omega = \text{AAI}$
 $1 \leq i \leq n$
 }

 Authenticate the received and computed RH_c against the original computed RH

 If ($\text{RH}_c == \text{RH}$)

 {
 Inform the DO that the integrity of the data is undamaged.
 }

```

Else
{
    Reject by emitting FALSE and inform the DO
}
End event

```

4.4.6 Data Dictionary

This subsection of the research defines the most data objects of each user in the database and the entire system. This definition helps various users to know most of the objects which exist in the database and other parts of the system and who can access it.

Data dictionary is used to give detailed information about data elements used in designing the new system, their meanings and allowable values. It gives information about each attribute of a data model. Seven entities from our data model generate the data dictionary for this new system as depicted in Table 4.1 to Table 4.7.

Table 4.1: Assessment

Column Name	Data Type	Constraints (Keys)	Notes
ID	int(11)	Primary	It identifies individual assessment file with respect to its use and auditing.
user_id	int(11)	Foreign	It identifies each user to its assessment record
assignment	varchar(10)	Not Null	It keeps CA values
Test	varchar(100)	Not Null	It keeps CA values
Exam	varchar(100)	Not Null	It keeps exam values
Total	int(11)	Not Null	It keeps individual assessment records

Table 4.2: Audit

Column Name	Data Type	Constraints (Keys)	Notes
ID	int(11)	Primary	It keeps track of audit jobs for the system.
file_id	int(11)	Foreign	It identifies individual file/block for auditing.

user_id	int(11)	Foreign	It identifies each user to its audit record
Pk	text	Not Null	It keeps the secured root key
Type	varchar(5)	Not Null	It keeps track of file type been audited either uploaded or generated file
Date	datetime	Not Null	It keeps track of the time the audit is done

Table 4.3: Data

Column Name	Data Type	Constraints (Keys)	Notes
ID	int(11)	Primary	It keeps track of split blocks of files for the system
file_id	int(11)	Foreign	It identifies individual file block for storage and auditing.
Sn	int(11)	Not Null	It keeps track of block index
Block	longtext	Not Null	It keeps the actual encrypted blocks

Table 4.4: Education

Column Name	Data Type	Constraints (Keys)	Notes
ID	int(11)	Primary	It keeps track of education details files for the system
user_id	int(11)	Foreign	It identifies each user to its educational record
registration_no	varchar(100)	Not Null	It tracks students' registration number
Level	varchar(10)	Not Null	It tracks students' academic level
department	varchar(200)	Not Null	It tracks students' department
faculty	varchar(400)	Not Null	It tracks students' faculty

Table 4.5: File

Column Name	Data Type	Constraints (Keys)	Notes
ID	int(11)	Primary	It keeps track of the files before splitting into blocks for the system
user_id	int(11)	Foreign	It is a candidate key that identifies individual file and its blocks for storage and auditing.
filename	varchar(100)	Not Null	It keeps track of file names
Type	varchar(5)	Not Null	It keeps track of the type of files either uploaded or generated
Date	datetime	Not Null	It keeps track of time of operation for the files of the system

Table 4.6: Hash

Column Name	Data Type	Constraints (Keys)	Notes
SN	int(11)	Primary, Foreign	It keeps track of the files of the split blocks for the system
file_id	int(11)	Foreign	It is a candidate key that identifies individual block hash for storage and auditing.
Id	int(11)	Foreign	It maps each block to its hash as its index
hashdata	text	Not Null	It keeps track of the actual block hashes

Table 4.7: Personal

Column Name	Data Type	Constraints (Keys)	Notes
<i>ID</i>	int(11)	Primary	It keeps track of students' biodata file for the system
firstname	varchar(50)	Not Null	It keeps track of students' first name
lastname	varchar(50)	Not Null	It keeps track of students' last name
username	varchar(50)	Not Null	It keeps track of students' login identity
password	varchar(50)	Not Null	It keeps track of students' login password
Email	varchar(100)	Not Null	It keeps track of students' email address
Age	int(11)	Not Null	It keeps track of students' age
State	varchar(100)	Not Null	It keeps track of state of origin
photo	varchar(300)	Not Null	It keeps track of students' passport photograph
Date	datetime	Not Null	It keeps record of time operation is been performed on this file

4.5 System Flowchart

Figure 4.5 shows the system flowchart in which all the new system processes are depicted in graphical form for easy understanding and subsequent implementation. System flowchart symbols are adequately utilized with appropriate connectors, on-page connectors are used to show the connection between the Auditor and Admin login decision symbol and login timestamp of which the system uses to track multiple suspicious access attempts. Process symbols are used to depict all the processes while label symbols are used to label appropriate functions of the system. Internal and external storage symbols are used to depict both internal and cloud storage respectively. Audit path generation, data auditing and audit report generation are done by the auditor and the report is sent to the admin. Data capturing and preprocessing, key generation and management, dynamic data operation and system access control are handled by admin which is the DO. The timestamp logs any failed login attempt by thrice as an intruder. Access to the system is granted only with valid login parameters controlled by decision symbols.

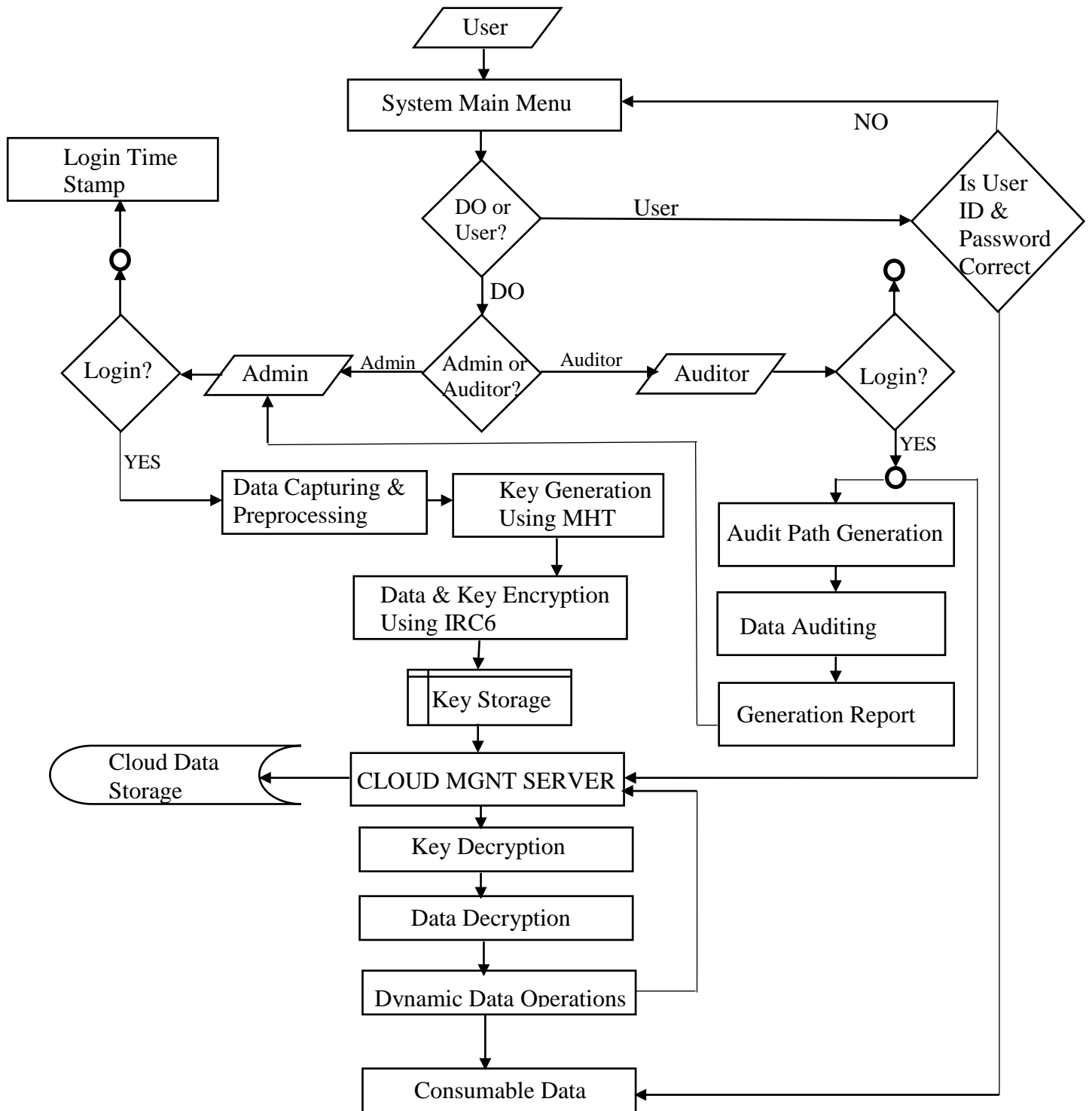


Figure 4.5: System Flowchart

4.6 System Implementation

The new system after proper analysis and design was implemented to ascertain the authenticity of the outsourced data to the cloud thereby restoring the integrity of cloud storage services, and bestow confidence to the prospective clients especially the Nigerian Federal Universities.

4.6.1 Proposed System Requirements

4.6.1.1 Hardware Requirements

Hardware	Specification
Processor Speed:	1.70 GHz and Above
Main Memory:	512 MB (minimum).
Hard Disk:	40 GB.
Disk Space:	100 MB and Above.
Keyboard:	ANY
Mouse:	ANY
Monitor:	ANY

4.6.1.2 Software Requirements

Software	Specification
Operating System:	Windows 8.1
Browser:	Firefox.
Software (Prog. languages used):	Java
IDE:	Nebeans
Administrative Software tool:	phpMyAdmin, and Apache IO collections.
DataBase:	MySql
Local Server:	XAMPP

4.6.2 Program Development

Various tools, procedures and methods required to build software, manage its complexity and its subsequent maintenance are discussed as program development. This will be considered under choice of programming environment and justification for the language used.

4.6.2.1 Choice of Programming Environment

NetBeans Integrated Development Environment (IDE) is used to develop this system together with Java programming language from Oracle Java Development Kit (JDK 1.7). Windows Software Development Kit (SDK) environment is also used to support the running of this new system which is a .NET Framework Software Development Kit (SDK) from Microsoft with useful components such as documentations, header files, libraries and tools required for developing this new system. To successfully run this application on Windows, Maven Protocol Buffers Plugin, which is a tool that helps generate Java source files from .proto which is a protocol buffer definition was also adapted.

4.6.2.2 Language Justification

The programming used to develop this new system is Java programming language because adapted MHT architecture was effectively implemented using java programming language. Also for effective improvement over RC6 core architecture and its integration with MHT to achieve the research objectives, Java Programming language is still the option. Though there are IDEs for Java development, NetBeans IDE supports scripting language like PHP and HTML5, which is key in the development of Web UI for this application. It also supports Maven and is cross-platform that runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM. Windows SDK was installed to help build windows native component (winutils.exe). Other notable advantages of Java language as choice for developing this system over other languages are:

- i. **Platform- Independent:** The key characteristic of java is its ability to build applications that can run on variety of computer systems (i.e. develop on one platform, run on any) which makes it platform independent.
- ii. **Simple:** Java is a simple, close to human language, elegant and easy programming language compared to other object oriented programming languages such as C++. Java replaced the complexity of multiple inheritances in C++ with a simple structure called interface. It also uses automatic memory allocation and garbage collection. It has a clean syntax which makes it easy to read, compile, debug, learn and write.
- iii. **Object oriented:** Java is object oriented because it is centred in creating objects, manipulating objects and making objects work together. Object oriented programming provides greater flexibility, modularity and reusability.

- iv. **Robust:** Java puts a lot of emphasis on early checking for possible errors. Java compilers provide several levels of additional checks to identify type mismatches and other inconsistencies. The Java runtime system duplicates many of the checks performed by the compiler and performs additional checks to verify that the executable byte-codes form a valid Java program.
- v. **Secure:** The Java language compatible browsers, compiler, and interpreter all contain several levels of security measures that are designed to reduce the risk of security compromise, loss of data and program integrity, and damage to system users. Java platform allows users to download insecure code over a network and run it in a secure environment without it causing any damage.
- vi. **Portable:** Java can run on any platform without having to be recompiled. The Java environment is portable to new hardware and operating systems. It ensures that there are no platform-specific features on the java language specification. For example the size of integer in Java is the same on every platform, as is the behaviour of arithmetic.
- vii. **Dynamic:** In Java, new methods and properties can be added freely in a class without it affecting their clients. Java can load classes as needed at runtime i.e. An application can decide as it is running what classes it needs and can load the when it needs them.

4.6.3 System Testing:

4.6.3.1 Test Plan:

Introduction: This section of the research will describe the scope, approach, resources and schedule of intended test activities. It will identifies amongst others test items, the features to be tested, the testing tasks, which human resource will do each task, extent of tester independence, the test environment etc. it is a record of the test planning process.

- i. **Test Item:** Hybrid Model for Remote Dynamic Data Auditing Application Software version 1.0.
- ii. **Features to be tested:** Output display through Student module and View Students sub-module in Admin module, data capturing through Add student module, Create Student Data/Scores module, and Upload Data module. Dynamic Data Auditing functionality, Adversary mode. Check Data Authenticity in Auditor Module. Login and Logout features, and Change login features.

iii. **Approach:** The method adopted in this test plan include master test plan in which some modules of the new system to be tested constitutes the major functions of the system. White box testing type is equally adopted which specifies that the internal modules of the new system are working according to specification.

Table 4.8: Item Pass/Fail Criteria:

Items	Pass Criteria	Fail Criteria
Output display through student module	Decrypted data shown properly	If blank
View Students sub-module in Admin module	Decrypted data shown properly	If blank
Data capturing through Add student module	If the captured data uploaded successful to the its database	Data not seen in the database
Create student data/scores module	If the captured data uploaded successful to its database	Data not seen in the database
Upload data module	If the preprocessed data file uploaded successfully	File Id not seen in the database
Dynamic data auditing module	Modifications on uploaded file authenticates correctly	Modifications on uploaded file fails to authenticate correctly
Adversary mode	Data authentication fails due to alteration of the original root key	Data authentication go successfully even with alteration of the original root key
Check data authenticity	Authenticate successfully with correct root key	Fails to authenticate if the root key is invalid
Login and Logout features	Grant access to the new system if the valid login parameters is supplied	Deny access to the new system if the invalid login parameters is supplied
Change login parameter	Grant opportunity to the change existing login parameters only if old valid login parameters is supplied	Deny or grant opportunity to the change existing login parameters if old invalid login parameters is supplied

Cryptosystem	If the main root key in the audit panel is displayed in cypher text	If the main root key in the audit panel is displayed in plain text
MHT preprocessing module	If the file content is displayed in blocks of data in the audit panel.	If the file content is not displayed in blocks of data in the audit panel.

Test Environment: The new system will be tested locally using phpMyAdmin version 5.4.1, XAMPP server version 3.2.2, MySql database, Window 8.1 Operating System running on Core i7 Dell Inspiron 15R with one terabyte of internal storage and four gigabyte RAM memory.

4.6.3.2 Test Data:

This is essentially the input data given to this new system. It represents data that spurs the system to function according to specification. Some data may be used for positive testing, typically to verify that a given set of input to a given function produces an expected result.

Selected test data is a word document with filename called “Eddy.docx” shown in table 4.9.

Table 4.9: Test Data (Eddy.docx)

1.1 Background of the Study

Database Creation and Management is a broad research area in the field of Computer Science which include Cloud Data Storage and Management. Cloud computing is a service delivery model whereby shared resources such as hardware, software, platforms, and information are provided to consumers electronically as a utility over an internet network (Wang *et al*, 2009). Several trends are opening up the era of cloud computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the mainstream computing architectures such as Software-as-a-Service (SaaS) Platform-as-a Service (PaaS) and Infrastructure-as-a-Service (IaaS), are transforming data centers into pools of computing service on a huge scale (Ben, 2011). The increasing network bandwidth and reliable, yet flexible network connections make it even possible that users can now subscribe for high quality services from data and software that reside solely on remote data centers. Moving data into the cloud offers great convenience to users since they do not have to care about the complexities of direct hardware management. Examples of such well known services include Amazon Simple Storage Service (S3), and Amazon Elastic Compute Cloud (EC2). While these internet-based online services do provide huge amounts of storage space and customizable computing resources, these computing platform shifts, however, are eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are at the mercy of their cloud service providers (CSP) for the availability and integrity of their data (Wang *et al*, 2009).

This research focuses on sensitive data generated from organizations globally such as IT industries, banks, private and cooperate organizations, and high institutions with Federal University Wukari (FUW) as a case study. Academic data for example which are the life wire of such organizations and other sensitive records such as students and staff records are being generated from different departments and units to be stored and managed internally. Due to some known risks posed to these data such as domestic accidents like fire and liquid hazards, conflicts of interest, political intent, and others, it has become so important for this research to propose outsourcing of these data to the cloud where it will be devoid of these risks.

4.6.3.3 Actual Test Result Versus Expected Test Result:

- i. Output Display: Figure 4.6 shows an expected display result of the chosen test data before preprocessing and outsourcing while figure 4.7 shows the actual display test result of the chosen test data after reprocessing and decryption.

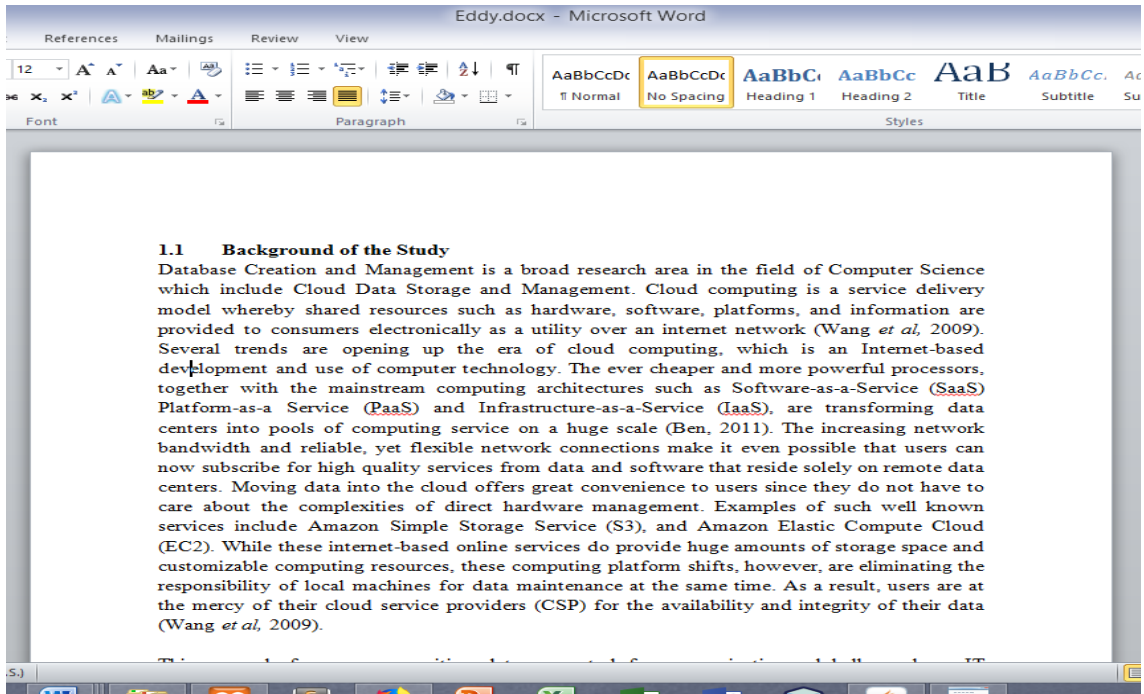


Figure 4.6: Expected Test Result

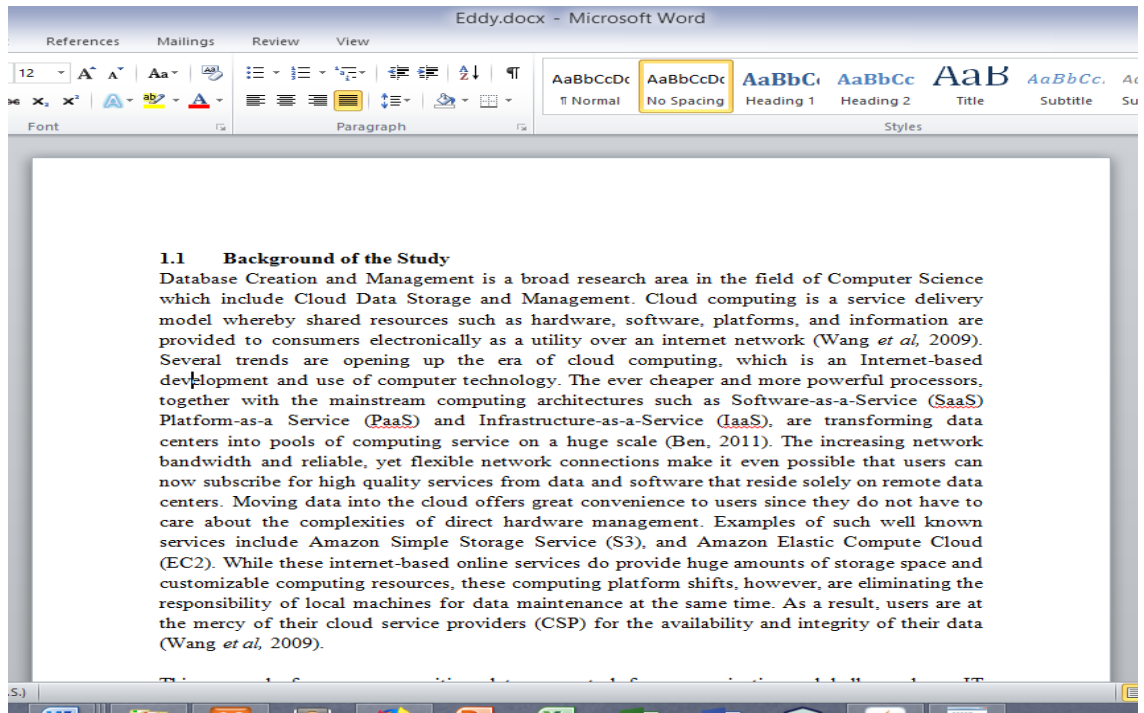


Figure 4.7: Actual Test Result

ii. View Students sub-module in Admin module: Figure 4.8 shows the actual display test result of the decrypted students' data which is also the same as the expected display test result.

S/N	FIRSTNAME	LASTNAME	ASSIGNMENT	TEST	EXAMS	TOTAL	DEPARTMENT	FACULTY
1	Emenike	Celine	5	18	62	85	Computer Science	Information and Communication Technology
2	Agu	David	8	19	67	94	Computer Science	Information and Communication Technology
3	Agu	Daniella	7	18	53	78	Computer Science	Information and Communication Technology
4	Agu	Daniel	8	17	62	87	Computer Science	Information and Communication Technology

Figure 4.8: Actual Test Result for View Students sub-module in Admin module

iii. Data Upload Module: Figure 4.9 shows the filename or the ID of the chosen test data file as stipulated in the data plan pass condition. This is the actual test result of the data upload module of the new system which is the same as the expected result.

ID	Filename	Action
1	Eddy.txt	Download
2	Elugwu P U Exco.txt	Download
3	Eddy.docx	Download

Figure 4.9: Actual Test Result for Data Upload Module

iv. Remote Data Auditing (Check data authenticity): This is the main function of this new system. The evaluation of this module is in two folds.

First is to check if the Auxiliary Authentication Information (AAI) or the audit path supplied from the cloud for the chosen test data corresponds to the expected AAI according to the MHT algorithm used.

Expected Test Result: If the chosen audit block is '8th block', the expected AAI is shown in figure 4.10.

Actual Test Result: Figure 4.10 shows the actual test result for the chosen audit block and its corresponding AAI.

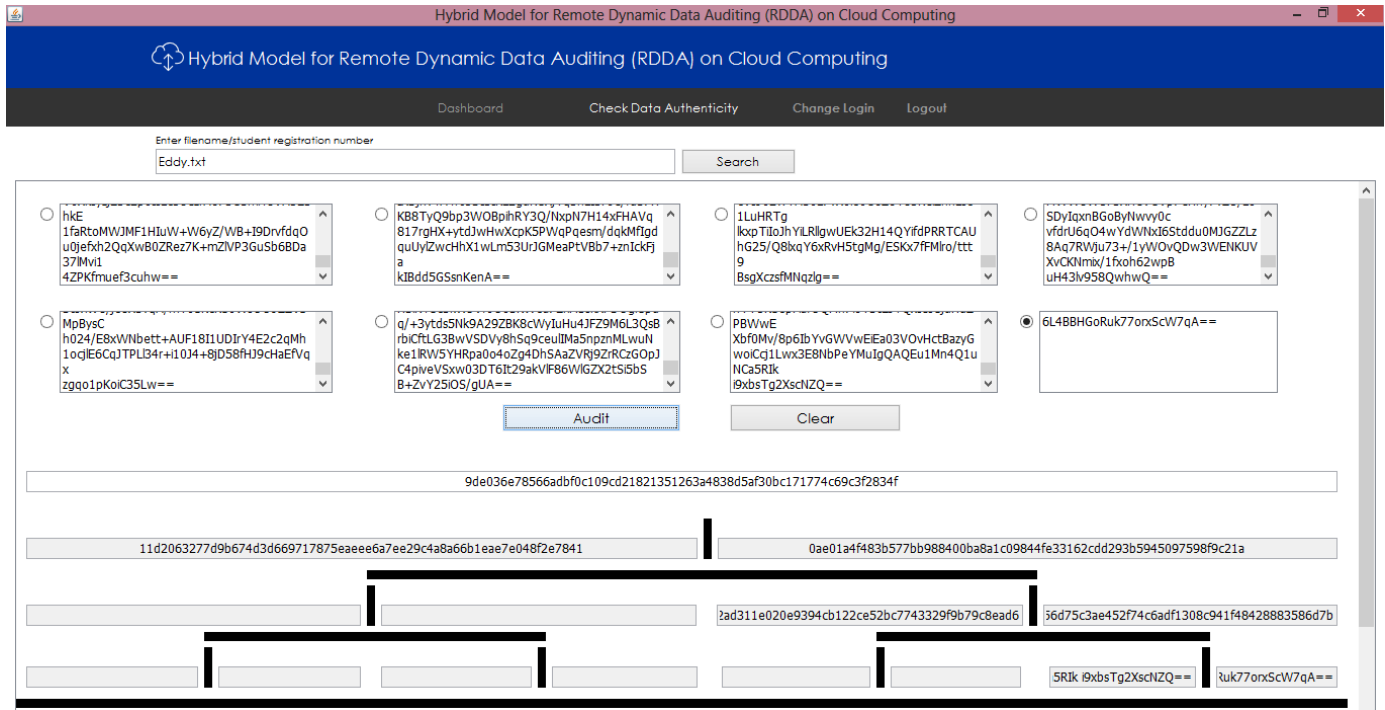
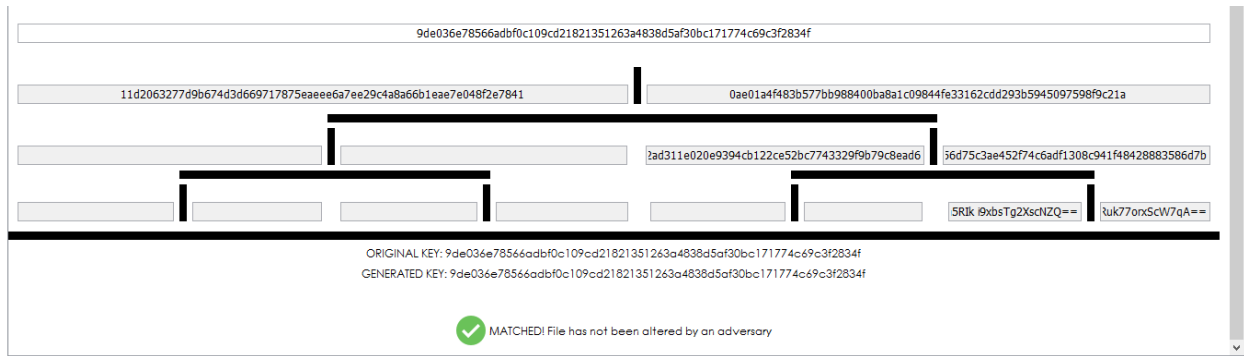


Figure 4.10: Actual Test Result for the Chosen Audit block and its Corresponding AAI.

The second aspect of this module evaluation is to check the authenticity of the remote data with respect to its IRC6 encrypted root key. The secured root will be generated using the AAI after which the data auditing will be carried out. The generated root key for the chosen test data is shown in figure 4.11. Figure 4.11 also shows the actual result of the remote data audit which is the same as the expected result, by showing the match result between the original key and the generated key.



ORIGINAL KEY: 9de036e78566adbf0c109cd21821351263a4838d5af30bc171774c69c3f2834f
GENERATED KEY: 9de036e78566adbf0c109cd21821351263a4838d5af30bc171774c69c3f2834f

✓ MATCHED! File has not been altered by an adversary

Figure 4.11: Actual Result of the Remote Data Audit

v. Dynamic Remote Data Auditing (Check Modified Data Authenticity):

Dynamic data integrity check achieved in this new system deals the capability of this new system to authenticate legitimately modified file after processing and decrypting it back to plain text. This is achieved through the download buttons implemented in the outsourced file menu which decrypts any selected file to its original content on issuing a download request. Figure 4.12 shows the actual test result obtained after encrypting and decrypting the modified file, which is still the same as the expected result.

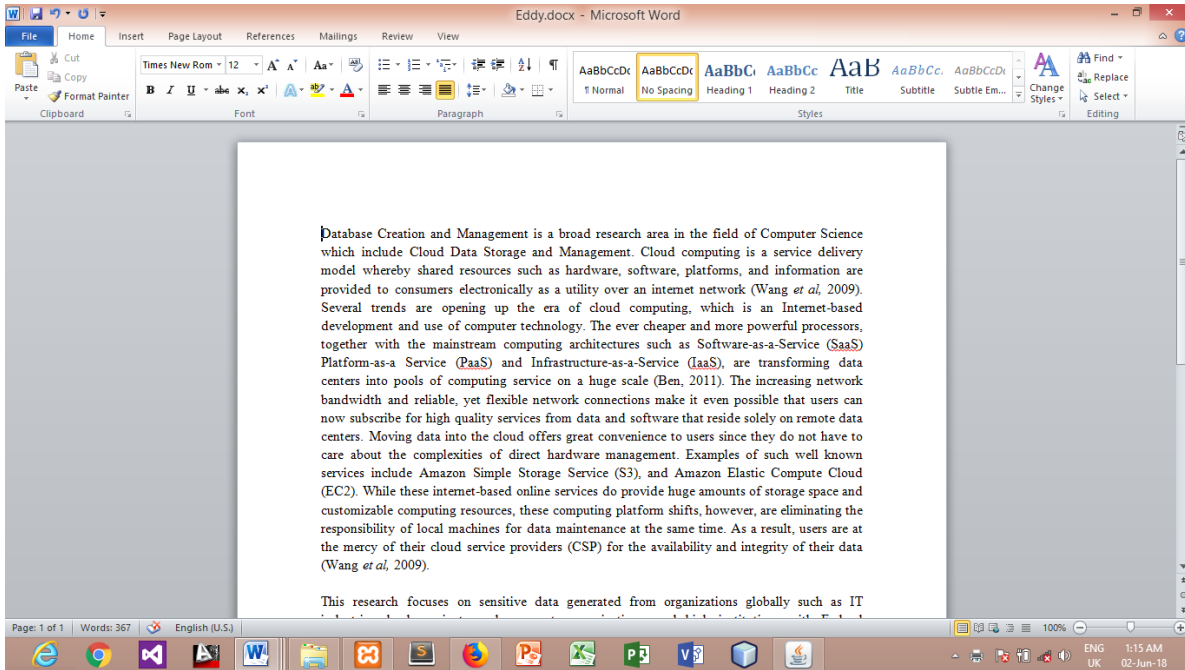


Figure 4.12 Actual Dynamic Audit Test Result Obtained after Decryption

Figure 4.13 also shows actual test result obtained after updating a block of data during dynamic operation.

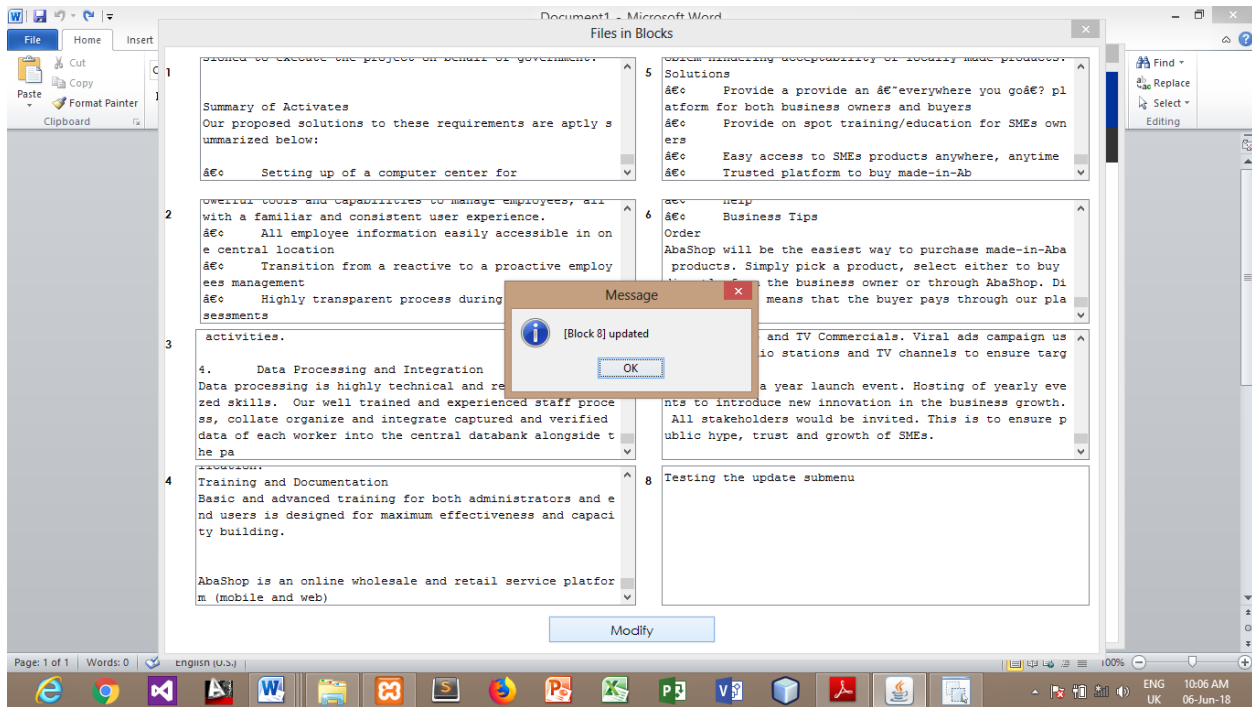


Figure 4.13 Actual Dynamic Block Update Test Result

vi. Adversary mode:

Figure 4.14 shows the actual test result of adversary mode operation obtained in which remote data authentication fails due to alteration in the root key. This test result is the same as the expected result of this module.

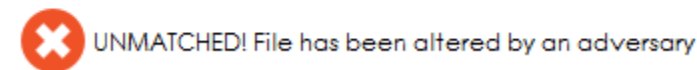
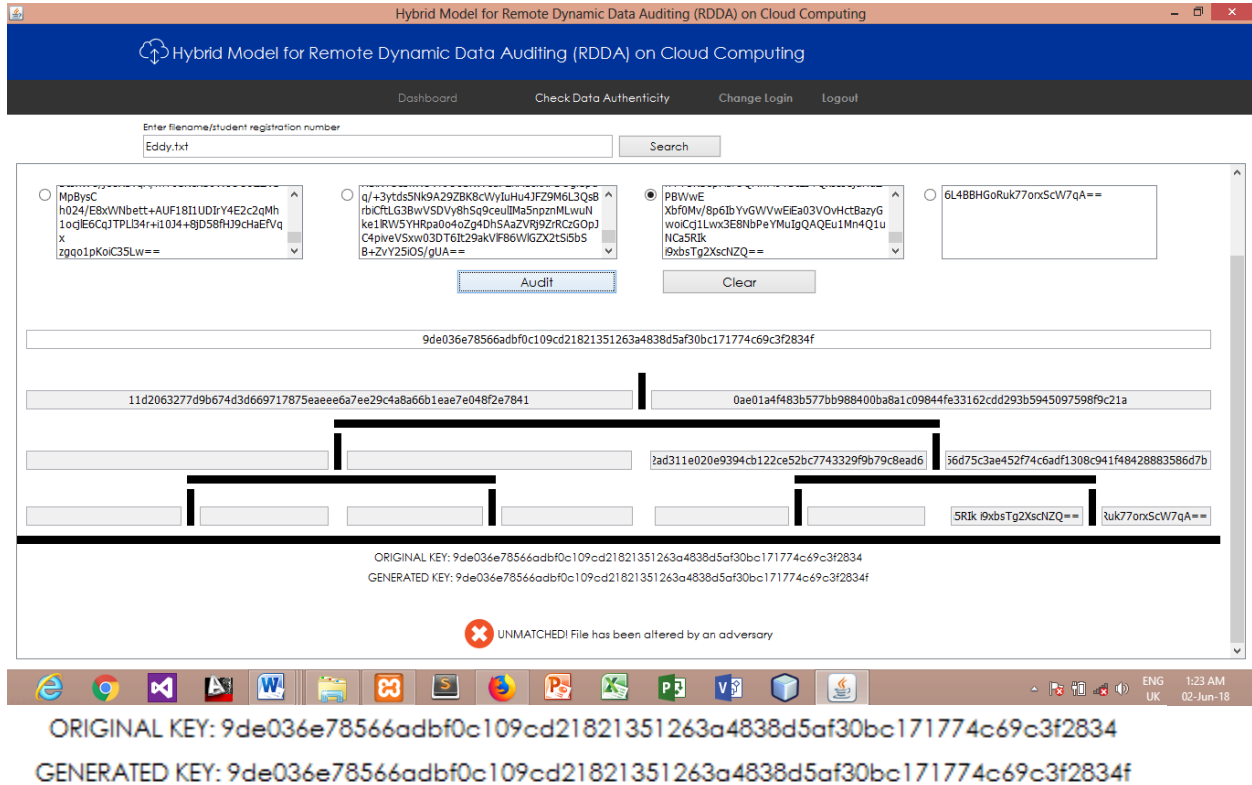


Figure 4.14 Actual Test Result of Adversary Mode Operation

vii. MHT preprocessing module:

Figure 4.15 shows the actual result of data file been preprocessed into data blocks according MHT algorithm. The expected result is also the same as the actual result obtained.

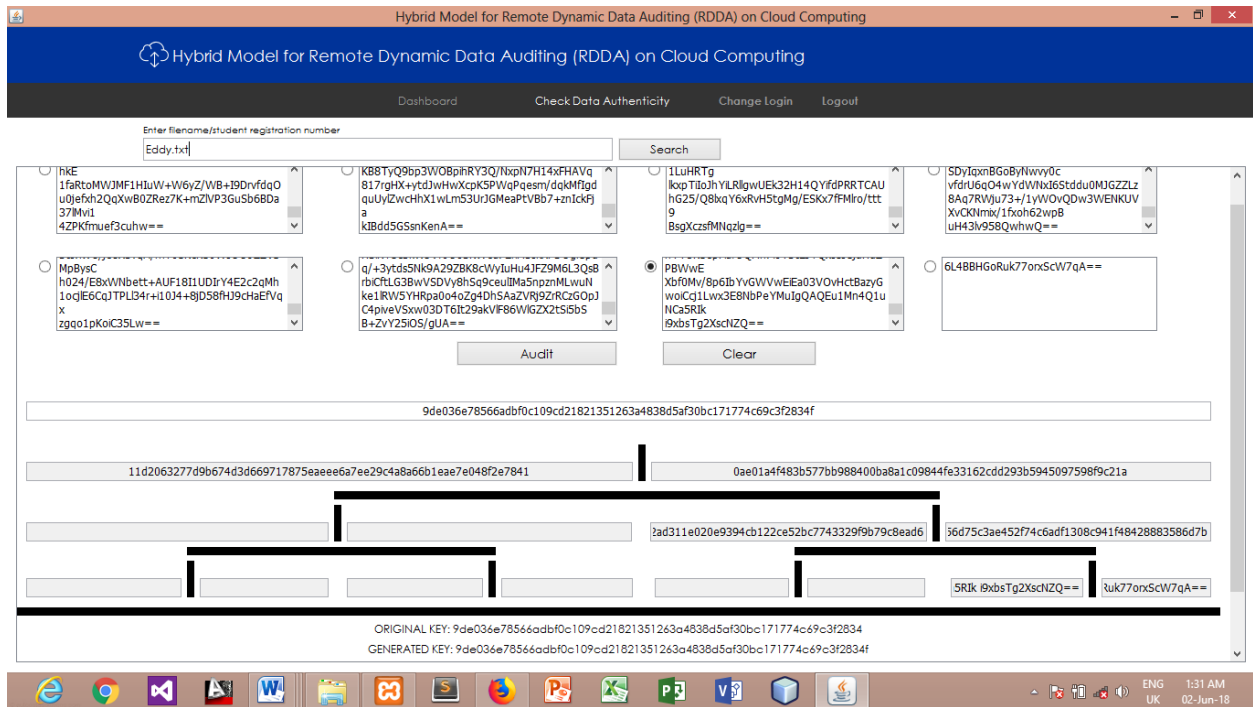


Figure 4.15 Actual Result of MHT Preprocessed Data Blocks

viii. IRC6 Cryptosystem:

Figure 4.16 shows that the main root key is encrypted using the IRC6, and it is masked against the auditor. Actual result of the encrypted root key which is the main audit parameter is the same as the expected result which is shown in figure 4.16.

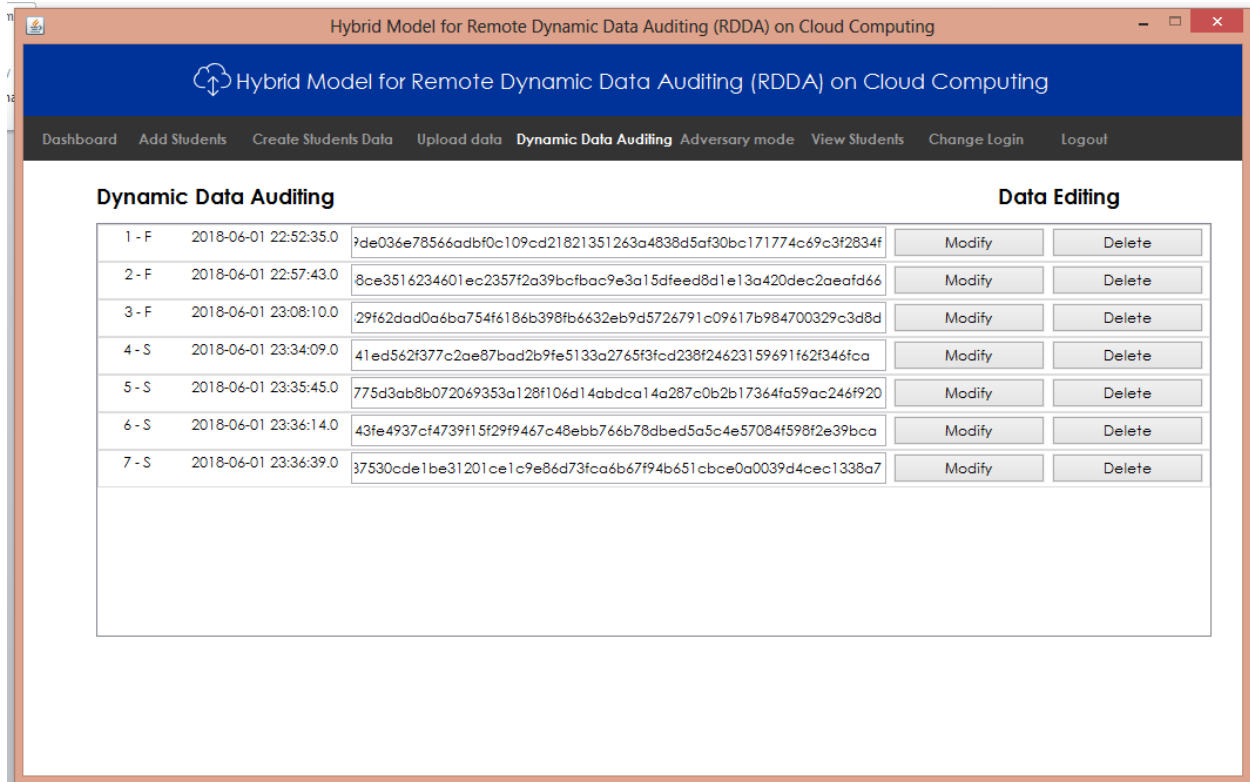


Figure 4.16 Actual Result of the Encrypted Root Key

ix. Login and Logout features:

Figure 4.17 and figure 4.18 show the actual test result of the login and logout features of the new system respectively.

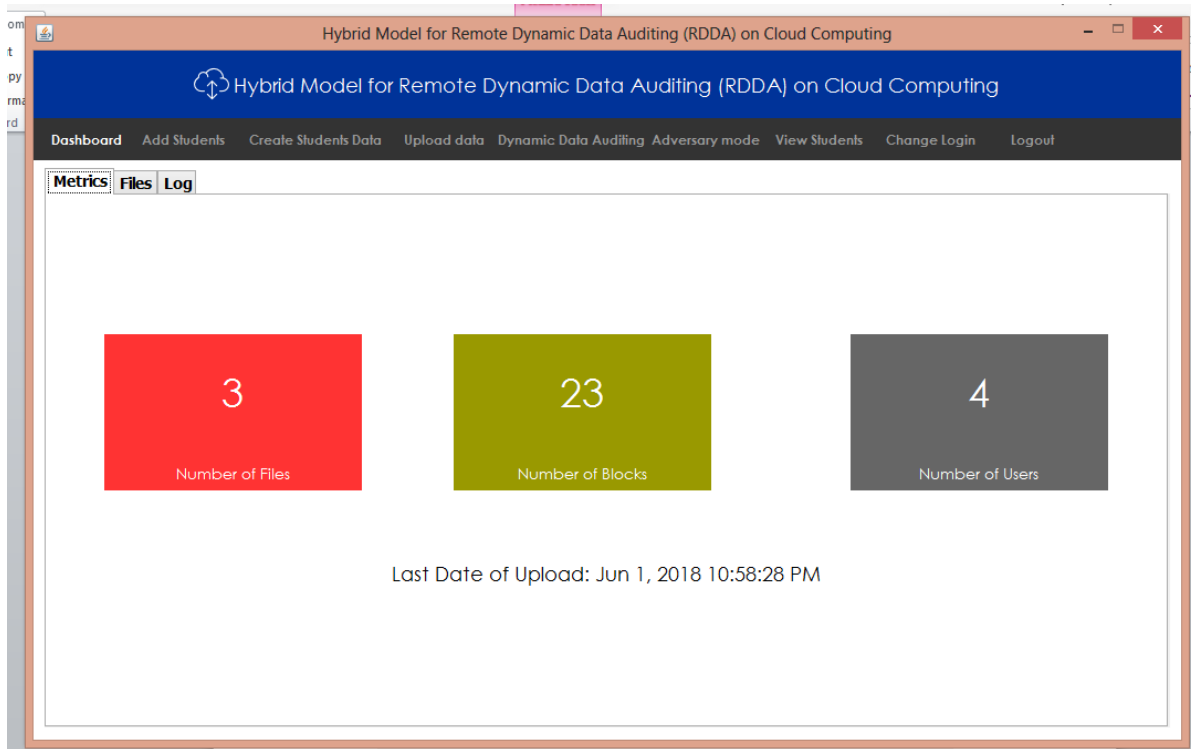


Figure 4.17: Actual Test Result of the Login Feature

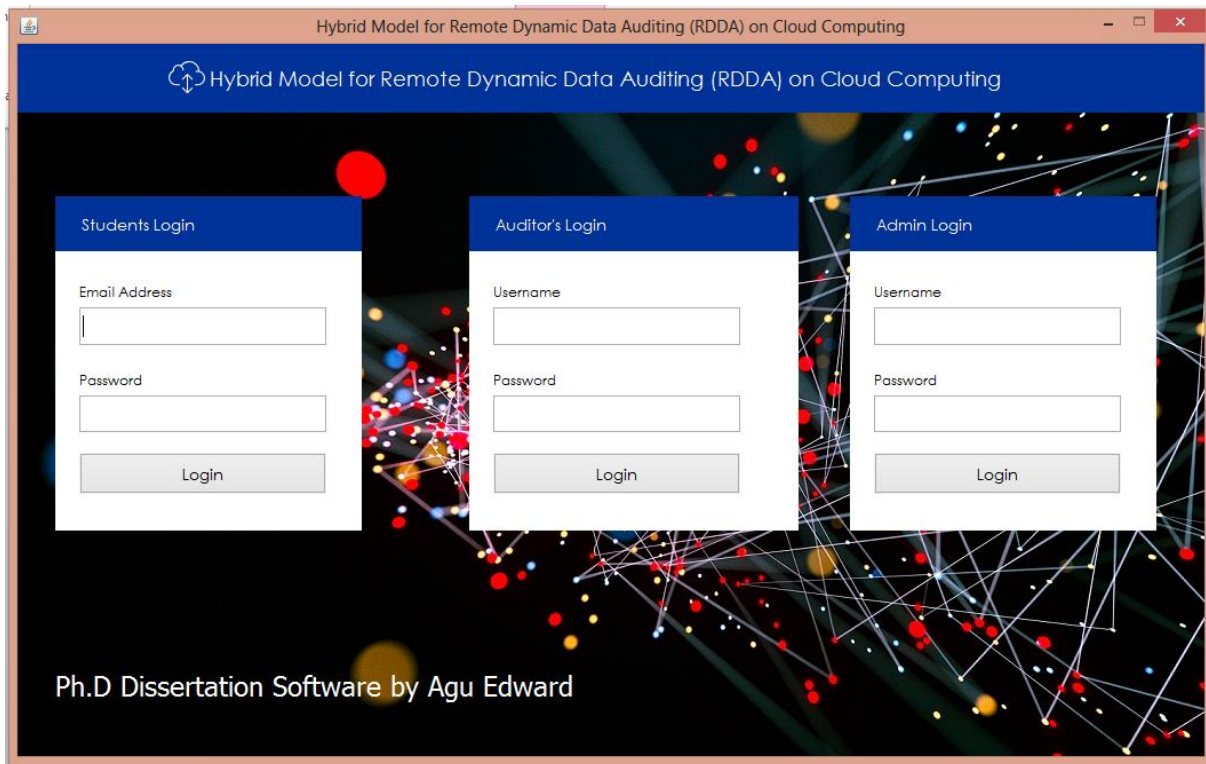


Figure 4.18: Actual Test Result of the Logout Feature

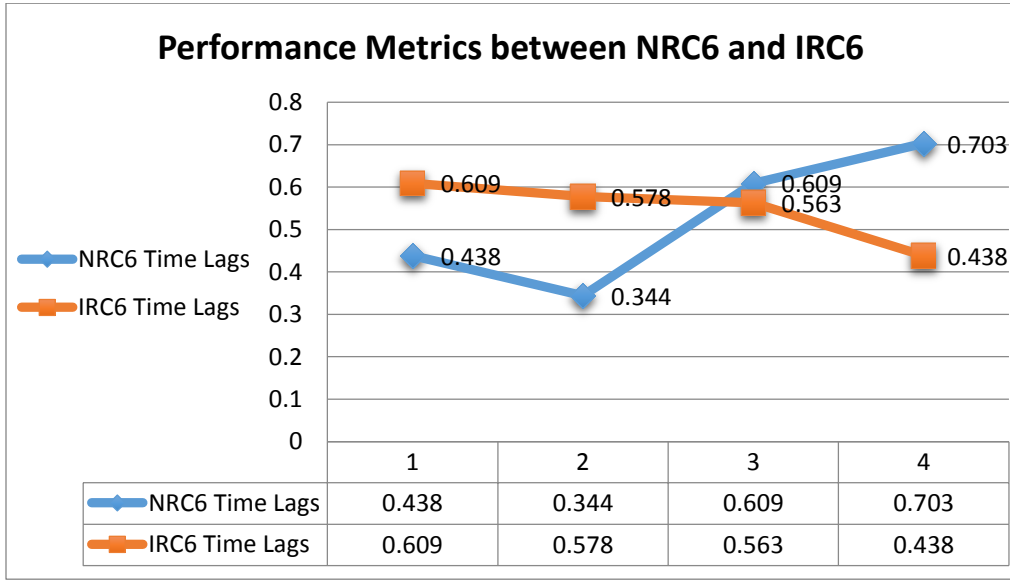


Figure 4.19: Performance Metrics between NRC6 and IRC6

Again, the encrypted cipher of IRC6 is quite different from that of NRC6. It is not transparent and survives crypto-analytical attacks during data transmission.

A. Comparison between the new method and other notable RDA protocols

The techniques adopted and implemented in this research is based on a random sampling strategy to reduce the workload on the CSP. The input file (F) is divided into numerous blocks (m) by the sampling techniques and randomly selects a number of challenge blocks (c) to perform batch processing.

Evaluating the experimental result, Table 4.11 shows a comparison between the new method and other notable RDA protocols based on three important parameters which includes; Communication Cost, Computation Cost of data auditing, and Computation Cost of dynamic data update:

- i. Communication cost: the feature of communication cost shows the amount of data transfer between the Data Owner, the Auditor and the Server in different phases of the auditing scheme.
- ii. Computation cost of data auditing: each step of the data auditing method is responsible for performing a specific task, and this puts some computational weight on the auditor or server. From the verifier’s point of view, computational cost of data auditing indicates the computational

resources that are used by the auditor to verify the integrity of the outsourced data. From the server's point of view, the computational cost of data auditing indicates the required time to process and generate the proof message in the response step.

iii. Computation cost of dynamic data update: dynamic data auditing methods allow data owners to update the outsourced data by using insert, delete, append, and modify operations. During dynamic data update operation, the data owner needs to accomplish some tasks such as finding the location of a requested block, generating new tag, and re-balancing the applied data structure based on the update operation. As a result, the required time to execute the update operations, such as insert, delete, append, and modify is called the computation cost of dynamic data update. In the reviewed work of Wang *et al* (2011) scheme, the maximum computational overhead is incurred during dynamic data update because the MHT data structure is used to check the integrity or perform the update operations on the outsourced data blocks. Also in the reviewed work of the Yang *et al* (2012) scheme, it improves the previous works of Wang *et al* (2011) to $O(c)$ with respect to modifying and appending a block. But to insert a block after i or delete a specific block ($f[i]$), the verifier must shift $(m-i)$ entities in the data structure. As a result of this, the computational overhead of this method during the insert and delete operations is $O(m)$. To address this problem and improve the auditing scheme, Sookhak *et al* (2015) designed a new Divide and Conquer Table data structure (DCT) to reduce computational overhead. To insert or delete a data block using their scheme, the verifier has to shift a part of the outsourced data blocks $(n/k - i)$ that incurs $O(n/k)$ computational overhead on the verifier. It is important to mention that to find a block ($f[i]$) in the DCT structure, the verifier has to divide the location of a block to k and find the appropriate DCT which also incurs computational overhead on the verifier though less when compare to the earlier scheme. To reduce this dynamic computational overhead cost on server to a minimum point, we implemented a system that will support offline-online dynamic update using MHT techniques. In the implemented scheme, the specific signature of block and the encrypted block to be updated is located, retrieved from the server and decrypted after which the update is affected offline without any cost on the server, after which it will be uploaded to the server.

Table 4.11 shows performance analysis comparison of data auditing schemes by Wang *et al*, Yang *et al* (2012), Sookhak *et al* (2015) and our new scheme, where m is the number of blocks, n is the number sectors of a block, c is the number of challenge blocks in each auditing query,

and k is the number of the DCTs structure as applied in Sookhak scheme. c_p in our scheme indicates that the update is applied only to the selected challenged blocks offline with limited or no cost.

Table 4.11: Performance comparison of different RDA schemes

Metric	Schemes				
		Wang <i>et al</i> (2011)	Yang <i>et al</i> (2012)	Sookhak <i>et al</i> (2015)	Our Scheme
Communication		$O(c \log m)$	$O(c)$	$O(c)$	$O(c_p)$
Computation Auditing	Server	$O(c \log m)$	$O(cn)$	$O(cn)$	$O(c_p)$
	Verifier	$O(c \log m)$	$O(c)$	$O(c)$	$O(c_p)$
Computation Modify	Verifier	$O(c \log m)$	$O(c)$	$O(c)$	$O(c_p)$
Computation Insert	Verifier	$O(c \log m)$	$O(m)$	$O(m/k)$	$O(c_p)$
Computation Delete	Verifier	$O(c \log m)$	$O(m)$	$O(m/k)$	$O(c_p)$
Computation Append	Verifier	$O(c \log m)$	$O(c)$	$O(c)$	$O(c_p)$

B. Algorithm to show the $O(C_p)$ Computation

1. Finding C_p (F, n, k)
2. Begin
3. Selected_block = block (n)
4. $K=1$
5. Repeat
6. Choose corresponding blocks
7. Selected_block = block($n - 1$)
8. $N = n - 1$
9. $K = k + 1$
10. Until ($k = 3$)
11. End

Consider statement line (7) in the above algorithm. The selection process is backward substitution. Also, statement line (3) is executed once before the loop, the time complexity is $O(1)$.

Therefore,

$$C_p(n) = 1 + C_p(n - 1) \quad (1)$$

For the second iteration

$$C_p(n - 1) = 1 + C_p(n - 2) \quad (2) \quad \text{[statement line (8) shows that } n \text{ decreases for each iteration]}$$

Third iteration

$$C_p(n - 2) = 1 + C_p(n - 3) \quad (3)$$

Combining equation (1) and (2)

$$\begin{aligned} C_p(n) &= 1 + 1 + C_p(n - 2), && \text{remember that } C_p(n - 1) = 1 + C_p(n - 2) \\ &= 2 + C_p(n - 2) && (4) \end{aligned}$$

Combining equation (3) and (4)

$$\begin{aligned} &= 2 + 1 + C_p(n - 3) \\ &= 3 + C_p(n - 3) \\ &= k + C_p(n - k) \end{aligned}$$

But

$$C_p(n - k) = 1 + C_p(n - k - 1)$$

Hence

$$\begin{aligned} n - k &= 1 + n - k - 1 \\ n - k &= 1 + 8 - 3 - 1 \\ n - k &= 5 \\ 8 - k &= 5 \end{aligned}$$

$$K = 3$$

Hence

$$\begin{aligned} k &= (3) \\ &= O(k) \\ &= O(Cp) \\ &= O(3) \end{aligned}$$

Therefore, the general time slice spent per block retrieval is $O(1)$. Whereby the previous model $O(C) = O(n/2)$ which implies their $k = 4$ while our $k = 3$, shows there is improvement over our model.

4.6.3.5 Limitations of the System:

The limitation observed in this research system includes a little increase in communication overhead cost during initial data upload.

4.6.4 System Security:

The security of this system encompasses the processes and procedures involve in keeping the developed system itself, its data and information it contains confidential, available, and assuring its integrity. It also deals with:

4.6.4.1 Password Protection

- i. Access controls, which prevent unauthorized personnel from entering or accessing the system using strong password security.
- ii. Unauthorized access to the system is guard against with the implementation of strong Triangular Security Handshake (TSH) between the user, the auditor and cloud service provider using strong password security.

4.6.4.2 Authentication

The integrity of the outsourced data to the cloud is equally periodically authenticated using MHT.

4.6.4.3 Cryptography

- i. Outsourced data blocks from the system are equally protected using cryptosystem against any form of network attack as it is being transmitted across the network medium.

- ii. The confidentiality and integrity is further achieved by masking the auditor from the real audit token which is encrypted using the developed IRC6.

In securing the new developed system, an adversary data authentication model was development and used to ascertain the effectiveness of the developed model. An intrusion detection mechanism was equally implementation through access log for access control. Infrastructural security measures such as setting up and maintaining firewalls and antivirus engines are also put in place. The system security also includes discovery of security breaches, as well as their proper documentation.

4.6.5 Training:

The Training will empower the end-users in terms of the knowledge, skills, and/or abilities required to support the new roles and/or technology. It will ensure that all impacted staff receives relevant training to prepare them for any new working practices. Training method to be adopted in this research is Blended Training Approach which includes Instructor-Led training approach and Job-aids training approach. This method of training enhances learners' retentive memory or retention of learning. It is obvious that blend of training delivery methods will best meet the needs of our project.

It is recommended that there will be one Training lead from the project team, and one from the deployment firm. They will be responsible for completing and managing the training program, including the development of instructional materials and training delivery.

The following section describes the distinct training environments:

- i. Training Development Environment: will be used for creating training materials; this environment is for the exclusive use of the project team.
- ii. Training Production Environment: will be used to deliver Instructor-Led Classroom Training
- iii. Training Practice Environment: will be used by end-users to practice in the new system; concurrently with the deployment of e-learning.

To evaluate the effectiveness of training delivery, information will be obtained from the following areas:

- i. The outcomes of competency tests completed by trainees at the end of each module.
- ii. Feedback from trainees on confidence level at the end of each module.

- iii. Feedback from trainers on training problems or individuals with who have experienced learning difficulties.

4.6.6 Documentation:

This new system allows users a simple way to protect and manage the outsourcing of their internal generated data/information from a remote device. This documentation describes the instructions for installing and accessing the system.

The local installation of this newly developed application is done by installing and configuring a XAMP version 3.2.2 local server and it is running on MySQL database, with Netbeans version 8.2 IDE.

To open the system on a computer, first step is to turn on the local server. Next step is to launch the system interface by double clicking the application icon. Access to the system is granted to only authorize users through the login main menu as shown in figure 4.20.

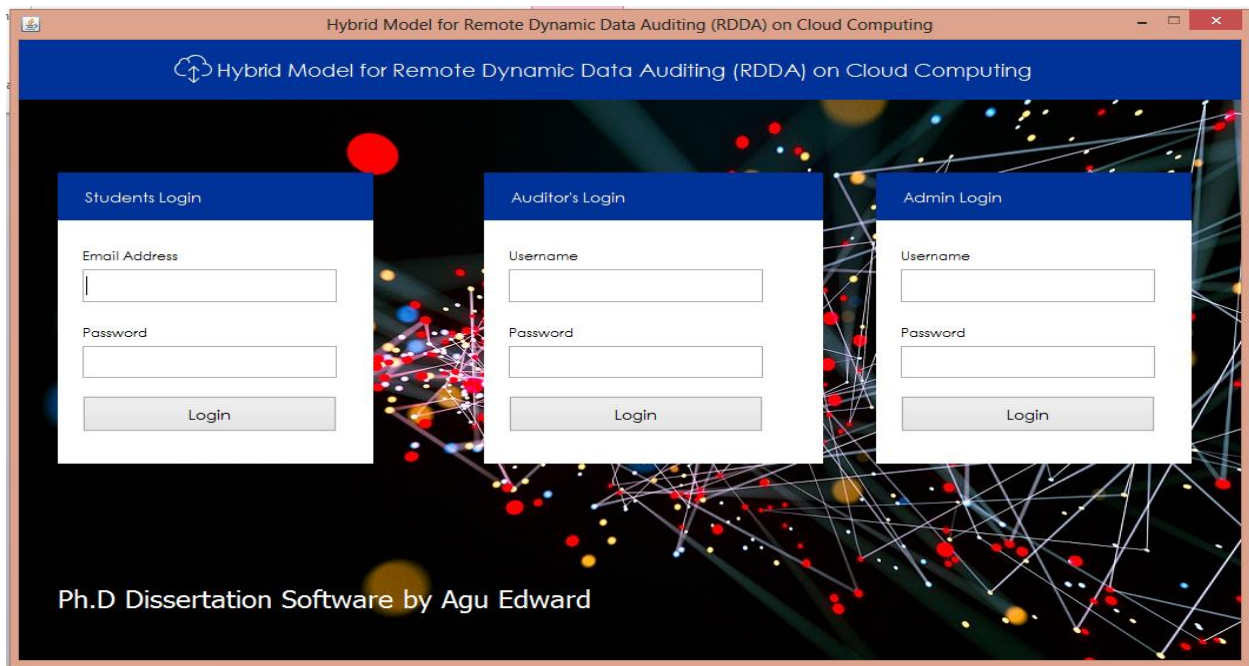


Figure 4.20: System Main Menu

The admin logs into the system through admin login interface with his login credentials to unlock admin system functionalities as depicted in figure 4.21.

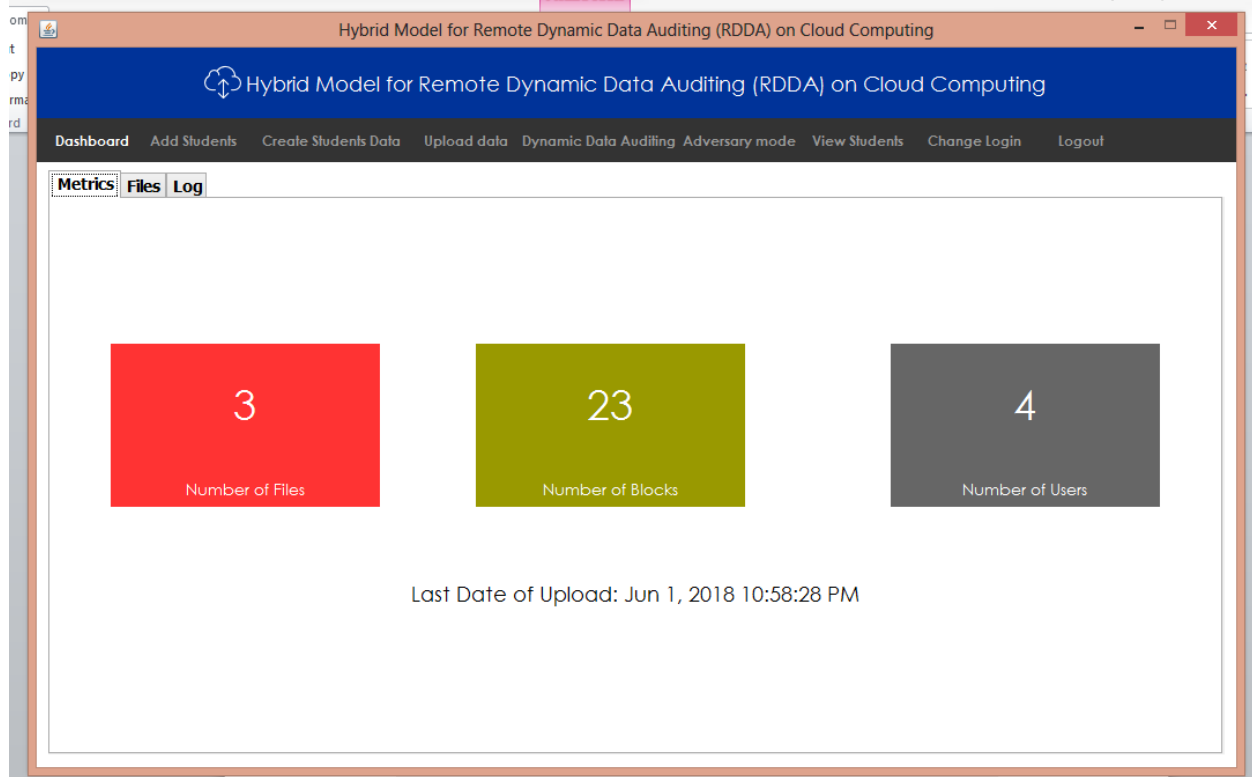


Figure 4.21: Admin Functionality window

Once a user has being granted legitimate access as the admin, the user has the power to create and manage students data which include their bio-data and results, preprocess these data and upload, execute dynamic data auditing operations and carry out adversary mode operation to ascertain when the system is in its right state of operation. All these functionalities are performed by the admin user through its respective submenus as indicated with the following figures.

The screenshot shows a web browser window with the title "Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing". The browser's address bar and the page's header both display this title. Below the header is a navigation menu with the following items: Dashboard, Add Students, Create Students Data, Upload data, Dynamic Data Auditing, Adversary mode, View Students, Change Login, and Logout. The main content area is titled "Add Student" and contains a form with the following fields and values:

Field	Value
Firstname	Emenike
Lastname	Celine
Email Address	celine@yahoo.com
State of Origin	Imo
Age	28
Faculty	Information and Communication Technology
Department	Computer Science
Level	400L
Password	••••••
Registration Number	UR20130001

At the bottom center of the form is a button labeled "Add Student".

Figure 4.22: Add Student Submenu

This submenu depicted by figure 4.22 helps the admin to enroll students into the new system and providing the students with legitimate access parameters.

Figure 4.23 shows a confirmation message after adding a student successfully.

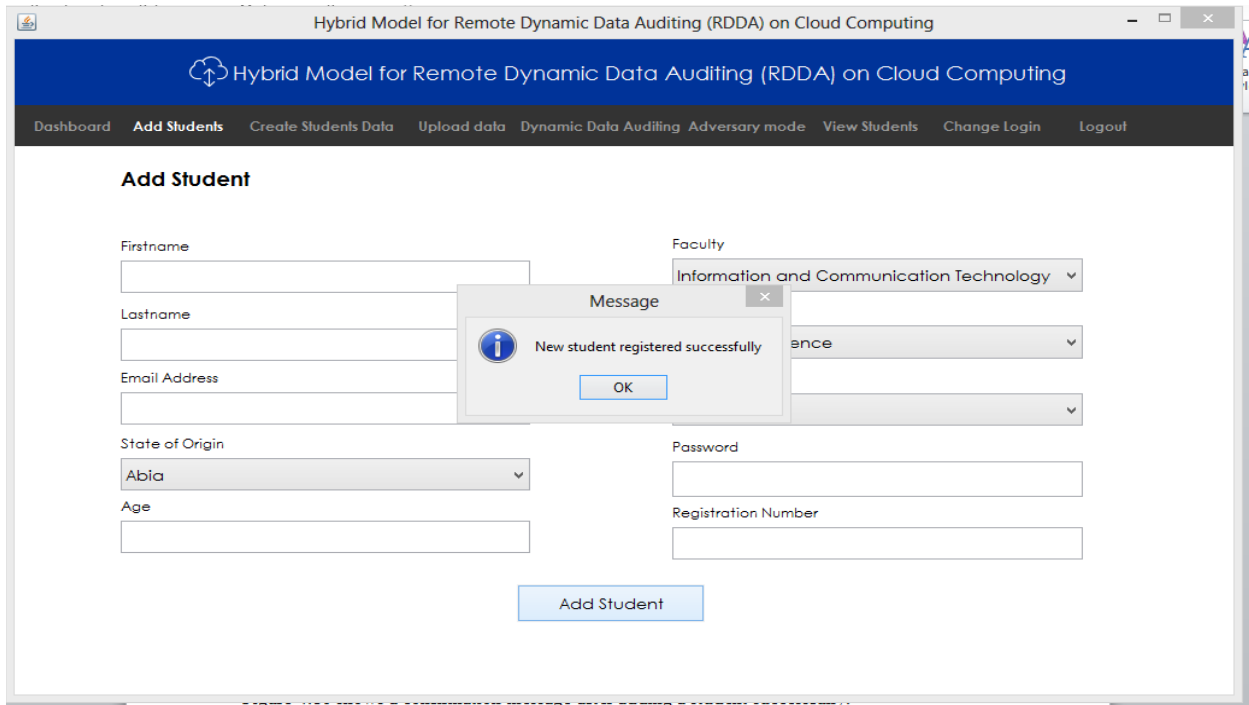


Figure 4.23: Add Student Confirmation window

Acquisition of the students' assessment data is done through figure 4. 24.

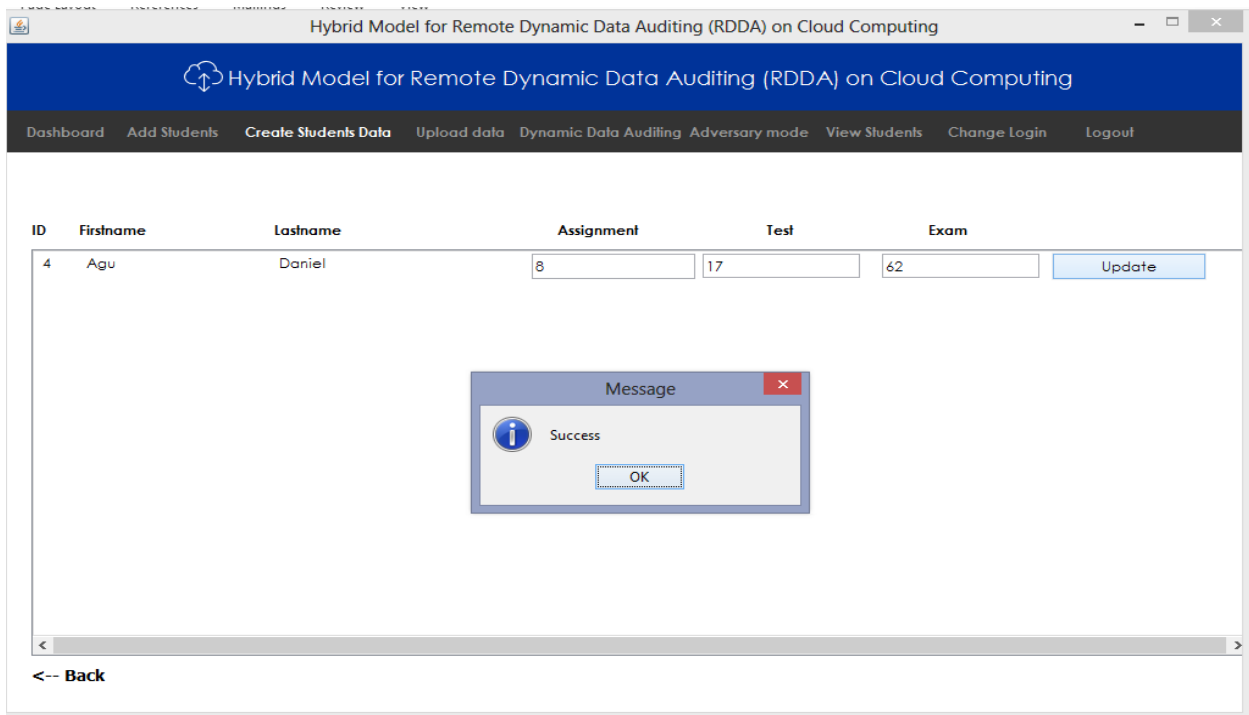


Figure 4.24: Create Students Data/Scores Submenu

Figure 4.24 submenu helps the admin during data capturing. Students' data are acquired through this submenu.

Figure 4.25 shows the upload interface while figure 4.26 shows the confirmation that file has been uploaded successfully.

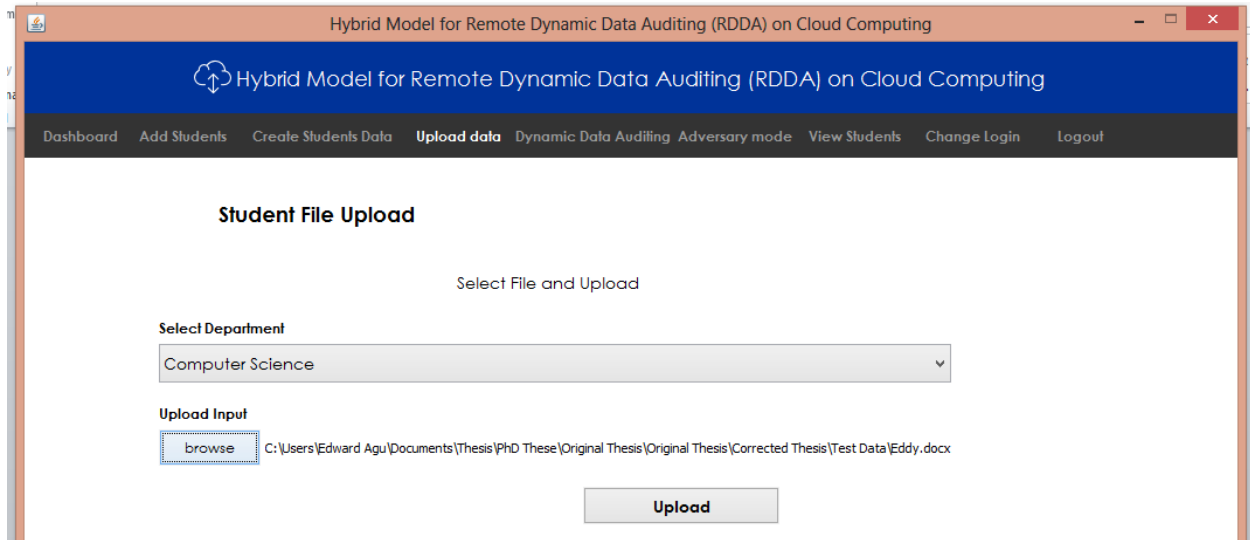


Figure 4.25: Data Upload Submenu

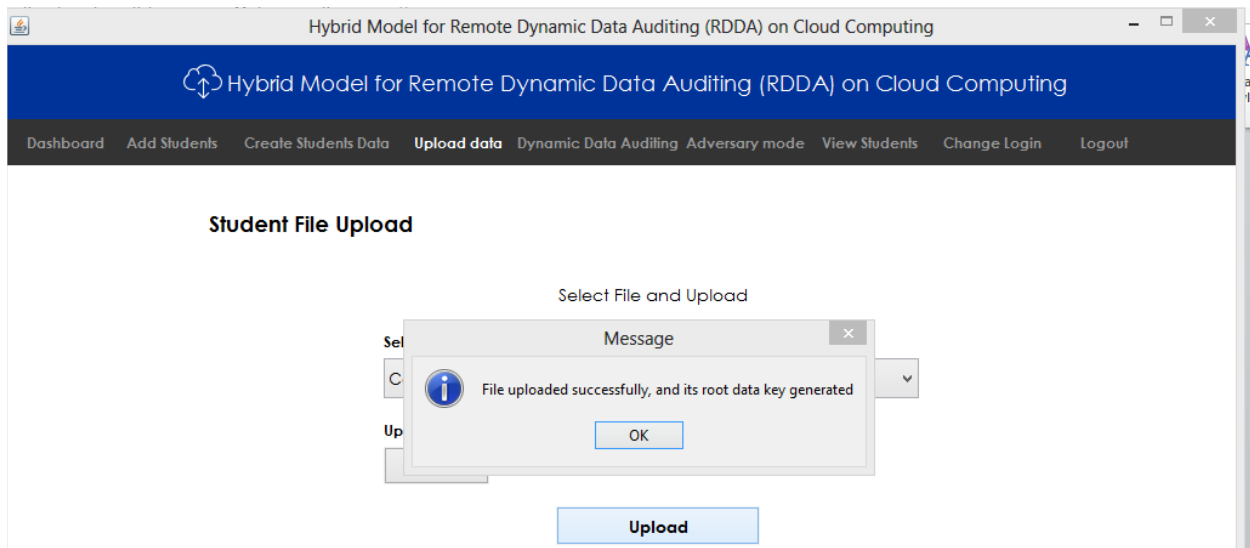


Figure 4.26: File Upload Confirmation

Figure 4.26 Submenu is used as an interface to preprocessed and outsourced data to the cloud. Data uploaded through this medium is preprocessed into blocks of data outsourced to the cloud.

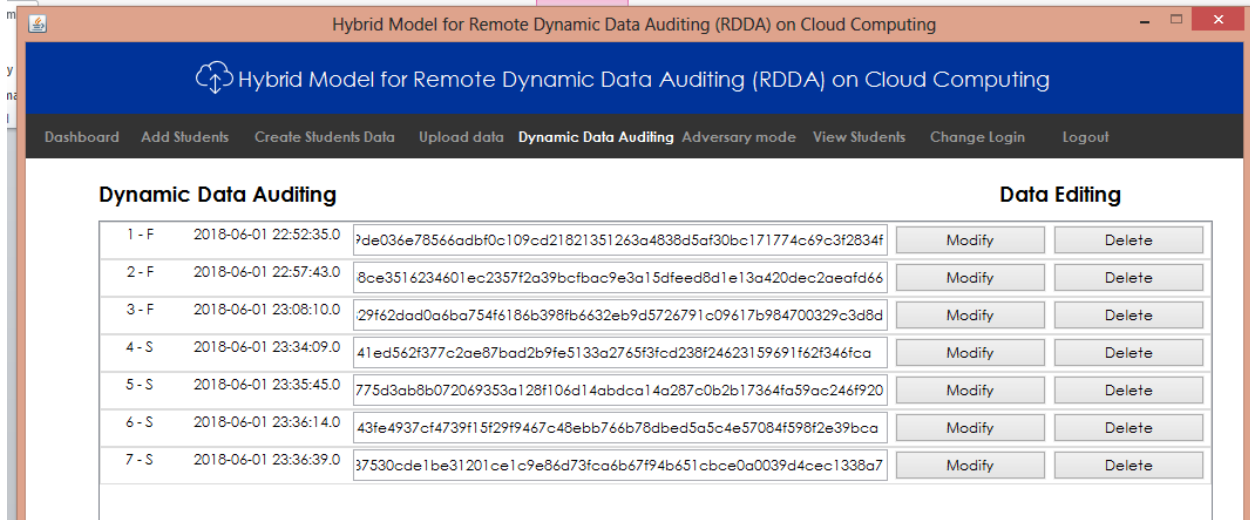


Figure 4.27: Dynamic Data Auditing Submenu

The dynamic data auditing operation as depicted in figure 4.27 is done through this window. Each of the outsourced data can be modified further through this submenu re-outsourced with valid audit key.

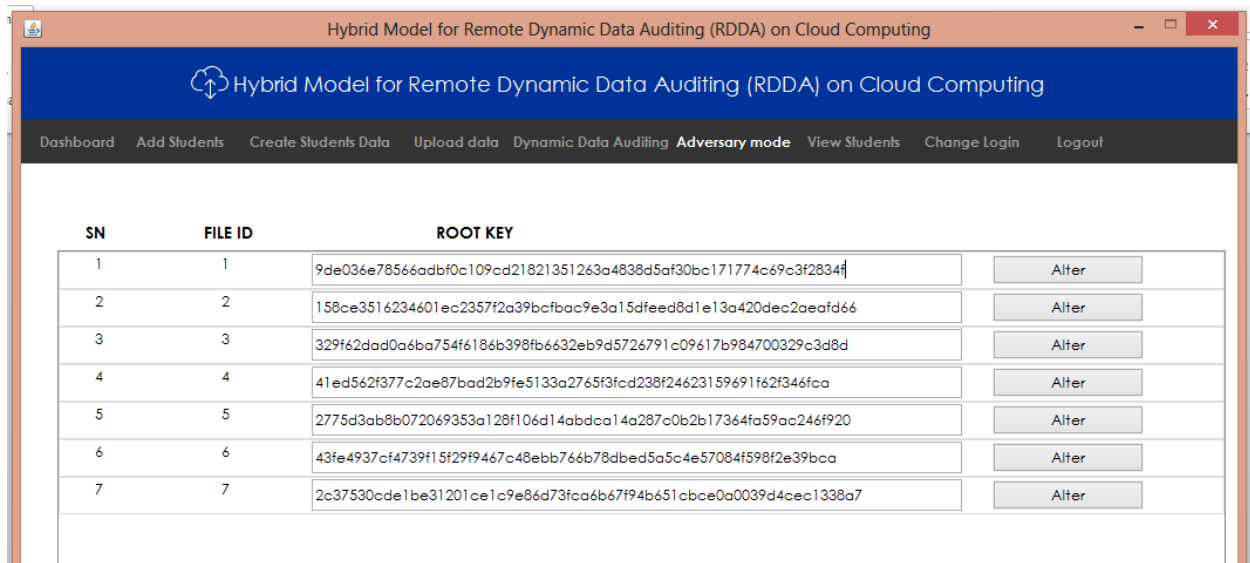


Figure 4.28: Adversary Mode Submenu

Figure 4.28 is the adversary mode submenu which is used to keep track of the system normal working state. If any of the file main audit key is altered through this submenu, it supposed to reflect in the system audit result; else the system integrity is questionable.

S/N	FIRSTNAME	LASTNAME	ASSIGNMENT	TEST	EXAMS	TOTAL	DEPARTMENT	FACULTY
1	Emenike	Celine	5	18	62	85	Computer Science	Information and Communication Technology
2	Agu	David	8	19	67	94	Computer Science	Information and Communication Technology
3	Agu	Daniella	7	18	53	78	Computer Science	Information and Communication Technology
4	Agu	Daniel	8	17	62	87	Computer Science	Information and Communication Technology

Figure 4.29: View Students Submenu

Figure 4.29 is used to have overview of the students' data on the system, while Figure 4.30 shows the dashboard or menu through which the Auditor checks the integrity of the outsourced data to cloud.

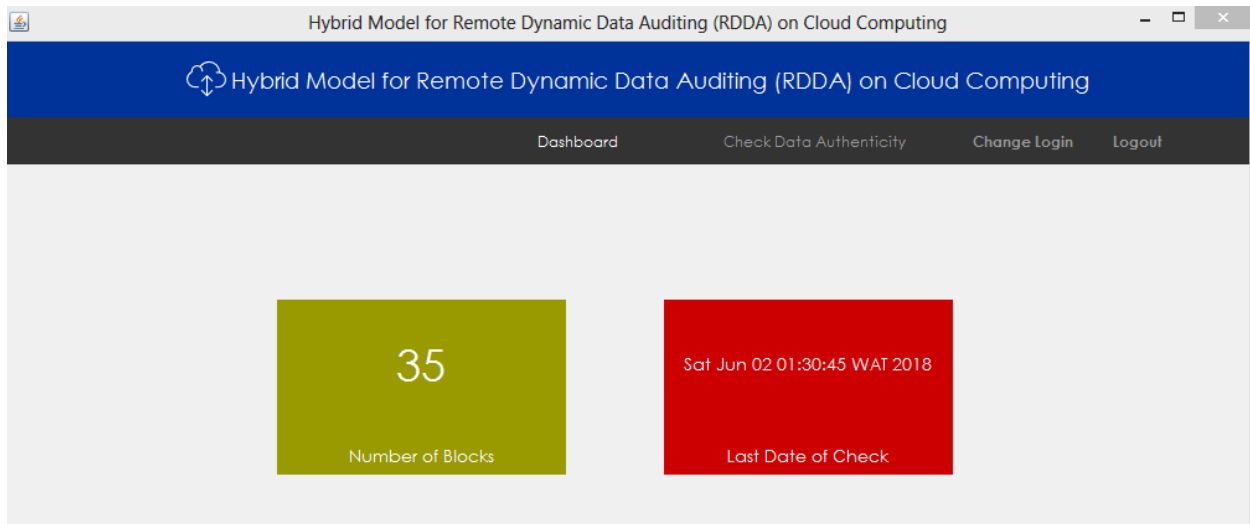


Figure 4.30: Auditor Menu

Figure 4.31 is the auditor submenu called Check Data Authenticity. The process of data auditing is done through this submenu by selecting the file to be audited.

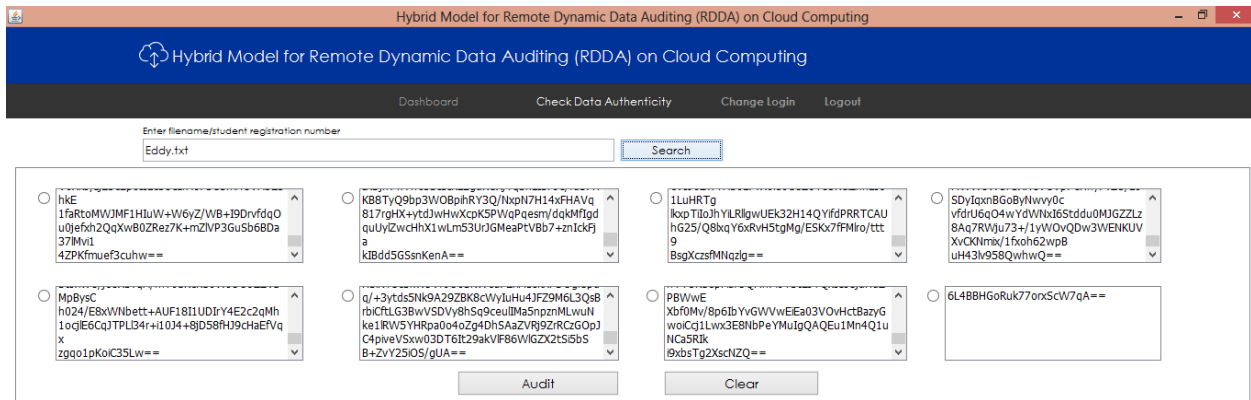


Figure 4.31: Auditor Submenu

Figure 4.32 show the selected audit path through which the main root key is generated.

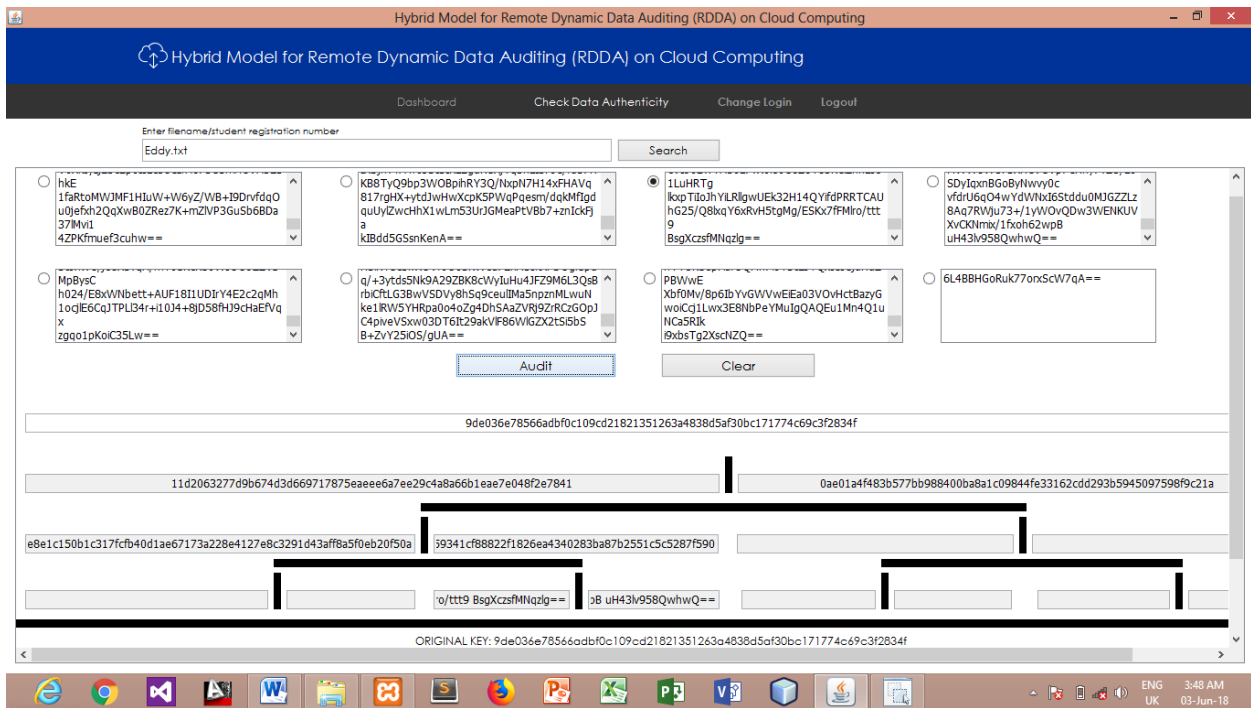


Figure 4.32: Audit Path

Figure 4.33 displays students' successful login page with its bio-data, while figure 4.34 shows the successfully logged in students' scores.

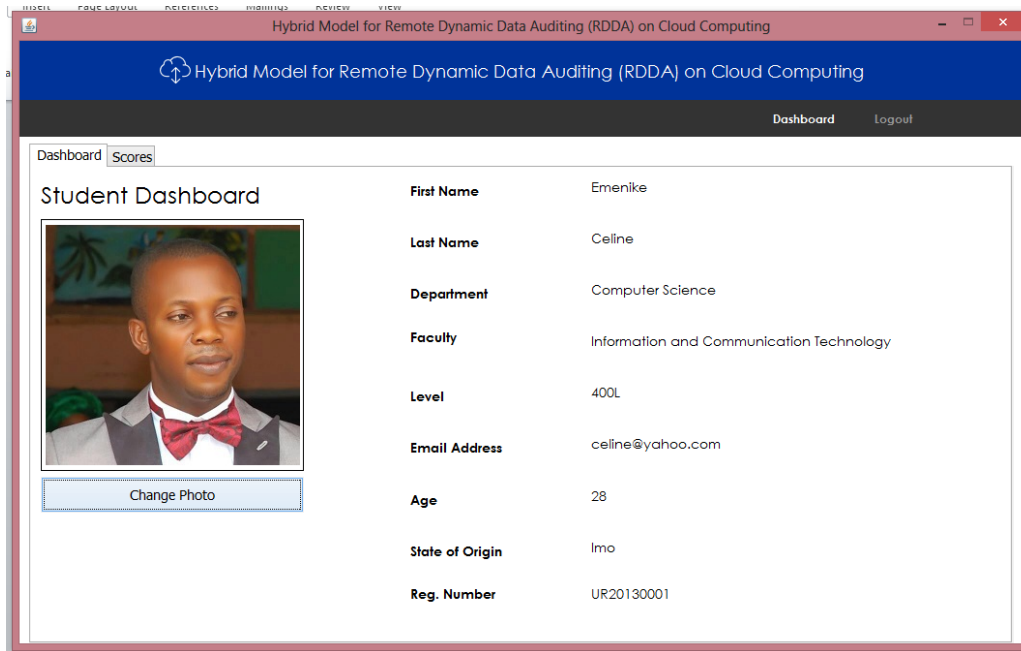


Figure 4.33: Students' login page Dashboard

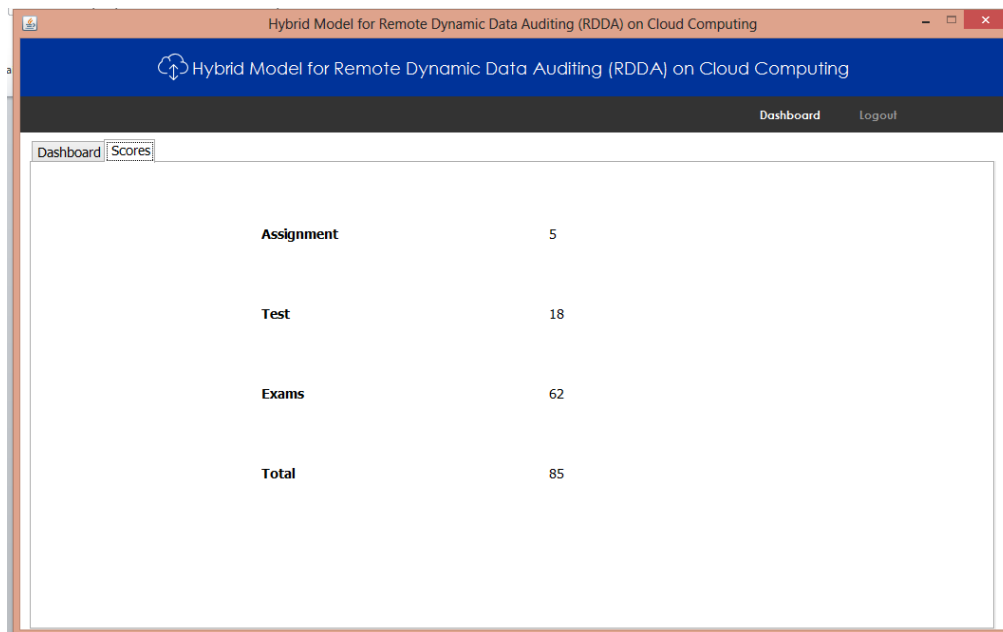


Figure 4.34: Students' logged in Scores

Figure 4.35 shows the list of files currently outsourced to cloud and the download operational buttons attached to each file is used to decrypt the files their original contents when required.

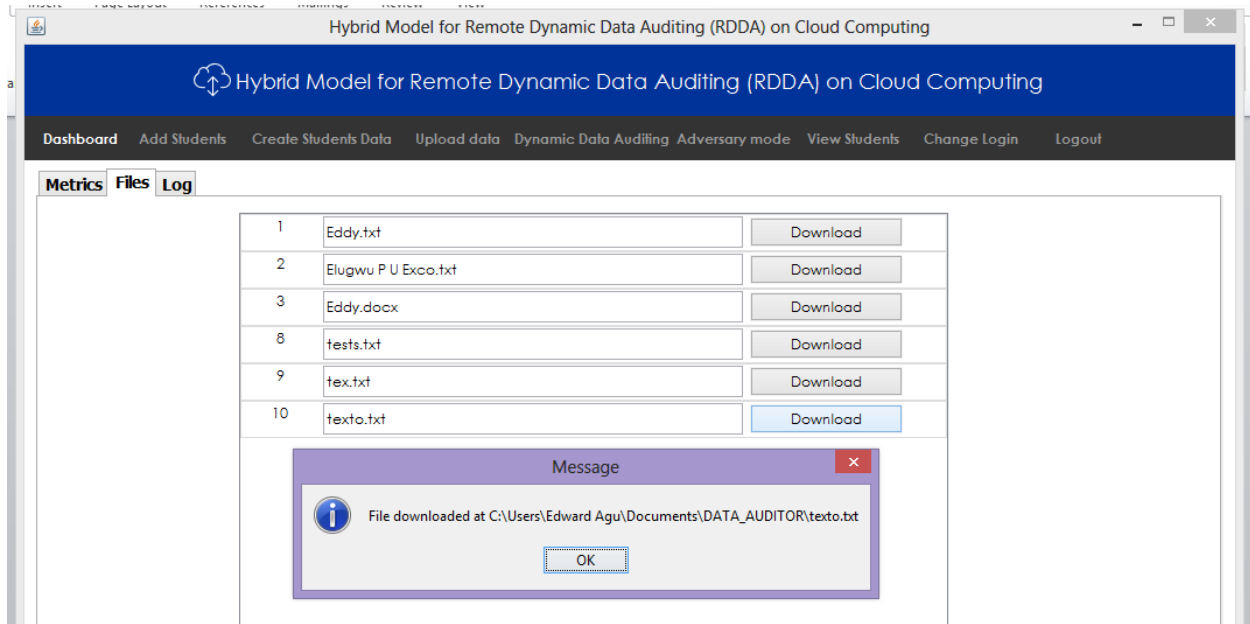


Figure 4.35: Outsourced File List

Figure 4.36 shows the log of all attempted access to the system. This is one of the security measures embedded to secure this system. It logs the user out after three failed attempts.

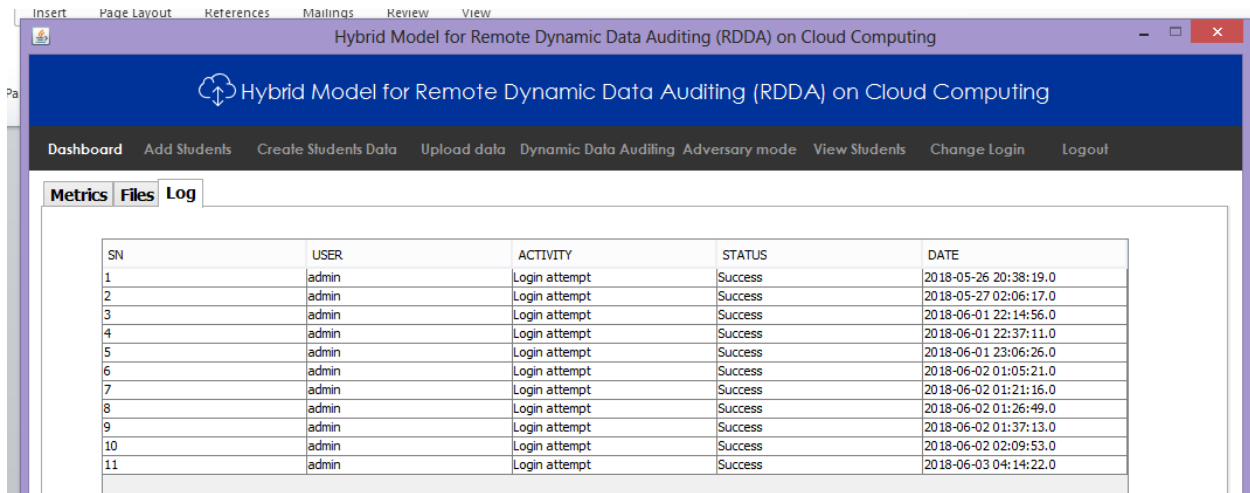


Figure 4.36: System Access Logs

Figure 4.37 is the new system metric window which shows the number of files outsourced, number of preprocessed blocks already in existence and number of registered users within the new system.

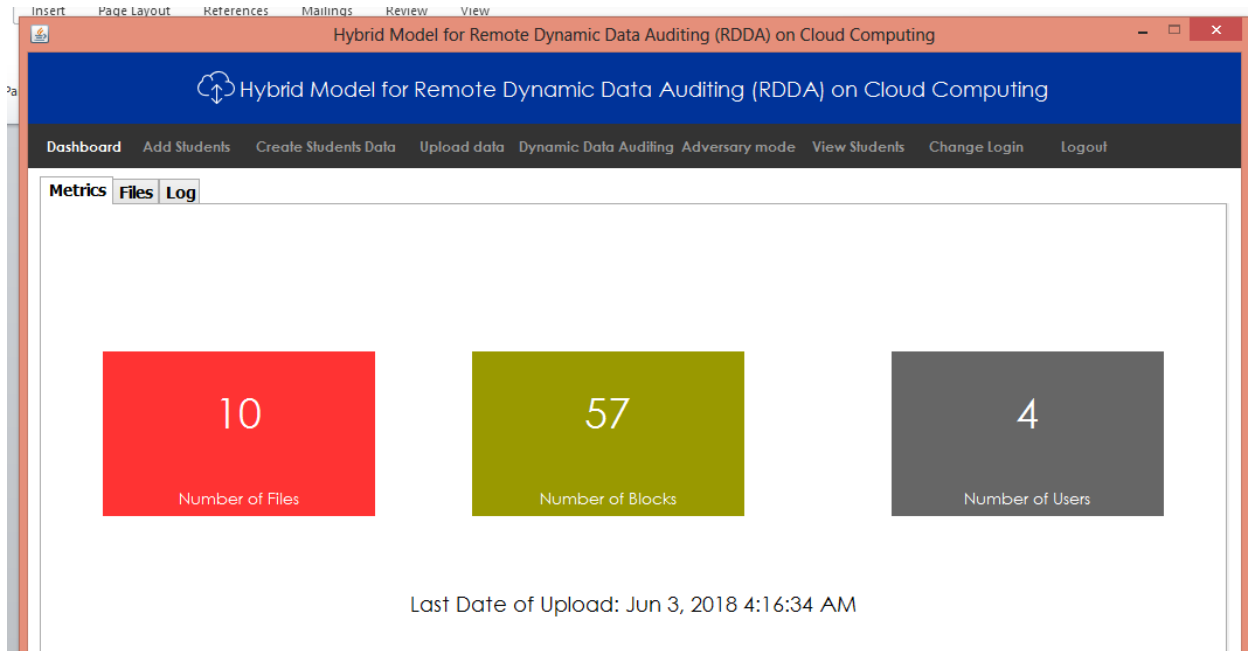


Figure 4.37: System Metrics

4.6.7 System Conversion:

4.6.7.1 Changeover Procedures:

There are five procedures or plans for converting from the old system to the new system and they are listed as follows:

- i. Direct changeover.
- ii. Parallel conversion.
- iii. Gradual, or phased, conversion.
- iv. Modular conversion.
- v. Distributed conversion.

4.6.7.2 Recommended Procedure:

The recommended changeover method or plan to be adopted in this research is gradual or phased conversion plan. This method of conversion is chosen by putting into consideration, the sensitiveness of the data to be outsourced. The method attempts to combine the best features of direct and parallel conversion plans, without incurring all the risks. In this plan, the volume of transactions handled by the new system is gradually increased as the system is phased in. The advantages of this approach include allowing users to get used to the system gradually, the

possibility of detecting and recovering from errors without a lot of down time, and the ability to add features one-by-one.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

Previous models for Remote Data Auditing (RDA) were found to be weak to security attacks due to insecurity of their rootkey, vulnerability of Third Party Auditor (TPA) and lack of dynamic data auditing. Huge amount of sensitive data are generated and managed internally on a daily basis by different organizations globally. Cloud computing services provide huge amounts of storage space and customizable cheap and very easy to adapt computing resources. But it eliminates the responsibility of local machines for data maintenance and auditing. As a result, the availability and integrity of clients' outsourced data are solely determined by Cloud Service Providers (CSP) which require periodic remote integrity audit, hence this study.

The aim of this research was to develop an enhanced hybrid model for dynamic remote data auditing on cloud computing. The objectives of the study were to; present the design of an Enhanced Model for Dynamic remote Data Auditing; develop an enhanced hybrid system to support dynamic remote data auditing and data dynamic operations in the cloud by maintaining data integrity and availability even if users modify, delete, insert or update their data files in the cloud; build and apply an adversary data authentication model to evaluate the effectiveness of the system; and compare performance of the new system with the existing system.

Object Oriented Analysis and Design Methodology (OOADM) was employed for systematic study of the existing system and implementation of the secured hybrid dynamic remote data audit model. Merkle Hash Tree (MHT) authentication data technique was used to develop a model that implements dynamic remote data auditing and dynamic remote data operations. Improved Revester Code version 6 (IRC6) cryptographic technique was used to secure the MHT rootkey which is the main auditing key. MHT techniques was also employed to develop an adversary data authentication model; vulnerabilities associated with TPA services were eliminated using combined MHT and IRC6 techniques. Maven Protocol Buffers Plugin was used to implement access log as intrusion detection mechanism.

An enhanced hybrid model for dynamic remote data auditing was developed to ensure availability and integrity of outsourced data; IRC6 subsystem was developed to secure the MHT rootkey which is the main auditing parameter; an adversary data authentication model was

developed to audit the activities of the internal data auditor; a model that bypasses the risk and cost of adopting vulnerable TPA services was also implemented; an intrusion detection mechanism through access log and a Triangular Security Handshake (TSH) with timestamp was implemented to improve access control to the new system. the performance of the new system was evaluated to be an improvement over the existing systems.

The implementation of the secured hybrid dynamic remote data auditing model was able to strengthen the confidence and business relationship between the CSP and its prospective clients. It also improved the security of the outsourced data and strengthened the security of the previous RDA models with IRC6. The developed IRC6 was evaluated to survive any crypto-analytical attack at little or no cost. This work is recommended for organizations where huge amount of data are generated on a daily basis and requires high level of economic means of preservation and security.

5.2 Conclusion

Benefits of cloud computing platform are now extended as confidence has being restored on more organizations who intend to outsource their sensitive organizational data to the cloud as a result of the development of this new improved data auditing system. The outlined problem statements in this research were achieved as the research objectives deliverables were implemented as part of the result discussed.

5.3 Recommendation

5.3.1 Application Areas

This research is applicable in data security areas such as organizations like higher institutions, financial institutions, government parastatals and corporate bodies where large amount of sensitive data are generated on a daily basis and requires high level of economic means of preservation and security.

5.3.2 Suggestion for Further Research

We suggest further development of a byzantine failures tracking reporting sub system that will be reporting directly from the cloud to the auditor in terms of arbitrary deviations of a process from its assumed behavior based on the algorithm it is supposed to be running and the inputs it receives. Byzantine failures sometimes occur due to a software bug, hardware malfunction, or a malicious attack.

5.4 Contribution to Knowledge

- i. Implementation of an enhanced hybrid cloud data storage auditing model that support both public auditability and dynamic data operation with MHT and IRC6 encryption algorithms.
- ii. Implementation of IRC6 to secure the root key, its performance evaluation in terms of cost against security achieved
- iii. Implementing the security of MHT root key which is the strength of the auditing scheme.
- iv. Implementation of a model that bypass the risk and cost of adopting vulnerable TPA.

References

- Alan, D., and Barbara, H. W. (2008). *Systems Analysis and Design*, 4th Edition, John Willey, New York
- Anmol, A., Priyanka, V., Rajeshwari, and A., Meera, C., (2015). Public Auditing for the Shared Data in the Cloud, *International Journal of Computing and Technology (IJCAT)*, Volume 2, Issue 4, ISSN : 2348 – 6090, www.IJCAT.org
- Armbrust M., Fox A., Griffith R., Joseph A., Katz R., Konwinski A., Lee G., Patterson D., Rabkin A., and Stoica I., (2009). Above the clouds: a Berkeley view of cloud computing, Technical Report UCB/EECS-2009-28.
- Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., and Song, D., (2007). Provable data possession at untrusted stores. *Cryptology ePrint Archive*, Report 2007/202.
- Avinash, C., Hasritha, P., and Abhinay J., (2015). A Dynamic K-means Algorithm for Searching Conserved Encrypted Data in a Cloud, *International Journal of Computer Applications* (0975 – 8887) Volume 129 – No.5, Pp 33-38
- Ben, K., (2011). *Understanding the Cloud Computing Stack SaaS, PaaS, IaaS*, Diversity LTD, A white paper developed by CloudU and sponsored by Rackspace Inc. United States of America.
- Bowers, K. D., Juels, A., and Oprea, A., (2009). Proofs of Retrievability: Theory and Implementation. *Proceedings of ACM Workshop Cloud Computing Security (CCSW)*, pp. 43-54.
- Cong, W., Qian, W., Kui, R., and Wenjing L. (2012). Towards Secure and Dependable Storage Services in Cloud Computing. *IEEE Transactions on Services Computing*.
- Curtmola, R., Khan, O., Burns, R., and Ateniese, G. (2008). MR-PDP: Multiple-replica provable data possession. *Proceedings of ICDCS'08*. IEEE computer society Washington DC, USA, pp. 411-420.
- Dave, R., (2012). What is Cloud Computing? – A tutorial, Leverhawk
<http://leverhawk.com/what-is-cloud-computing-tutorial-2012120519>. Accessed on 10th May, 2015 by 19:12 GMT.

- Educause, (2009). 7 things you should know about cloud computing. From <http://creativecommons.org/licenses/by-nc-nd/3.0/educause.edu>. Accessed on 25th May, 2017
- Filho, D. L. G., and Barreto, P. S. L. M., (2007). Demonstrating data possession and uncheatable data transfer. IACR Cryptology ePrint Archive 2006/150.
- Fredrickson, N., (1998). Up-front Investment for OOAD Can Pay Off, *Electronic Engineering Times*,
- Gosavi H. A., and Umale M. R., (2014). Public Auditing and Data Dynamics for Storage Security in Cloud Computing. *International Journal of Engineering Trends and Technology (IJETT) – Volume 11 Number 4*.
- Juels, A., and Kaliski, B., (2007). PORs: Proofs of retrievability for large files. *ACM CCS*, pp. 584-597.
- Karthikeyan B., ((2015). Merkle Hash Trees for Distributed Audit Logs. Retrieved from <https://www.enseignement.polytechnique.fr/informatique/INF441/projets/merkle/merkle.pdf> on 28th May, 2016 by 03:57 GMT.
- Karthika R., Archana D. R., Suganya T., and Nivethitha K., (2015), Efficient Authentication Using Merkle Hash Tree Algorithm In Jelastic Server, *International Journal of Engineering Research-Online*, Vol.3., Issue.2, ISSN: 2321-7758
- Kayalvizhi S., and Jagadeeswari S., (2012), Data Dynamics for Storage Security and Public Auditability in Cloud Computing, *Journal of Computer Applications* ISSN: 0974 – 1925, Volume-5, Issue EICA2012-1,
- Khaba M.V., and Santhanalakshmi M., (2013). Remote Data Integrity Checking in Cloud Computing. *International Journal on Recent and Innovation Trends in Computing and Communication*, ISSN 2321 – 8169 Volume: 1 Issue: 6
- Lillibridge, M., Elnikety, S., Birrell, A., Burrows, M., and Isard, M., (2003). A Cooperative internet Backup Scheme. *Proceedings of the 2003 USENIX Annual Technical conference (General Track)*, Vol. 34, pp. 29-41.
- Martin, J., (1992). *Rapid Application Development*. Englewood Cliffs, Prentice-Hall, New Jersey, USA.
- Merkle R. C., (1980). Protocols for public key cryptosystems, *Proceedings of IEEE Symposium on Security and Privacy*, pp. 122–133.

- Paul B.D., (1998). Rapid Application Development: A Review and Case Study, Kane Thompson Centre.
- Paul F., James M., and Peter H., (2001). System Development Life Cycle Models and Methodologies: Canadian Society for International Health Certificate Course in Health Information System, Canada.
- Peter M., and Timothy G., (2011). The NIST Definition of Cloud Computing, NIST Special Publication 800-145. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930.
- Qian W., Cong W., Jin L., Kui R., and Wenjing L., (2008). Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. Illinois Institute of Technology, Chicago IL 60616, USA, {qwang,cwang,jin.li,kren}@ece.iit.edu.
- Ren, K., Wang, C., and Wang, Q., (2012). Security Challenges for the Public Cloud. IEEE Internet Computing, vol. 16, no. 1, pp. 69-73.
- Rhee M. Y., (2003). Internet security: cryptographic principles, algorithms, and Protocols, John Wiley, John Wiley & Sons Ltd, The Atrium, South Date, Chichestre, West Sussex PO09 8SQ, England. Pp 26-298.
- Sadoya A.S., Ibrahim S. A., and Ajayi O. B., (2004). The State Of Information Security In South-Western Nigerian Educational Institutions, Proceedings of the International Conference on Science and National Development.
- Sam, J., (2009). Cloud Computing Architecture. Web resource from https://en.wikipedia.org/wiki/cloud_computing#/media/file:cloud_computing.svg
- Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N., (2009). The case for VM-based cloudlets in mobile computing, IEEE Pervasive Computing 8, pp.14–23.
- Satyanarayanan, M., (1996). Fundamental challenges in mobile computing, in: Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, PODC'96, ACM, New York, NY, USA, pp. 1–7.
- Satyanarayanan M. (1993), Mobile computing, IEEE Pervasive Computing 26, pp. 81–82.
- Sasaki Y., and Shibata Y., (2012), A disaster information sharing method by the mobile servers in challenged networks, in: Advanced Information Networking and Applications Workshops, WAINA, 26th International Conference on, pp. 1048–1053.
- Schwarz, T.S.J., and Miller, E.L., (2006). Store, Forget, and Check: Using Algebraic Signatures

- to Check Remotely Administered Storage. Proceedings of ICDCS'06, Vol. 26, pp. 12-12.
- Shacham, H., and Waters, B., (2008). Compact proof of retrievability. Cryptology ePrint Archive, Report 2008/073.
- Shah, M. A., Baker, M., Mogul, J. C., and Swaminathan, R., (2007). Auditing to Keep Online Storage Service Honest. Proc 11th USENIX Workshop on Hot Topics in Operating systems (HOTOS'07), Vol. 13, pp.1-6.
- Siegele, L., (2008). Let it rise: a special report on corporate it, <http://www.economist.com/node/12411882>. Accessed on 25th May 2017.
- Snehal, V. Z., Amruta, V. T., Shivangi, S. S., Rajashree, C. B., (2014). Data Integrity Checking Protocol with Data Dynamics and Public Verifiability for Secure Cloud Computing, International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 5 (3), Pp 4062-4064.
- Sookhak, M., Gani, A., Khan, M. K., and Buyya, R., (2015) Dynamic remote data auditing for securing big data storage in cloud computing, Information Sciences, Elsevier, pp 1-16, <http://dx.doi.org/10.1016/j.ins.2015.09.004>
- Srijanya, A. K., and Kasiviswanath, N., (2013). Data Integrity Verification by Third Party Auditor in Remote Data Cloud. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-5.
- Wang, C., Wang, Q., Ren, K., and Lou W., (2009). Ensuring Data Storage Security in Cloud Computing. Proc. 17th Int'l Workshop Quality of Service pp. 1-9.
- Wang, S., (1996). Two MIS Analysis Methods: An Experimental Comparison, Journal of Education for Business, Vol. 71 Issue 3, p136.
- Wang, Q. A., Wang C., Ren K., Lou W. J., and Li, J., (2011). Enabling public auditability and data dynamics for storage security in cloud computing, IEEE Trans. Parallel Distr. pp 847–859.
- Webster, S., (1996). Focus groups. RAD, JAD and DSDM Heathrow, London, Unicom Seminars.
- Yang, K., and Jia, X., (2012) An efficient and secure dynamic auditing protocol for data storage in cloud computing, IEEE Trans. Parallel Distrib. PP (2012) 1717–1726.
- Zion, D. G., and Kayitha, D., (2012). Remote Sensing Data as a Service in Hybrid Clouds:

Security Challenges and Trusted Third Party Auditing Mechanisms. International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 7.

Appendix A: Program Listings

MerkleTrees.java

```
import java.security.MessageDigest;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author user
 */
public class MerkleTrees {
    // transaction List
    List<String> txList;
    // Merkle Root
    String root;
    //holds all nodes of the merkle tree
    private List<String> nodes;
    //default constructor
    public MerkleTrees(){

/**
 * constructor
 * @param txList transaction List
 */
    public MerkleTrees(List<String> txList) {
        this.txList = txList;
        root = "";
        this.nodes = new ArrayList<>();
    }
    /**
     * execute merkle_tree and set root.
     */
    public void merkle_tree() {
        List<String> tempTxList = new ArrayList<String>();
        for (int i = 0; i < this.txList.size(); i++) {
            tempTxList.add(this.txList.get(i));
        }

        print(tempTxList);
        List<String> newTxList = getNewTxList(tempTxList);
        while (newTxList.size() != 1) {
            print(newTxList);
            newTxList = getNewTxList(newTxList);
        }
        this.root = newTxList.get(0);
        nodes.add(this.root);
    }

    private void print(List<String> newTxList){
        //System.out.print(newTxList.size()+" - \t");
        for(String tx: newTxList){
            nodes.add(tx);
            //System.out.print(tx+"\t");
        }
        //System.out.println();
    }
}
```

```

}
/**
 * return Node Hash List.
 * @param tempTxList
 * @return
 */
private List<String> getNewTxList(List<String> tempTxList) {
    List<String> newTxList = new ArrayList<String>();
    int index = 0;
    while (index < tempTxList.size()) {
        //left
        String left = tempTxList.get(index);
        index++;

        //right
        String right = "";
        if (index != tempTxList.size()) {
            right = tempTxList.get(index);
        }
        //sha2 hex value
        String sha2HexValue = getSHA2HexValue(left + right);
        newTxList.add(sha2HexValue);
        index++;
    }
    return newTxList;
}
/**
 * Return hex string
 * @param str
 * @return
 */
public static String getSHA2HexValue(String str) {
    byte[] cipher_byte;
    try{
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(str.getBytes());
        cipher_byte = md.digest();
        StringBuilder sb = new StringBuilder(2 * cipher_byte.length);
        for(byte b: cipher_byte) {
            sb.append(String.format("%02x", b&0xff) );
        }
        return sb.toString();
    } catch (Exception e) {
        e.printStackTrace();
    }

    return "";
}

/**
 * Get Root
 * @return
 */
public String getRoot() {
    return this.root;
}

```

```

public List<String> getNodes(){
    return this.nodes;
}
}

```

RC6Algorithm.java

```

/**
 *
 * @author user
 */
public interface RC6Algorithm {

    public static String ALGORITHM_NAME = "RC6";
    public static String MODE_OF_OPERATION = "ECB";
    public static String PADDING_SCHEME = "PKCS5Padding" ;
    public final static int RC6_KEYLENGTH = 128;

    public void setKey(String password);
    public String rc6Encrypt(String encryptedText, String key) throws Exception;
    public String rc6Decrypt(String decryptedText, String key) throws Exception;
}

```

RC6_Normal.java

```

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import javax.crypto.Cipher;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Modifier;
import java.util.Map;

import java.io.UnsupportedEncodingException;
import java.security.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class RC6_Normal implements RC6Algorithm{
    static {
        Security.addProvider(new BouncyCastleProvider());
    }
}

```



```

}

private SecretKey secretKey;

private String startKey, startEncrypt, startDecrypt;
private String endKey, endEncrypt, endDecrypt;

private DateFormat df;

public RC6_Normal(){
    fixKeyLength();
    df = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss.SSS");
    startKey = ""; startEncrypt = ""; startDecrypt = "";
    endKey = ""; endEncrypt = ""; endDecrypt = "";

    try {
        String plaintext = "edy";
        String key = "admin";

        String cipher = rc6Encrypt(plaintext, key);

        Date sK = df.parse(startKey);
        Date eK = df.parse(endKey);
        long duration = eK.getTime() - sK.getTime();
        double ms = ((double)duration)/1000.0 % 60.0;

        System.out.println("\n\ntime for key generation: ");
        System.out.println("start time: "+startKey);
        System.out.println("end time: "+endKey);
        System.out.println("diff. in time: "+ms+" secs");

        Date sE = df.parse(startEncrypt);
        Date eE = df.parse(endEncrypt);
        duration = eE.getTime() - sE.getTime();
        ms = ((double)duration)/1000.0 % 60.0;

        System.out.println("\n\ntime for encryption: ");
        System.out.println("start time: "+startEncrypt);
        System.out.println("end time: "+endEncrypt);
        System.out.println("diff. in time: "+ms+" secs");
        System.out.println("cipher: "+cipher);

        String text = rc6Decrypt(cipher, key);

        Date sD = df.parse(startDecrypt);
        Date eD = df.parse(endDecrypt);
        duration = eD.getTime() - sD.getTime();
        ms = ((double)duration)/1000.0 % 60.0;

        System.out.println("\n\ntime for decryption: ");
        System.out.println("start time: "+startDecrypt);
        System.out.println("end time: "+endDecrypt);
        System.out.println("diff. in time: "+ms+" secs");
    }
}

```

```

        System.out.println("plaintext: "+text);
    } catch (Exception ex) {
        Logger.getLogger(RC6_Normal.class.getName()).log(Level.SEVERE, null, ex);
    }
}

@Override
public void setKey(String password)
{
    startKey = df.format(new Date());
    byte[] digestOfPassword = password.getBytes();//sha.digest(key);
    byte[] keyBytes = Arrays.copyOf(digestOfPassword, RC6_KEYLENGTH);
    secretKey = new SecretKeySpec(keyBytes, ALGORITHM_NAME);
    endKey = df.format(new Date());
}

@Override
public String rc6Encrypt(String toEncrypt, String key) throws Exception{
    setKey(key);
    startEncrypt = df.format(new Date());
    Cipher cipher = Cipher.getInstance(ALGORITHM_NAME + "/" + MODE_OF_OPERATION + "/" +
PADDING_SCHEME);
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);
    byte[] byteCipherText = cipher.doFinal(toEncrypt.getBytes());
    String strCipherText = new BASE64Encoder().encode(byteCipherText);
    endEncrypt = df.format(new Date());
    return strCipherText;
}

@Override
public String rc6Decrypt(String encryptedText, String key) throws Exception {
    setKey(key);
    startDecrypt = df.format(new Date());
    Cipher cipher = Cipher.getInstance(ALGORITHM_NAME + "/" + MODE_OF_OPERATION + "/" +
PADDING_SCHEME);
    cipher.init(Cipher.DECRYPT_MODE, secretKey);
    byte[] byteEncryptedText = new BASE64Decoder().decodeBuffer(encryptedText);
    byte[] decrypted = cipher.doFinal(byteEncryptedText);
    endDecrypt = df.format(new Date());
    return new String(decrypted);
}

public static void main(String[] args){
    new RC6_Normal();
}

public void fixKeyLength() {
String errorString = "Failed manually overriding key-length permissions.";
int newMaxKeyLength;
try {
    if ((newMaxKeyLength = Cipher.getMaxAllowedKeyLength("AES")) < 256) {
        Class c = Class.forName("javax.crypto.CryptoAllPermissionCollection");
        Constructor con = c.getDeclaredConstructor();
        con.setAccessible(true);
    }
}
}

```

```

    Object allPermissionCollection = con.newInstance();
    Field f = c.getDeclaredField("all_allowed");
    f.setAccessible(true);
    f.setBoolean(allPermissionCollection, true);

    c = Class.forName("javax.crypto.CryptoPermissions");
    con = c.getDeclaredConstructor();
    con.setAccessible(true);
    Object allPermissions = con.newInstance();
    f = c.getDeclaredField("perms");
    f.setAccessible(true);
    ((Map) f.get(allPermissions)).put("*", allPermissionCollection);

    c = Class.forName("javax.crypto.JceSecurityManager");
    f = c.getDeclaredField("defaultPolicy");
    f.setAccessible(true);
    Field mf = Field.class.getDeclaredField("modifiers");
    mf.setAccessible(true);
    mf.setInt(f, f.getModifiers() & ~Modifier.FINAL);
    f.set(null, allPermissions);

    newMaxKeyLength = Cipher.getMaxAllowedKeyLength("AES");
}
} catch (Exception e) {
    throw new RuntimeException(errorString, e);
}
}
if (newMaxKeyLength < 256)
    throw new RuntimeException(errorString); // hack failed
}
}

```

RC6_Modified.java

```

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import javax.crypto.Cipher;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Modifier;
import java.util.Map;

import java.io.UnsupportedEncodingException;
import java.security.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

```

```

public class RC6_Modified implements RC6Algorithm{

    static {
        Security.addProvider(new BouncyCastleProvider());
    }

    private SecretKey secretKey;

    private String startKey, startEncrypt, startDecrypt;
    private String endKey, endEncrypt, endDecrypt;

    private DateFormat df;

    /*
    class constructor
    */
    public RC6_Modified(){
        fixKeyLength();
        df = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss.SSS");
        startKey = ""; startEncrypt = ""; startDecrypt = "";
        endKey = ""; endEncrypt = ""; endDecrypt = "";

        try {
            String plaintext = "Eddy improving RC6";
            String key = "admin";

            String cipher = rc6Encrypt(plaintext, key);

            Date sK = df.parse(startKey);
            Date eK = df.parse(endKey);
            long duration = eK.getTime() - sK.getTime();
            double ms = ((double)duration)/1000.0 % 60.0;

            System.out.println("\n\ntime for key generation: ");
            System.out.println("start time: "+startKey);
            System.out.println("end time: "+endKey);
            System.out.println("diff. in time: "+ms+" secs");

            Date sE = df.parse(startEncrypt);
            Date eE = df.parse(endEncrypt);
            duration = eE.getTime() - sE.getTime();
            ms = ((double)duration)/1000.0 % 60.0;

            System.out.println("\n\ntime for encryption: ");
            System.out.println("start time: "+startEncrypt);
            System.out.println("end time: "+endEncrypt);
            System.out.println("diff. in time: "+ms+" secs");
            System.out.println("cipher: "+cipher);

            String text = rc6Decrypt(cipher, key);

            Date sD = df.parse(startDecrypt);
            Date eD = df.parse(endDecrypt);

```

```

        duration = eD.getTime() - sD.getTime();
        ms = ((double)duration)/1000.0 % 60.0;

        System.out.println("\n\ntime for decryption: ");
        System.out.println("start time: "+startDecrypt);
        System.out.println("end time: "+endDecrypt);
        System.out.println("diff. in time: "+ms+" secs");
        System.out.println("plaintext: "+text);
    } catch (Exception ex) {
        Logger.getLogger(RC6_Modified.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/*
generate key
@param password
*/

@Override
public void setKey(String password)
{
    startKey = df.format(new Date());
    try{
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(password.getBytes());
        byte[] digest = md.digest();
        String secret = DatatypeConverter.printHexBinary(digest).toString();

        byte[] key = secret.getBytes("UTF-8");
        MessageDigest sha = MessageDigest.getInstance("SHA-1");
        byte[] digestOfPassword = sha.digest(key);
        byte[] keyBytes = Arrays.copyOf(digestOfPassword, RC6_KEYLENGTH);
        secretKey = new SecretKeySpec(keyBytes, ALGORITHM_NAME);
    }
    catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    endKey = df.format(new Date());
}

```

```

/*
encrypt plaintext
@param toEncrypt
@param key
*/
@Override
public String rc6Encrypt(String toEncrypt, String key) throws Exception {
    setKey(key);
    startEncrypt = df.format(new Date());
    Cipher cipher = Cipher.getInstance(ALGORITHM_NAME + "/" + MODE_OF_OPERATION + "/" +
PADDING_SCHEME);
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);
    byte[] byteCipherText = cipher.doFinal(toEncrypt.getBytes());
    String strCipherText = new BASE64Encoder().encode(byteCipherText);
    endEncrypt = df.format(new Date());
    return strCipherText;
}

/*
decrypt ciphertext
@param encryptedText
@param key
*/
@Override
public String rc6Decrypt(String encryptedText, String key) throws Exception {
    setKey(key);
    startDecrypt = df.format(new Date());
    Cipher cipher = Cipher.getInstance(ALGORITHM_NAME + "/" + MODE_OF_OPERATION + "/" +
PADDING_SCHEME);
    cipher.init(Cipher.DECRYPT_MODE, secretKey);
    byte[] byteEncryptedText = new BASE64Decoder().decodeBuffer(encryptedText);
    byte[] decrypted = cipher.doFinal(byteEncryptedText);
    endDecrypt = df.format(new Date());
    return new String(decrypted);
}

```

```

/*
main method
*/
public static void main(String[] args){
    new RC6_Modified();
}

/*
fixes key-length permissions. to be run once at the beginning of the program
*/
public void fixKeyLength() {
String errorString = "Failed manually overriding key-length permissions.";
int newMaxKeyLength;
try {
    if ((newMaxKeyLength = Cipher.getMaxAllowedKeyLength("AES")) < 256) {
        Class c = Class.forName("javax.crypto.CryptoAllPermissionCollection");
        Constructor con = c.getDeclaredConstructor();
        con.setAccessible(true);
        Object allPermissionCollection = con.newInstance();
        Field f = c.getDeclaredField("all_allowed");
        f.setAccessible(true);
        f.setBoolean(allPermissionCollection, true);

        c = Class.forName("javax.crypto.CryptoPermissions");
        con = c.getDeclaredConstructor();
        con.setAccessible(true);
        Object allPermissions = con.newInstance();
        f = c.getDeclaredField("perms");
        f.setAccessible(true);
        ((Map) f.get(allPermissions)).put("/*", allPermissionCollection);

        c = Class.forName("javax.crypto.JceSecurityManager");
        f = c.getDeclaredField("defaultPolicy");
        f.setAccessible(true);
        Field mf = Field.class.getDeclaredField("modifiers");
        mf.setAccessible(true);

```

```

        mf.setInt(f, f.getModifiers() & ~Modifier.FINAL);
        f.set(null, allPermissions);

        newMaxKeyLength = Cipher.getMaxAllowedKeyLength("AES");
    }
} catch (Exception e) {
    throw new RuntimeException(errorString, e);
}
if (newMaxKeyLength < 256)
    throw new RuntimeException(errorString); // hack failed
}
}

```

DatabaseHelper.java

```

import Algorithm.RC6;
import Model.FileBlocks;
import Model.FileInfo;
import Model.RootKey;
import Model.ScoreData;
import Model.Student;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

```



```

/**
 *
 * @author user
 */
public class DatabaseHelper {
    private static final String classname = "com.mysql.jdbc.Driver";
    private static final String database = "DATA_AUDITOR_DB";
    private static final String url = "jdbc:mysql://localhost:3306/"+database;
    private static final String username = "root";
    private static final String password = "";

    private static final String PERSONAL = "personal";
    private static final String EDUCATION = "education";
    private static final String ASSESSMENT = "assessment";
    private static final String FILES = "files";
    private static final String AUDIT = "audit";
    private static final String DATA = "data";
    private static final String HASH = "hash";
    private static final String LOG = "log";

    private static final String PERSONAL_TABLE = "create table if not exists "+PERSONAL+" (ID int
auto_increment not null primary key, firstname varchar(50), lastname varchar(50), username varchar(50), password
varchar(50), email varchar(100), age int, state varchar(100), photo varchar(300), date datetime)";
    private static final String EDUCATION_TABLE = "create table if not exists "+EDUCATION+" (ID int
auto_increment not null primary key, user_id int, registration_no varchar(100), level varchar(10), department
varchar(200), faculty varchar(400), FOREIGN KEY (user_id) REFERENCES "+PERSONAL+"(ID) ON DELETE
CASCADE ON UPDATE CASCADE)";
    private static final String ASSESSMENT_TABLE = "create table if not exists "+ASSESSMENT+" (ID int
auto_increment not null primary key, user_id int, assignment varchar(10), test varchar(100), exam varchar(100),
total int, FOREIGN KEY (user_id) REFERENCES "+PERSONAL+"(ID) ON DELETE CASCADE ON UPDATE
CASCADE)";
    private static final String FILES_TABLE = "create table if not exists "+FILES+" (ID int auto_increment not null
primary key, user_id int, filename varchar(100), type varchar(5), date datetime, FOREIGN KEY (user_id)
REFERENCES "+PERSONAL+"(ID) ON DELETE CASCADE ON UPDATE CASCADE)";

```

```
private static final String AUDIT_TABLE = "create table if not exists "+AUDIT+" (ID int auto_increment not null primary key, file_id int, user_id int, pk TEXT, type varchar(5), date datetime, FOREIGN KEY (user_id) REFERENCES "+PERSONAL+(ID) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (file_id) REFERENCES "+FILES+(ID) ON DELETE CASCADE ON UPDATE CASCADE);";
```

```
private static final String DATA_TABLE = "create table if not exists "+DATA+" (ID int auto_increment not null primary key, file_id int, sn int, block LONGTEXT, FOREIGN KEY (file_id) REFERENCES "+FILES+(ID) ON DELETE CASCADE ON UPDATE CASCADE);";
```

```
private static final String HASH_TABLE = "create table if not exists "+HASH+" (SN int auto_increment not null primary key, file_id int, id int, hashdata TEXT, FOREIGN KEY (file_id) REFERENCES "+FILES+(ID) ON DELETE CASCADE ON UPDATE CASCADE);";
```

```
private static final String LOG_TABLE = "create table if not exists "+LOG+" (SN int auto_increment not null primary key, user varchar(100), activity varchar(100), status varchar(50), date datetime);";
```

```
private Connection connection;
```

```
private RC6 rc6;
```

```
public DatabaseHelper(RC6 rc6){
```

```
    this.rc6 = rc6;
```

```
    try{
```

```
        Class.forName(classname);
```

```
        //creating database file if not exist
```

```
        connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mysql", username, password);
```

```
        PreparedStatement pst = connection.prepareStatement("create database if not exists "+database);
```

```
        pst.execute();
```

```
        pst.close();
```

```
        connection = DriverManager.getConnection(url, username, password);
```

```
        Statement stmt = connection.createStatement();
```

```
        stmt.execute(PERSONAL_TABLE);
```

```
        stmt.execute(EDUCATION_TABLE);
```

```
        stmt.execute(ASSESSMENT_TABLE);
```

```
        stmt.execute(FILES_TABLE);
```

```
        stmt.execute(AUDIT_TABLE);
```

```
        stmt.execute(DATA_TABLE);
```

```
        stmt.execute(HASH_TABLE);
```

```
        stmt.execute(LOG_TABLE);
```

```

        stmt.close();
    }catch(ClassNotFoundException c){
        JOptionPane.showMessageDialog(null, c.getLocalizedMessage());
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

```

```

public boolean isStudent(String regNo){
    String sql = "select * from "+EDUCATION+" where registration_no = '"+regNo+"' ";
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            if(regNo.equalsIgnoreCase(rs.getString("registration_no")))
                return true;
        }
        rs.close();
        stmt.close();
    }catch(SQLException sq)
    {
        sq.printStackTrace();
    }
    return false;
}

```

```

public boolean isUser(String username, String password){
    String sql = "select * from "+PERSONAL+" where username = '"+username+"' and password = '"+password+"' ";
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            if(username.equalsIgnoreCase(rs.getString("username")) && password.equals(rs.getString("password")))
                return true;
        }
        rs.close();
    }
}

```

```

        stmt.close();
    }catch(SQLException sq)
    {
        sq.printStackTrace();
    }
    return false;
}

/*

*/

public int addStudentPersonalInfo(String firstname, String lastname, String username, String password, String
email, String age, String state){
    String sql = "insert into "+PERSONAL+" (firstname, lastname, username, password, email, age, state, photo,
date) values("+firstname+", "+lastname+", "+username+", "+password+", "+email+", "+age+", "+state+", ",
now())";

    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
    return getLastId(PERSONAL, "id");
}

private int getLastId(String table, String id){
    String sql = "select * from "+table;
    int sn = 1;
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        if(rs.last()){
            sn = rs.getInt(id);
        }
        rs.close();
    }
}

```

```

        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
    return sn;
}

public void addStudentEducationInformation(int user_id, String level, String department, String faculty, String
registration_no){
    String sql = "insert into "+EDUCATION+" (user_id, level, department, faculty, registration_no)
values("+user_id+", "+level+", "+department+", "+faculty+", "+registration_no+")";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void addStudentAssessmentInformation(int user_id, int assignment, int test, int exams){
    int total = assignment + test + exams;
    String sql = "insert into "+ASSESSMENT+" (user_id, assignment, test, exam, total) values("+user_id+",
"+assignment+", "+test+", "+exams+", "+total+")";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public int addFileToStudentAccount(int user_id, String filename, String type){
    String sql = "insert into "+FILES+" (user_id, filename, type, date) values("+user_id+", "+filename+",
"+type+", now())";

```

```

try{
    Statement stmt = connection.createStatement();
    stmt.executeUpdate(sql);
    stmt.close();
}catch(SQLException s){
    JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
}
return getLastId(FILE, "id");
}

public void addAuditFile(int file_id, int user_id, String pk, String type){
    String sql = "insert into "+AUDIT+" (file_id, user_id, pk, type, date) values("+file_id+", "+user_id+", "+pk+",
"+type+", now())";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void addHash(int file_id, int index, String hash){
    String sql = "insert into "+HASH+" (file_id, id, hashdata) values("+file_id+", "+index+", "+hash+"));";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void addBlock(int file_id, int id, String block){
    String sql = "insert into "+DATA+" (file_id, sn, block) values("+file_id+", "+id+", "+block+"));";
    try{
        Statement stmt = connection.createStatement();

```

```

        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        s.printStackTrace();
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void addLog(String user, String activity, String status){
    String sql = "insert into "+LOG+" (user, activity, status, date) values('"+user+"', '"+activity+"', '"+status+"',
now())";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        s.printStackTrace();
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public Student getStudent(String email){
    Student student = new Student();
    String sql = "select "+PERSONAL+".firstname, "+PERSONAL+".lastname, "+PERSONAL+".username,
"+EDUCATION+".department, "+EDUCATION+".faculty, "+EDUCATION+".level, "
        +PERSONAL+".age, "+PERSONAL+".email, "+PERSONAL+".state, "+PERSONAL+".photo,
"+EDUCATION+".registration_no, "+PERSONAL+".id from "+PERSONAL+" left join "+EDUCATION+" on
"+PERSONAL+".id = "+EDUCATION+".user_id"
        + " where "+PERSONAL+".username = '"+email+"'";
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            student.setFirstname(rs.getString(1));
            student.setLastname(rs.getString(2));
            student.setUsername(rs.getString(3));
            student.setDept(rs.getString(4));

```

```

        student.setFaculty(rs.getString(5));
        student.setLevel(rs.getString(6));
        student.setAge(rs.getString(7));
        student.setEmail(rs.getString(8));
        student.setState(rs.getString(9));
        student.setPhoto(rs.getString(10));
        student.setRegNo(rs.getString(11));
        student.setUser_id(rs.getInt(12));
    }
    rs.close();
    stmt.close();
} catch(SQLException s){
    JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
}
return student;
}

```

```

public int getNumberOfBlocks(){
    String sql = "select * from "+DATA;
    int total = 0;
    try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next()){
            total += 1;
        }
        rs.close();
        st.close();
    } catch(SQLException sq){
        sq.printStackTrace();
    }
    return total;
}

```

```

public int getFileId(String filename){
    String sql = "select * from "+FILES+" where filename = '"+filename+"'";
    int id = -1;

```



```

try{
    Statement st = connection.createStatement();
    ResultSet rs = st.executeQuery(sql);
    while(rs.next()){
        id = rs.getInt("id");
    }
    rs.close();
    st.close();
}catch(SQLException s){
    JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
}
return id;
}

```

```

public String getRootKey(int file_id){
    String sql = "select * from "+AUDIT+" where file_id = "+file_id+"";
    String key = "";
    try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next()){
            key = rs.getString("pk");
        }
        rs.close();
        st.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
    return key;
}

```

```

public String getRegNo(int user_id){
    String sql = "select * from "+EDUCATION+" where user_id = "+user_id+"";
    String key = "";
    try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);

```

```

        while(rs.next()){
            key = rs.getString("registration_no");
        }
        rs.close();
        st.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
    return key;
}

```

```

public ScoreData getScoreData(int user_id){
    String sql = "select * from "+ASSESSMENT+" where user_id = "+user_id+"";
    ScoreData score = new ScoreData();

    try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next()){
            score.setAssignment(rs.getString("assignment"));
            score.setTest(rs.getString("test"));
            score.setExams(rs.getString("exam"));
        }
        rs.close();
        st.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
    return score;
}

```

```

public int getNumberOfUsers(){
    String sql = "select * from "+PERSONAL;
    int total = 0;
    try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);

```

```

        while(rs.next()){
            total += 1;
        }
        rs.close();
        st.close();
    }catch(SQLException sq){
        sq.printStackTrace();
    }
    return total;
}

```

```

public int getNumberOfFiles(){
    String sql = "select * from "+FILES;
    int total = 0;
    try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next()){
            total += 1;
        }
        rs.close();
        st.close();
    }catch(SQLException sq){
        sq.printStackTrace();
    }
    return total;
}

```

```

public ArrayList<String> getFiles(){
    String sql = "select * from "+FILES;
    ArrayList<String> files = new ArrayList<>();
    try{
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while(rs.next()){
            files.add(rs.getString("filename"));
        }
    }
}

```

```

        rs.close();
        st.close();
    }catch(SQLException sq){
        sq.printStackTrace();
    }
    return files;
}

public List<ScoreData> getStudentsScoresByEducation(String level, String dept, String faculty){
    List<ScoreData> data = new ArrayList<>();
    String sql = "select "+PERSONAL+".id, "+PERSONAL+".firstname, "+PERSONAL+".lastname,
"+ASSESSMENT+".assignment, "+ASSESSMENT+".test, "+ASSESSMENT+".exam "
        + " from "+PERSONAL+" left join "+ASSESSMENT+" on "+PERSONAL+".id =
"+ASSESSMENT+".user_id JOIN "+EDUCATION+" on "+PERSONAL+".id = "+EDUCATION+".user_id "
        + " where "+EDUCATION+".level = '"+level+"' AND "+EDUCATION+".department = '"+dept+"' AND
"+EDUCATION+".faculty = '"+faculty+"' ";
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            ScoreData obj = new ScoreData();
            obj.setId((int)rs.getObject(1));
            obj.setFirstname((String)rs.getObject(2));
            obj.setLastname((String)rs.getObject(3));
            obj.setAssignment((String)rs.getObject(4));
            obj.setTest((String)rs.getObject(5));
            obj.setExams((String)rs.getObject(6));

            data.add(obj);
        }
        rs.close();
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
    return data;
}

```

```

public List<RootKey> getAuditData(){
    String sql = "select * from "+AUDIT;
    List<RootKey> data = new ArrayList<>();
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            RootKey key = new RootKey(rs.getInt("id"), rs.getInt("file_id"), rs.getInt("user_id"), rs.getString("pk"),
(String)rs.getObject("date").toString(), rs.getString("type"));
            data.add(key);
        }
        rs.close();
        stmt.close();
    }catch(SQLException sq){
        JOptionPane.showMessageDialog(null, sq.getLocalizedMessage());
    }
    return data;
}

public List<FileInfo> getAllFiles(){
    String sql = "select * from "+FILES+" where type = 'F'";
    List<FileInfo> data = new ArrayList<>();
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            FileInfo key = new FileInfo(rs.getInt("id"), rs.getInt("user_id"), rs.getString("type"),
rs.getString("filename"), (String)rs.getObject("date").toString());
            data.add(key);
        }
        rs.close();
        stmt.close();
    }catch(SQLException sq){
        JOptionPane.showMessageDialog(null, sq.getLocalizedMessage());
    }
    return data;
}

```

```
}
```

```
public String getFile(int file_id){  
    String sql = "select * from "+DATA+" where file_id = "+file_id+"";  
    StringBuilder sb = new StringBuilder();  
    try{  
        Statement stmt = connection.createStatement();  
        ResultSet rs = stmt.executeQuery(sql);  
        while(rs.next()){  
            sb.append(rc6.rc6Decrypt(rs.getString("block"), "admin"));  
        }  
        rs.close();  
        stmt.close();  
    }catch(Exception e){  
        JOptionPane.showMessageDialog(null, e.getLocalizedMessage());  
    }  
    return sb.toString();  
}
```

```
public List<RootKey> getAuditData(int file_id){  
    String sql = "select * from "+HASH+" where file_id = "+file_id+"";  
    List<RootKey> data = new ArrayList<>();  
    try{  
        Statement stmt = connection.createStatement();  
        ResultSet rs = stmt.executeQuery(sql);  
        while(rs.next()){  
            RootKey key = new RootKey(rs.getInt("id"), rs.getInt("file_id"), rs.getInt("sn"), rs.getString("hashdata"));  
            data.add(key);  
        }  
        rs.close();  
        stmt.close();  
    }catch(SQLException sq){  
        JOptionPane.showMessageDialog(null, sq.getLocalizedMessage());  
    }  
    return data;  
}
```

```

public List<FileBlocks> getFileBlocks(int file_id){
    String sql = "select * from "+DATA+" where file_id = "+file_id+"";
    List<FileBlocks> data = new ArrayList<>();
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            FileBlocks blocks = new FileBlocks(rs.getInt("sn"), rs.getInt("file_id"), rs.getString("block"));
            data.add(blocks);
        }
        rs.close();
        stmt.close();
    }catch(SQLException sq){
        JOptionPane.showMessageDialog(null, sq.getLocalizedMessage());
    }
    return data;
}

```

```

public final DefaultTableModel getTable(String table){
    Vector columnNames = new Vector(), data = new Vector();
    columnNames.clear();
    data.clear();
    try {
        Statement stmtt = connection.createStatement();
        ResultSet rs = stmtt.executeQuery("select * from "+table);
        ResultSetMetaData mdx = rs.getMetaData();
        int columnsx = mdx.getColumnCount();
        for (int i = 1; i <= columnsx; i++) {
            columnNames.addElement(mdx.getColumnName(i).toUpperCase());
        }
        while (rs.next()) {
            Vector rowx = new Vector(columnsx);
            for (int i = 1; i <= columnsx; i++) {
                rowx.addElement(rs.getObject(i));
            }
            data.addElement(rowx);
        }
    }
}

```

```

rs.close();
stmt.close();

} catch (SQLException ex) {

    JOptionPane.showMessageDialog(null, ex, "Report", JOptionPane.ERROR_MESSAGE);
}

return new DefaultTableModel(data, columnNames);
}

public final DefaultTableModel getStudentAssessment(){
    String sql = "select "+PERSONAL+".firstname, "+PERSONAL+".lastname, "+ASSESSMENT+".assignment,
"+ASSESSMENT+".test, "+ASSESSMENT+".exam, "+ASSESSMENT+".total, "
        +EDUCATION+".department, "+EDUCATION+".faculty from "+PERSONAL+" left join
"+ASSESSMENT+" on "+PERSONAL+".id = "+ASSESSMENT+".user_id JOIN "+EDUCATION+" on
"+PERSONAL+".id = "+EDUCATION+".user_id ";

    Vector columnNames = new Vector(), data = new Vector();
    columnNames.clear();
    data.clear();
    try {
        Statement stmtt = connection.createStatement();
        ResultSet rs = stmtt.executeQuery(sql);
        String[] metadata = new String[]{"S/N", "Firstname", "Lastname", "Assignment", "Test", "Exams", "Total",
"Department", "Faculty"};
        for (int i = 0; i < metadata.length; i++) {
            columnNames.addElement(metadata[i].toUpperCase());
        }
        int x = 1;
        while (rs.next()) {
            Vector rowx = new Vector(metadata.length);
            rowx.addElement(x);
            for (int i = 1; i <= metadata.length - 1; i++) {
                rowx.addElement(rs.getObject(i));
            }

```



```

        data.addElement(rowx);
        x++;
    }
    rs.close();
    stmt.close();

} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, ex, "Report", JOptionPane.ERROR_MESSAGE);
}

return new DefaultTableModel(data, columnNames);
}

```

```

public boolean isAssessment(int id){
    String sql = "select * from "+ASSESSMENT+" where user_id = "+id+"";
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            if(id == (rs.getInt("user_id")))
                return true;
        }
        rs.close();
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
    return false;
}

```

```

public boolean isFile(String filename){
    String sql = "select * from "+FILES+" where filename = '"+filename+"";
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            if(filename.equals(rs.getString("filename")))

```

```

        return true;
    }
    rs.close();
    stmt.close();

} catch(SQLException s){
    JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
}
return false;
}

public void deleteFile(int id){
    String sql = "delete from "+FILES+" where id = "+id+"";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
        JOptionPane.showMessageDialog(null, "File deleted");
    } catch(SQLException sq){
        JOptionPane.showMessageDialog(null, sq.getLocalizedMessage());
    }
}

public void deleteStudent(int id){
    String sql = "delete from "+PERSONAL+" where id = "+id+"";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
        JOptionPane.showMessageDialog(null, "Student and its details deleted");
    } catch(SQLException sq){
        JOptionPane.showMessageDialog(null, sq.getLocalizedMessage());
    }
}

public void clearLog(){
    String sql = "TRUNCATE TABLE "+LOG;

```

```

try{
    Statement stmt = connection.createStatement();
    stmt.executeUpdate(sql);
    stmt.close();
}catch(SQLException sq){
    JOptionPane.showMessageDialog(null, sq.getLocalizedMessage());
}
}

public void updateAssessment(int id, String assignment, String test, String exams, int total){
    String sql = "UPDATE "+ASSESSMENT+" SET assignment = '"+assignment+"', test = '"+test+"', exam =
"+exams+"', total = '"+total+"' WHERE user_id = "+id+"";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void updateBlock(int id, int file_id, String block){
    String sql = "UPDATE "+DATA+" SET block = '"+block+"' WHERE sn = "+id+" AND file_id = "+file_id+
";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void updatePhotoPath(int user_id, String path){
    String sql = "UPDATE "+PERSONAL+" SET photo = '"+path+"' where id = "+user_id+"";
    try{
        Statement stmt = connection.createStatement();

```

```

        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void updateRootKey(int file_id, String data){
    String sql = "UPDATE "+AUDIT+" SET pk = '"+data+"' where file_id = "+file_id+"";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
        //JOptionPane.showMessageDialog(null, classname);
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void updateHash(int id, int file_id, String hashdata){
    String sql = "UPDATE "+HASH+" SET hashdata = '"+hashdata+"' WHERE id = "+id+" AND file_id = "+file_id+"";
    try{
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(sql);
        stmt.close();
    }catch(SQLException s){
        JOptionPane.showMessageDialog(null, s.getLocalizedMessage());
    }
}

public void closeConnection(){
    try {
        connection.close();
    } catch (SQLException ex) {
        Logger.getLogger(DatabaseHelper.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

```
}  
}
```

Student.java

```
/**  
 *  
 * @author user  
 */  
public class Student {  
    private int user_id;  
    private String firstname, lastname, username, password, email;  
    private String state, dept, faculty, level, regNo, age, photo;  
  
    public Student(){  
  
    }  
  
    public Student(int user_id, String firstname, String lastname, String username, String password, String email,  
String state, String dept, String faculty, String level, String regNo, String age, String photo) {  
        this.user_id = user_id;  
        this.firstname = firstname;  
        this.lastname = lastname;  
        this.username = username;  
        this.password = password;  
        this.email = email;  
        this.state = state;  
        this.dept = dept;  
        this.faculty = faculty;  
        this.level = level;  
        this.regNo = regNo;  
        this.age = age;  
        this.photo = photo;
```

```
}

public int getUser_id() {
    return user_id;
}

public void setUser_id(int user_id) {
    this.user_id = user_id;
}

public String getFirstname() {
    return firstname;
}

public void setFirstname(String firstname) {
    this.firstname = firstname;
}

public String getLastname() {
    return lastname;
}

public void setLastname(String lastname) {
    this.lastname = lastname;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}
```

```
public void setPassword(String password) {  
    this.password = password;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```
public void setEmail(String email) {  
    this.email = email;  
}
```

```
public String getState() {  
    return state;  
}
```

```
public void setState(String state) {  
    this.state = state;  
}
```

```
public String getDept() {  
    return dept;  
}
```

```
public void setDept(String dept) {  
    this.dept = dept;  
}
```

```
public String getFaculty() {  
    return faculty;  
}
```

```
public void setFaculty(String faculty) {  
    this.faculty = faculty;  
}
```

```
public String getLevel() {
    return level;
}

public void setLevel(String level) {
    this.level = level;
}

public String getRegNo() {
    return regNo;
}

public void setRegNo(String regNo) {
    this.regNo = regNo;
}

public String getAge() {
    return age;
}

public void setAge(String age) {
    this.age = age;
}

public String getPhoto() {
    return photo;
}

public void setPhoto(String photo) {
    this.photo = photo;
}

}
```

ScoreData.java


```

/**
 *
 * @author user
 */
public class ScoreData {
    private int id;
    private String firstname, lastname;
    private String assignment, test, exams;

    public ScoreData(){

    }

    public ScoreData(int id, String firstname, String lastname, String assignment, String test, String exams) {
        this.id = id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.assignment = assignment;
        this.test = test;
        this.exams = exams;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
}

```

```
public String getLastname() {
    return lastname;
}

public void setLastname(String lastname) {
    this.lastname = lastname;
}

public String getAssignment() {
    return assignment;
}

public void setAssignment(String assignment) {
    this.assignment = assignment;
}

public String getTest() {
    return test;
}

public void setTest(String test) {
    this.test = test;
}

public String getExams() {
    return exams;
}

public void setExams(String exams) {
    this.exams = exams;
}
}
```

RootKey.java

```
/**
 *
 * @author user
 */
public class RootKey {
    private int id, file_id;
    private String datakey;
    private String date;
    private int user_id;
    private String type;

    public RootKey(){

    }

    public RootKey(int id, int file_id, int user_id, String datakey, String date, String type) {
        this.id = id;
        this.file_id = file_id;
        this.datakey = datakey;
        this.date = date;
        this.user_id = user_id;
        this.type = type;
    }

    public RootKey(int id, int file_id, int user_id, String datakey) {
        this.id = id;
        this.file_id = file_id;
        this.datakey = datakey;
        this.user_id = user_id;
        //this.type = type;
    }

    public int getId() {
        return id;
    }
}
```

```
public void setId(int id) {
    this.id = id;
}

public String getDatakey() {
    return datakey;
}

public void setDatakey(String datakey) {
    this.datakey = datakey;
}

public String getDate() {
    return date;
}

public void setDate(String date) {
    this.date = date;
}

public int getUser_id() {
    return user_id;
}

public void setUser_id(int user_id) {
    this.user_id = user_id;
}

public int getFile_id() {
    return file_id;
}

public void setFile_id(int file_id) {
    this.file_id = file_id;
}

public String getType() {
```

```
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
}
```

FileBlocks.java

```
/**
 *
 * @author user
 */
public class FileBlocks {
    private int id;
    private int file_id;
    private String blocks;

    public FileBlocks(){

    }

    public FileBlocks(int id, int file_id, String blocks) {
        this.id = id;
        this.file_id = file_id;
        this.blocks = blocks;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
```

```
        this.id = id;
    }

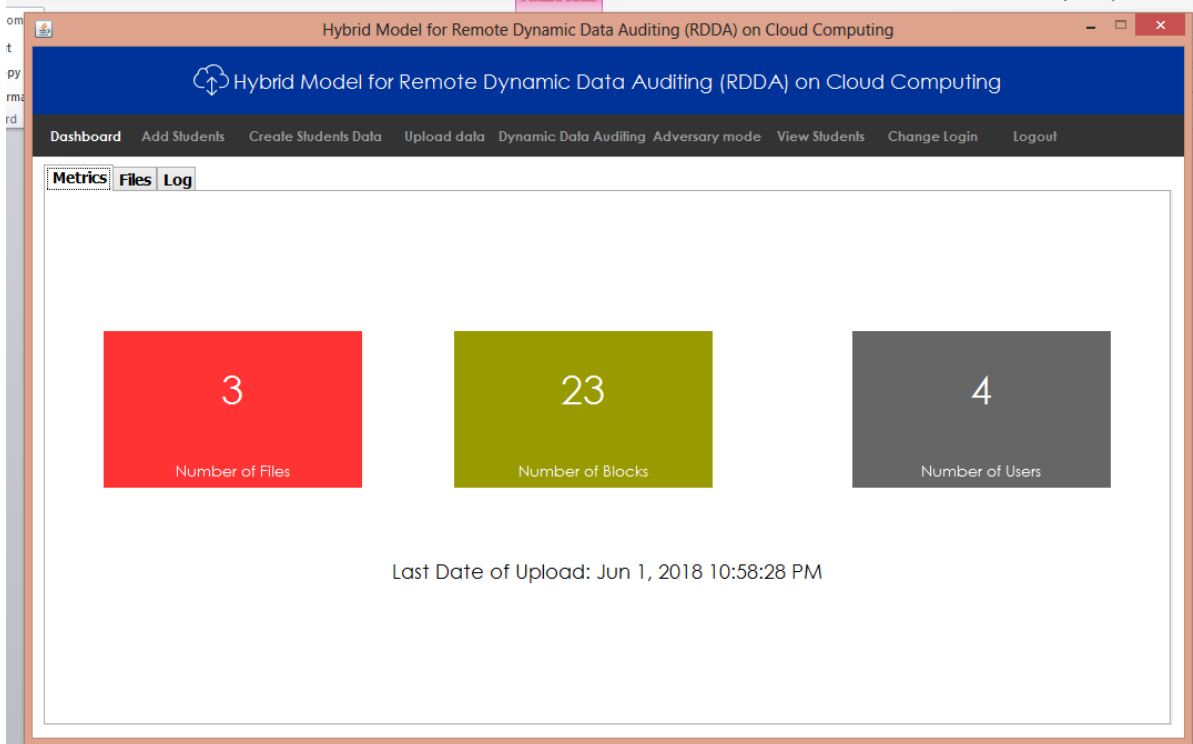
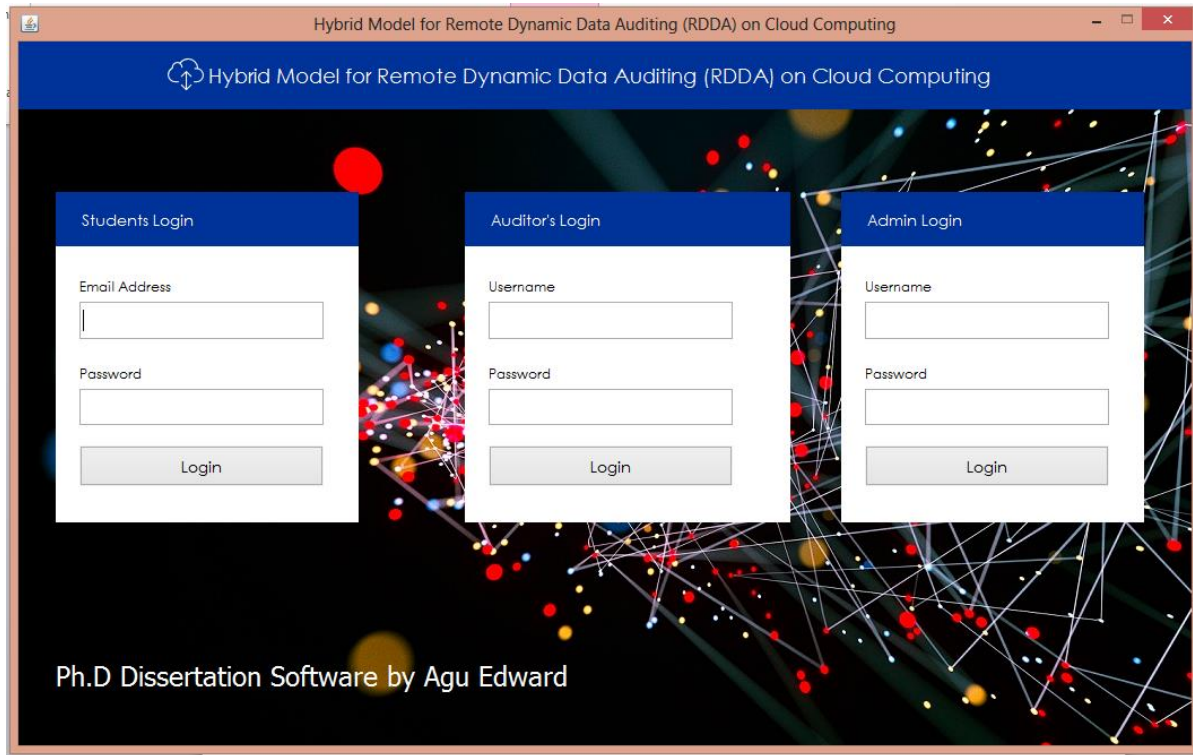
    public int getFile_id() {
        return file_id;
    }

    public void setFile_id(int file_id) {
        this.file_id = file_id;
    }

    public String getBlocks() {
        return blocks;
    }

    public void setBlocks(String blocks) {
        this.blocks = blocks;
    }
}
```

Appendix B: Sample Outputs



Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard **Add Students** Create Students Data Upload data Dynamic Data Auditing Adversary mode View Students Change Login Logout

Add Student

Firstname <input type="text" value="Emenike"/>	Faculty <input type="text" value="Information and Communication Technology"/>
Lastname <input type="text" value="Celine"/>	Department <input type="text" value="Computer Science"/>
Email Address <input type="text" value="celine@yahoo.com"/>	Level <input type="text" value="400L"/>
State of Origin <input type="text" value="Imo"/>	Password <input type="text" value="*****"/>
Age <input type="text" value="28"/>	Registration Number <input type="text" value="UR20130001"/>

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard **Add Students** Create Students Data Upload data Dynamic Data Auditing Adversary mode View Students Change Login Logout

Add Student

Firstname <input type="text"/>	Faculty <input type="text" value="Information and Communication Technology"/>
Lastname <input type="text"/>	Department <input type="text" value="Computer Science"/>
Email Address <input type="text"/>	Level <input type="text"/>
State of Origin <input type="text" value="Abia"/>	Password <input type="text"/>
Age <input type="text"/>	Registration Number <input type="text"/>

Message ✕

New student registered successfully

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Add Students **Create Students Data** Upload data Dynamic Data Auditing Adversary mode View Students Change Login Logout

ID	Firstname	Lastname	Assignment	Test	Exam	
4	Agu	Daniel	<input type="text" value="8"/>	<input type="text" value="17"/>	<input type="text" value="62"/>	<input type="button" value="Update"/>

Message

Success

<-- Back

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Add Students Create Students Data **Upload data** Dynamic Data Auditing Adversary mode View Students Change Login Logout

Student File Upload

Select File and Upload

Select Department

Upload Input

C:\Users\Edward Agu\Documents\Thesis\PhD These\Original Thesis\Original Thesis\Corrected Thesis\Test Data\Eddy.docx

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Add Students Create Students Data **Upload data** Dynamic Data Auditing Adversary mode View Students Change Login Logout

Student File Upload

Select File and Upload

Message

File uploaded successfully, and its root data key generated

OK

Upload

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Add Students Create Students Data Upload data **Dynamic Data Auditing** Adversary mode View Students Change Login Logout

Dynamic Data Auditing

1 - F	2018-06-01 22:52:35.0	9de036e78566adb0c109cd21821351263a4838d5af30bc171774c69c3f2834f	Modify	Delete
2 - F	2018-06-01 22:57:43.0	8ce3516234601ec2357f2a39bcfbac9e3a15dfeed8d1e13a420dec2aeafd66	Modify	Delete
3 - F	2018-06-01 23:08:10.0	29f62dad0a6ba754f6186b398fb6632eb9d5726791c09617b984700329c3d8d	Modify	Delete
4 - S	2018-06-01 23:34:09.0	41ed562f377c2ae87bad2b9fe5133a2765f3cd238f24623159691f62f346fca	Modify	Delete
5 - S	2018-06-01 23:35:45.0	775d3ab8b072069353a128f106d14abdca14a287c0b2b17364fa59ac246f920	Modify	Delete
6 - S	2018-06-01 23:36:14.0	43fe4937cf4739f15f29f9467c48ebb766b78dbed5a5c4e57084f598f2e39bca	Modify	Delete
7 - S	2018-06-01 23:36:39.0	37530cde1be31201ce1c9e86d73fca6b67f94b651cbce0a0039d4cec1338a7	Modify	Delete

Data Editing

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Add Students Create Students Data Upload data Dynamic Data Auditing **Adversary mode** View Students Change Login Logout

SN	FILE ID	ROOT KEY	
1	1	9de036e78566adb0c109cd21821351263a4838d5af30bc171774c69c3f2834f	Alter
2	2	158ce3516234601ec2357f2a39bcfbac9e3a15dfeed8d1e13a420dec2aeafd66	Alter
3	3	329f62dad0a6ba754f6186b398fb6632eb9d5726791c09617b984700329c3d8d	Alter
4	4	41ed562f377c2ae87bad2b9fe5133a2765f3cd238f24623159691f62f346fca	Alter
5	5	2775d3ab8b072069353a128f106d14abdca14a287c0b2b17364fa59ac246f920	Alter
6	6	43fe4937cf4739f15f29f9467c48ebb766b78dbed5a5c4e57084f598f2e39bca	Alter
7	7	2c37530cde1be31201ce1c9e86d73fca6b67f94b651cbce0a0039d4cec1338a7	Alter

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Add Students Create Students Data Upload data Dynamic Data Auditing Adversary mode **View Students** Change Login Logout

S/N	FIRSTNAME	LASTNAME	ASSIGNMENT	TEST	EXAMS	TOTAL	DEPARTMENT	FACULTY
1	Emenike	Celine	5	18	62	85	Computer Science	Information and Communication Technology
2	Agu	David	8	19	67	94	Computer Science	Information and Communication Technology
3	Agu	Daniella	7	18	53	78	Computer Science	Information and Communication Technology
4	Agu	Daniel	8	17	62	87	Computer Science	Information and Communication Technology

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Check Data Authenticity Change Login Logout

35

Number of Blocks

Sat Jun 02 01:30:45 WAT 2018

Last Date of Check

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Check Data Authenticity Change Login Logout

Enter filename/student registration number

Eddy.txt

hKE
1fARtoMWJMF1HhUw+W6yZ/WB+19DnrvfdQ
u0Jefkh2QqXw80ZRez7K+mZVP3GuSb6BDA
37IMv1
4ZPKfmuef3cuhw==

KB8TyQ9bp3V0BpRrY3Q/NxpM7H14xPHAVq
817rghX+yrDjWwXcKSP1qPcesmJq6MfPd
quUyZwchX1wLmS3Uj0GMeapTV8b7+znIckfj
a
kBDd5GSnKenA==

1LuHRtg
kcp7l0ajhYLRlgwUEk32H14QYifdPRRTCAU
hg25/Q8xqY6xRvH5tgMg/ESKx7FMlro/ttt
9
BsgXczsfMqzlg==

5DyIqxn8G6ByNwvy0c
vfdU6qQ4wYdWxJ6Sstdu0MJGZZLz
8Aq7RvUj73+/1YWOvQDw3WENKUV
XvCKNmz/1fxoh62wvB
uH43lv958QwhwQ==

MpBySc
h024/E8xWnbett+AUF1811UD1Y4E2c2qMh
1oqE6cqJTPLB4++i04+8jD58H99chaEVq
x
zqqo1pKoiC35Lw==

q/+3ytdsNk9A29ZBK8cVYUHu4JF29M6L3Qs8
rbCTLG3BwVSDVY8h5q9ceUjMa5mpmMLwUW
ke1RW5YH9p0o4oz249h5AazV89ZRCzGOpJ
C4pweV5xw13D761t29akVF86W/GZQZt55b5
B+2vY25IOJ/gUA==

PBWwE
Xbf0Mv/8p6ibYvGWwVEE903VOvHctBazyG
woicg1LwX3E8HbPeYMuJgQAQE1Mn4Q1u
NCs5Rk
8xbsTg2XstZQ==

6L4BBHGoruk77onxScw7qA==

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Check Data Authenticity Change Login Logout

Enter filename/student registration number
Eddy.txt Search

hKE
1faRtoMWJMF1HjW+W6yZ/WB+19DvfdQ
u0Jefxh2QqXw80ZReZ7K+mZVP3Gu5b68Da
37Mwi
4ZPKfmuef3cuhw==

KB8TYQ9bp3W0BphRY3Q/NxpN7H14xvHAVq
817gHX+yt1JWHwXcpKSPWqPaesm/dqkMfJgd
quUyIZwchX1wLn53UJGMeaPtVbB7+znIckFj
a
kTBdd5GSnKenA==

1LuHRTg
kxpTloJhYLRlgwUEK32H14YfdPRRTCAU
hG25/Q8qY6xRvH5tgMg/ESkx7FFMro/ttt
9
BsgXczsFMNqzlg==

SDyIaxnBGeByltwyy0c
vfdRU6a04wYdWlnJ6Stddu0MJGZLz
8Aq7RWUj73+1yW0VQDw3WENKUV
XvCKNmbv1fxoh62wp8
uH43lv958QwhwQ==

MpBy9C
h024/EBxWNBett+AUf1811UDrY4E2c2qMh
10qjE6CqjTPLB4r+H104+8jD58fH9ChEfvQ
x
zgqo1pKoC35Lw==

q/+3yrd5Nk0A2QZ8K8c1YluHu4JFZ9M6L3Q8
rbCfL63BwVSDVY8hSg9cseuIMa5nznMluWn
ke1RW5YHRpa004oZg4DhSaaZVRj9Z/RcZGopJ
C4piveVSw03D6tlt29akVf86WfGZx2tS6S5
B+ZvY2SiOS/gUA==

PBWWE
XbF0Mv/8p6BbYvGWVwEEa03V0vHctBazyG
woiCq1Lw3E8NBPeYMuIqQAQEu1Mn4Q1u
Nca5Rik
9xbsTg2XsclZQ==

6L48BHGoRuk77onScW7qA==

Audit Clear

9de036e78566adbfc109cd21821351263a4838d5af30bc171774c69c3f2834f

11d2063277d9b674d3d669717875eae6a7e29c4a8a66b1eae7e048f2e7841 | 0ae0124f483b577bb988400ba8a1c09844fe33162cdd293b5945097598f9c21a

e8e1c150b1c317fcb40d1ae67173a228e4127e8c3291d43aff8a5f0eb20f50a | 99341cf88822f1826ea4340283ba87b2551c5c5287f590

o/ttt9 BsgXczsFMNqzlg== | jB uH43lv958QwhwQ==


ORIGINAL KEY: 9de036e78566adbfc109cd21821351263a4838d5af30bc171774c69c3f2834f

Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Logout

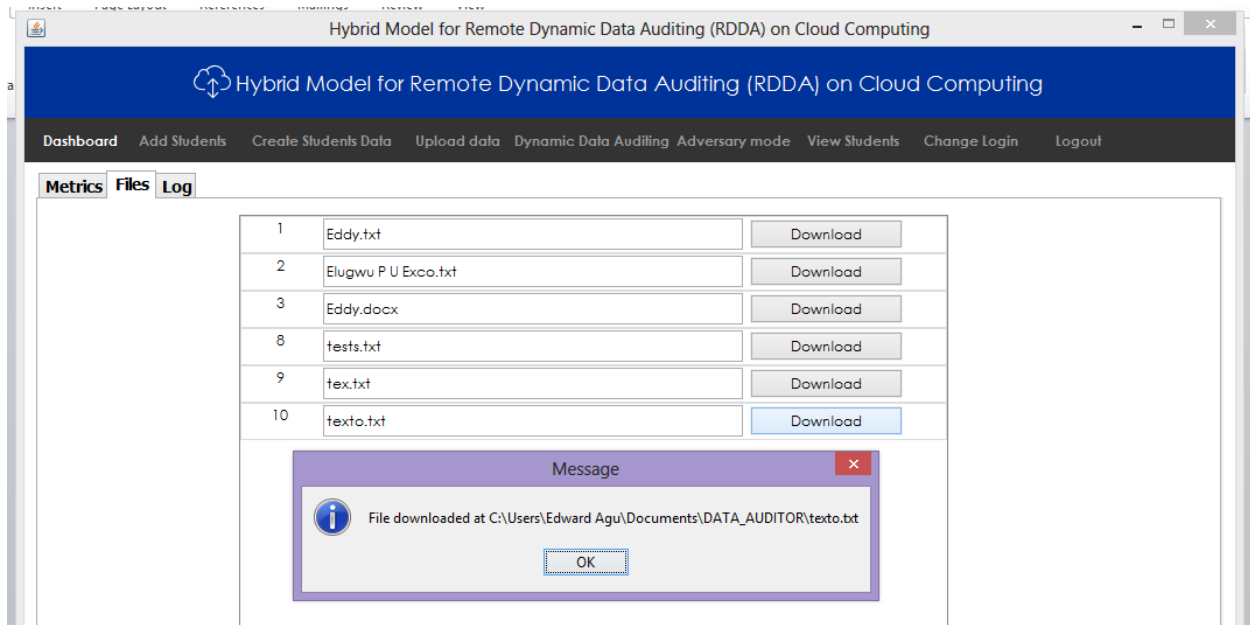
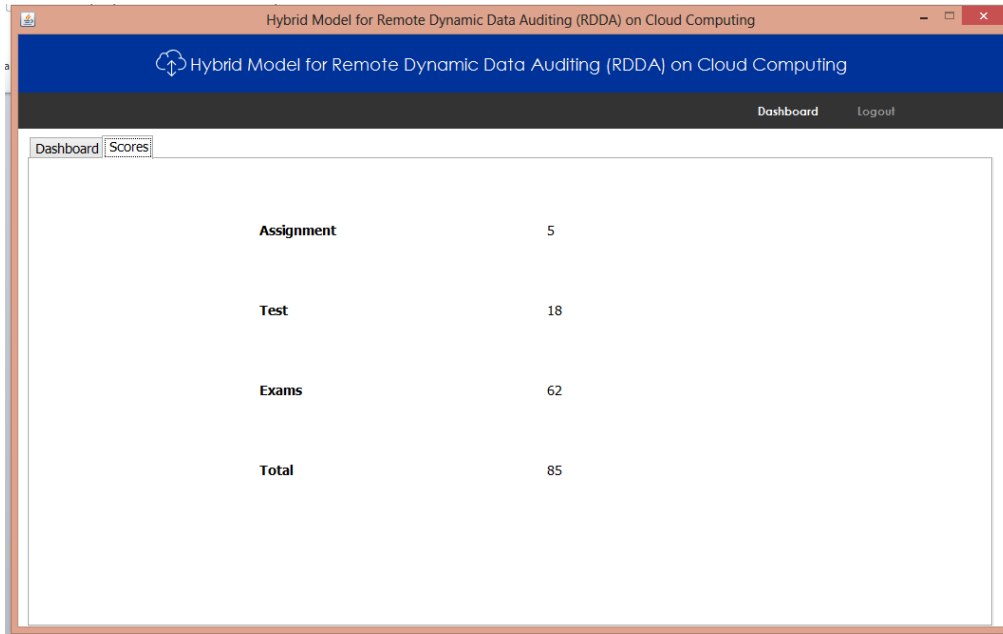
Dashboard Scores

Student Dashboard



Change Photo

First Name	Emenike
Last Name	Celine
Department	Computer Science
Faculty	Information and Communication Technology
Level	400L
Email Address	celine@yahoo.com
Age	28
State of Origin	Imo
Reg. Number	UR20130001



Hybrid Model for Remote Dynamic Data Auditing (RDDA) on Cloud Computing

Dashboard Add Students Create Students Data Upload data Dynamic Data Auditing Adversary mode View Students Change Login Logout

Metrics Files Log

SN	USER	ACTIVITY	STATUS	DATE
1	admin	Login attempt	Success	2018-05-26 20:38:19.0
2	admin	Login attempt	Success	2018-05-27 02:06:17.0
3	admin	Login attempt	Success	2018-06-01 22:14:56.0
4	admin	Login attempt	Success	2018-06-01 22:37:11.0
5	admin	Login attempt	Success	2018-06-01 23:06:26.0
6	admin	Login attempt	Success	2018-06-02 01:05:21.0
7	admin	Login attempt	Success	2018-06-02 01:21:16.0
8	admin	Login attempt	Success	2018-06-02 01:26:49.0
9	admin	Login attempt	Success	2018-06-02 01:37:13.0
10	admin	Login attempt	Success	2018-06-02 02:09:53.0
11	admin	Login attempt	Success	2018-06-03 04:14:22.0

