UNIVERSITY OF NAIROBI

School of Computing and Informatics

IMPLEMENTING DATA INTEGRITY AND CONFIDENTIALITY IN MOBILE PHONE BASED CYBER FORAGING SYSTEM

BY:
ALFAYO OYUGI ADEDE

SUPERVISOR:
PROF. WILLIAM OKELO-ODONGO

Submitted in partial fulfillment of the requirement for the award of Masters of Science in Computer Science.

October 2014

DECLARATION

This project is my original work and to the best of my knowledge it has not been presented in any University or College.

SIGNED: _ _ _ _ _ _ __ _ _ __ _ _ __ _ _   DATE: _ _ _ _ _ _ __ _ _ __ _ _ __ _ _

ALFAYO OYUGI ADEDE

REG. No: P58/73066/2009

This report has been submitted as part of fulfillment of the requirements for the award of Masters of Science in Computer Science at the School of Computing and Informatics, University of Nairobi, with my approval as a University supervisor.

SIGNED: _ _ _ _ _ _ __ _ _ __ _ _ __ _ _   DATE: _ _ _ _ _ _ __ _ _ __ _ _ __ _ _

PROF. WILLIAM OKELO-ODONGO

SCHOOL OF COMPUTING AND INFORMATICS

UNIVERSITY OF NAIROBI (UON)

DEDICATION

This project is dedicated to my wife Christine Kagonya and my daughter Kayla Akoth for their tremendous support and understanding when I had to put in long hours and effort into this project.

ACKNOWLEDGEMENT

This project would not have been possible without the support of many people. It is with immense gratitude that I express my sincere appreciation to my supervisor Prof. William Okelo-Odongo, for his ideas, tireless effort and commitment in reading and correcting my work.

I am also grateful to Mr. Eric Ayienga, Dr. Robert Oboko and Prof. Elijah Owenga for their guidance and positive criticism during project presentation, which helped me to improve.

I am also grateful to every other persons whose ideas were applied in this project study.

Finally, I thank the Almighty God for giving me the grace to finish this project.

ABSTRACT

Following the emergence of cyber foraging systems small resource constrained mobile phone devices are able to offload some of their resource intensive work to computationally powerful surrogate computers accessible on LAN. However, mobile phone based cyber foraging system also threatens control over data ownership, distribution and management. Users cannot be guaranteed that surrogate computers do not share data with unauthorized entities. Likewise, storing data on a mobile device causes user to lose control over that data when the device is stolen or lost; hence cannot prevent data from being compromised. This project examines the challenge of data integrity and confidentiality arising from using mobile phone based cyber foraging systems. We implemented and integrated data integrity and confidentiality enforcing mechanism based on Remote Access Control and Auditing (RACA) framework into an open source mobile phone based cyber foraging system prototype using use case approach. Experimental approach method is applied to measure and evaluate execution time overhead cost attributed to RACA integration. Result obtained indicated that RACA not only enhances data integrity and confidentiality but also imposes an insignificantly compromise on task offloading execution time. However, for enhanced confidentiality and availability we recommend the use of SSL protocol for data transmission and deployment of surrogate computers through defensive responses to Denial of Service (DoS) attacks respectively.

***Key words**: Cyber foraging systems, data integrity, data confidentiality*

TABLE OF CONTENT PAGES

ACRONYMS AND DEFINITIONS

AES	Advanced Encryption System algorithm
AESTET	Automated Estimation System of Task Execution
AOP	Aspect-oriented programming
ApectJ	Simple and practical aspect-oriented extension to Java™.
API	Application Programming Interface
CASE	Computer Aided Software Engineering
COCA	Computation Offload to Clouds using AOP
CODA	Constant Data Availability Files System
COMET	Code Offload by Migrating Execution Transparently
CRC	Classes-Responsibility and Collaboration
DRAM	Dynamic random-access memory
DSM	Distributed Shared Memory
EAI	Enterprise application Integration
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
Java EE6	Java Enterprise Edition Version 6
JSP	Java Server Pages
JSTL	Java Server Pages Standard Tag Library
JVM	Java Virtual Machine
LAN	Local Area Network
LBFS	Low Bandwidth File System
MAUI	Mobile Assistance Using Infrastructure
MCC	Mobile Cloud Computing
MCO	Mobile Computation Offloading
NFS	Network File System
OMT	Object Modeling Technique
OOA	Object Oriented Analysis

OOAD	Object Oriented Analysis and Design
OOSE	Object Oriented Software Engineering
ORM	Object Relational Mapping
OS	Operating System
OSGi	Open Services Gateway initiative
PGP	Pretty Good Privacy
RACA	Remote Access Control and Auditing
R-OSGI	Remote Remote OSGi
SDK	Software Development Kit
SFS	Secure File System
SRAM	Static random-access memory
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User Interface
UML	Unified Modeling Language
USB	Universal Serial Bus
VM	Virtual Machine
WAN	Wide Area Network
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

A side channel attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms (compare cryptanalysis). For example, timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information which can be exploited to break the system. Some side-channel attacks require technical knowledge of the internal operation of the system on which the cryptography is implemented,

A cold boot attack (or to a lesser extent, a **platform reset attack**) is a type of side channel attack in which an attacker with physical access to a computer is able to retrieve encryption keys from a running operating system after using a cold reboot to restart the machine. The attack relies on the data remanence property of DRAM and SRAM to retrieve memory contents which remain readable in the seconds to minutes after power has been removed

Android platform. A Google Company Inc.'s open and free software stack that includes an operating system, middleware and also key application for use on mobile devices, including smart phones.

Cloud computing Internet based computing, whereby shared resources, software, and information are provided to computers and other devices on demand. It is a natural evolution of the widespread adoption of virtualization, Service-oriented architecture and utility computing. Details are abstracted from consumers, who no longer have need for expertise in, or control over, the technology infrastructure "in the cloud" that supports them.

Cyber foraging Opportunistic use of available computing resources whereby mobile devices offload some of their resource intensive work to computationally powerful surrogate computers within LAN.

Data confidentiality Limiting data access and disclosure to authorized users/entities.

Data integrity Maintaining and assuring the accuracy and consistency of data.

Decryption The inverse of encryption.

Encryption The translation of data into a secret code.

Key management Refers Management of cryptographic keys in a cryptosystem. It entails generation, exchange, storage, use, and replacement of keys. It also includes cryptographic protocol design, key servers, user procedures, and other relevant protocols.

Man-in-the middle of attack. A form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker must be able to intercept all messages going between the two victims and inject new ones.

Mobile cloud computing Combination of cloud computing and mobile networks to bring benefits for mobile users, network operators, as well as cloud computing providers

Network bandwidth is a measurement of bit-rate of available or consumed data communication resources expressed in bits per second or multiples.
Network latency also called network *delay*, is an expression of how much time it takes for a packet of data to get from one designated point to another.
Offloading. Migration of data and programs to be executed on remote computers

Pervasive Ubiquitous/Computing is one possible potential direction toward our future computing lifestyle, in which computer systems seamlessly integrate into our everyday lives providing services and information at any time and any place. It is envisaged that ultimately different kinds of computer will become such a natural part of our environments that people will not even be aware of their existence.
Self-signed SSL/TLS certificate an identity certificate that is signed by the same entity whose identity it certifies.
***Surrogat*e** Computers within LAN used in execution of offloaded tasks.

***Trusted Computing*.** Computing where the computer will consistently behave in expected ways, and those behaviors will be enforced by computer hardware and software. Enforcing this behavior is achieved by loading the hardware with a unique encryption key inaccessible to the rest of the system.

Use case approach. Methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

Virtualization Refers to the act of creating a virtual (rather than actual) version of something, including but not limited to virtual computer hardware platform, operating system (OS), storage device, or computer network resources.

Wire-tapping. The practice of connecting a listening device to a telephone line to secretly monitor a conversation.

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

1.1 Background

Satyanarayanan (2001) coined the term Cyber foraging system and defined it as opportunistic use of available computing resources by computationally deprived devices such as mobile phone devices that offload some of their resource intensive work to computationally powerful surrogate computers within LAN. However, the mobile phone device should not only be able to use surrogates when available, but also be able to solve tasks own its own when no surrogates are available i.e. the application does not cease to function when the mobile device is on its own as affirmed by Balan et al. (2002).

Mobile phoned based Cyber foraging system envisioned a scenario in which when a mobile phone device enters a neighborhood, it detects the presence of potential surrogates and negotiates their use. Communication with a surrogate is via short-range wireless peer-to-peer technology. When an intensive computation has to be performed, mobile phone device offload the computation to the surrogate; the latter may cache data on its local disk in performing the computation. Alternatively, the surrogate may have staged data ahead of time in anticipation of the user's arrival in the neighborhood. When mobile computer leaves the neighborhood, its surrogate bindings are broken, and any data staged or cached on its behalf are discarded.

Offloading goal is accomplished by following six steps. First mobile client discovers surrogates available within WLAN. Once the surrogate discovery is completed the application is partitioned into either locally or remotely executable tasks. This is followed by selection of the best execution strategy that decides on which tasks and where tasks will be executed in current environment by weighing performance cost penalties in undertaking such decision. Next a trust relationship is established between mobile client and the surrogate computer. Once these entire prerequisite steps are completed; the execution of tasks on surrogate computers takes place. The task to be executed can either be pre-installed or migrated on-demand. Last steps involve mobile client constant monitoring of their execution environment and adapting the cyber foraging process accordingly.

1.2 Motivation for Cyber Foraging

Mobile computing devices, such as smart phone, are becoming ubiquitous and an increasing number of users are carrying such a device at all time. Barton et al. (2006) predicted that the smart phones are quickly advancing to provide full-fledged personal computing due to rapid improvement in their display size, network connectivity and input methods. However, relatively powerful, mobile devices remain constrained in terms of physical size, thus leading to limitations in their computing and communication capabilities, battery lifetime, as well as screen and keyboard size. These constraints inhibit mobile devices from fully supporting increasingly demanding mobile applications. Furthermore, despite rapid improvement in processing capabilities of mobile devices, battery energy density that is critical resource on mobile devices has experienced the slowest improvement trend according to Paradiso and Starner (2005) illustrated in figure 1 below.

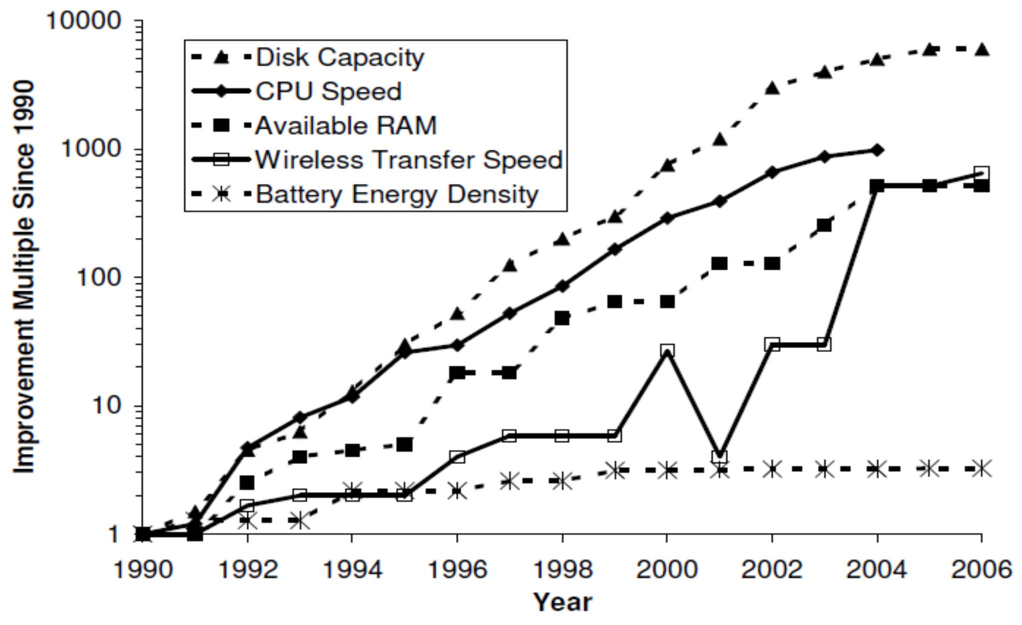


Figure 1: Trend of processing capabilities of mobile devices
Source: Adapted from Paradiso and Starner (2005)

Cyber foraging is one of the effective ways to deal with this problem as it attempts to reconcile the contradictory requirements of having longer battery life while at the same time meeting the ever-growing expectations of mobile users to execute intensive computation and data manipulation applications that are well beyond those of a lightweight mobile computing device with long battery life.

Another form of pervasive computing model closely related to cyber foraging system that has gained popularity in recent years is mobile cloud computing. However, its failure to address the following challenges continues to inspire interest in cyber foraging system as a possible alternative.

i. **Monetary cost.** When using a mobile device, accessing the Internet comes at a cost while accessing a locally available Wi-Fi network does not.

ii. **Network latency.** Whereas network bandwidth when connecting to the Internet has increased rapidly recently, the latency has not been reduced sufficiently thus making cloud computing infeasible for deploying applications where response time is crucial.

iii. **Internet bandwidth.** While mobile Internet speeds are increasing they are still slower than the speed of local WLAN networks. This renders cloud computing useless in data intensive scenarios such as image manipulation, speech recognition.

iv. **Energy efficiency.** Finally, by using the unused resources of machines that are already running, cyber foraging uses little or no extra power, whereas cloud computing applications are run in large data centers where hundreds of power hungry dedicated servers may be running 24/7.

Notwithstanding the aforementioned challenges; Satyanarayanan (2009) suggested that mobile cloud computing can be used in some scenarios to complement cyber foraging whereby task schedulers would consider local execution, execution at available surrogates, and execution in the cloud when choosing where to perform a task.

1.3 Data Integrity and Confidentiality

Avizienis et al. (2004) defined computer security as a composite confidentiality, integrity and availability (also called CIA) attributes. They defined confidentiality as absence of unauthorized information disclosure; Integrity as absence of improper (meaning unauthorized) system and underlying data alteration while availability as continued readiness for authorized actions. Avizienis et al. (2004) maintained that a system with appropriate security should maximize the balance of the these three attributes. Moreover, privacy is defined as a subset of confidentiality and integrity. In other words, users have the right to be sure that their data is not disclosed.

This project focuses on implementing Data integrity and Confidentiality attributes in mobile phone based cyber foraging system. Whereas the availability attribute is not covered we recommend that the same be enforced through deployment of surrogate computers in manner that is defensive to responses of Denial of Service attacks.

1.4 Problem Statement

Even though offloading some of resource intensive work to powerful surrogate computers reachable on LAN is advantageous, cyber foraging systems do compromise users' control over data integrity and confidentiality.

Present solutions to enforce data integrity and confidentiality rely on data protection systems that are implemented using encrypted file systems such as Microsoft BitLocker, Apple OS X FileVault, PGP Whole Disk Encryption and TrueCrypt systems as observed by Casey and Stellatos (2008). However, this is only possible because conventional in-house computing environment offers trusted, flexible customization and certainty in enforcement of data security policies. Regrettably, Mobile phone based cyber foraging systems exhibit opposite of these properties i.e. untrusted computing environments as data and applications physically migrate to "insecure" surrogate computers; inflexible customization as regards to assured data deletion coupled with lack of uniform access control policies management.

Existing encrypted file systems do not provide remote auditing capabilities thus security breach may go undetected. This is because they focus on data exposure prevention yet ignore data exposure detection. They also rely on locally stored key that is protected by user’s passphrase that may be insecure. Furthermore, encrypted file systems potentially compromise integrity and confidentiality as users find it difficult to create, remember, and manage passphrases or keys.

We propose to address the challenge of data integrity and confidentiality by implementing Remote Access Control and Auditing (RACA) mechanism that augment existing solutions based on encrypted file systems as suggested by Geambasu et al. (2011). RACA mechanism combines encryption, remote key storage and audit server hence capable of augmenting encrypted file system with auditability and remote data control. It provides file audit that provides explicit evidence on whether or not a file access was made. It also allows users to disable file access by configuring an audit server to refuse to return a particular file decryption key.

1.5 Justification of the Study

Geambasu et al. (2011) applied RACA technique christened Keypad in auditing file system/forensic mechanism for theft-prone devices, such as mobile phones, laptops, tablets, and USB stick. Keypad provided explicit evidence on whether or not files in such devices were accessed after device loss. Similarly, Rahim and Saravanan (2013) used RACA technique in mobile cloud computing to implement secure cloud storage system that achieves policy-based access control and file assured deletion together with an information accountability cloud framework to track actual usage of client data.

However, in existing literature on cyber foraging, use of RACA technique appears non-existent thus the relevance of this work. Furthermore, virtually all related works in cyber foraging have had a focus on how to design and implement a functional cyber foraging system with insignificant attempt made to address security challenges arising therein.

Our initial assessment revealed that RACA technique is suitable in cyber foraging domain because mobile phone devices used in cyber foraging system are highly vulnerable to theft and loss; an issue that greatly compromises data integrity and confidentiality. Secondly, the technique effectively augments existing solutions based on encrypted file systems by addressing their shortcomings in untrusted, inflexible customization and uncertain enforcement of data security policies environment such as in cyber foraging.

1.6 Objectives of the Study

1.6.1 General Objectives

To enhance security of mobile phone based cyber foraging system by implementing a mechanism that enforce data integrity and confidentiality.

1.6.2 Research Objectives

i. To examine existing mobile phone based cyber foraging systems with a focus on data integrity and confidentiality security challenges.
ii. To design and implement data integrity and confidentiality enforcing mechanism in mobile phone based cyber foraging system based on Remote Access Control and Auditing framework.
iii. To integrate the implemented mechanism in objective (ii) above into an open source mobile phone based cyber foraging system prototype.
iv. To evaluate offloading performance overhead costs attributed to the integration of integrity and confidentiality enforcing mechanism in mobile phone based cyber foraging system prototype.

1.7 Research Questions

i. How can RACA framework be integrated into mobile based cyber foraging system?
ii. Does the integration of RACA mechanism significantly undermine task offloading execution time performance in mobile phone based cyber foraging system?

1.8 Proposed Solution

Conventional data-protection systems such as Microsoft BitLocker, PGP Whole Disk Encryption and TrueCrypt system enforce data integrity and confidentiality through encrypted file system. We propose a solution that entails the use of Remote Access Control and Auditing (RACA) framework originally described by Geambasu et al. (2011). The proposed solution augments traditional encrypted file systems by providing remote access control and auditing in a mobile phone based cyber foraging system as illustrated in figure 2 and 4 (***shaded parts depict RACA integration***). Figure 3 illustrates threats/vulnerability a typical mobile phone based cyber foraging is exposed to. Our proposed solution addressed threats arising from disclosure of information to unauthorized individuals, potential inaccuracy and inconsistency of data compromises. The threat of interception of data during communication is mitigated upon through transmission based on SSL protocol.

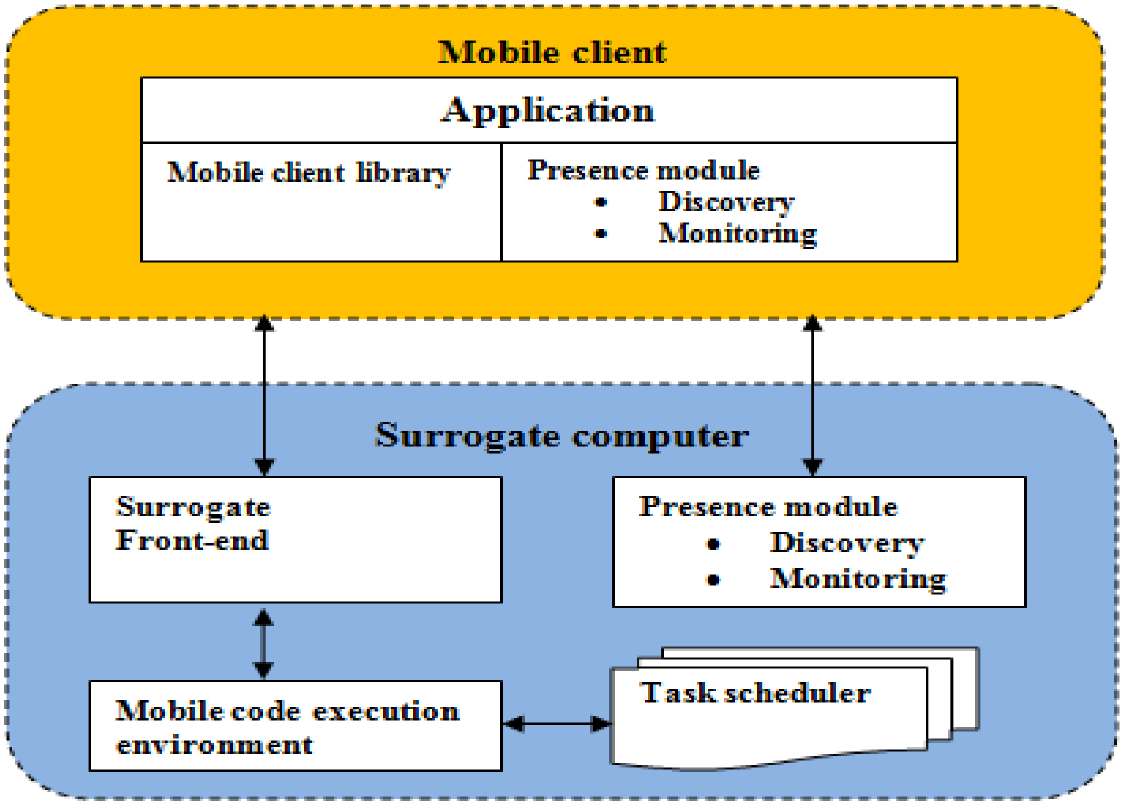
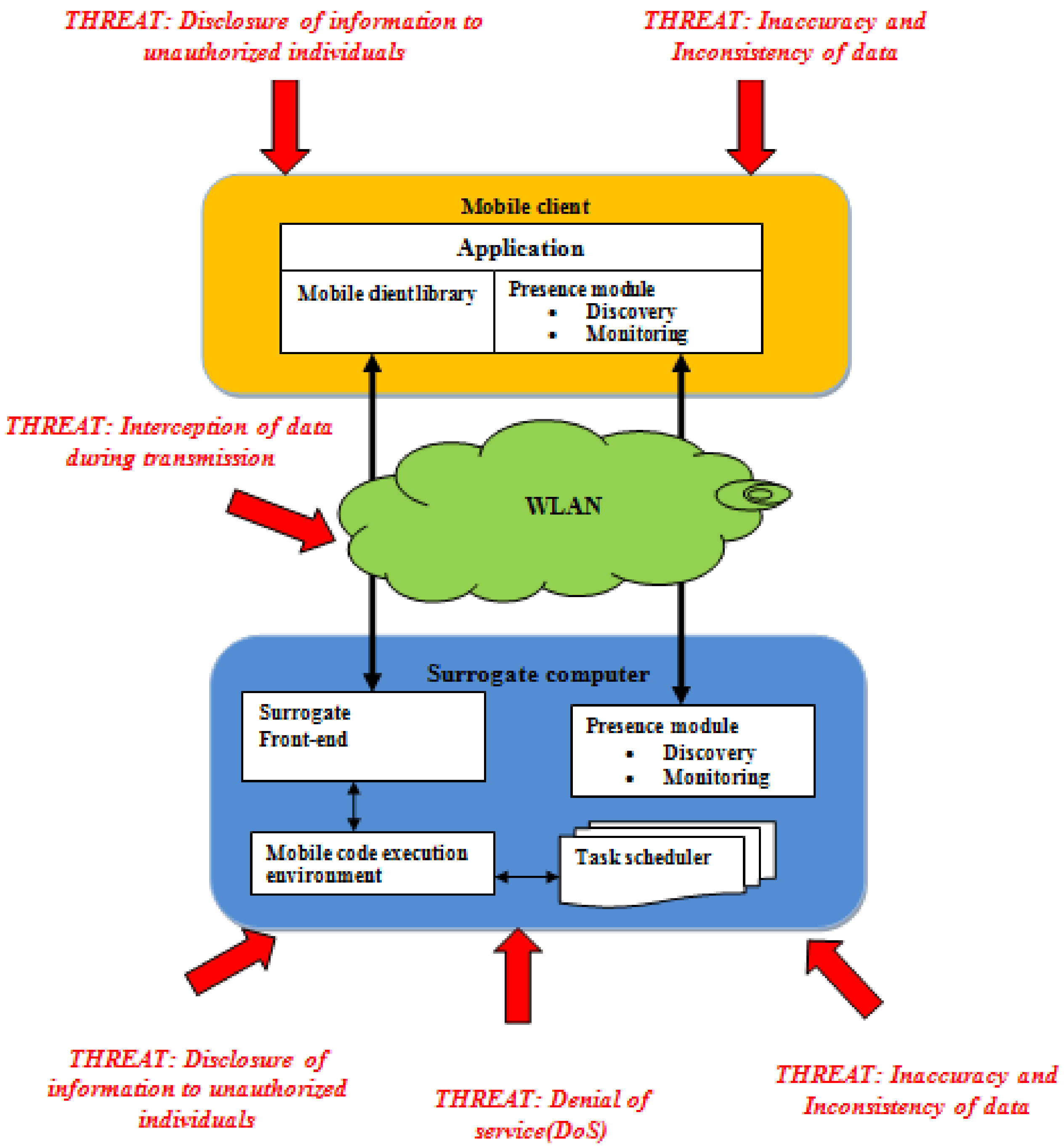


Figure 2: A typical high-level cyber foraging architecture

Source: Adapted from Mads and Bouvin (2010)


THREAT: Disclosure of information to unauthorized individuals
THREAT: Inaccuracy and Inconsistency of data
Mobile client
Application
Mobile client library
Presence module
Discovery
Monitoring
THREAT: Interception of data during transmission
WLAN
Surrogate computer
Surrogate Front-end
Presence module
Discovery
Monitoring
Mobile code execution environment
Task scheduler
THREAT: Disclosure of information to unauthorized individuals
THREAT: Denial of service(DoS)
THREAT: Inaccuracy and Inconsistency of data

NB: Threat on Denial of service(DoS) is not covered within the scope of the system.

Figure 3: Security threats facing cyber foraging system
Source: Author's compilation

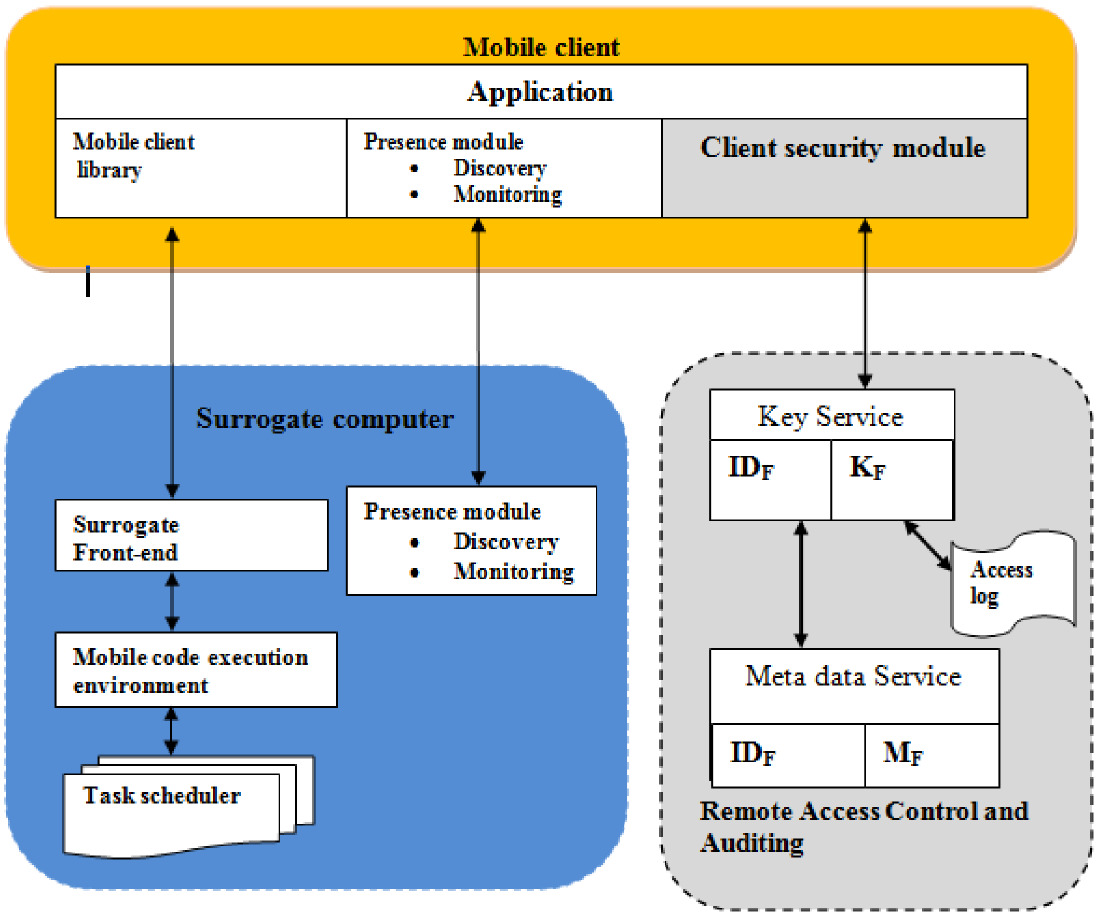


Figure 4: Integrating RACA mechanism in a typical cyber foraging architecture

Source: *Author's compilation*

The proposed solutions is motivated by the fact that traditional file encryption systems do fail to guarantee data integrity and confidentiality as users find it difficult to create, remember and manage passphrases or keys. Furthermore encrypted file systems often rely on a locally stored key that is protected by a user's passphrase that may be insecure and are easily vulnerable to physical attacks through "cold-boot attack". In addition when encrypted file system is compromised, it does so "silently" without providing data owners with a means of discovering the access, thus they may lead to a false sense of protection to mobile device users. Figure 5 illustrates RACA conceptual mechanism.

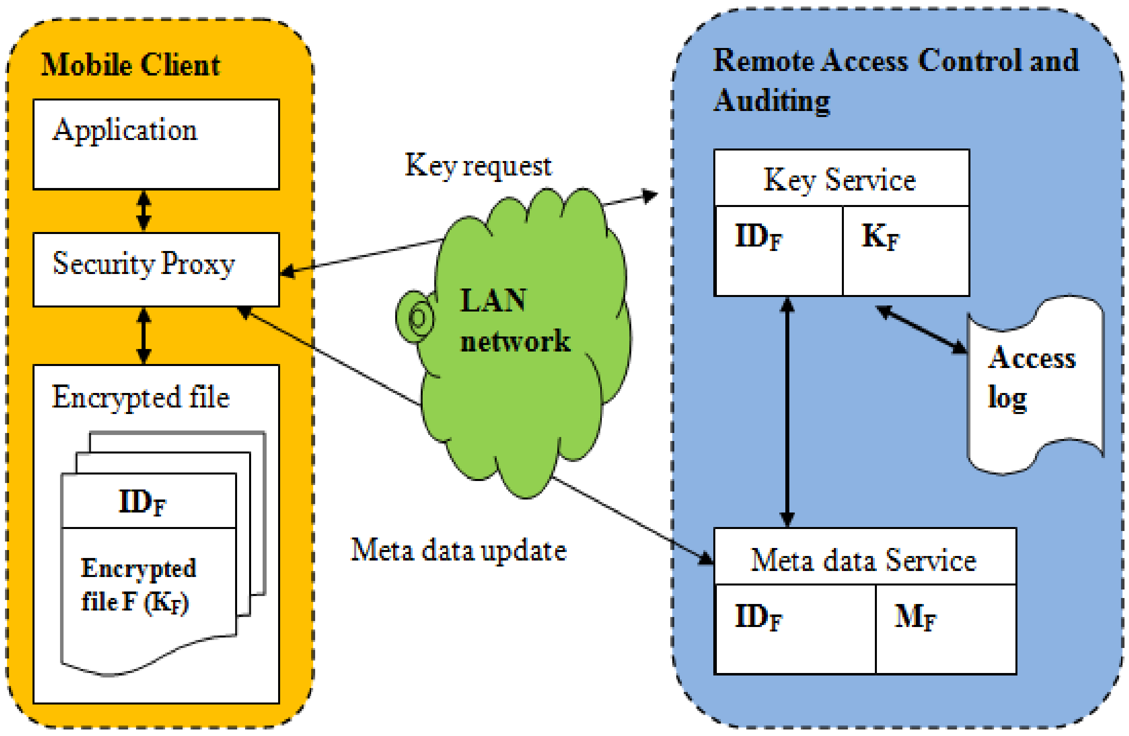


Figure 5 : Conceptual illustration of RACA framework
Source: Adapted from Geambasu et al. (2011)

On the mobile client device, each file **F** has a unique identifier (**ID_F**) and the file's data is encrypted with a unique symmetric key, **K_F** . A remote key service maintains the mappings between audit IDs and keys. When an application wants to read or write a file, RACA looks up the file's audit ID and requests the associated key from the service. Before responding to the request, the service durably logs the requested ID and a timestamp. In addition to the key service, RACA contains a metadata service that maintains information needed by users to interpret the logs. The file metadata (**M_F**) information includes a file's path, the process that created it, and the file's extended attributes.

According to Geambasu et al. (2011), RACA combines encryption, remote key storage and an audit server to provide two important properties. First is a fine grained file auditing that offers explicit evidence on whether or not a file access was made. Secondly, it allows users to disable future file access on the device once the device is lost by allowing a configuration on the audit server to refuse to return a particular file key.

Hence, RACA augments encrypted file systems with auditability and remote data control. This is achieved by (1) encryption of each file with its own symmetric key, (2) storage of all keys on a remote audit service, (3) downloading the key for a file each time it is accessed, and (4) destruction of the key immediately after use. By configuring the audit service to log all storage accesses, we obtain fine-grained auditability; by disabling all keys associated with a stolen device on the service, we prevent further data access.

1.9 Assumptions

i. Remote Access Control and Auditing mechanism should be able to prevent unaudited access.
ii. Once mobile device is stolen, the victim should be able to disable access to protected file after the device is lost.
iii. File access latency and throughput arising from the use of RACA should be acceptable.

1.10 Target Audience

The research targets individuals who own smart mobile phone devices and would like to benefit from using cyber foraging systems but are dissuaded from doing so due to inadequate security mechanism in such systems. These are individuals in possession of confidential data mainly from security based agencies (Intelligence Officers, Military Strategist), product developers and academic researchers among others owing to fact that unwarranted compromise on data integrity and confidentiality can result into devastating consequences. The target group is expected to benefit significantly from this research as cyber foraging systems do send users' data and programs to servers which are not necessarily under users’ control.

CHAPTER TWO: LITERATURE REVIEW

2.1 Cyber Foraging Systems

2.1.1 Research Evolution

Research in cyber foraging has evolved considerably. According to Kumar et al. (2012), studies on cyber foraging systems can be traced from the year 1996. These studies focused mainly on exploring cyber foraging feasibility, designing efficient offloading decisions algorithms and development of offloading infrastructures as summarized in table 1 below.

Period	**Research focus**
1996-2000	Offloading feasibility
2000-2005	Suitable offloading decision algorithms
2005-date	Provision of offloading infrastructure

Table 1: Previous cyber foraging research focus
Source: Adapted from Kumar et al. (2012)

During the period between 1996-2000, the struggle to overcome limitation on wireless network as a result of low bandwidth was primarily the driving force, thus majority studies primarily focused on exploring offloading feasibility.

Between 2000-2005 majority of studies shifted and focused on how to design an efficient decision making algorithm aimed at objectively determining whether offloading would benefit mobile users. Researchers during this period designed both static and dynamic based offloading decision making algorithms.

Since 2005-to date, improvement in virtualization technologies, increased network bandwidth and emergence of cloud computing infrastructure continue to significantly influence research studies on cyber foraging. Hence the shift has been on how to leverage on these infrastructures while building cyber foraging systems.

Whereas the trend indicates considerable efforts directed towards the design and implementation of a functional cyber foraging system, it also reaffirms the assertion that no substantial effort have been made to address security related challenges arising in these systems as elucidated below. Hence, the significance of this research as an attempt to address the gap.

2.1.2 Security Challenge

Flinn et al. (2001) and Balan et al (2003) pioneered the development of cyber foraging systems by developing Spectra and Chroma sytstems respectively. They focused on partitioning the application into modules and calculating the optimal offloading strategy. These partitioning schemes require significant modification of the original application and lacked mechanism to enforce security and privacy needs.

Cuervo et al. (2010) developed the MAUI system that provide fine-grade code offload. Their result showed that minimizing the size of the application state that is sent over the network significantly improves performance by allowing more methods to be offloaded. However, MAUI serializes the application state using XML which is slower than binary serialization. It also modifies the .NET bytecode and creates two separate code bases:- one for the mobile device and another one for the server thus making debugging more difficult. Furthermore, no security enforcement mechanism is provided.

Chun et al. (2011) developed CloneCloud system that uses virtualization technology to migrate Dalvik Virtual Machine (VM) from an Android phone on which the application is running to a backend server. It does not require developer’s intervention for offloading since this is accomplished at the Operating System (OS) level. Its major shortcoming is the overhead required to migrate the entire VM from the mobile device to the server. Unlike other systems, it attempts to enforce security mechanism by encrypting data during transmission, but the cloned stored files are unencrypted thus remains vulnerable.

Kosta et. al (2012) developed ThinkAir system that performs method level code offloading but focuses more on scalability issues through parallel execution of offloaded tasks. Like CloneCloud, it attempts to enforce security mechanism by encrypting data during transmission, but the stored files are unencrypted.

Kemp et al (2010) developed Cuckoo offloading framework that leverages the existing activity/service model in Android platform to offload computation intensive portions of the applications. Developers are forced to write offloadable methods twice; one for local computations and another one for remote computations resulting into to unnecessary code duplication. No security enforcement mechanism is provided.

Chen et al. (2012) developed Computation offload to clouds using AOP (COCA) system that uses AspectJ to offload Android applications to the cloud environment. The system offloads pure functions without transferring application state and does not implement any security mechanism.

Giurgiu et al. (2009) proposed "Calling the Cloud" middleware platform that can automatically distribute different layers of an application between the phone and the server by optimizing a variety of objective functions. It requires the application to be partitioned into several software modules using the R-OSGi. No security enforcement mechanism is provided.

Mark et al (2012) proposed Code Offload by Migrating Execution Transparently (COMET) system that leverage Distributed Shared Memory (DSM) to provides a virtual shared memory space that is accessible by threads on different machines without any work on the part of the developer. It offloads fine-grained parallel algorithms to multiple machines, resulting in improved performance. Its shortcoming is that developers remain oblivious that a simple memory access can result in a network call thus resulting into inefficient applications that do not scale. No security enforcement mechanism is provided.

Recently, Verbelen (2013) developed AIOLOS cyber foraging system on Android platform based on OSGi components replicated on remote server. At runtime method calls are forwarded either to the local or remote OSGi component instance. Similarly, Mads (2010) developed Scavenger system aimed at providing developers with a complete cyber foraging toolbox that can ease the process of developing applications that utilize cyber foraging. Though these two systems are robust, Verbelen (2013) and Mads (2010) did not address any security challenges.

Lastly, Griera (2013) implemented a simplified open source Mobile Computation Offloading (MCO) system based on Android platform. MCO utilizes Automated Estimation System of Task Execution (AESTET) in offloading decision making. It also implements basic authentication mechanism as part of its security mechanism. It is of interest in our study since unlike the aforementioned systems it is freely available and further enhancement can done on it to achieve our project objectives.

2.2 Remote Access Control and Auditing (RACA)

According to Geambasu et al. (2011) Remote Access Control and Auditing (RACA) mechanism is related to work in three areas: (1) theft-protection systems, (2) data-protection systems, and (3) distributed /network file systems.

2.2.1 Theft-Protection Systems

Theft-Protection Systems such as MobileMe and Adeona rely on software running on a device that can monitor file accesses, report device locations and file accesses to a remote trusted server. However, these systems are vulnerable to hardware and disconnection attacks as determined attacker can circumvent these protections and analyze the device’s media using his own hardware, without the associated monitoring software installed. RACA provides a strong auditing support and data-destruction capabilities even against thieves who use their own hardware and software to attack a protected file system or (temporarily) block the device’s access to the network. Unlike in MobileMe and Adeona, where the audit log for a file access occurs after the fact, the audit log in RACA is produced before the access can occur, making it mandatory.

2.2.2 Data-Protection Systems

Data-Protection Systems such as Microsoft BitLocker, Pretty Good Privacy (PGP) Whole Disk and TrueCrypt are based on encrypted file systems. However, none provide remote auditing capabilities; therefore a security breach may go undetected. On the other hand RACA provide a stronger barrier to access and a forensic trail whenever that barrier is breached. Geambasu et al. (2011) argued that data-protection systems differ from RACA system as the former focus on data exposure prevention, whereas RACA focuses on data exposure detection should prevention systems fail and thus the two should be considered as complementary rather than competitors.

2.2.3 Networked File Systems (NFS)

NFS allow a server to share directories and files with clients over a network as if they are stored locally. Notable benefit of NFS is that data that would otherwise be duplicated on each client can be kept in a single location and accessed by clients on the network. Examples of some of existing NFS are Constant Data Availability Files System (CODA) developed by Satyanarayanan et al. (1991), Low-Bandwidth Network File System (LBFS) developed by Muthitacharoen et al. (2001) and Secure File System (SFS) developed by Mazieres et al. (1999).

CODA supports disconnected file operations through data catching. It also supports encrypted communication but not encrypted storage. LBFS focuses on minimizing communication bandwidth between clients and servers by compressing file data on transit to reduce latency for interactive file access over slow wide-area networks. SFS provides secure file transfer by embedding public key in file pathnames.

In general these systems neither encrypt data stored on the disk, nor provide auditing mechanism as it is in RACA. Furthermore, RACA is concerned with encryption key management and its transfer between a file system and a remote server. On the other hand existing NFS are concerned with the transfer of file data between the client and the server. Geambasu et al. (2011) maintain that RACA uniqueness is evident through the integration of both encryption and audit logging; thus demonstrating the advantage of separating encryption and key management to enforce auditing in a distributed file system.

CHAPTER THREE: METHODOLOGY

3.1 Introduction

The goal of research methodology is to provide a standard method and guidelines to ensure that project is completed on time and is conducted in a disciplined, well-managed, and consistent manner that promotes the delivery of quality product and results. This section presents an overview of the methods used in the study.

This project consists of four (4) main tasks. The first task is to analyze and examine existing mobile phone based cyber foraging systems with a focus on how data integrity and confidentiality security challenges is addressed. Second task is to design and implement data integrity and confidentiality enforcing mechanism in mobile phone based cyber foraging systems derived from RACA framework. Third task is to integrate implemented RACA framework in an open source mobile phone based cyber foraging system while the last task is to examine actual offloading performance overhead costs attributed to the integration of RACA mechanism in mobile phone based cyber foraging system.

3.2 Research Approach

This research begins by studying existing theories and techniques related to research problem area hence deductive approach is used. In order to implement, integrate data integrity and confidentiality enforcing mechanism into prototype mobile phone based cyber foraging systems we use a use case approach as proposed by Jacobson et al. (1994). Finally, to evaluate the resulting execution overhead cost attributed to the integration of RACA framework an experimental method is used with extractive text summarization application adopted as a test case scenario.

3.3 RACA Framework Implementation

Implementation of RACA framework to mitigate on security threats facing mobile phone based cyber foraging highlighted in figure 3, follows the following steps.

STEP 1 : Registration/Encryption of a file

File to be secured is first registered into the system. This entails encrypting the file using 128-bit Advanced Encryption System (AES) algorithm. A symmetric key, encrypted file and Access Control List (ACL) of the file is generated as output. Encrypted file name, secret key among other attributes are recorded and stored on local database of the mobile phone.

STPEP 2: Synchronization of access control and auditing records

This step is achieved through database synchronization service that execute automatically on the background ensuring that any changes on access control and audit log of a registered file is consistently reflected on both mobile phone device and at surrogate computer.

STEP 3 : Decryption of encrypted file

This step is invoked whenever, an encrypted file is accessed on the mobile phone device. The system first interrogate file Access Control List (ACL) to ascertain whether access to the file is allowed. If it is permitted secret key is returned from the key registry that is used to decrypt the file. Otherwise, no secret key is returned and file access is denied, this prevents **disclosure of information to unauthorized individuals** and also ***Inaccuracy and Inconsistency of data*** that could arise in the event unauthorized individual access the file. Figure 6 in the next page illustrate the steps above.

It is worth pointing out that RACA framework presented in figure 5 above does not address security threats posed by interception of data during transmission and denial of service attack (DoS). To mitigate on interception threat we use HTTPS for transmission. Threat of DoS is not covered within the scope of this study.

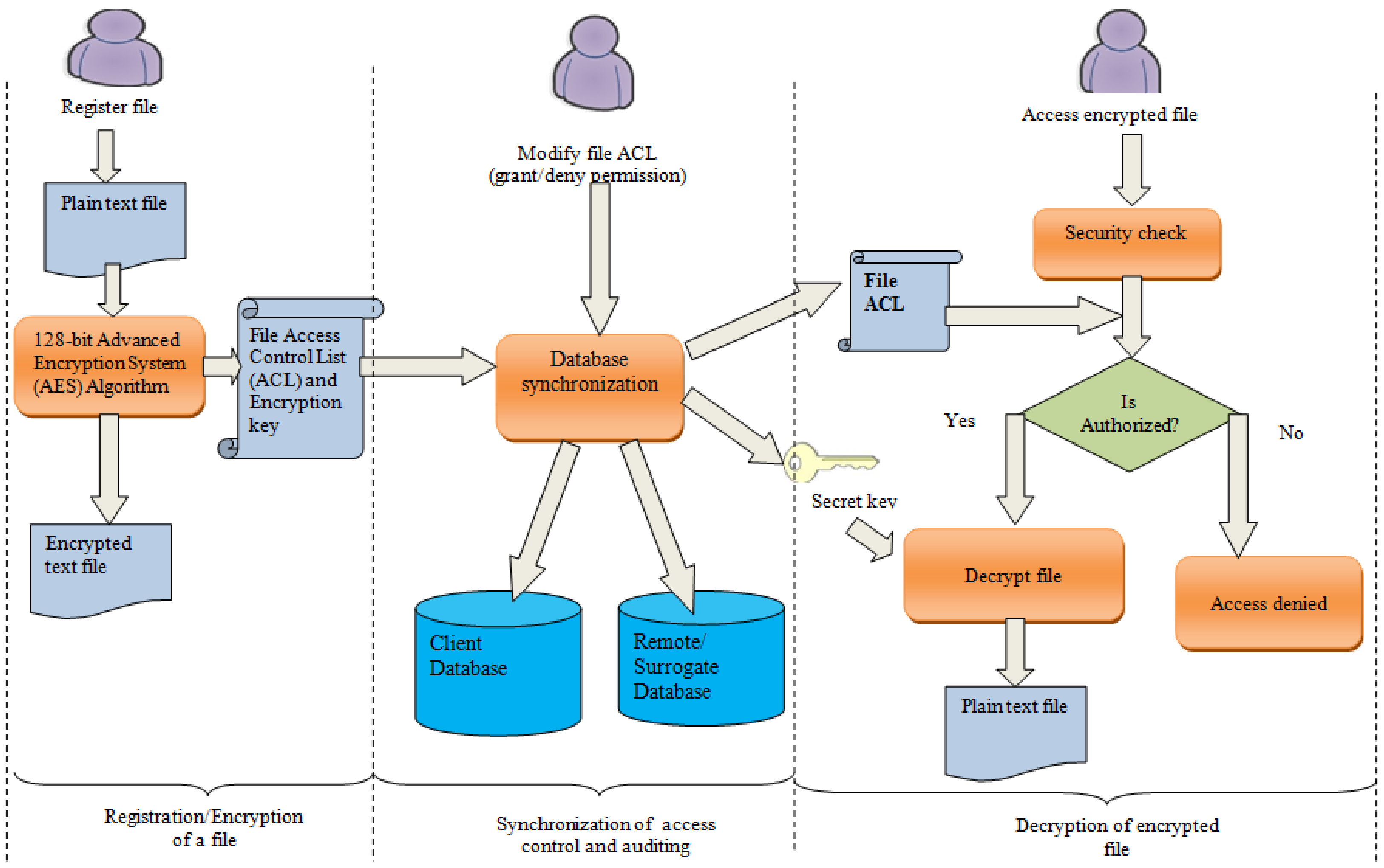

Register file
Plain text file
128-bit Advanced Encryption System (AES) Algorithm
File Access Control List (ACL) and Encryption key
Encrypted text file
Modify file ACL (grant/deny permission)
Database synchronization
Client Database
Remote/ Surrogate Database
File ACL
Secret key
Access encrypted file
Security check
Is Authorized?
Yes
No
Decrypt file
Access denied
Plain text file
Registration/Encryption of a file
Synchronization of access control and auditing
Decryption of encrypted file

Figure 6 : RACA process

Source: Author's compilation

3.4 Prototype System Development

The prototype system development is done using use case approach methodology for Object Oriented Software Engineering (OOSE). This involve identification of functional requirements from which use case artifacts is developed. Thereafter the dynamic and static behaviour of the system is analysed and modelled. The modelling of static behaviours is done through identification of objects and classes which are represented using Unified Modelling Language (UML) diagrams. The dynamic aspects of the system are modelled using sequence, interactive, state diagrams and collaboration diagrams.

RACA implementation is done using Java programming language on both Android based mobile phone and surrogate computer module. Two **raca** databases are maintained on the mobile phone and surrogate computer from which database synchronization is implemented in order to enforce access control and auditability requirements In addition, the integration of RACA into an open source mobile phone cyber foraging system is done at application, database and user interface levels.

As regards to implementation prototyping technique is followed. The prototype development entails four phases namely functional selection, construction, evaluation, and further use as suggested by Floyd (1984) depicted in Figure 7 below.

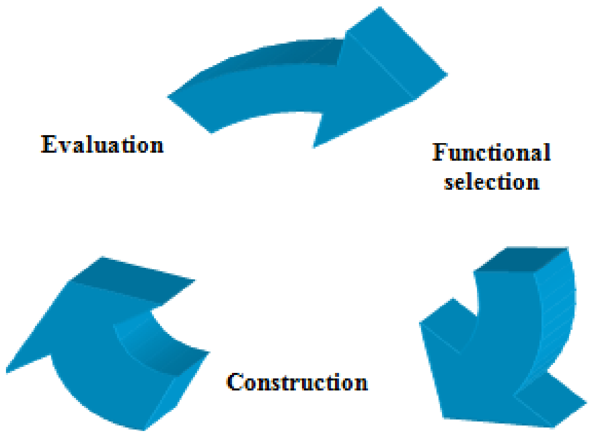


Figure 7: Prototyping phases

Source: Adapted from Floyd (1984).

i. **Functional selection** refers to the functionality chosen for the prototype. In general, the chosen functionality should be a subset of the functionality one would expect to exist in the final product. Within functional selection Floyd identifies two differing ways of prototyping: *vertical prototyping*, where the implemented functionality is presented in its intended final form, but only a small subset of the total functionality is included. Alternatively *horizontal prototyping* can be employed, where the entire functionality is represented, but the functions are not implemented in detail. In our case we followed vertical prototyping.

ii. **Construction** refers to the actual implementation of the prototype. The effort involved constructing a prototype should be much smaller than that involved in building the final product. We use Integrated Development Environment (IDE) and other CASE tools in this phase.

iii. **Evaluation** is the phase where the implemented prototype is tested and evaluated in order to inform the development process of the final product. This is done through Use case testing and statistical analysis (Descriptive and Inferential statistics) of execution time performance.

iv. **Further use** this may vary depending on the kind of prototype being developed. In some projects the prototype is used exclusively for learning purpose, and is thus thrown away after prototyping. Other prototypes may be matured and then used fully or partially as a component in the final product. The developed prototype is used exclusively for learning purpose.

3.5 Execution Time Evaluation

In order to evaluate the execution time overhead attributed to RACA integration; we use experimental approach and extractive text summarization is experimented upon as test case scenario. Extractive text summarization is executed on both mobile phone and surrogate computer. We implement execution performance introspection mechanism that records various variables values namely ***overallTime, realServerTime, overhead, estServerRuntime, estOffloadingTime, decryptionTime and estAndroidRuntime*** in miliseconds during extractive text summarization. These values are stored in the database and thereafter execution runtime analysis is carried out and displayed using bar chart and line graph for visual interpretation.

In order to generate the adequate samples for evaluation, we used Microsoft word document text files whose sizes are 20kb, 40 kb, 60 kb, 80kb and 100 kb as input from which relevant parameters outlined above are recorded. This is carried out for both encrypted and unencrypted files with and without extractive text summarization offloading.

The experimental approach is preferred in this project because integration of RACA in cyber foraging demands imposes additional execution time. This is because data encryption/decryption comes at a significant offloading execution overhead cost that might considerably undermine the goal of offloading resource intensive tasks in a cyber foraging environment.

Furthermore, experimental designs allow us to test for causal relationships between variables instead of just correlation relationships. Controlling an experimental situation also allows us to minimize or eliminate confusing effects from variables other than our variables of interest.

3.6 Analysis and Interpretation Results

Descriptive analysis on how the proposed RACA mechanism enhances confidentiality and integrity and RACA shortcoming in enforcing the same will be provided.

A comparative analysis of execution time overhead associated with RACA mechanism integration vis-à-vis its absence of in a mobile cyber foraging system is also provided. This is achieved through using bar chart and line graph for visual interpretation. This is carried out for both local and offloaded task execution times**.** Local execution time is the time taken to solve the task when no LAN connection is available or when offloading is infeasible while offloading task execution time is the actual time that the whole process of offloading the task takes.

Finally, Chi-Square (X^2) and Persons correlation coefficient (r) are applied as part of inferential statistics to rigorously ascertain occurrence of experimental control variability and also establish existence of casual relationship between the input file size and execution time categorized under encrypted and unencrypted input files.

CHAPTER FOUR: REQUIREMENT SPECIFICATIONS

4.1 Introduction

In order to achieve project objectives specified as in Chapter one, section 1.6; requirement specification, analysis, design, implementation of prototype system was undertaken. This chapter describes system requirements for the implemented prototype system.

Schach (2011) asserted that the any system requirements can be classified as either functional or non-functional requirement; both of which technical design and actual implementation of the system must satisfy. The functional requirement analysis was carried out by adopting user-case approach proposed by Jacobson et al. (1994) together with use case scenarios as advocated by Alexander et al. (2009). As regards non-functional requirement, various categories such as architectural requirements, structural requirements, availability requirements and performance requirements were specified as constraints.

4.2 Functional Requirements

i) The system should be able to offload execution of computationally intensive tasks (extractive text summarization in our case) to surrogate computer within WLAN.

ii) The mobile phone client should be able to execute task on its own when surrogate computers are not available within its range.

iii) The system should be able encrypt and decrypt a text file using Advanced Encryption System (AES) algorithm on Android based mobile phone deployed as mobile phone client in cyber foraging system.

iv) The system should be able to associate the secret encryption/decryption (symmetric) key to the encrypted file.

v) The system should provide Remote Access Control mechanism that can disable or enable access to encrypted file on mobile phone client.

vi) The system should provide explicit evidence of any file access from mobile phone at the remote auditing server.

vii) The system should be able to synchronize Access Control changes between mobile phone client and the Remote Access Control server.

viii) The system should be able to record performance metrics during execution of offloaded task (extractive text summarization) at a remote server.

ix) The system should be able to provide graphical interpretation (bar charts and line graphs) of performance metrics for both offloaded and non-offloaded task in mobile phone based cyber foraging system.

4.3 Non-functional Requirements

❑ **Performance**

a) Response Time: The system should be able to provide response within 60 seconds after being issued with the command.

b) Memory: Runtime Memory for the system should be within 32 MB for mobile client while 1 GB RAM for Surrogate computers.

❑ **Dependability:**

a) Robustness: The system should not crash for bad inputs and for that matter under any circumstance (unless the reason is external to the system like OS failure).

b) Reliability: The system should generate correct response for at least 99 out of randomly selected 100 inputs.

❑ **Cost**

a) Development man power : Should be restricted to 1 masters student under the supervision of 1 lecturer.

b) Resources:

- The hardware resources should be restricted to The Galaxy Grand Duos (GT-19082) on Android v4.1.2 (Jelly Bean) OS with Wi-Fi support, One (1) Toshiba Laptop running on Intel (R) Core i5-3230 M @ 2.60 GHz with 8 GB RAM on Windows 8 x64 bit and TP-LINK, 3G/4G Wireless N Router TL-MR3220.

- Software resources used to develop and deploy the system should be restricted to freeware/open source software. It will include Java SDK version 1.6_25, Android Development Tools build 22.6.2, Netbeans IDE version 6.9.1, Apache Tomcat version 6.0.20, SQLite database system among others.

❑ **Maintenance:**

a) The document/code should be self explanatory with comments in codes wherever necessary.

b) The code should follow the Java coding standards.

❑ **End-User:**

a) Usability: A naïve end-user should be able to understand and use the system given within five minutes of training.

❑ **Pseudo Requirements**

a) The code should be written in Java.

b) The code written should be compiled using Android SDK version 4.4.2 while the Surrogate server mobile should be compiled using Java SDK Version 1.6_25.

c) The documents for the code written should be generated using Java doc.

4.4 Requirement Model

The requirement model was carried out using *'A use case driven approach'* as advocated by Jacobson et al. (1994) in development of Object-Oriented Software Engineering *(OOSE)*. A use case driven approach focuses on specification of actors, use cases, interface descriptions and problem domain objects while all relevant notations are drawn using Unified Modeling Language (UML).

Subsequently, a set of functional, object and dynamic models were developed to facilitate better understanding of the system. The choice of use-case driven approach during the prototype system development was motivated by existence of seamless incremental transition between development stages and models as depicted in figure 8 and 9.

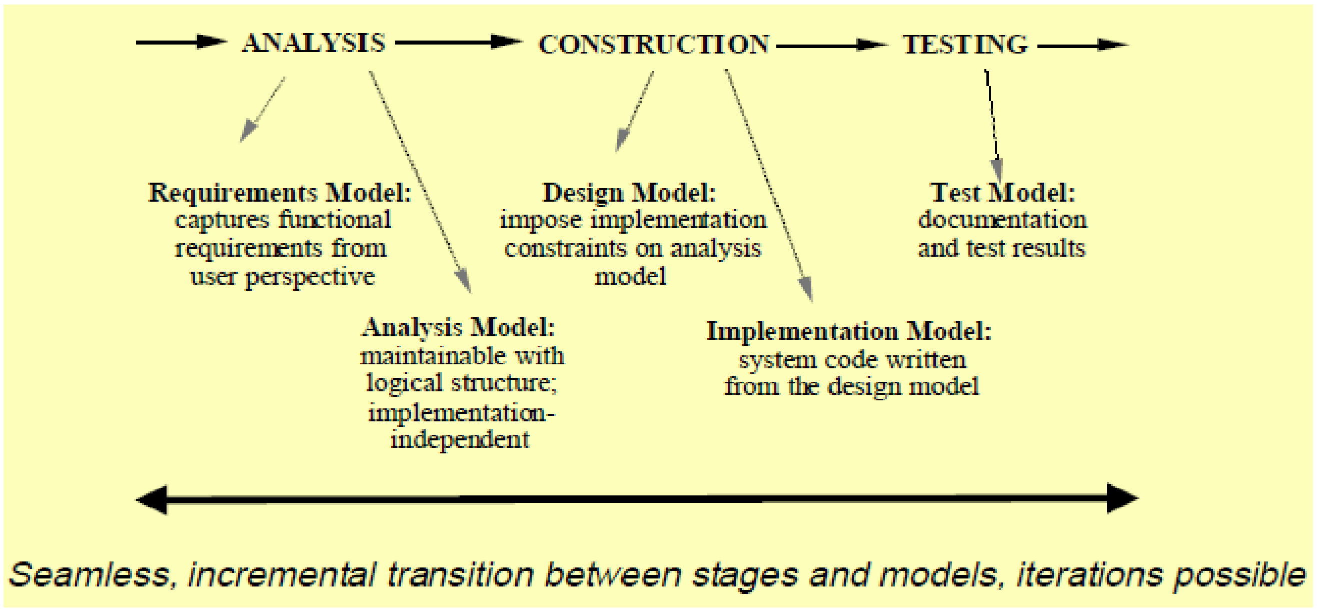

ANALYSIS
CONSTRUCTION
TESTING
Requirements Model: captures functional requirements from user perspective
Design Model: impose implementation constraints on analysis model
Test Model: documentation and test results
Analysis Model: maintainable with logical structure; implementation-independent
Implementation Model: system code written from the design model
Seamless, incremental transition between stages and models, iterations possible

Figure 8: System development as a construction of models

Source: Adapted from Jacobson et al. (1994)

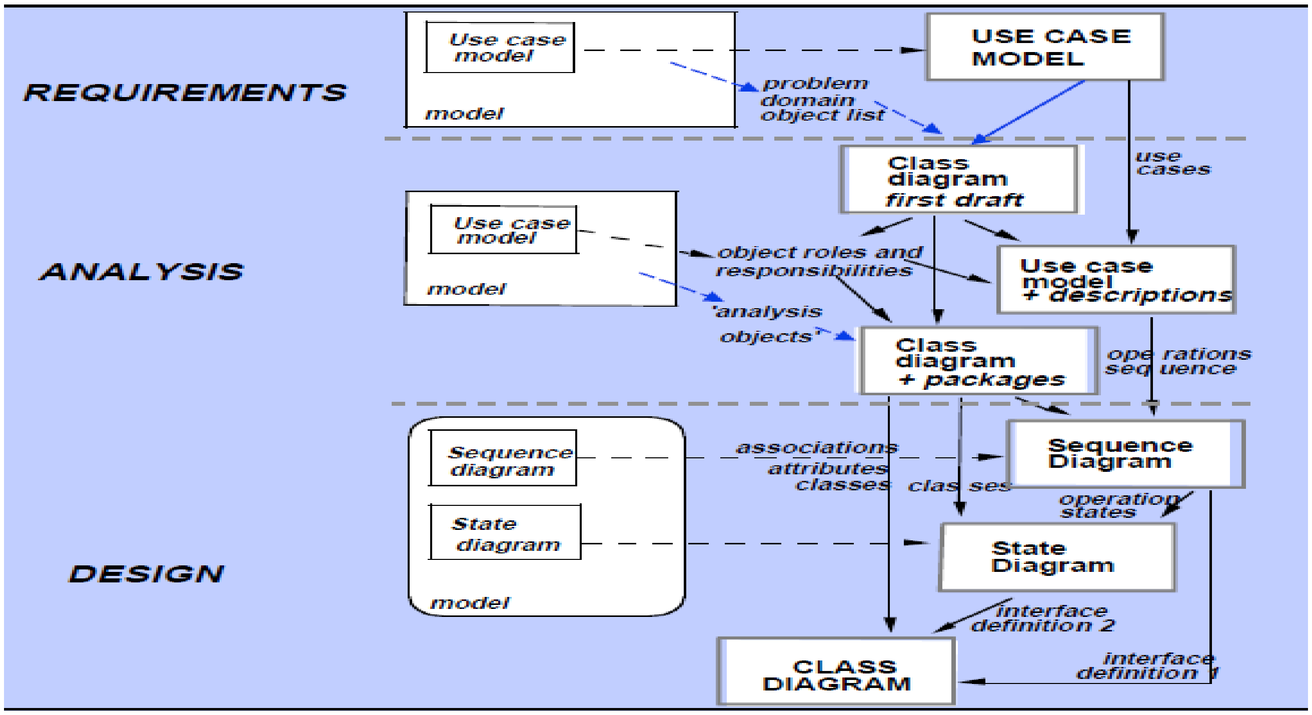


Figure 9: Incremental transition between from Requirement, Analysis and Design phases

Source: Adapted from Jacobson et al. (1994)

Other popular Object-Oriented Software Development approaches include Object Modeling Technique (OMT) by Rumbaugh (1996), The Booch Method by Booch (1994), Object-Oriented Analysis (OOA) by Coard and Yourdon (1991) and also Classes-Responsibility and Collaboration (CRC) approach by Wirfs-Brock (1989) among others. None of these approaches provide seamless incremental transition from requirement analysis, design, implementation, testing and deployment phases comparable to use-case approach. Nevertheless, an attempt to address the aforementioned limitation was made in the year 1999 by Jacobson, Booch and Rumbaugh (1999) through the development of a Unified Software Development Process.

4.5 Functional Model

The functional model was represented with the help of use case diagrams and describes the functionality of the system from users' perspective.

4.5.1 Use Case Diagram

Figure 10 below illustrates the users' view of prototype system and their interactions with the system.

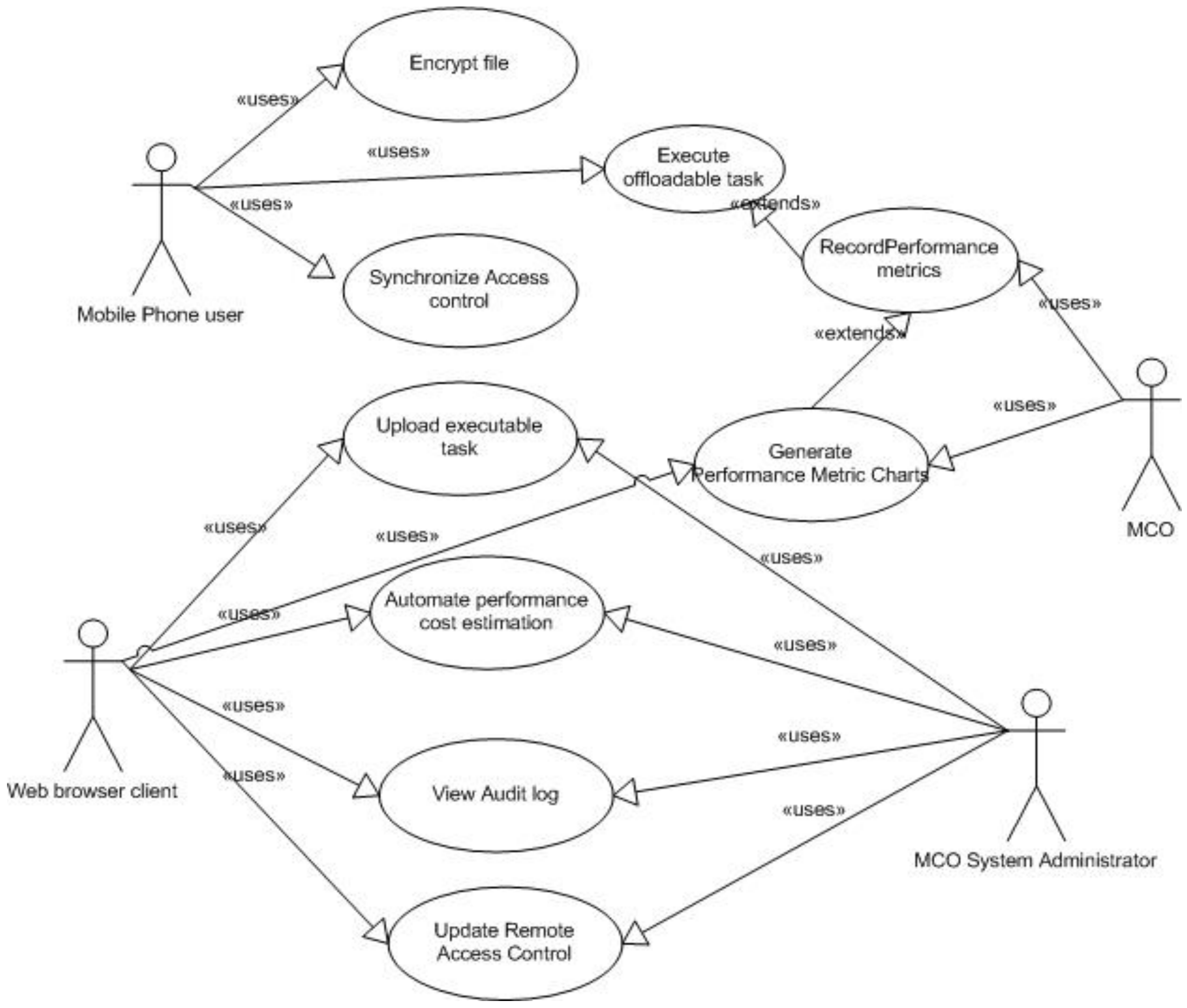


Figure 10 Use Case Diagram

Source: Author's compilation

4.5.2 Use Case Template

4.5.2.1 Encrypt File

Use Case Name: Encrypt file

Participating Actor: Initiated by Mobile Phone User

Entry Condition:

i) Mobile Phone User is ready to select a Microsoft word document that will be used as input in the system.

Flow of Events:

ii) Mobile Phone User clicks on "**File Registration**" button on Mobile phone enable cyber foraging application and select a file using a file browser pop up on an Android based phone.

iii) The system encrypt the file and save encrypted file in the disk storage by appending **encrypted_**{***file_name***}

iv) The System generates an Audit log and Access Control records and save the same on the mobile phone.

v) The System initializes Access Control for encrypted file as unrestricted and launches Remote Access Control and Auditing synchronization service in the background.

Exit Condition:

vi) The Mobile Phone User is notified of the outcome of the encryption.

Special Requirements:

A plain text file to be encrypted must be uploaded prior to the invocation of Encrypt file use case..

4.5.2.2 Execution of an Offloadable Task

Use Case Name: Execution of an offloadable task

Participating Actor: Initiated by Mobile phone User

Communicates with ***Offloading Engine***, ***Security***, ***Persistence*** and ***Performance Metrics*** modules.

Entry Condition:

i) Mobile phone User is ready to select an encrypted file as an input into for potentially offloadable task.

Flow of Events:

ii) Mobile phone user select an encrypted file and click on "**Task execution**" button.

iii) The request is processed by ***Persistence*** and ***Security modules*** to make a determination on allowed access control. If access is allowed; input file is decrypted and handed over to ***Offloading Engine*** else the execution of task is terminated and an access denied message displayed to mobile user.

iv) The ***Offloading Engine*** makes a determination on whether to execute the task locally or remotely.

v) ***Performance Metrics*** module captures the execution times and stores them on the mobile phone.

vi) ***Security module*** logs the Audit log for the file access locally.

iv) The result obtained is displayed to mobile phone user transparently i.e. without him being aware on whether the task was executed locally or remotely.

Exit Condition:

v) Mobile phone user receives results of task execution.

Special Requirements:

Access Control record must be in existence and associated with the selected file in Mobile Phone device prior to execution of potentially offloadable task.

4.5.2.3 Database Tables Synchronization Service

Use Case Name: Synchronization of Access Control, Audit log and Performance metrics

Participating Actor: Initiated by the Mobile Computing Offloading (MCO) system or Mobile Phone User.

Communicates with ***Offloading Engine*** and ***Persistence*** modules.

Entry Condition:

i. Existence of dissimilar Access Control, Audit log and Performance metrics records in either Mobile Phone client and offloading server.

Flow of Events:

ii. Database synchronization service polls Access Control, Audit log and Performance metrics records from the mobile phone.

iii. Performs a ”**wget**” operation to determine the availability of the server. If it is available it hand the records to server synchronization module one record at a time.

iv. The server synchronization module performs database synchronization and returns the updated record and a instruction indicating whether an update is required on the client side or not.

v. Mobile phone database synchronization module updates local database ones instructed by the server to do so, otherwise the update is ignored.

vi. Database synchronization service resumes polling Access Control, Audit log and Performance metrics records and repeats the (i-v) above.

Exit Condition:

Existence of similar Access Control, Audit log and Performance metrics records in both Mobile Phone client and offloading server.

4.5.2.4 View Performance Metrics Graphs

Use Case Name: Generate Performance Metrics charts

Participating Actor: Web-browser client user and MCO

Communicates with ***Persistence***, ***Performance Metrics and Chart*** modules.

Entry Condition:

i. A potentially offloadable task has been executed (either locally or remotely) and the database table records in the server have been synchronized.

Flow of Events:

ii. Web-browser client user clicks on a link “**View chart**” on the web page.

iii. The System processes the request by using Persistence module to Query the database and ***Performance metrics module*** to generate required input to the ***Chart module.***

iv. ***The Chart module generates*** the graph a visual representation of the same to the mobile client.

Exit Condition:

v. Web-browser client user view performance metric chart (bar graph or line graph).

Special Requirements:

Existence of task execution time metrics records at the server.

4.5.2.5 Remote Access Control Management

Use Case Name: Update Remote Access Control

Participating Actor: Web-Browser client user and MCO Administrator

Entry Condition:

i. Existence of Access Control record at the server.

Flow of Events:

ii. Web-Browser client user clicks on a link **"Access Control"** on the application's web page.

iii. The System generates a table of all Access Control records together with their updated access permission status.

iv. Web-Browser client user enables or disables access permission status "**Check box**" of any of the Access Control records he/she deems necessary. The user can also disable or enable all permission on Access Control from a single click.

v. The Web-Browser client user clicks on "**Save button**".

vi. The system updates server-side Access control record and returns a confirmation message to the user.

vii. The Database Synchronization module schedules updated Access Control record for client-side synchronization.

Exit Condition:

viii. Modified Access Control record at the server side.

4.5.2.6 Audit File Access

Use Case Name: View Audit log

Participating Actor: Web-Browser client user and MCO Administrator

Entry Condition:

i. Existence of Audit record at the server.

Flow of Events:

ii. Web-Browser client user clicks on a link **"Audit"** on the application's web page.

iii. The System generates a table of all Audit log records

iv. Web-Browser client user views the record and exit the page.

Exit Condition:

v. File Access log audited.

4.5.3 Scenario: Extractive Text Summarization

4.5.3.1 Encrypt File

Use Case Name: Encrypt file

Participating Actor: John: Mobile Phone User

Flow of Events:

i) John has mobile phone and would like to secure a Microsoft word document in it containing confidential text. He would also like to control the document access remotely incase his mobile phone device is lost.

ii) He clicks on "**File Registration**" button on Android phone enable mobile phone based cyber foraging application and select a file using a file browser.

iii) The system encrypt the file and save encrypted file in the disk storage by appending **Encrypted_{*file_name*}**

iv) The System generates an Audit log and Access Control records and save the same on the mobile phone.

v) The System initializes Access Control for encrypted file as unrestricted and launches Remote Access Control and Auditing synchronization service in the background.

vi) The system notifies John on the outcome of file encryption.

4.5.3.2 Execution of Extractive Text Summarization

Use Case Name: Execution of an offloadable task

Participating Actor: John: Mobile phone User

Communicates with ***Offloading Engine***, ***Security***, ***Persistence*** and ***Performance Metrics*** modules.

Flow of Events:

i) John select an encrypted file, enters the number of sentences expected in output text summary and click "**Summarize**" button.

ii) The request is processed by ***Persistence*** and ***Security modules*** to make a determination on allowed access control. If access is allowed, the input file is decrypted and handed over to ***Offloading Engine*** else the task is terminated and an access denied message displayed to John.

iii) The ***Offloading Engine*** makes a determination on whether to execute the task locally or remotely. This is done transparently and John is not aware of it.

iv) ***Performance Metrics*** module captures the execution times and stores them on the mobile phone.

v) ***Security module*** logs the Audit log for the file access locally.

iv) The result obtained from is displayed to the John transparently without him being aware on whether the task was executed locally or remotely.

4.5.3.3 File Access Control Management

Use Case Name: Update Remote Access Control

Participating Actor: John: Web-Browser client user

Flow of Events:

i. John has lost his mobile phone device but the device is within WLAN. He clicks on "**Access Control"** link on the application's web page.

ii. The System generates a table of all Access Control records together with their updated access permission status.

iii. John disables access permission status "**Check box**" of a specific file he considers sensitive and clicks on "**Save button**".

iv. The system updates server-side Access control record and returns a confirmation message to the John.
v. The Database Synchronization module schedules updated Access Control record for client-side synchronization. Consequently locking out access to the file.

4.5.3.4 Audit file Access

Use Case Name: View Audit log
Participating Actor: John: Web-Browser client user
Flow of Events:

i. John clicks on a link **“Audit”** on the application’s web page.
ii. The System generates a table of all Audit log records
iii. John undertakes a forensic Audit and exit the page.

4.5.3.5 View Performance Metrics Graphs

Use Case Name: Generate Performance Metrics charts
Participating Actor: John: Web-browser user
Flow of Events:

i. John clicks on a link “**View chart**” on the web page.
ii. The System processes the request, generates a graph (Bar and line chart) and displays it to John.
iii. John studies the graph (Bar and line chart) and exit the page.

4.6 Summary

This chapter adopted use-case approach to specify system requirements for Secure Mobile Phone based Cyber foraging prototype system. Despite numerous object-oriented related references made in this chapter, the use-case approach is also applicable when constructing software systems based on non object-oriented techniques. The perspective provided by use cases reinforces the ultimate goal of software engineering by assisting involved-analysts and end-users to arrive at a common, shared vision of what the product they are specifying will do. This is a key aspect in constructing quality software as asserted by Karl (1997).

CHAPTER FIVE: ANALYSIS AND DESIGN

5.1 Introduction

In this chapter we will convert use cases specified during requirement specification in Chapter four of this document into analysis and design models. This is achieved through use-case analysis and design approach as proposed by Jacobson et al. (1994). This entailed.

i. Identification of classes which perform a use case's flow of events.
ii. Distribution of use case behavior to those classes, using use-case realizations.
iii. Identification of responsibilities, attributes and associations of the classes.
iv. Identification of usage of architectural mechanisms.

5.2 Analysis

The following steps were followed from which various analysis models were developed:-

STEP I: Supplementing the Use-Case Descriptions

STEP II: For each use case realization

- o Finding Analysis Classes from Use-Case Behavior
- o Distribution of Behavior to Analysis Classes

STEP III: For each resulting analysis class

- o Description of Responsibilities
- o Description of Attributes and Associations
- o Definition of Attributes
- o Establishment of Associations between Analysis Classes
- o Description of Event Dependencies between Analysis Classes
- o Qualification of Analysis Mechanisms

STEP IV: Evaluation of Results of Use-Case Analysis

STEP I: Was carried out to capture additional information needed in order to understand the required internal behavior of the system that were missing from the use-case description.

STEP II: Was carried out to identify a candidate set of model elements (analysis classes modeled as **object models**) which were capable of performing the behavior described in use cases. Similarly use-case behavior in terms of collaborating analysis classes and responsibilities of analysis classes were determined. This were then expressed in terms of **Sequence** and **Collaboration** model diagrams

STEP III: Was carried out to enrich analysis classes identified by identifying attributes, methods and association for each class. Furthermore, Event Dependencies between Analysis Classes were identified for objects that needed to know when an event occurs in some "***target***" object, without the "***target***" having to know all the objects which require notification when the event occurs. Event Dependencies were modeled using **Subscribe-Association** model. Thereafter, we qualified Analysis Mechanisms (special information that were of interest) for each class that could substantially influence architectural design. Analysis Mechanisms identified were captured **using a note** attached to a diagram to convey the information.

5.2.1 Analysis Model

Figure 11 to 28 illustrates various analysis model identified in the prototype system.

5.2.1.1 Object Model

i. **Encrypt file**

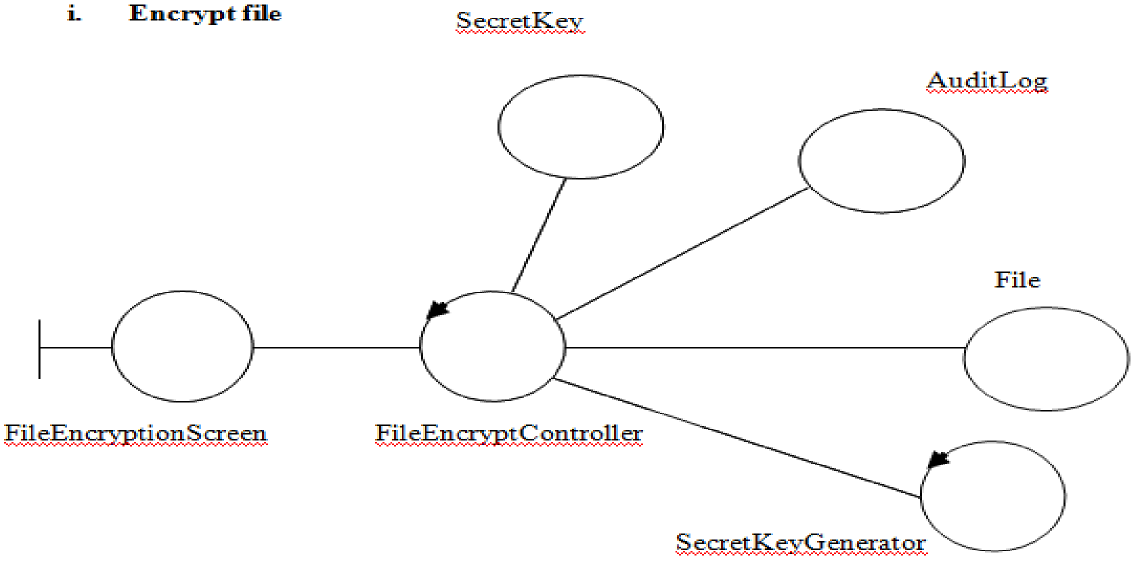


Figure 11: File Encryption object model

Source: Author's compilation

ii. Execute offloadable method (Extractive text summarization)

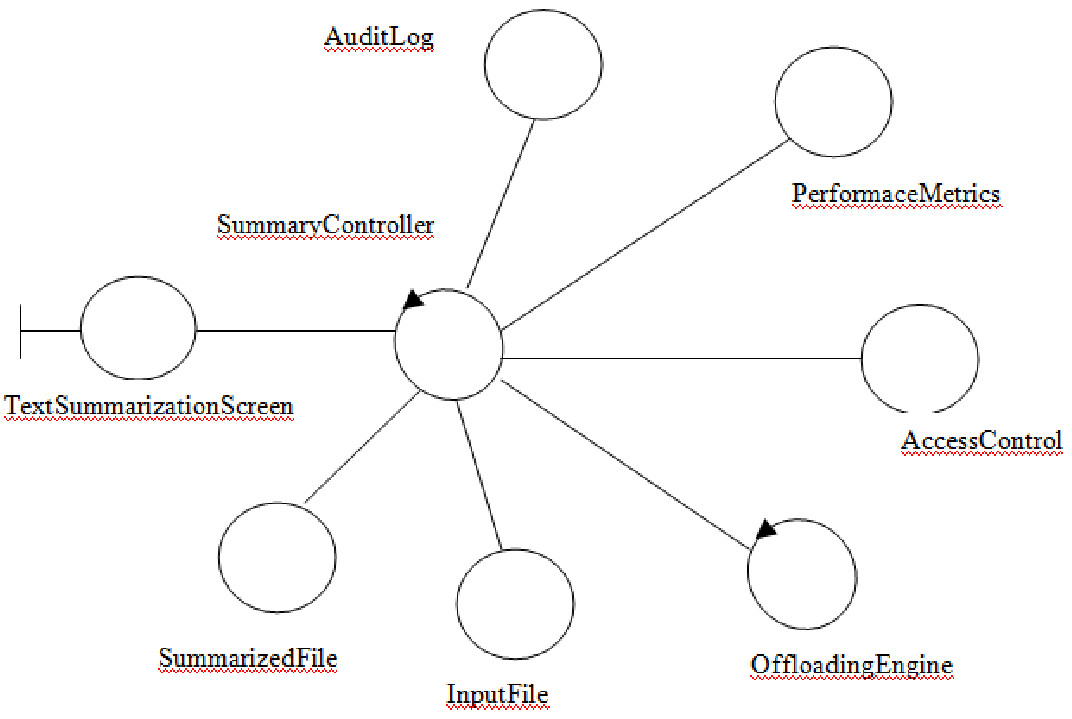


Figure 12 : Extractive text summarization object model

Source Author's compilation

iii. Synchronize Access Control

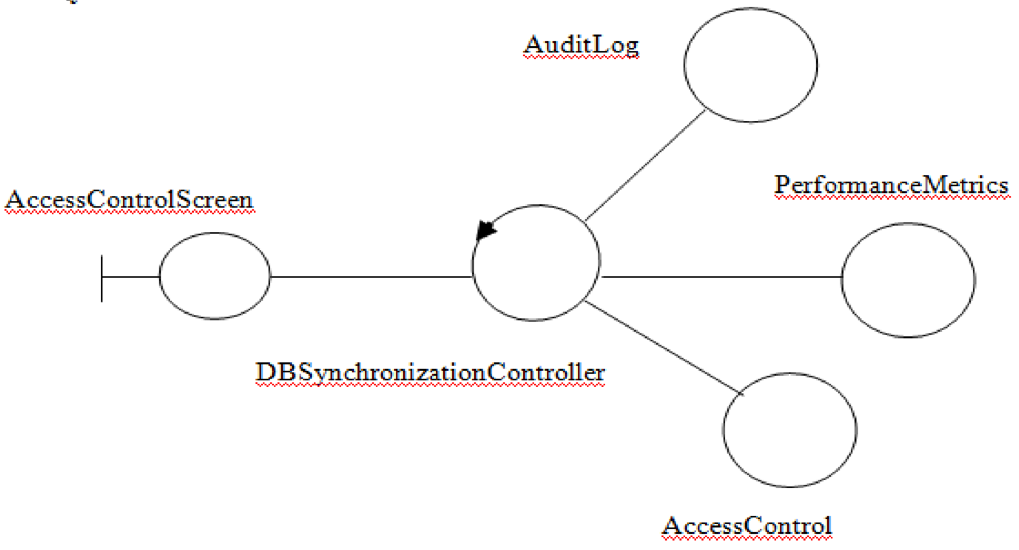


Figure 13: Access Control Synchronization object model

Source: Author's compilation

iv. Update Access Control management

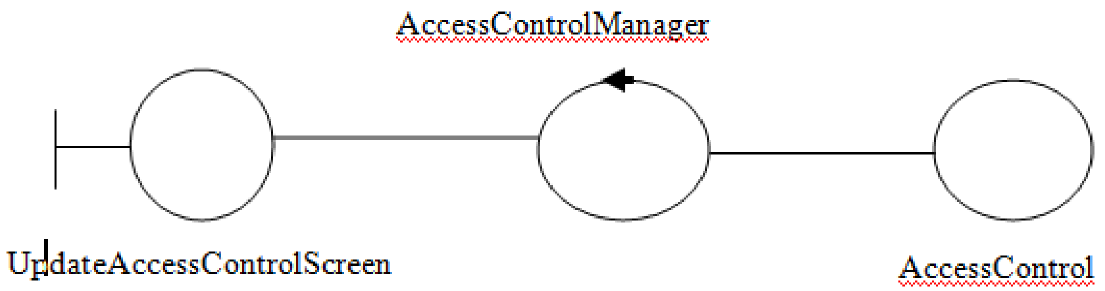


Figure 14: Update Access Control object model

Source: Author's compilation

v. View Audit log

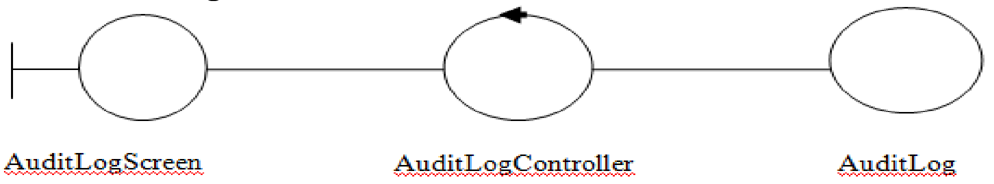


Figure 15 : View Audit log object model

Source: Author's compilation

vi. View Performance Metrics

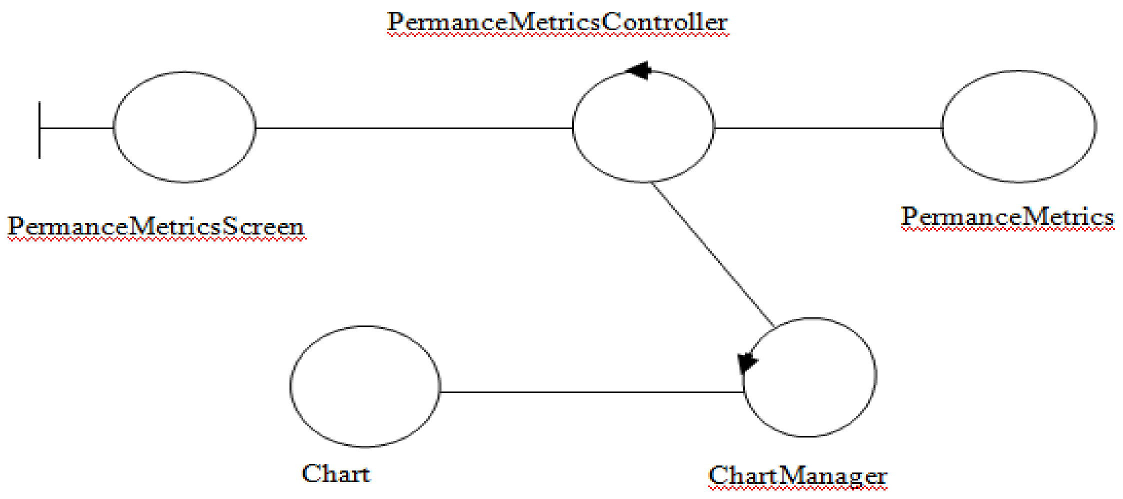


Figure 16 : View Performance Metrics object model

Source: Author's compilation

5.2.1.2 Sequence Diagrams

i. File Encryption

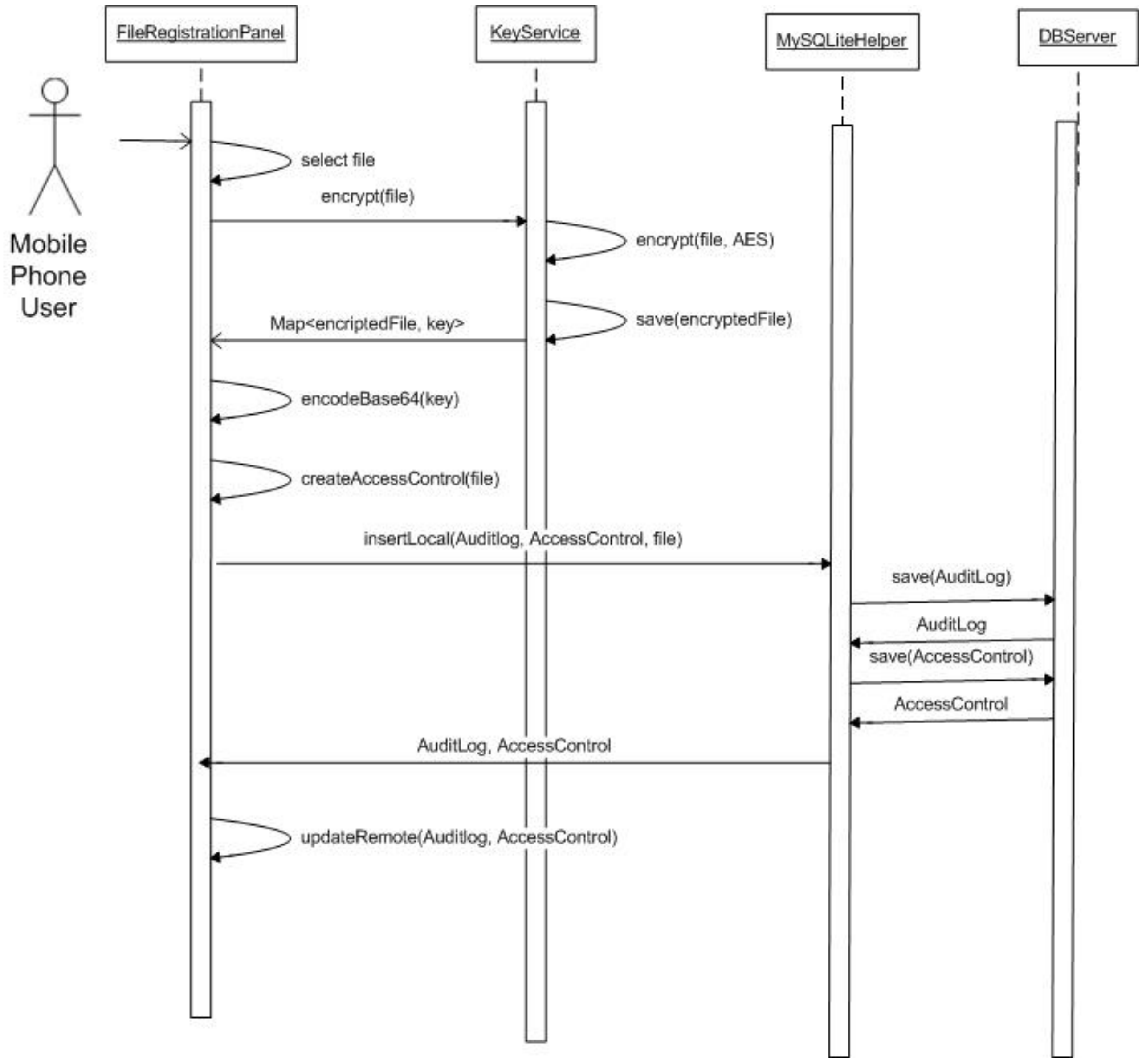


Figure 17 : File Encryption Sequence diagram

Source: Author's compilation

ii. Extractive Text Summarization

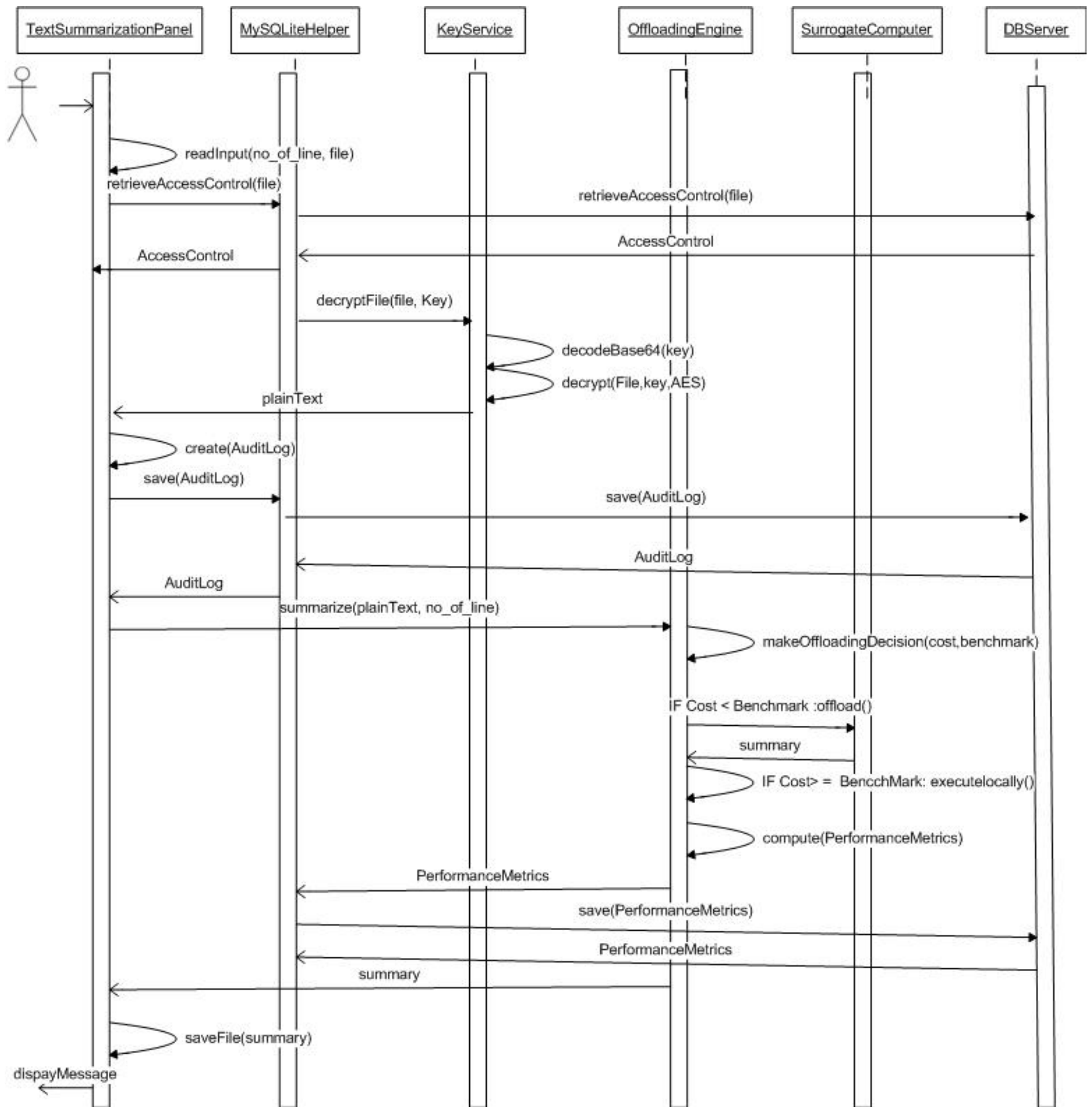


Figure 18: Extractive text summarization sequence diagram

Source: Author's compilation

iii. Update Access Control

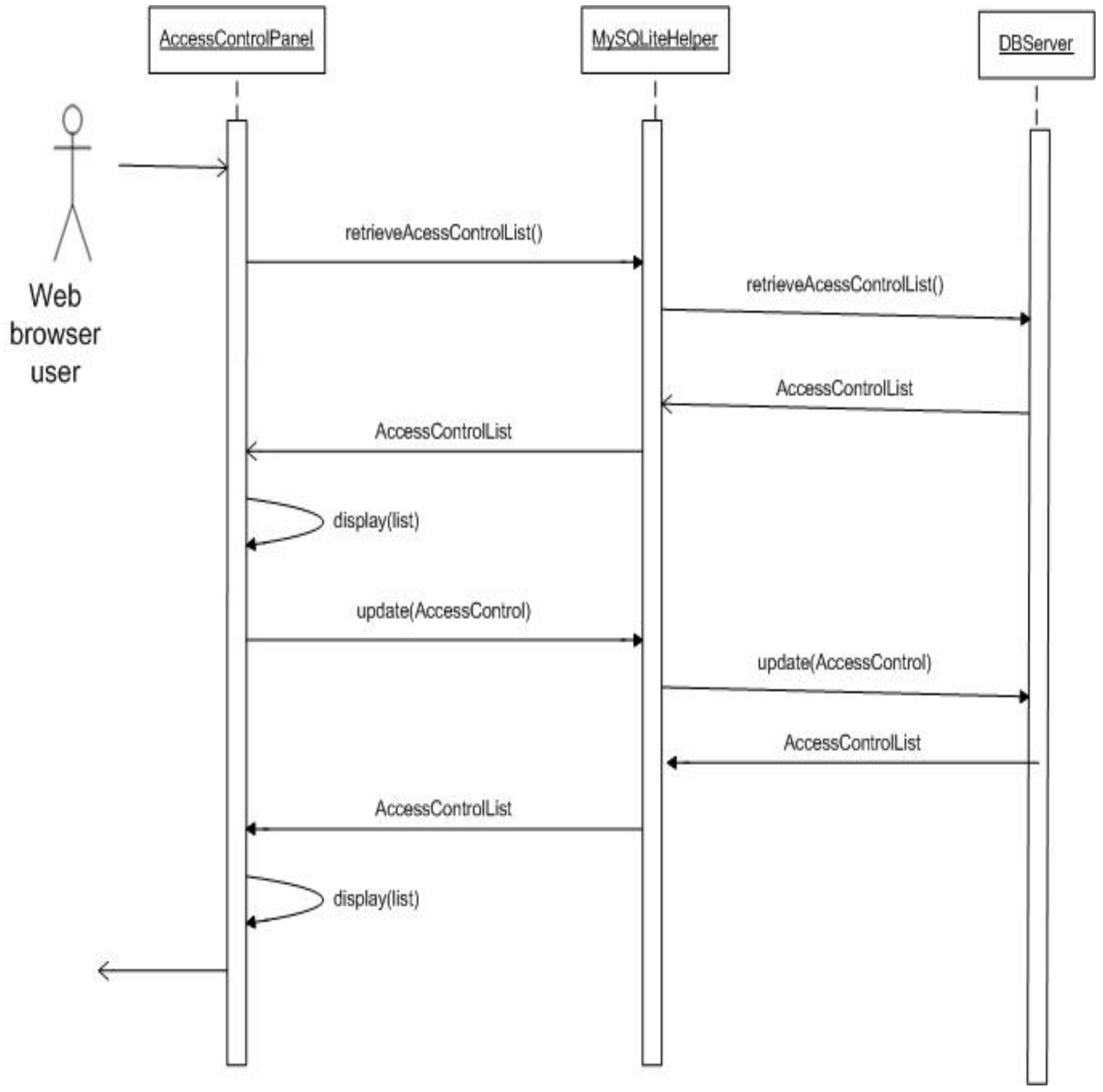


Figure 19 : Manage Access Control Sequence diagram

Source: Author's compilation

iv. View Audit log

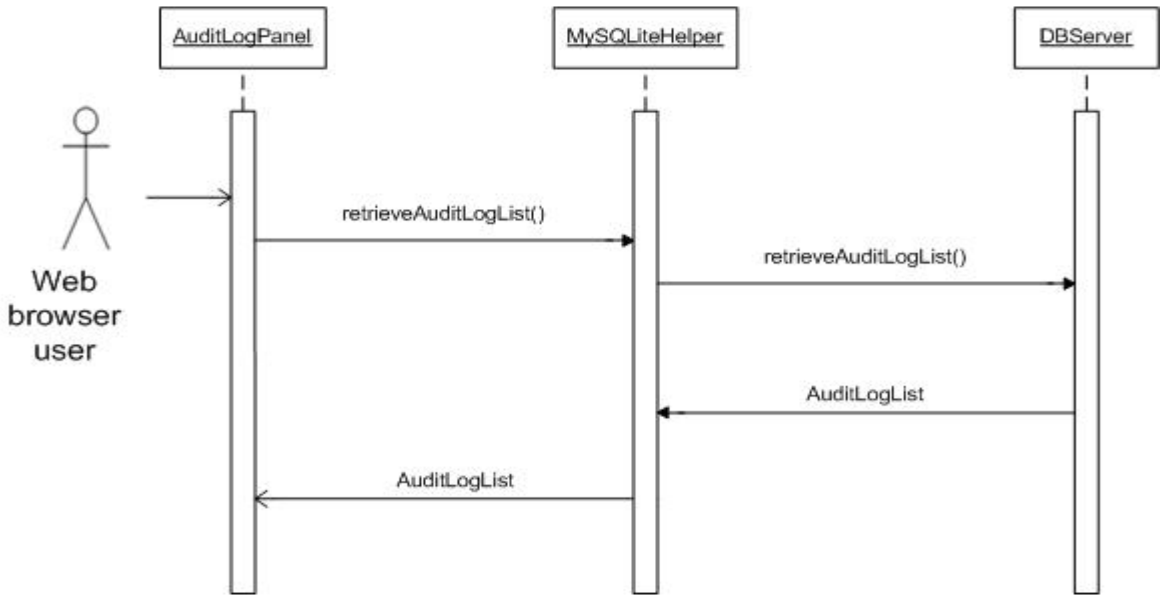


Figure 20 : View Audit log Sequence diagram

Source: Author's compilation

v. View Performance metrics

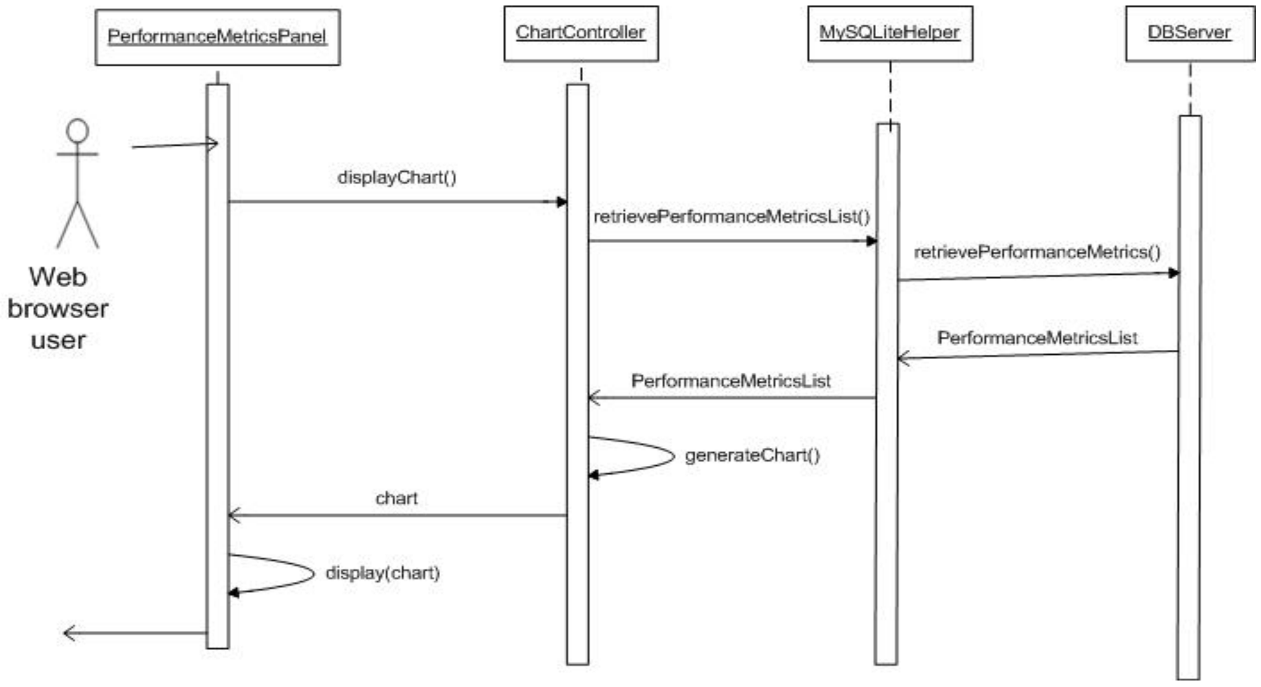
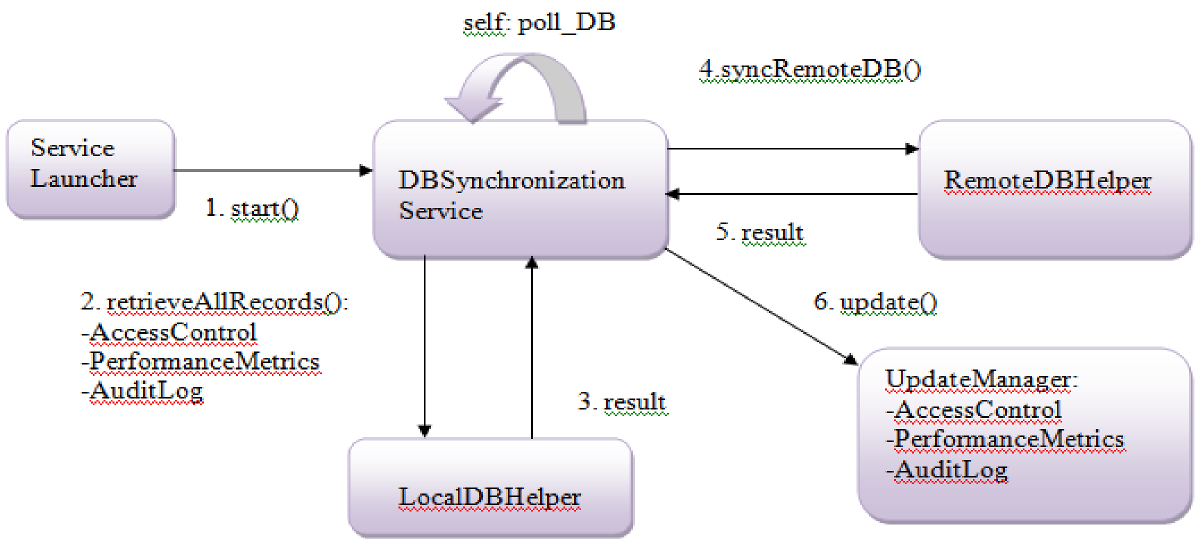


Figure 21 : View Performance metrics

Source: Author's compilation

5.2.1.3 Collaboration Diagrams

i. Client side: synchronize Access Control

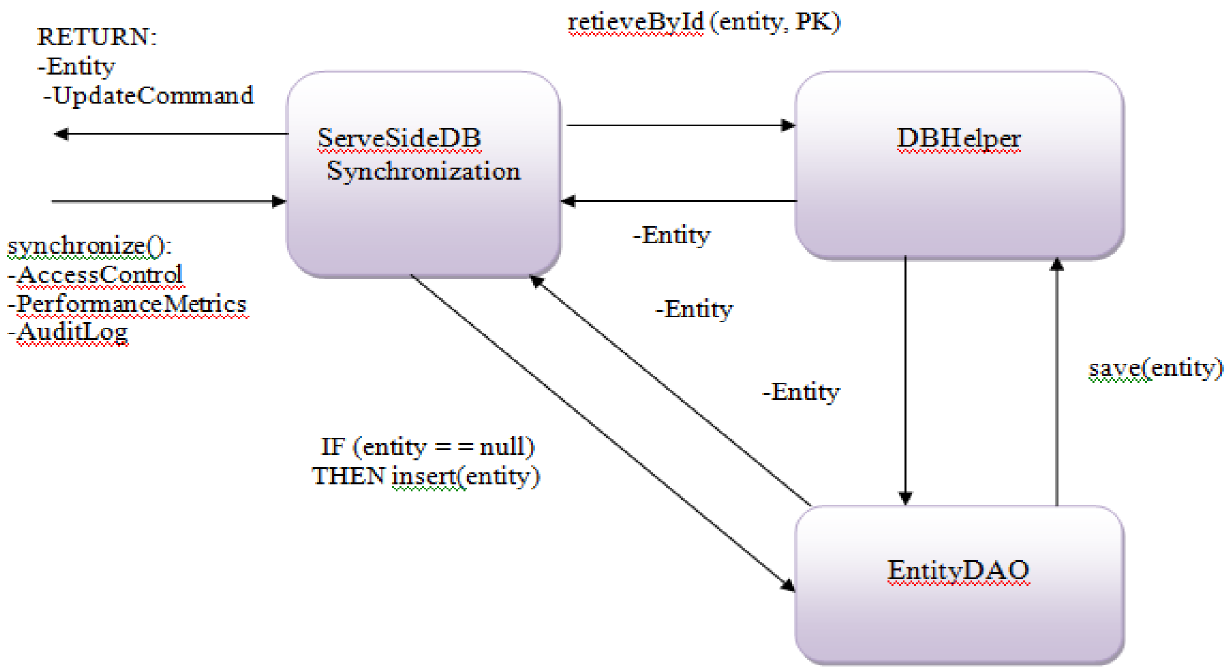


self: poll_DB
4.syncRemoteDB()
Service Launcher
1. start()
DBSynchronization Service
RemoteDBHelper
5. result
2. retrieveAllRecords(): -AccessControl -PerformanceMetrics -AuditLog
6. update()
UpdateManager: -AccessControl -PerformanceMetrics -AuditLog
3. result
LocalDBHelper

Figure 22 : Client side Access Control Synchronization Collaboration diagram

Source: Author's compilation

ii. Server side: synchronize Access Control



RETURN: -Entity -UpdateCommand
retieveById (entity, PK)
ServeSideDB Synchronization
DBHelper
-Entity
synchronize(): -AccessControl -PerformanceMetrics -AuditLog
-Entity
-Entity
save(entity)
IF (entity = = null) THEN insert(entity)
EntityDAO

Figure 23 : Server side Access Control Synchronization Collaboration diagram

Source: Author's compilation

5.2.1.4 Activity Diagrams

i. Extractive text summarization

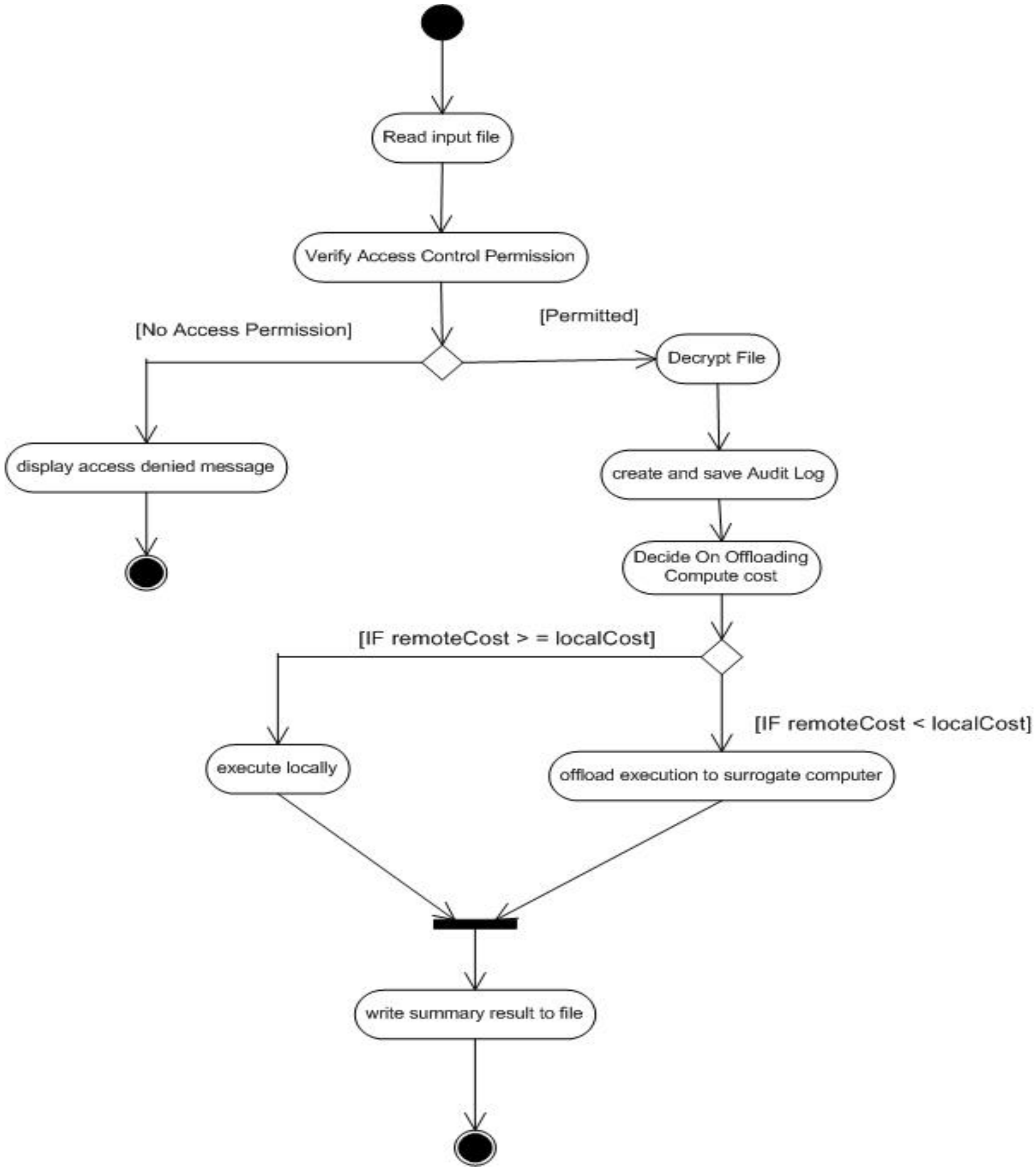

Read input file
Verify Access Control Permission
[No Access Permission]
[Permitted]
Decrypt File
display access denied message
create and save Audit Log
Decide On Offloading Compute cost
[IF remoteCost > = localCost]
[IF remoteCost < localCost]
execute locally
offload execution to surrogate computer
write summary result to file

Figure 24 : Extractive text summarization activity diagram

Source: Author's compilation

ii. Synchronize Access Control

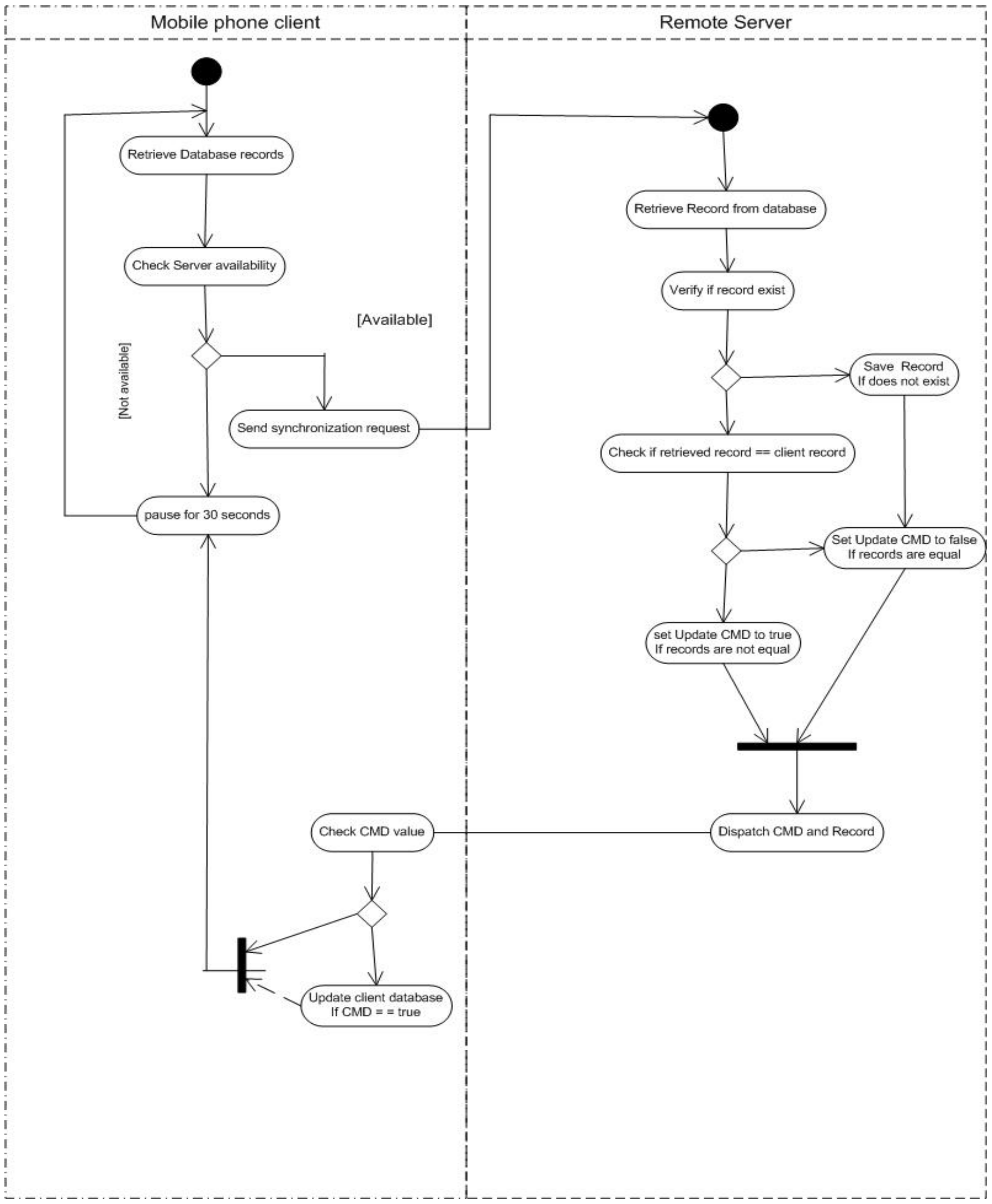

Mobile phone client
Remote Server
Retrieve Database records
Check Server availability
[Available]
[Not available]
Send synchronization request
pause for 30 seconds
Retrieve Record from database
Verify if record exist
Save Record If does not exist
Check if retrieved record == client record
Set Update CMD to false If records are equal
set Update CMD to true If records are not equal
Dispatch CMD and Record
Check CMD value
Update client database If CMD = = true

Figure 25 : Synchronize Access Control activity diagram

Source: Author's compilation

iii. File encryption activity

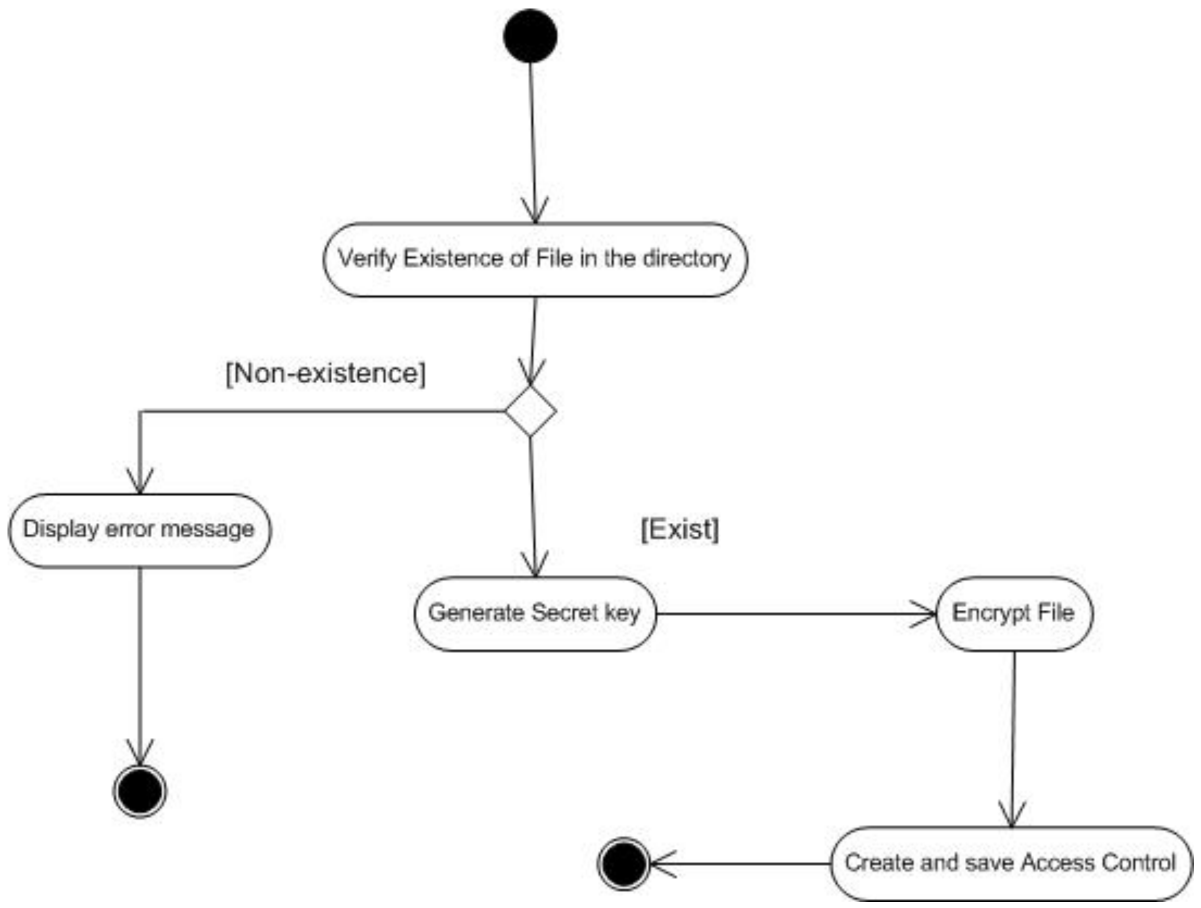


Figure 26 : File encryption activity

Source: Author’s compilation

5.2.1.5 State Machine Diagrams

To model the dynamic behavior of the entire system we used the following state machines.

i. Client side (Mobile Phone)

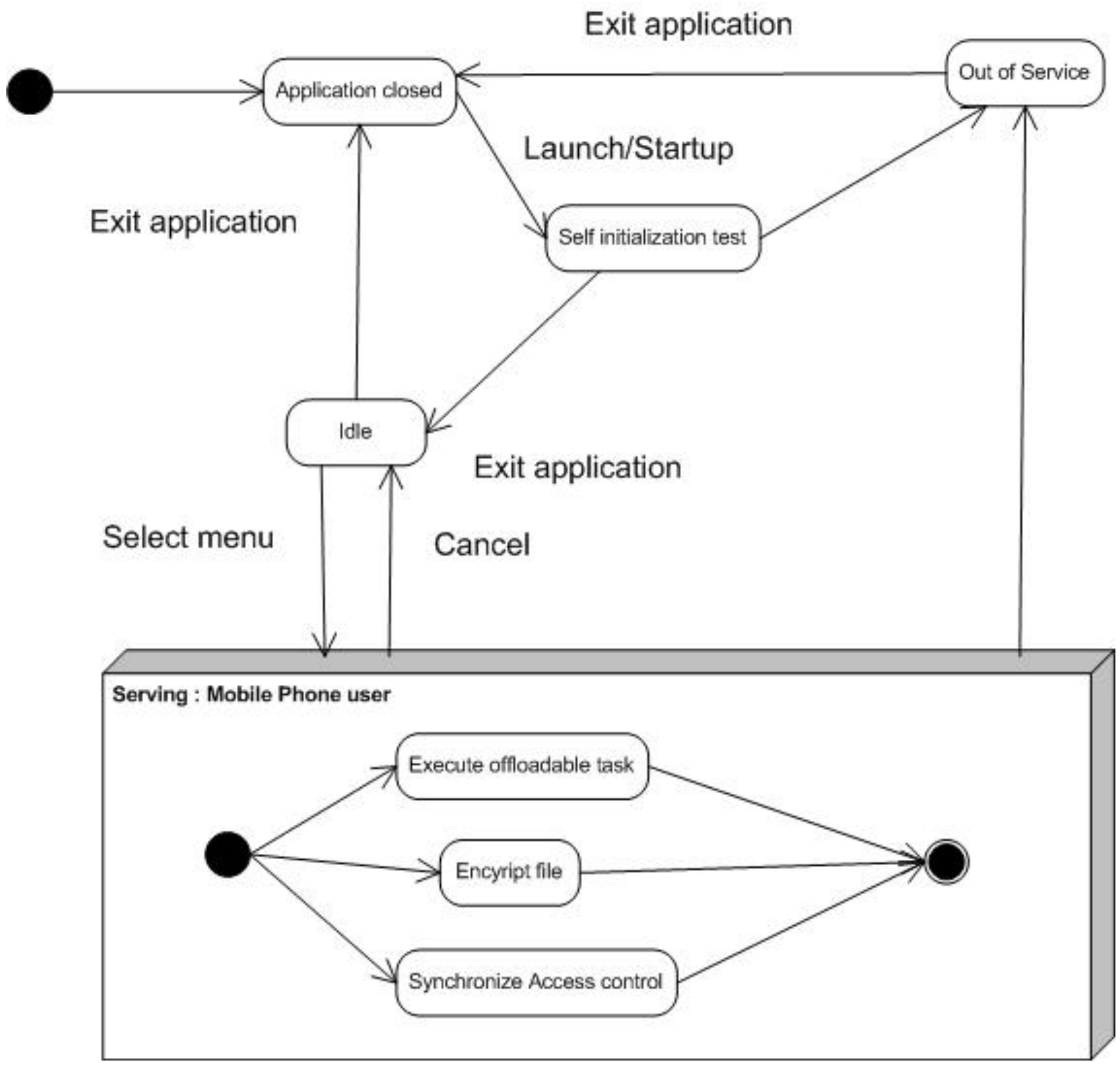


Figure 27 : Client side state machine diagram

Source : Author's compilation

ii. Server side (Surrogate computer)

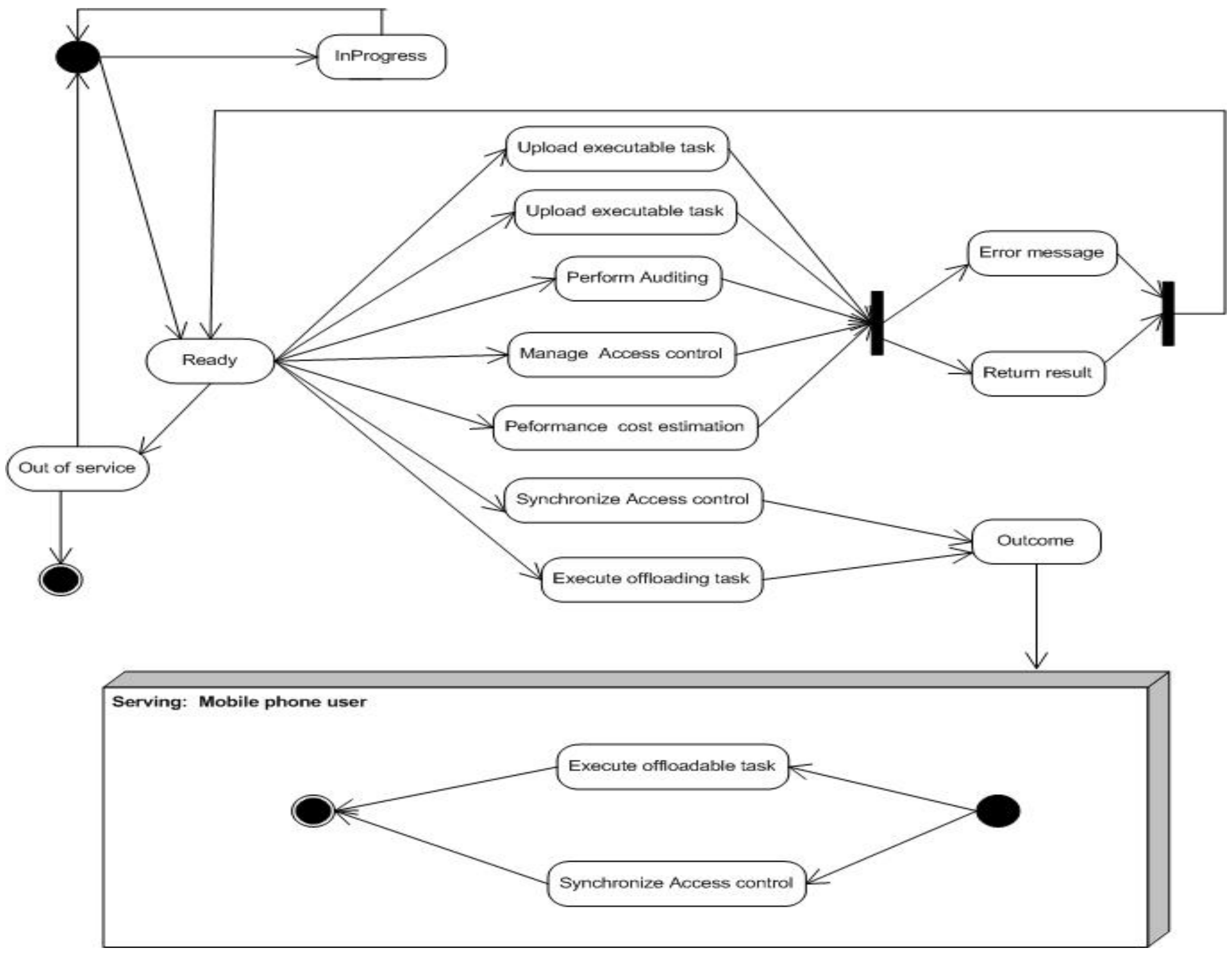


Figure 28: Server side state machine diagram

Source : Author's compilation

5.3 Design

5.3.1 Overall System Design

The prototype system (**Secure Mobile phone based Cyber foraging system**) hereafter christened as **Secure-MCO** is made up of many interactive software systems which collectively present a complex structure of components. In order to clearly envision the overall blueprint of the system, multiple views or the overall system are needed. These views were classified as either infrastructure architecture or application architecture. Figure 29 in the next page gives an overview of these views.

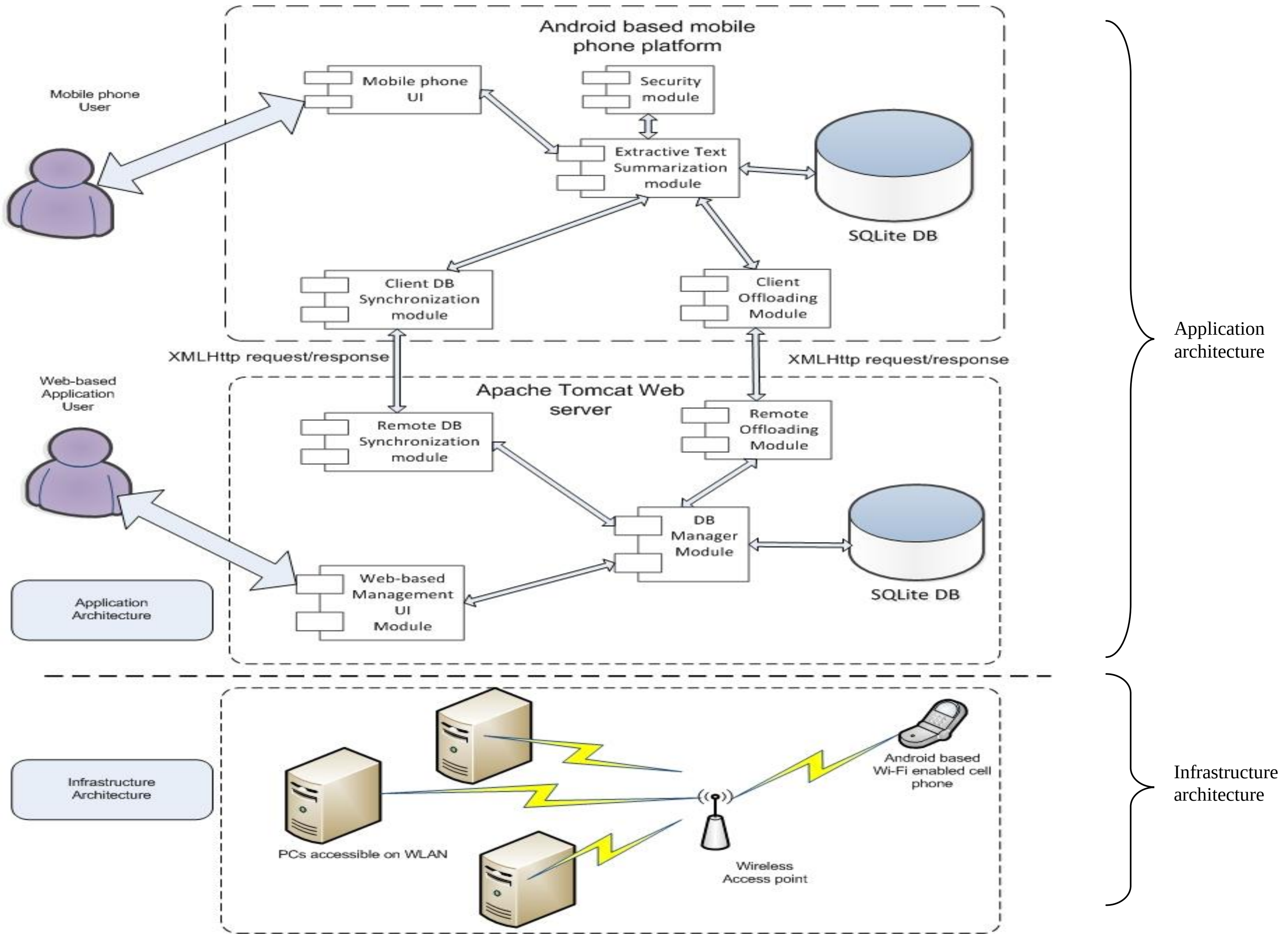


Figure 29 : Overall system design
Source: Author's compilation

Application Architecture as illustrated in figure 29 the previous page, provides the view of how various modules of **Secure-MCO** work together to support mobile phone based cyber foraging capabilities while at the same time enforcing Remote Access Control and Auditing (RACA) mechanism thus enhancing data integrity and confidentiality. These modules and components are categorized as:

i. Front-end User Interface (e.g. Mobile phone UI and Web-based UI)

ii. Application-level Communication Protocol (e.g XMLHttpRequest/XMLHttpResponse)

iii. Core Application (e.g. Client/ Remote Offloading Engine, Client/Remote Database Synchronization module)

iv. Back-end data storage (e.g Flat file and SQLite Database)

The Infrastructure Architecture illustrates a view of how the hardware and network topology are deployed to support **Secure-MCO** system. This includes Android based Wi-Fi enabled smart phone, Wireless access point (Wireless-Router), Several Desktop Computers configured to be accessible on WLAN.

5.3.2 Architectural Design

5.3.2.1 Infrastructure

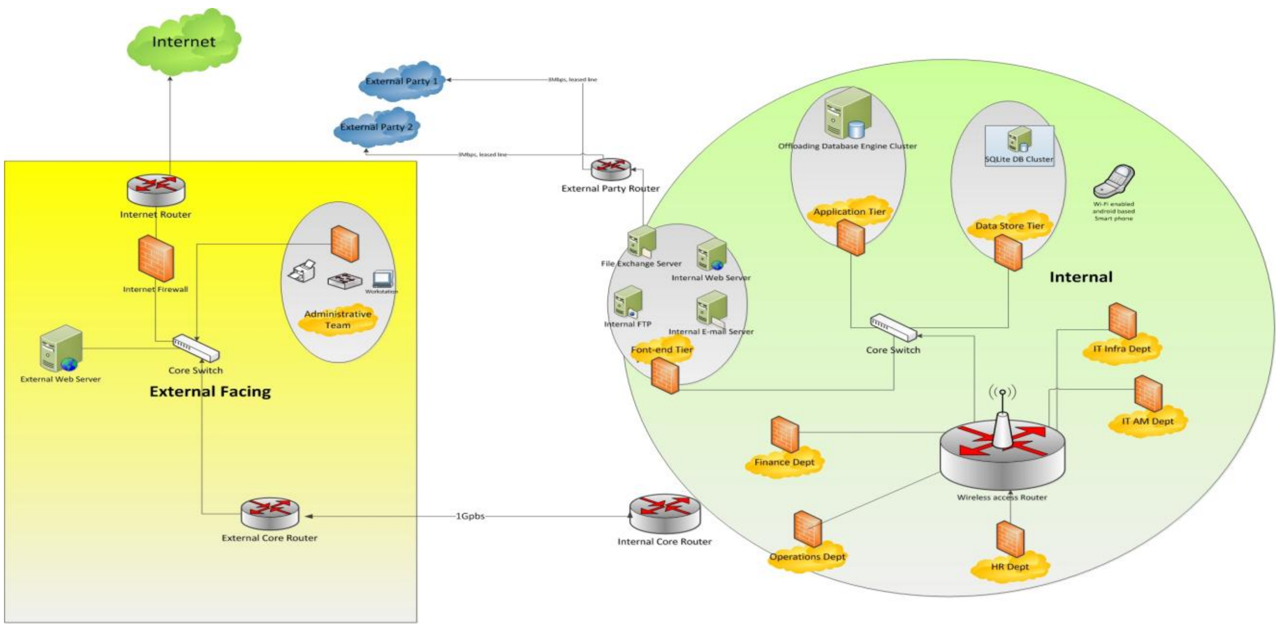


Figure 30: Infrastructure architecture
Source: Author's compilation

As regards Infrastructure design illustrated in figure 30 no special configuration is required except a wireless access point from which a mobile phone user can access WLAN to execution offloadable tasks on surrogate computers. In addition a web application user should be able to access the system both within and outside LAN.

5.3.2.2 Application

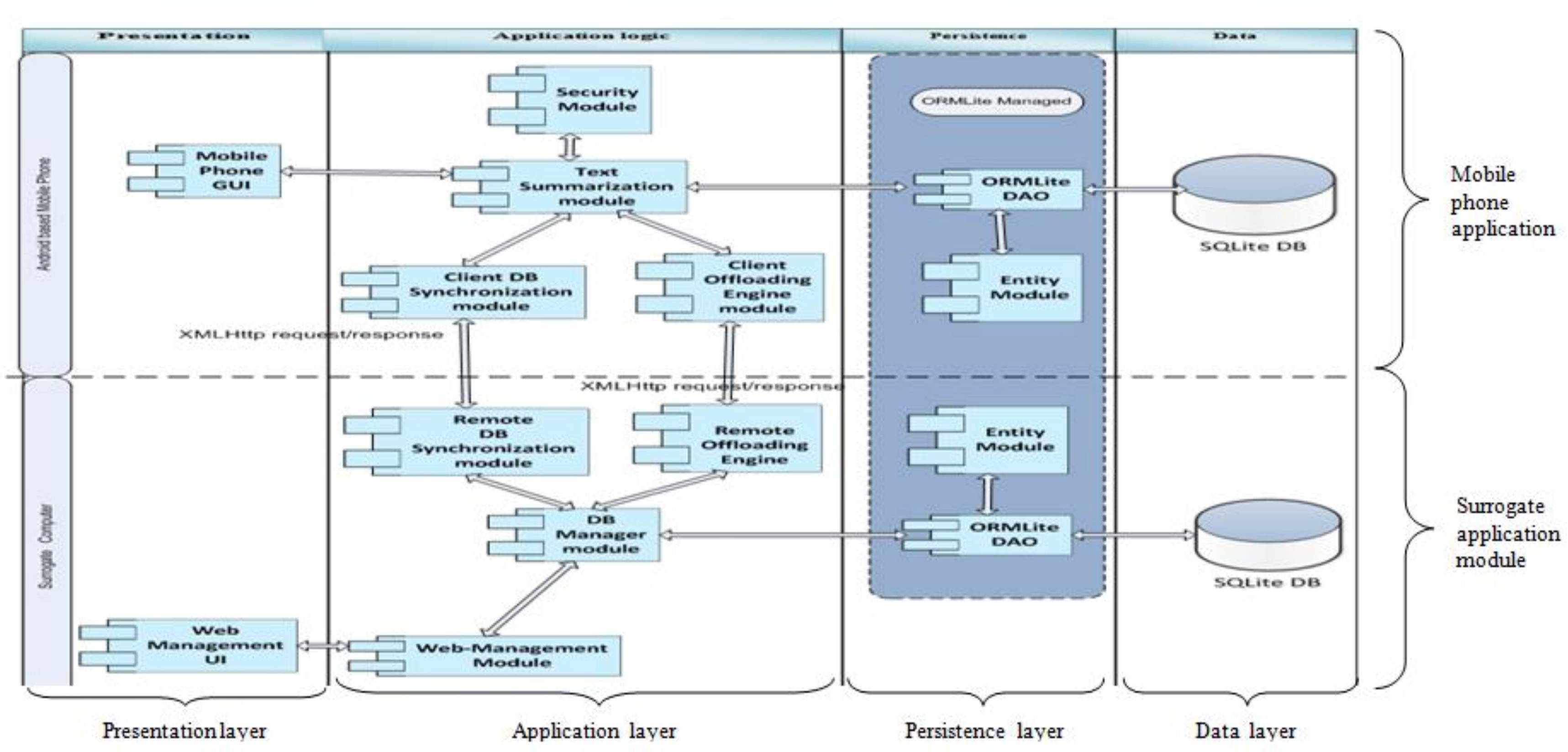


Figure 31 : Application architectural design
Source: Author's compilation

Figure 31 in the previous page illustrates application architectural design. We used layered software architectural pattern in designing the application. This enabled **Secure-MCO** to be robust, reliable, user-friendly and high-performance. This approach enables extension of the application and reusability of components across various modules. An overview of components in each tier is elucidated below.

i. Presentation Tier

Presentation tier contains components directly interfacing with end-users. These users include Mobile Phone User and Web-Based Users.

ii. Application Logic Tier

Application Logic tier is the backbone and integrator of various components of Secure-MCO. Though depicted as dependent with Remote modules located in Surrogate computer, both client-side and server-side application modules were designed to run independently and the presence or absence of any module does not prevent the other module from running. Another notable aspect of the application layer above is the communication between mobile phone application and the server through XML over HTTPS protocol (XMLHttpRequest and XMLHttpResponse) this allows easier access of surrogate computer located behind a firewall. **Client and Remote Database Synchronization module** together with **Security module** implement RACA mechanism. Furthermore, **Client and Remote offloading Engine modules** were integrated into the system to support offloading capability feature.

iii. Persistence layer

This tier connects the application logic and the data layer in **Secure-MCO** system. In order to bridge the disconnect between Object-Oriented (OO) and Relational Database we opted to use **ORMLite** library as an Object Relational Mapping (ORM). The choice of ORMLite was made due to its light-weight and Android platform support; contrary to existing heavy weight ORM such as **Hibernate, NHibernate and Oracle Toplink** ORM among others**.** F*igure 32 and 33* in the next page illustrate high-level architecture and java based implementation of an ORM.

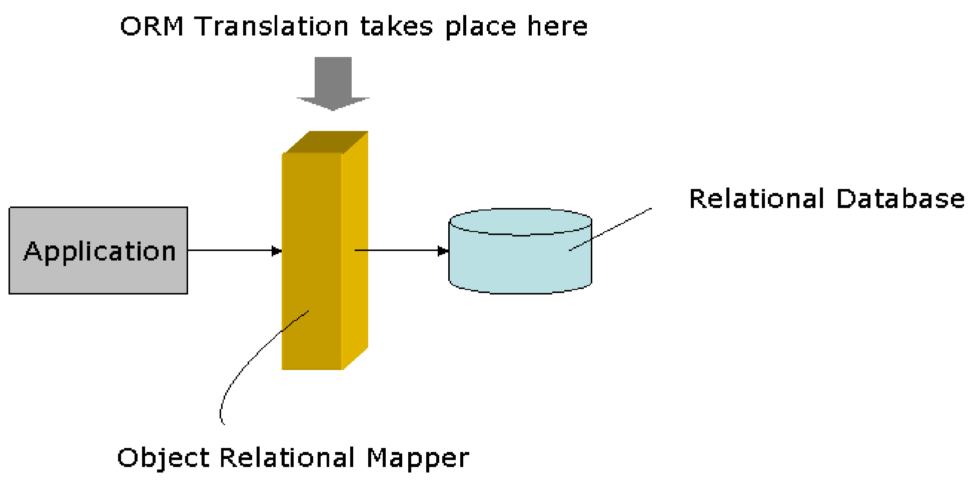


Figure 32 : High-level architecture of an ORM

Source: Author's compilation

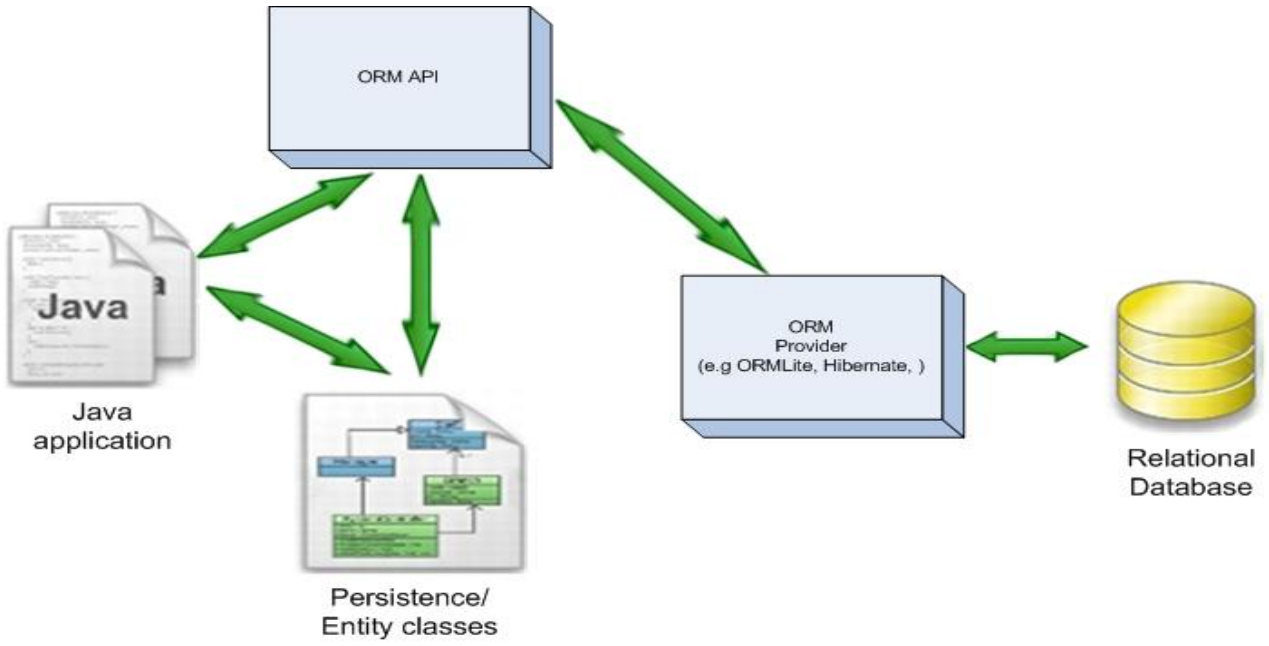

ORM API
Java
Java application
Persistence/ Entity classes
ORM Provider (e.g ORMLite, Hibernate,)
Relational Database

Figure 33 : Using ORM in java application

Source : Author's compilation

iv. Data Tier

Data tier contains the back-end data store for **Secure-MCO**. The database used in this tier was **SQLite** owing to its light-weight with minimal memory foot print (it successfully starts with memory footprint starts at about 50 kilobyte) thus suitable for resource constrained mobile phone.

5.3.3 Detailed Design

This section undertakes a detailed design of **Secure-MCO** with an aim of addressing inadequacy arising from Analysis and high-level architectural design phases above.

5.3.3.1 User Interface

The user interface is divided into two (for Mobile phone application and for Web-based application) as illustrated in **Appendix D.**.

5.3.3.2 Class Diagram

Detailed class diagrams for both mobile phone based application and the Web-based application offloading application designed and implemented in the system is illustrated in **Appendix C.**

5.3.3.3 Package Diagram

Figure 34 and figure 35 illustrates package design for mobile application module and web application module respectively.

i. **Mobile phone application packages**

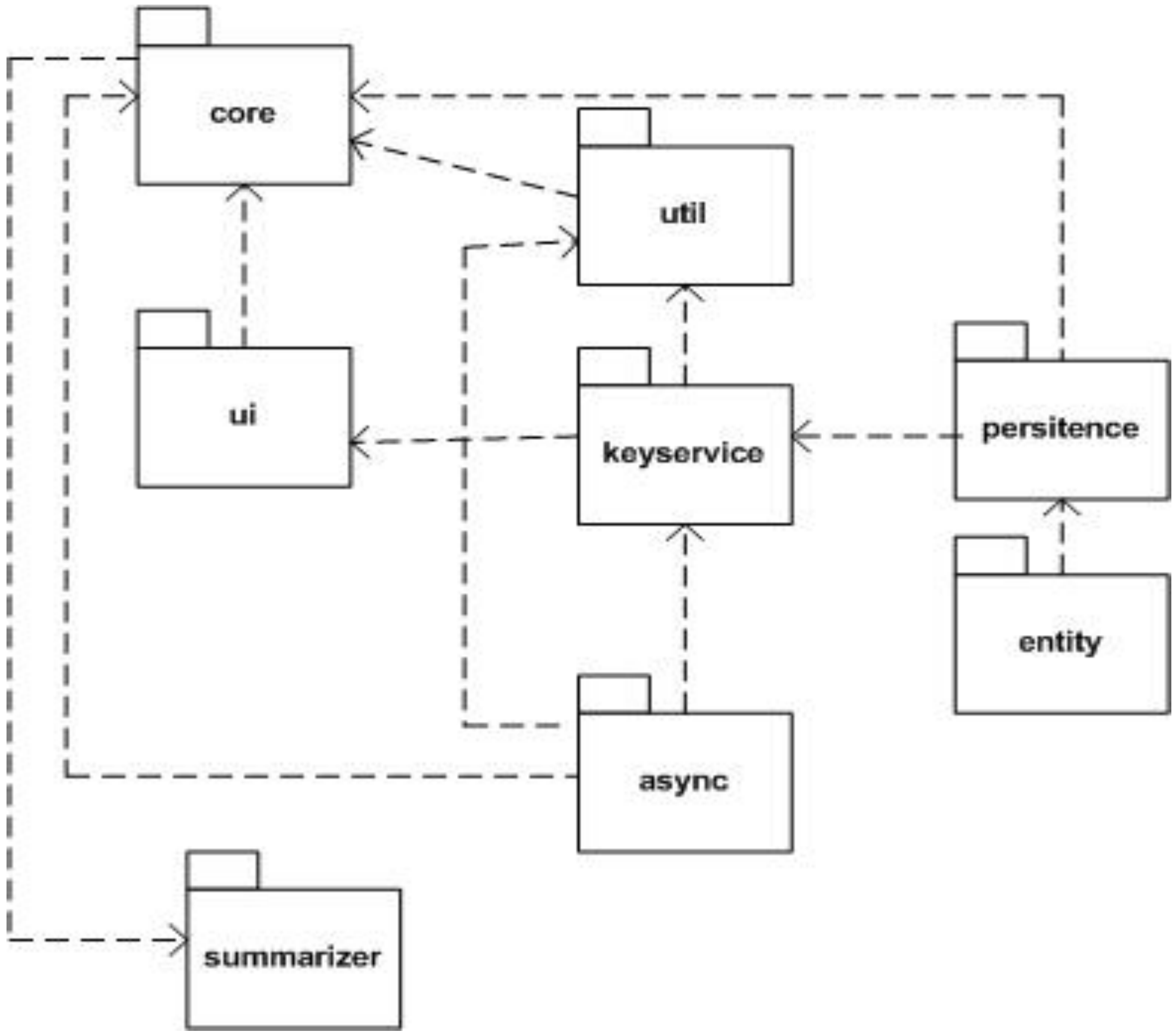


Figure 34 : Mobile phone application packages

Source: Author's compilation

ii. Web- management system packages

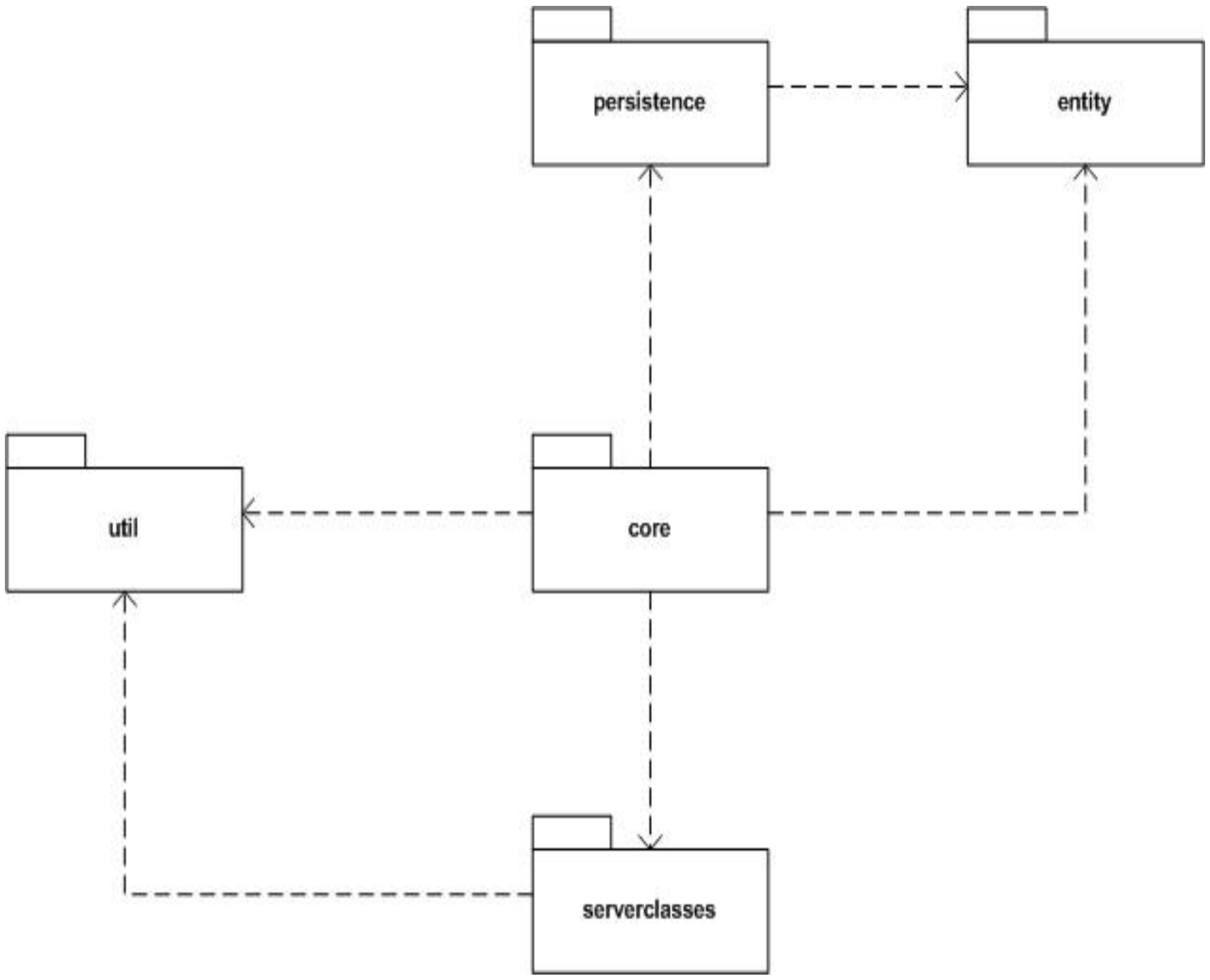


Figure 35 : Web- management system packages

Source : Author's compilation

5.3.3.4 Database Design

We followed Chen (1976), Entity-Relationship (ER) model in modeling our data base design as depicted in the below.

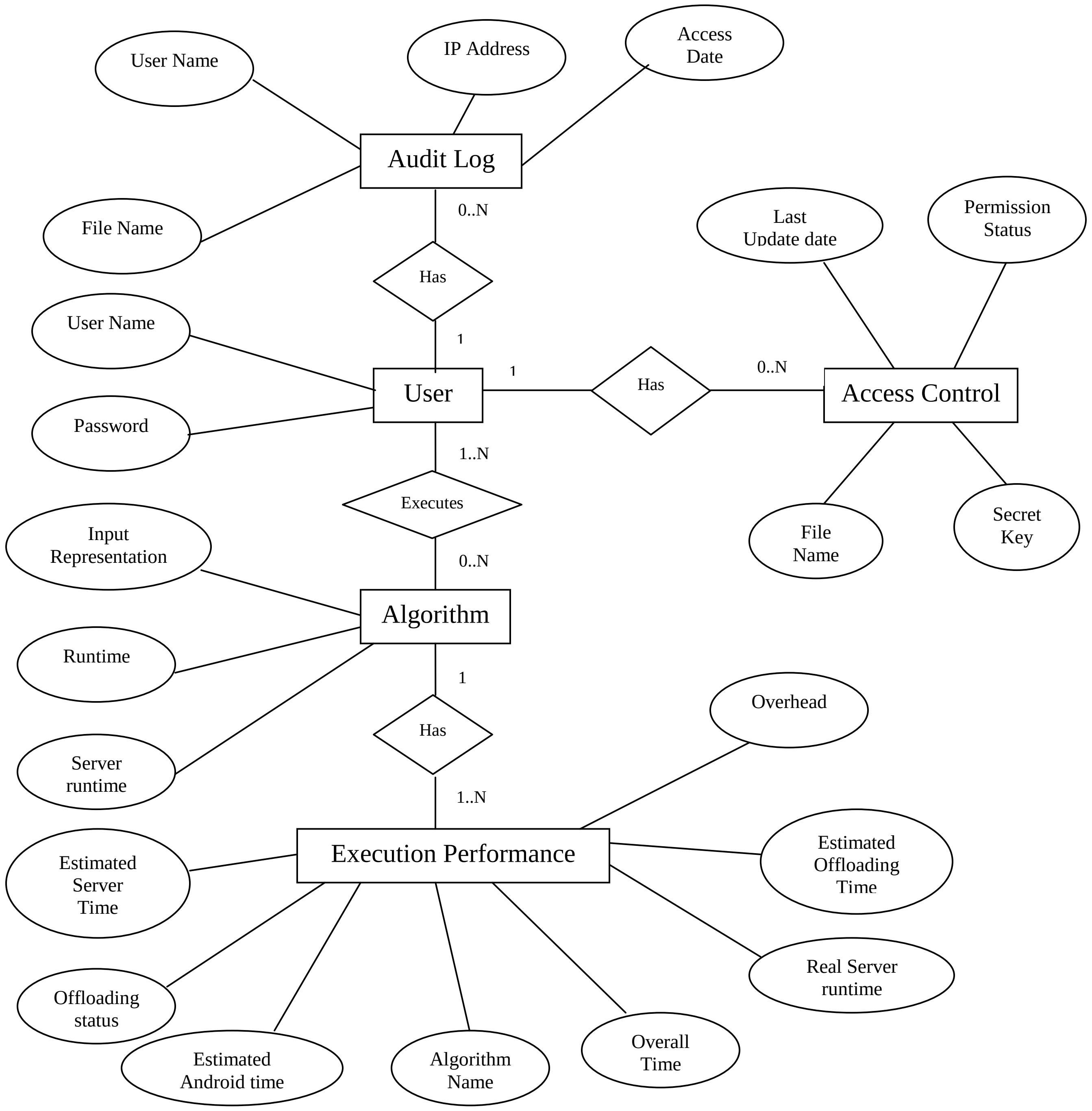


Figure 36 : Entity-relationship model
Source : Author's compilation

5.3.3.5 Integration Design

Linthicum (2000), described four types of Enterprise Application Integration (EAI) namely:- Data-level integration, Application-level integration, Method-level integration and User interface (UI)-level integration. In our project we used Data-level integration, Application-level integration and User interface (UI)-level integration as follows.

i. Application-level Integration

Application-level integration was used to integrate extractive text summarization module with an open Mobile Computation Offloading (MCO) system based on Android platform developed by Griera (2013). This is depicted below.

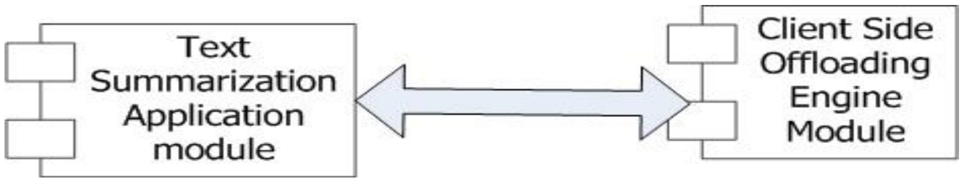

Text Summarization Application module
Client Side Offloading Engine Module

Figure 37 : Application-level integration
Source : Author's compilation

ii. Data-level Integration

Data level integration was used to integrate Client-side Security module running on Android based application and Remote-side security module of running on Apache Tomcat web server. This is depicted below.

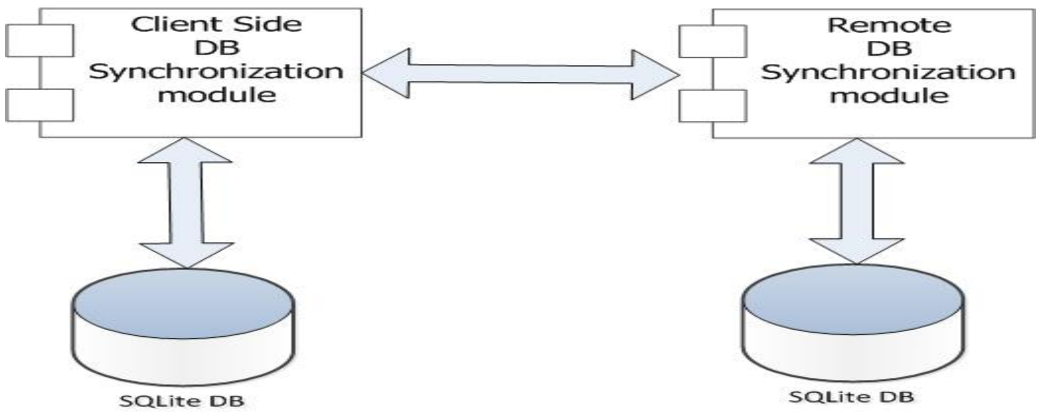

Client Side DB Synchronization module
Remote DB Synchronization module
SQLite DB
SQLite DB

Figure 38 : Data level integration
Source : Author's compilation

iii. **User-Interface (UI) level Integration**

Griera (2013) implemented an open source Mobile Computation Offloading (MCO) system, that has a web management interface; its integration with RACA at the User-Interface (UI) level is illustrated below.

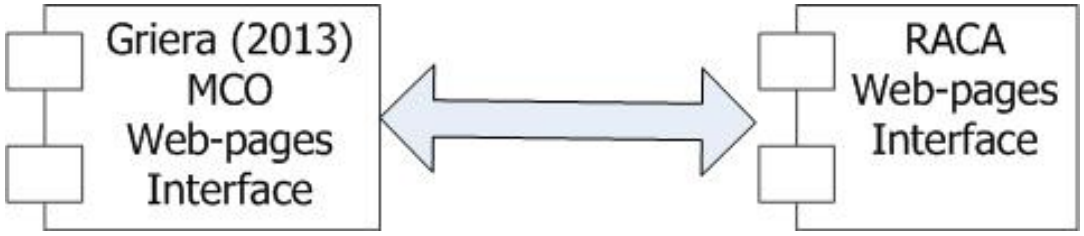


Figure 39 : User-Interface level integration
Source : Author's compilation

5.4. Conclusion

In this chapter we undertook Object Oriented Analysis and Design (OOAD) that involved identification and modeling both static and their dynamic aspects of the **Secure-MCO** prototype system following a use-case approach. However, OOAD carried out is partly an extension of use-cases modeled during requirement specifications in Chapter four; thus a justification of seamless incremental transition of development phases described in Chapter four, Section 4.4.

CHAPTER SIX: PROTOTYPE IMPLEMENTATION AND TESTING

6.1 Introduction

This chapter provides a detailed description on how various design artifacts explored in Chapter five were implemented. It also describes an implementation of RACA framework originally proposed by Geambasu et al. (2011). Thereafter we describe the integration of the same into an open source Mobile Computation Offloading (MCO) system based on Android platform developed by Griera (2013); with **Secure-MCO prototype system** as the resulting product thus effectively fulfilling our set out project specific objective number (ii) and (iii) respectively as outlined in Chapter One, Section 1.6.2.

6.2 Implementation Environment

6.2.1 Implementation Platform

Mobile phone application was implemented on Android platform v4.1.2 (Jelly bean) while the server side was implemented on Java EE6 web platform (based on Servlet 2.5 API).

6.2.2 Programming Languages

The system was implemented using mainly using Java programming language though JSP, JSTL, XML and HTML were also used. Java programming was used to implement complete Mobile phone application components on Android platform. It was also used to implement the Remote application logic and Persistence layer as per the application architectural design in *figure 32*.

The choice of Java was made because it has the following features:-

i. *Simple* - It is easy to write and has a concise, cohesive set of features that makes it easy to learn and use.
ii. *Secure* - It provides a secure means of creating Internet applications and also provides secure way to access web applications.
iii. *Portable* - Java programs can execute in any environment for which there is a Java run-time system (JVM).

iv. *Object-oriented* - Java programming is object-oriented programming language.

v. *Robust* - It encourages error-free programming by being strictly typed and performing run-time checks.

vi. *Multithreaded* - It provides integrated support for multithreaded programming.

vii. *Architecture-neutral* - It is not tied to a specific machine or operating system architecture.

viii. *Interpreted* - Java supports cross-platform code through the use of Java bytecode.

ix. *Distributed* - It was designed with the distributed environment in mind hence can be transmitted, run over internet.

x. *Dynamic* - Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve accesses to objects at run time.

6.2.3 CASE Tools

Among case tools used were:-

- *Android Developer Tools version 22.6.2-1085508* and *Netbeans IDE v. 6.9.1* were used as Integrated Development Environment (IDE).
- *Microsoft Visio* 2007 was used to draw design artifacts of the prototype.
- Microsoft Excel 2007 was used to analyze data and draw graphs and charts
- *SqliteMan v. 1.2.2* was used to browse SQLite database.
- *The ObjectAid UML Explorer for Eclipse* was used for generating class diagrams.
- *Macromedia Fireworks MX 2004* was used for image editing

6.2.4 Application Servers

- Apache Tomcat v. 6.0.20 was used as web server
- SQLite Database was used as a data storage.

6.2.5 Core Development Libraries/API

The following were among key API/libraries were used in the implementation of prototype system:-

i. **Core API**
   - *Java Security API* from *javax.crypto* package to implement Advanced Encryption System (AES) algorithm
   - *Simple XML Serialization API* for Unmarshalling and Marshalling XML and Java Object respectively.
   - *Java Reflection API* for inspecting classes, interfaces, fields and methods at runtime, without knowing the names of the classes, methods etc. at compile time. It was also used to instantiate new objects, invoke methods and get/set field values using reflection.

ii. **Libraries**
   - *http-client-4.0.3.jar* used to provide http communication within Java source code
   - *ormlite-core-4.48.jar* and ormlite-*android-4.48.jar* both were to used to provide ORM support in Remote sever application and Android application respectively.
   - *jfreechart-1.0.9.jar* used to generate bar and line chart of execution performance metrics within the Server-side application module
   - *classifier4j-0.6.jar* used implement extractive text summarization.

6.3 User Interface Implementation

The User interfaces were implemented derived from sketches depicted in section 3.3.3.1. Three fundamental principles taken into consideration while implementing the same were:-

i. **Organization**: provide the user with a clear and consistent conceptual structure
ii. **Economize**: do the most with the least amount of cues
iii. **Communicate**: match the presentation to the capabilities of the user.

Appendix E: (User Manual) contains sample illustration of both mobile application and Web application User interfaces implemented.

6.4 Database Implementation

The database was implemented on SQLite database. The schema of the **raca.db** database is illustrated below. Though it is possible to access **raca.db** from standard SQL language, the prototype system was implemented using **ORMLite ORM**, to facilitate Rapid Application Development (RAD) by transparently addressing type mismatch between our Object-Oriented application entity objects and the SQL statements required to query/update the SQLite database.

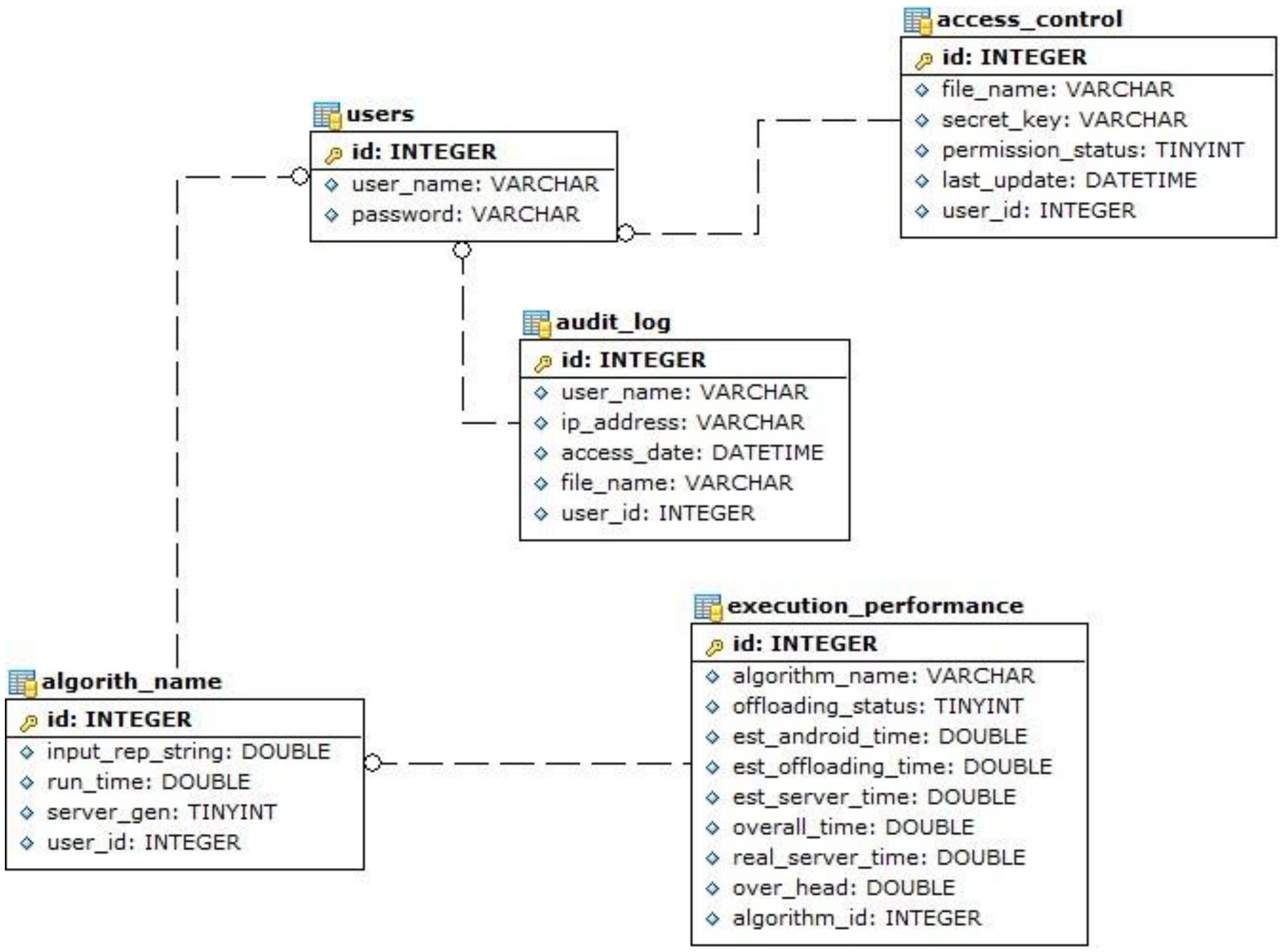


Figure 40 : RACA Database schema

Source : Author's compilation

6.5 Coding

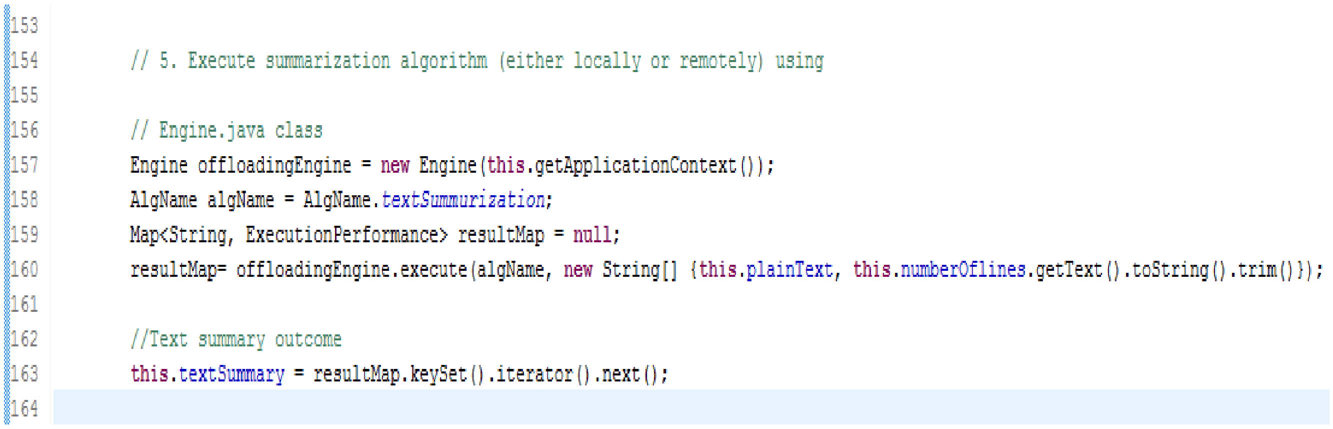
Java programming language was used to implement virtually the whole prototype except user interface of Web-management module that relied on JSP, JSTL, JavaScript and HTML for dynamic web pages implementation.

Appendix A contains sample source code snippets implemented in java. Coding in java was done by strictly following **Java Code Conventions** as described by Chaudhuri and Depradine (2003).

6.6 Integration

i. **Application-level Integration**

Open Mobile Computation Offloading (MCO) system based on Android platform developed by Griera (2013) provides three Java classes (`Engine.java`, `Algorithms.java` and `DataBaseHelper.java`) that a developer must add to his Android application in order to enjoy MCO functionality. We included them and invoked MCO API as shown in the code snippet below from `TextSummarizationActivity.java` class.



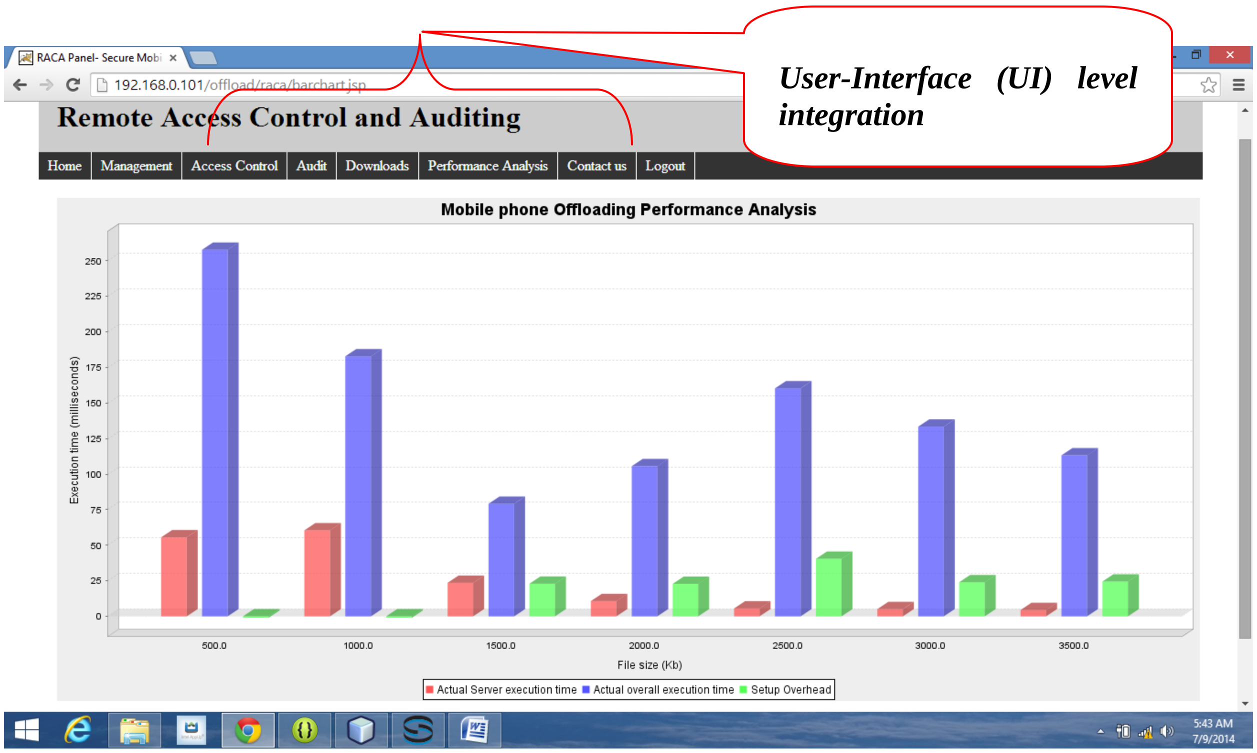
```

        // 5. Execute summarization algorithm (either locally or remotely) using

        // Engine.java class
        Engine offloadingEngine = new Engine(this.getApplicationContext());
        AlgName algName = AlgName.textSummurization;
        Map<String, ExecutionPerformance> resultMap = null;
        resultMap= offloadingEngine.execute(algName, new String[] {this.plainText, this.numberOflines.getText().toString().trim()});

        //Text summary outcome
        this.textSummary = resultMap.keySet().iterator().next();

```

ii. **Data-level Integration**

This was done through synchronization of **raca.db** database located on Android phone based application running Extractive text summarization and remote **raca.db** running Access Control Management application. The code snippet for the integration is illustrated in **Appendix A** under ***DatabaseService.java*** and ***SyncKeyServiceDatabase.java*** for mobile phone application while ***SyncKeyServiceRegistryServlet.java*** was used for Remote database synchronization.

iii. **User-Interface (UI) level Integration**

To integrate RACA Access Control pages into Open Mobile Computation Offloading (MCO) system based on Android platform developed by Griera (2013) ***Access Control, Audit and Performance Analysis*** links were added on the MCO link menu as depicted below.



RACA Panel- Secure Mobi
192.168.0.101/offload/raca/barchart.jsp
Remote Access Control and Auditing
Home
Management
Access Control
Audit
Downloads
Performance Analysis
Contact us
Logout
User-Interface (UI) level integration
Mobile phone Offloading Performance Analysis
Execution time (milliseconds)
File size (Kb)
500.0
1000.0
1500.0
2000.0
2500.0
3000.0
3500.0
0
25
50
75
100
125
150
175
200
225
250
Actual Server execution time
Actual overall execution time
Setup Overhead
5:43 AM
7/9/2014

Figure 41 : Illustration of User-interface level intergration

Source : Author's compilation

6.7 Testing

Three types of test were conducted during the implementation of the prototype system as highlighted below.

i. **Unit testing** – The functionalities implemented within each java classes written were tested to ensure they methods returns the expected results.

ii. **Integration testing** – This test was conducted to on **database synchronization modules** on **remote** and **mobile application** performs its functionality as expected. **Offloading Engine modules** on **remote** and **mobile application, Security module** with **Text Summarization modules** were also subjected to this test.

iii. **System testing** – This test was carried out after the entire prototype system was implemented to ascertain whether it was implemented as per the requirements specified in the Use-Cases. This was done by running the prototype to test for overall functional requirements specification provided in section 4.2 of Chapter Four.

Tabulated below is a summary of test plan conducted in the system.

	Test case description	**Expected result**	**Observed**	**Corrections**
1.	Test file encryption and decryption in the system	A word file (.doc) file is encrypted using AES algorithm and the decryption yield the same content results	Word file successfully encrypted and decrypted as expected.	NIL
2.	Test extractive text summarization in both mobile application and remote offloading engine	The system should generate a text summary as per the required number of lines.	Text summarization successfully performed. However, existence of stopping tokens within a sentence degrades the quality of summary.	Handling of stopping token remains unresolved.
3.	Test offloading engine to ensure computationally intensive tasks are offloaded to surrogate computer.	Computationally intensive tasks are offloaded.	Network monitoring status parameters within **Engine.java class** threw NullPointer Exeception owing initialization failure.	A method *public void updateServerStatus()* was implemented to fix the initialization bug. Hence Computationally intensive tasks were offloaded correctly.
4.	Test the synchronization of SQLite databases on both Mobile phone and surrogate computer	Whenever, there is a network connectivity the two databases should always be sync.	In presence of network connectivity the database service module	NIL

			synchronizes the databases transparently without users intervention.	
5.	Test remote access control functionality i.e users can enable or disable file access remotely.	Users can access remote access control interface and enable or disable file access on the mobile phone	Web application users were able to successfully enable and disable file access on the mobile phone.	NIL
6.	Test to ensure that all access to encrypted file on the mobile phone is logged on the remote auditing server whenever there is a network between mobile phone application and the audit server.	All access to encrypted file is logged on the audit server.	All access to encrypted file logged on audit server.	NIL
7.	Test to ensure that all task execution performance metrics recorded and saved into the surrogate computer.	All task execution performance metrics are recorded on surrogated computer.	Performance metrics recorded as expected.	NIL
8.	Test to ensure that performance metrics analysis is done and line and bar charts of the same generated	The system to perform performance metrics analysis and generate bar chart and line graph from the same	Analysis done, both line graph and bar chart generated as expected.	NIL

Table 2 : Test case plan

Source : Author's compilation

6.8 Conclusion

This Chapter described our implementation of RACA framework as proposed by Geambasu et al. (2011). This was achieved by implementing a file encryption/decryption key management mechanism within the Security module of the prototype. In order to support remote functionality database synchronization modules were implemented.

File encryption was done using Advanced Encryption System (AES) algorithm implemented in Java Security API from `javax.crypto package.` The generated secret key was encoded to **base64** to allow the key to stored a String data type thereafter stored in Access Control object and later used to control encrypted file access.

Remote Audit mechanism was implemented by ensuring that all file access made on encrypted file are recorded on Audit object on the phone application and immediately posted to the Audit Server whenever there was a network connectivity between the two. This was achieved through database synchronization modules.

Extractive text summarization was implemented as a scenario in our prototype system to test the integration of offloading functionality. Though overall text summarization goal was successfully achieved the quality of summary remained degraded due to lack of refined implementation of stopping tokens in ***Classifier4j-0.6.jar*** library that was used to in the prototype system.

Lastly, to facilitate the fulfillment of our specific objective (iv) outlined in Chapter One section 1.6.2, relevant parameters values of extractive text summarization execution time in the prototype system were recorded in the remote database. The database data was then retrieved, statistically analyzed and presented using bar and line graph.

CHAPTER SEVEN: RESULTS AND EVALUATION

7.1 Results

Whereas main outcome of the study was a **Secure-MCO prototype system** presented in Chapter four that fulfilled our specific objective number (ii) and (iii); the study further traced execution time of the prototype system and analyzed the resulting metrics to establish the impact of integrating RACA mechanism in order to fulfill our specific objective (iv). This chapter presents the results obtained and discussion of analysis conducted.

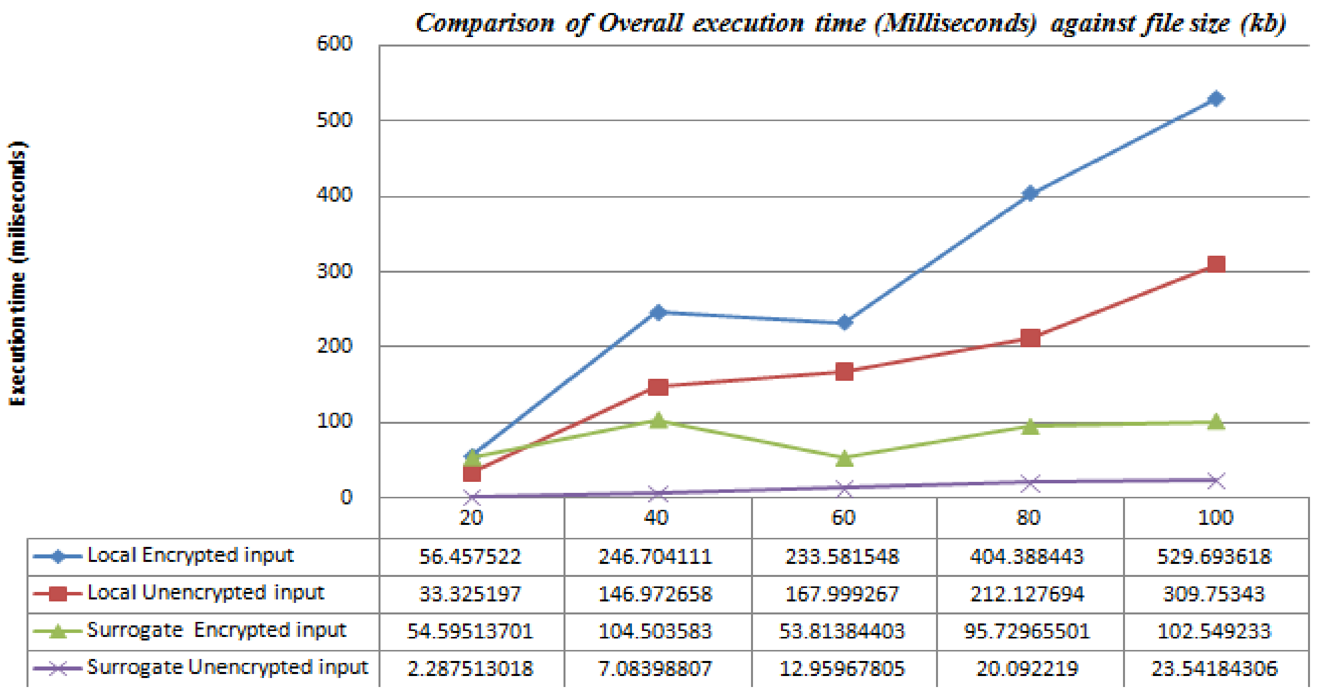
We set out the experiment by using the prototype system to summarize text content of Microsoft word document files whose sizes were 20kb, 40 kb, 60 kb, 80kb and 100 kb. This were carried out using both encrypted files and unencrypted files. The resulting execution time were recorded when extractive text summarizations were carried out in surrogate computer during offloading and when no offloading took place. **Appendix B** provides a tabulation of execution time data extracted from the database while tabulated in table 3 is a summarization of overall execution time taken. Figure 42 to figure 47 illustrate generated bar charts and corresponding line graph.

Overall execution time (Milliseconds)				
Input file size (Kb)	**Local/Mobile phone based execution**		**Offloaded/Surrogate computer execution**	
	Encrypted	**Unencrypted**	**Encrypted**	**Unencrypted**
20	56.4575220048428	33.3251969963312	54.59513701	2.287513018
40	246.704110994935	146.972658008337	104.503583	7.08398807
60	233.581547990441	167.99926699698	53.81384403	12.95967805
80	404.388442993164	212.127693995833	95.72965501	20.092219
100	529.693617984653	309.753429993987	102.549233	23.54184306

Table 3: Overall execution time (milliseconds)

Source: Author's compilation

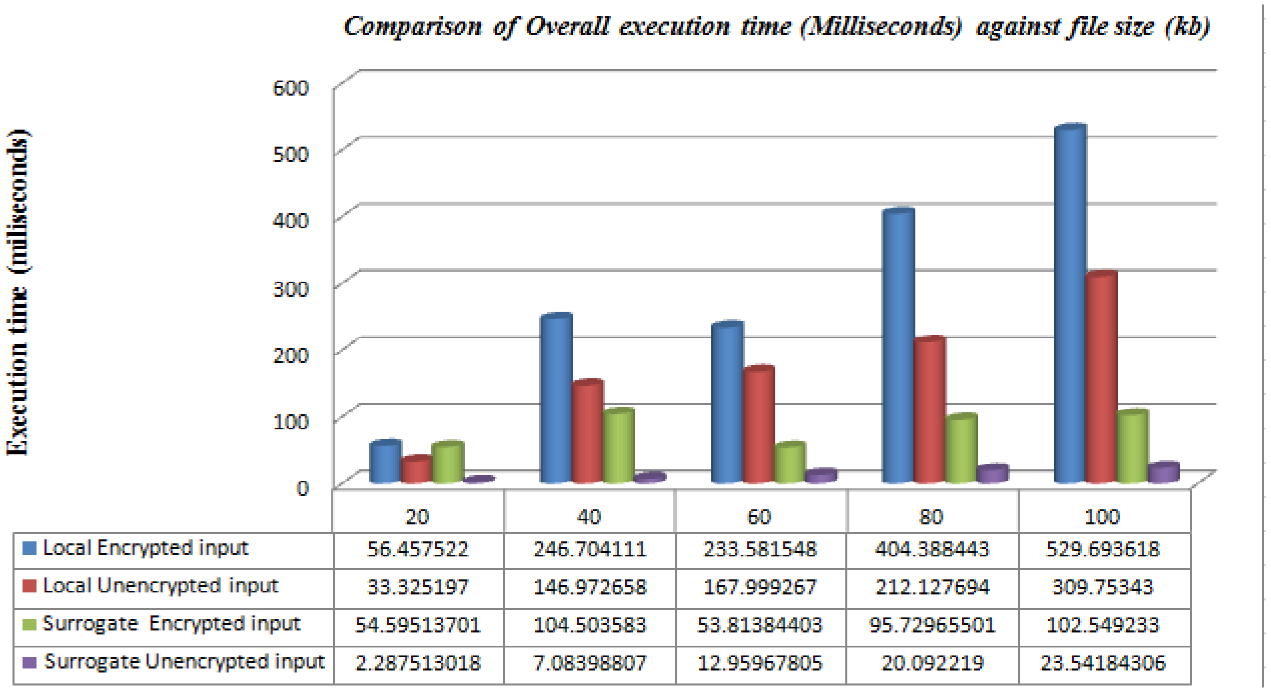
Overall execution time (Milliseconds) against file size is illustrated below:



	20	40	60	80	100
Local Encrypted input	56.457522	246.704111	233.581548	404.388443	529.693618
Local Unencrypted input	33.325197	146.972658	167.999267	212.127694	309.75343
Surrogate Encrypted input	54.59513701	104.503583	53.81384403	95.72965501	102.549233
Surrogate Unencrypted input	2.287513018	7.08398807	12.95967805	20.092219	23.54184306

Figure 42 : Overall execution time line graph

Source: Author's compilation



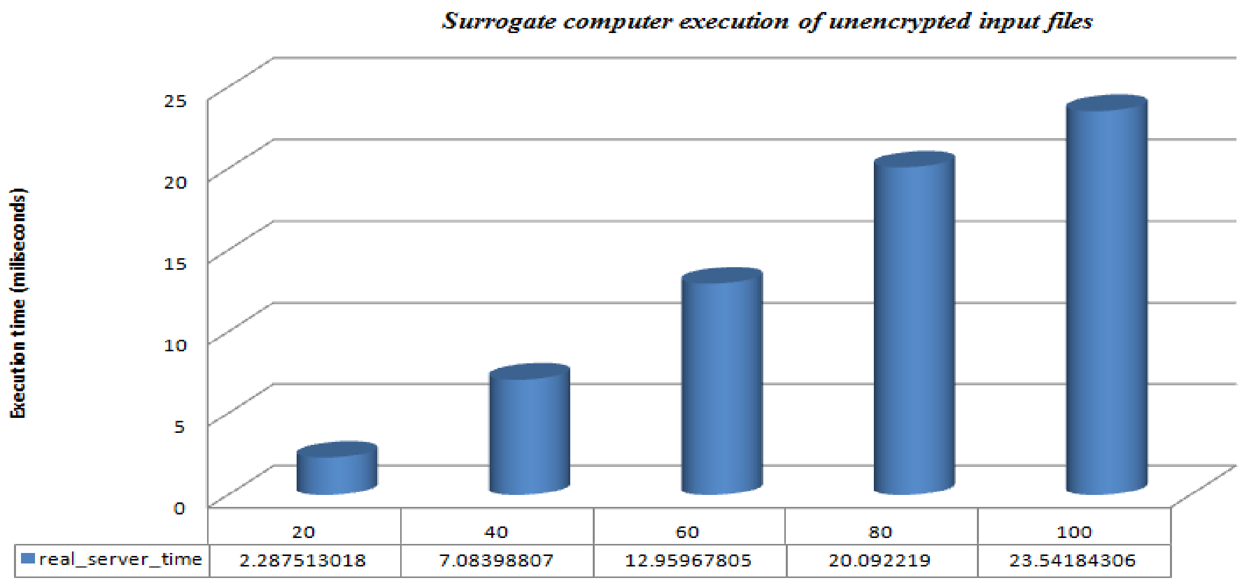
	20	40	60	80	100
🟦 Local Encrypted input	56.457522	246.704111	233.581548	404.388443	529.693618
🟥 Local Unencrypted input	33.325197	146.972658	167.999267	212.127694	309.75343
🟩 Surrogate Encrypted input	54.59513701	104.503583	53.81384403	95.72965501	102.549233
🟪 Surrogate Unencrypted input	2.287513018	7.08398807	12.95967805	20.092219	23.54184306

Figure 43 : Overall execution time bar chart

Source: Author's compilation

Depicted below are bar charts depicting performance metrics result obtained.

i. Surrogate computer execution of unencrypted input files

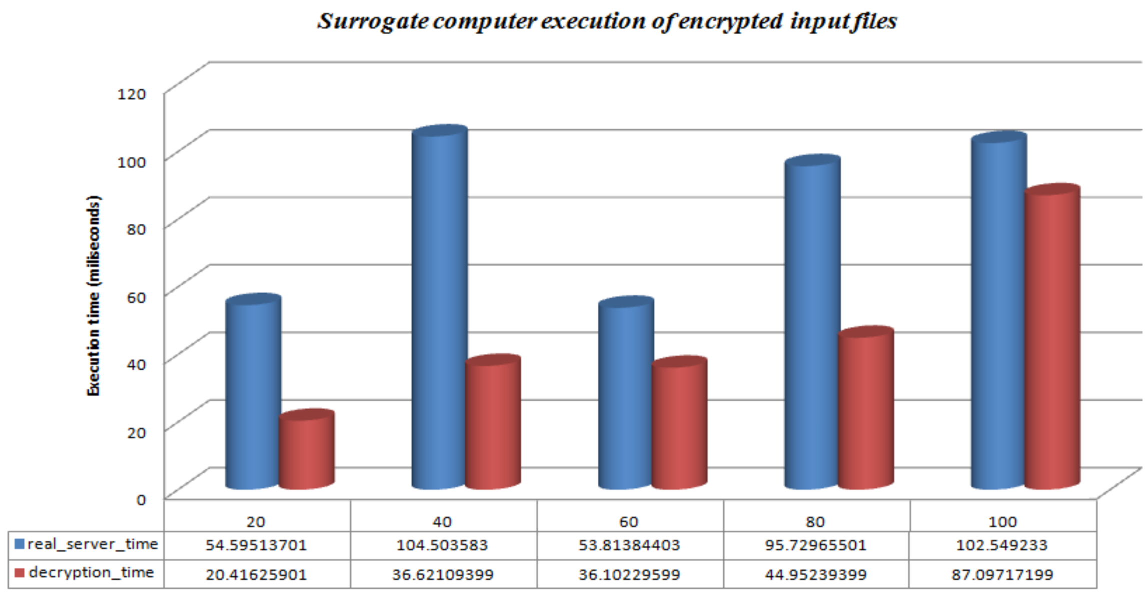


	20	40	60	80	100
🟦 real_server_time	2.287513018	7.08398807	12.95967805	20.092219	23.54184306

Figure 44: Surrogate unencrypted input execution chart

Source: Author's compilation

ii. Surrogate computer execution of encrypted input files

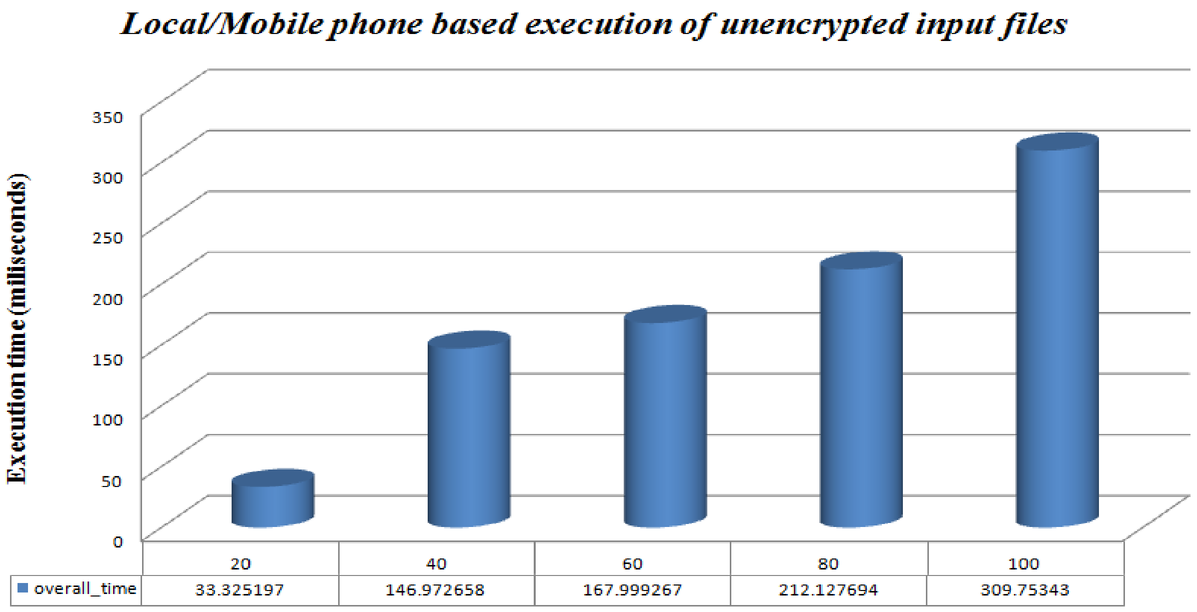


	20	40	60	80	100
🟦 real_server_time	54.59513701	104.503583	53.81384403	95.72965501	102.549233
🟥 decryption_time	20.41625901	36.62109399	36.10229599	44.95239399	87.09717199

Figure 45 : Surrogate encrypted input execution chart

Source: Author's compilation

iii. Local/Mobile phone based execution of unencrypted input files

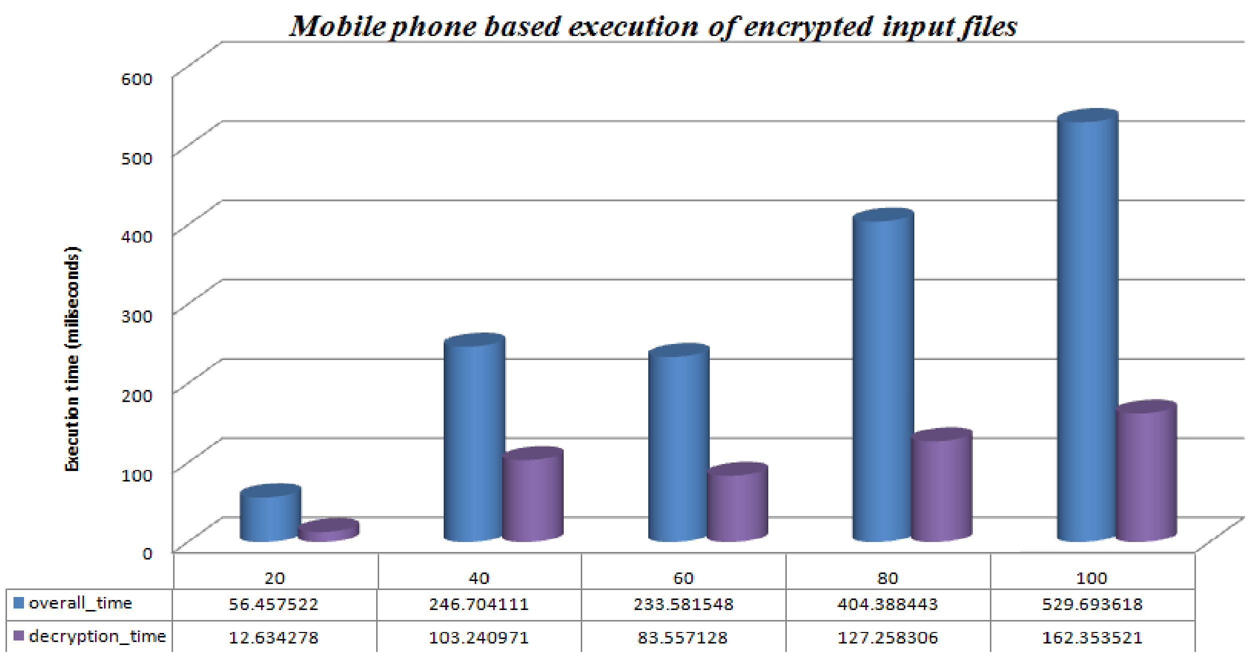


	20	40	60	80	100
overall_time	33.325197	146.972658	167.999267	212.127694	309.75343

Figure 46 : Mobile phone based unencrypted input execution chart

Source: Author's compilation

iv. Local/Mobile phone based execution of encrypted input files



	20	40	60	80	100
overall_time	56.457522	246.704111	233.581548	404.388443	529.693618
decryption_time	12.634278	103.240971	83.557128	127.258306	162.353521

Figure 47 : Mobile phone based encrypted input execution chart

Source: Author's compilation

7.2 Analysis of System results

Section 7.1 provided execution time **descriptive statistics** of **Secure-MCO prototype system** using bar chart; however adequate fulfillment of specific objective (iv) demanded a rigorous demonstration in order to ascertain the impact of RACA integration. Furthermore, it was necessary to ascertain whether or not a causal relationship existed between various variables involved. We applied **inferential statistics** as affirmed by Currell et al. (2009) to rigorously demonstrate whether or not there was a relationship between the input file size and execution time categorized under encrypted and unencrypted input files among other execution time related metrics. We formulated the following questions and the corresponding hypotheses:-

QUESTION I: *Is there a significant difference between the distributions of encrypted and unencrypted input file execution times outcomes for both surrogate and local execution? This question was formulated to test the presence of experimental control variability in the study.*

To answer this question we formulated the following hypothesis and tested it using **chi-square statistics** as per the following equation

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Here,

χ^2 = the chi-square statistic

O_i = the observed frequency

E_i = the expected frequency

i = the number of the cell (cell 1, cell 2, etc.)

Hypotheses:

- **Null (H_0):** No difference in conditional distributions/No real relationship
- **Alternate (H_1):** There is a real relationship.

i. **Local execution**

Input file size (Kb)	Local/Mobile phone based execution	
	Encrypted	**Unencrypted**
20	56.4575220048428	33.3251969963312
40	246.704110994935	146.972658008337
60	233.581547990441	167.99926699698
80	404.388442993164	212.127693995833
100	529.693617984653	309.753429993987

Table 4 : Local execution time (milliseconds)

Source: Author's compilation

Results:

Computed $\mathbf{X^2}$= 5.786698

Degrees of freedom = (Rows – 1)(Columns – 1) = 4

Computed $\mathbf{X^2}$= 5.786698 is less than tabulated $\boldsymbol{\chi 2}$ at 0.05, 0.01 and even 0.001 probability columns presented as 9.49, 13.28 and 18.47 respectively. *Hence we accepted* ***Null (H_o) hypothesis*** *and concluded that there were no errors/ experimental control variability in local execution scenario that interfered with the experiment outcome.*

ii. **Offloaded/Surrogate computer execution**

Input file size (Kb)	Offloaded/Surrogate computer execution	
	Encrypted	**Unencrypted**
20	54.59513701	2.287513018
40	104.503583	7.08398807
60	53.81384403	12.95967805
80	95.72965501	20.092219
100	102.549233	23.54184306

Table 5 : Surrogate execution time (milliseconds)

Source: Author's compilation

Results

Computed $\mathbf{X^2}$= 14.74009

Degrees of freedom = (Rows – 1)(Columns – 1) = 4

Computed $\mathbf{X^2}$= 14.74009 is greater than tabulated $\chi 2$ at 0.05, 0.01 but not at 0.0005 probability columns presented as 9.49 13.28 and 18. 86 respectively. *Hence we rejected* ***Null (H_o) hypothesis*** *and concluded that there was a 99% probability there were experimental control variability during surrogate execution scenario consequently impacting on experiment outcome.* However, this conclusion was made cognizance of the fact that **text summarization algorithm exhibit intractable execution time owing to the fact that it is an heuristic algorithm hence it doesn't necessarily execute following a particular distribution function.**

QUESTION II*: Is there a significant relationship between file input size and text summarization performance?*

To answer this question we formulated the following hypothesis, computed **Pearson's *r* correlation coefficient**, calculated the **t-ratio** and determined the hypothesis significance from t-tables. Moreover we computed **R-Squared** values to determine portion of variance in the execution time (dependant variable) that can be attributed to file input size (independent variable) as suggested by Currell et al. (2009).

Hypotheses:

- **Alternate (H_1):** The greater the file input size, the greater the text summarization execution time
- **Null (H_0):** There is no relationship between file input size and text summarization execution time.

Table five below tabulates the execution time of various encrypted input files obtained during the experiment.

Input File size (Kb) (X)	Execution time (Milliseconds) (y)
20	56.457522
20	54.59513701
40	246.704111

40	104.503583
40	7.08398807
60	167.999267
60	53.81384403
80	404.388443
80	95.72965501
100	20.092219
100	102.549233

Table 6 : Relationship between encrypted input file size and execution time

Source: Author's compilation

We computed

i. **Pearson's (r) correlation coefficient** using the following formulae.

$$r = \frac{\sum xy - N\bar{x}\bar{y}}{\sqrt{(\sum x^2 - N\bar{x}^2)(\sum y^2 - N\bar{y}^2)}}$$

ii. **t- ratio** using the following equation

$$t = \frac{r\sqrt{N-2}}{\sqrt{1-r^2}}$$

iii. R-Squared = r^2

Results

Computed r= 0.110211123 and ratio t= 0.470452094

Degrees of freedom (*df*) = N-2 = 10-2 = 8

Since our computed value of t is less than tabulated t-table values at 0.1 probability. We reject Alternate (H_1) and assert that the probability that a significant relationship exist between input file size and the execution time is less that 0.1%. **However, this can be attributed to the intractable nature of heuristic algorithm in text summarization algorithm used as far as its execution time is concerned.**

Computing R-Squared gives 0.012146491632921129, ***This was interpreted by stating that Input file size explains 1.21465% of the variance in execution time.***

CHAPTER EIGHT: DISCUSSIONS AND CONCLUSION

8.1 Achievement Objectives

This project achieved the overall objective it was set out to carry; this was to enhance security of mobile phone based cyber foraging system by implementing a mechanism that enforce data integrity and confidentiality. The study achieved its stated specific objectives as highlighted below.

OBJECTIVE I: To examine existing mobile phone based cyber foraging systems with a focus on data integrity and confidentiality security challenges.

This was achieved through detailed review of literatures on cyber foraging systems. However, we focused on integrity and confidentiality security challenges exhibited in the existing systems. Literature review revealed that related works in cyber foraging have had a focus on how to design and implement a functional cyber foraging system with insignificant effort made to address security challenges arising in such systems; notwithstanding obvious vulnerability such system are expose to owing to migration of data to remote surrogate computers. Furthermore, through literature review it become evident that the application of RACA framework to address such problem was non-existent thus establishing the foundation for relevance of the study.

OBJECTIVE II: To design and implement data integrity and confidentiality enforcing mechanism in mobile phone based cyber foraging system using Remote Access Control and Auditing model.

The project successfully implemented data integrity and confidentiality enforcing mechanism based on RACA framework originally developed by Geambasu et al. (2011). This was achieved by combining encryption based on 128-bit Advanced Encryption System (AES) algorithm, remote key storage and an audit server to provide two important properties. First is a file auditing that offers explicit evidence on whether or not a file access was made. Secondly, it allows users to disable future file access on the device once the device is lost by allowing a configuration on the audit server to refuse to return a particular file key.

OBJECTIVE III: To integrate the implemented mechanism in objective (ii) above into an open source mobile phone based cyber foraging system prototype.

RACA implementation was successfully integrated into an open source Mobile Computation Offloading (MCO) system based on Android platform developed by Griera (2013). This was done at Application, Data and User-interface levels as described in Chapter six Section 6.6 of this document.

OBJECTIVE IV: To evaluate offloading performance overhead costs attributed to the integration of RACA implementation in mobile phone based cyber foraging system prototype.

Using extractive text summarization as a use case scenario in the prototype system, we analyzed and evaluated execution time performance overhead arising from integration of RACA implementation into MCO. The result indicated that despite the intractable execution time of extractive text summarization algorithm owing to it heuristic properties, the execution time overhead arising from the integration was minimal hence integration of RACA into MCO was deemed feasible.

Lastly, in order to provide robust secure mechanism; the prototype system implemented a basic authentication while the communication between Mobile Phone application and Surrogate computer was implemented based on Hypertext Transfer Protocol Secure (HTTPS). HTTPS provides layering of Hypertext Transfer Protocol (HTTP) on top of the SSL/TLS protocol, thus adding the security capabilities of SSL/TLS to standard HTTP communications. The main motivation behind the use of HTTPS implementation was to prevent chances of wiretapping and man-in-the-middle attacks whenever the system offload a sensitive confidential data to surrogate computer. This feature is not covered in RACA framework.

Research questions were answered as follows:-

iii. *What are security challenges facing mobile phone based cyber foraging system?*

It was established that data integrity and confidentially remains unaddressed hence among major security challenges facing mobile phone based cyber foraging system.

iv. *Does RACA security model improve data integrity and confidentiality in mobile phone based cyber foraging system?*

RACA framework sufficiently augment traditional encryption based file system by combining encryption, remote key storage and an audit server to provide explicit evidence on whether or not a file access was made. It also allows users to disable future file access on mobile phone device once the device is lost by allowing a configuration on the audit server to refuse to return a particular file key. However, RACA need to be complemented with Secure transport between Mobile Phone device and Surrogate Computer based on SSL/TLS in order to prevent chances of wiretapping and man-in-the-middle attacks. This was achieved in the prototype system through the use of HTTPS.

v. *Can RACA mechanism be integrated into mobile phone based cyber foraging system?*

Yes. This can be effectively achieved at both data, application and User interface level integration.

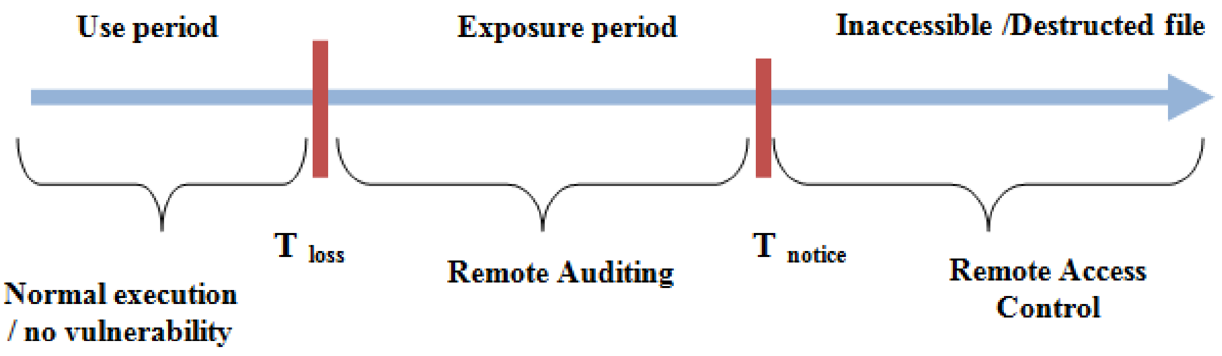
vi. *Does the integration of RACA mechanism significantly undermine task offloading performance in mobile phone based cyber foraging system?*

Execution time overhead penalty attributed to RACA mechanism integration is minimal hence the proposed solution is feasible.

8.2 Limitation of the study

Two major limitations were identified in the study.

i. ***Enforcing data integrity and confidentiality after loss/theft of mobile phone device.***



Key: **T loss** - Time when mobile phone device user losses control of the device

T notice - Time when user realizes he/she lost the device

Figure 48 : Timeline of theft of mobile phone device

Source: Author's compilation

Illustrated above in figure 48 is the timeline of theft/loss of mobile phone device deployed in the Cyber foraging system. The implemented solution fall short of enforcing 100% data integrity and confidentiality during loss/theft of a mobile phone device between **T loss** and **T notice** interval. This is because during this interval user would not have disable file access on the mobile device through remote access control functionality thus the file may remain accessible to a user who has basic authentication login access on the mobile phone application. Furthermore, file access on the mobile phone device can only be accessible when the device is accessible within WLAN.

No viable solution was found on how to mitigate on this limitation without contradicting the goal of cyber foraging that demands that the mobile phone application ought to be capable of executing offloadable tasks on its own whenever, surrogate computer remains inaccessible within WLAN as affirmed Balan et al. (2002). Hence this challenge remains open/unaddressed.

ii. Prevention of wire-tapping and man-in-the middle of attack

RACA framework addresses data integrity and confidentiality challenges at file system level. However, the framework does not provides mechanism of addressing data confidentiality and integrity vulnerability exposed to data during transmission. This becomes an hindrance/limitation whenever a distributed system such as cyber foraging is involved. This is because data transmission between mobile phone device and surrogate computer is susceptible to interception with a potential of compromise thereafter. This is particularly so through wire-tapping and man-in-the-middle attacks. We mitigated on this limitation by relying on HTTPS for communication between Mobile phone device and surrogate computer based on self-signed SSL/TLS certificates.

8.3 Recommendations

The ever-increasing amount of valuable digital data needs to be protected, since its irrevocable loss is unacceptable. In recent years, cyber foraging systems popularity has increased dramatically. However, individuals in possession of confidential data hesitate to entrust their data to cyber foraging systems since they fear that they will lose control over it. This is exacerbated by rudimental enforcement of computer security attributes namely confidentiality, integrity and availability (also called CIA) as recommended by Avizienis et al. (2004) in these systems.

In this study we have enforced confidentiality and integrity attributes using RACA framework. However, the framework exhibits several inadequacies such as lack of transport security during transmission and lack of a discretionary access control among others. Consequently, in order to improve RACA framework to sufficiently support CIA attributes we recommend the following:-

i. **System Access Controls**.

The framework should not only support strong *identification* and *authentication* mechanism but also encourage (and sometimes force) authorized users to be security-conscious–for example, by changing their passwords on a regular basis.

ii. **Data Access Controls.**

Whereas the framework supports Remote Access Control and Auditing features, there is need for enhancement so that it supports fine-grained discretionary access control by determining whether other people can read or change file data i.e. `-rwxrwxrwx` . The system might also support mandatory access controls; whereby the system determines access rules based on the security levels of the system users.

iii. **Encryption**

The framework supports encryption of stored data and not across-the-wire transmission. However, this can easily be mitigated upon by using secure transmission protocol based on self-signed SSL/TLS certificates.

8.4 Conclusion

The study implemented data integrity and confidentiality mechanism in Mobile phone based cyber foraging system based on RACA framework. The mechanism provides users with evidence on weather sensitive data/file was accessed or not. If file was accessed, it provide users with an audit log indicating the same. It also allows users to disable file access on lost devices. These goals were achieved through integration of encryption, remote key management and auditing. An analysis of our experimental results showed that this approach works properly and is feasible. Thus our contribution within the mobile phone based cyber foraging is twofold namely:-

i. Application of RACA framework in mobile phone based cyber foraging to enhance data integrity and confidentiality.
ii. Evaluation of execution time overhead cost attributed to integration of RACA mechanism in mobile phone based cyber foraging.

REFERENCES

A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr (2004). "Basic concepts and taxonomy of dependable and secure computing". *Dependable and Secure Computing, IEEE Transactions,* Vol.1, Iss.1, pages 11–33, 2004.

A. Muthitacharoen, B. Chen, and D. Mazières (2001). "A Low-bandwidth Network File System*". Proc. 18th Symposium on Operating Systems Principles*. Banff, Canada.

B. Chun, S. Ihm, P. Maniatis, M. Naik and A. Patti (2011). "CloneCloud: elastic execution between mobile device and cloud". *In Proceedings of the sixth conference on Computer systems (EuroSys '11).* ACM, New York, NY, USA, 301-314.

C. Floyd (1984)."A systematic look at prototyping". *Approaches to prototyping*, pages 1–18,
Casey E., & Stellatos G. J. (2008). "The impact of full disk encryption on digital forensics" . *ACM SIGOPS Operating Systems Review*, *42*(3), 93-98.

Chen, P. P. S. (1976). "The entity-relationship model- toward a unified view of data". *ACM Transactions on Database Systems (TODS)*, *1*(1), page 9-36.

D. Mazieres, M. Kaminsky, M. F. Kaashoek, and E. Witchel (1999). "Separating key management from file system security". *In Proceedings of the ACM Symposium on Operating Systems Principles (SOSP),*

David S. Linthicum (2000), *Enterprise Application Integration*, Addison-Wesley Professional
Depradine, C., & Chaudhuri, P. (2003). "P³: a code and design conventions preprocessor for Java". *Software: Practice and Experience*, Vol. *33*(1), page 61-76.

E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra and P. Bahl (2010). "MAUI: making smartphones last longer with code offload", *In Proceedings of the 8th international conference on Mobile systems,applications, and services (MobiSys '10).* ACM, New York, NY, USA, 49-62.

G. Booch (1994), *Using the Booch Method: A Rational Approach*, Benjamin Cummings Publishing Company.

G. Currell, A. Dowman and W. Blackwell (2009), *Essential Mathematics and Statistics for Science,* 2nd Edition , John Wiley & Sons, Ltd., Publication

H. Y. Chen, Y. H. Lin, and C. M. Cheng (2012). “COCA: Computation offload to clouds using AOP”. *In 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012),* May 2012, 466-473.

I. Alexander and Beus-Dukic Ljerka (2009), *Discovering Requirements: How to Specify Products and Services*, John Wiley Publisher.

I. Giurgiu, O. Riva, D. Juric, I. Krivulev and G. Alonso (2009). “Calling the cloud: enabling mobile phones as interfaces to cloud applications”. *In Proceedings of the ACM/IFIP/USENIX 10th international conference on Middleware (Middleware'09),* Jean M. Bacon and Brian F. Cooper (Eds.). Springer-Verlag, Berlin, Heidelberg, 83-102.

I. Jacobson, M. Christerson, P. Jonsson and G. Overgaad (1994), *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley Publisher.

J. Flinn, D. Narayanan and M. Satyanarayanan (2001). “Self-Tuned Remote Execution for Pervasive Computing”, *In Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HOTOS '01)*. IEEE Computer Society, Washington, DC, USA, 61-63.

J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West (1998). *Scale and performance in a distributed file system. ACM Transactions on Computer Systems (TOCS),*

J. J. Barton, S. Zhai, and S. B. Cousins (2006). “Mobile phones will become the primary personal computing devices”. In *WMCSA ’06: Proceedings of the Seventh IEEE Workshop on Mobile Computing Systems & Applications,* pages 3–9, Washington, DC, USA, 2006. IEEE Computer Society.

J. J. Kistler and M. Satyanarayanan (1991) “Disconnected operation in the Coda file system”. *In Proceedings of the ACM Symposium on Operating Systems Principles (SOSP),*

J. Rumbaugh (1996), *OMT Insights: Perspectives on Modeling from the Journal of Object-Oriented Programming,* SIGS Books & Multimedia publisher.

K. Kumar, J. Liu, Y. Lu and B. Bhargava (2012) “A Survey of Computation Offloading for Mobile Systems”, *Springer Science Business Media, LLC*

Karl E. Wiegers (1997), "Listening to the Customer's Voice", *Software Development Magazine,* March 1997, available at: http://www.processimpact.com/articles/usecase.html accessed on Wednesday 18th June 2014 at 1845 hours.

M. Blaze (1993). “A cryptographic file system for UNIX”. *In Proceedings of the ACM Conference on Computer and Communications Security* (CCS).

M. Griera (2013). “Improving the reliability of an offloading engine for Android mobile devices and testing its performance with interactive applications” MSc. Thesis, Department of Mathematics and Computer Science, Institute of Computer Science, Freie Universitat.

M. Satyanarayanan (2001). “Pervasive computing: vision and challenges. *Personal Communications IEEE”, Vol.* 8(4):10–17.

M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies (2009). “The case for VM based cloudlets in mobile computing”. *IEEE Pervasive Computing*, 8:14–23,
M.S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao and X. Chen (2012). “COMET: Code Offload by Migrating Execution Transparently”. *In Proceedings of OSDI.*

Mads D. K, Bouvin, N.O (2010), “Scavenger: Transparent Development of Efficient Cyber Foraging Applications”. In *Journal of Pervasive and Mobile Computing (PMC)*, Elsevier.

Mads D. K. (2010). “Empowering Mobile Devices Through Cyber Foraging: The Development of Scavenger, an Open, Mobile Cyber Foraging System” Ph.D dissertation, Faculty of Science, Aarhus University.

N. Rahim and K. Saravanan (2013). “Secured Image Sharing and Deletion in the Cloud Storage Using Access Policies”, *International Journal on Computer Science and Engineering (IJCSE)*

P. Coard and E. Yourdon (1991), *Object-Oriented Analysis*, (2nd ed.) Yourdon Press, Upper Saddle River, NJ USA.

R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang (2002) “The case for cyber foraging” . In *EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop: beyond the PC*, pages 87–92, New York,NY, USA, 2002. ACM Press.

R. Geambasu, J. P. John, S. D. Gribble, T. Kohno, and H. M. Levy (2011). “Keypad: An auditing file system for theft-prone devices”. *Proceedings of the ACM European Conference on Computer Systems (Eurosys)*

R. K. Balan, M. Satyanarayanan, S.Y Park and T. Okoshi (2003). “Tactics-based remote execution for mobile computing”. *In Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys '03).* ACM, New York, NY, USA, 273-286.

R. Kemp, N. Palmer, T. Kielmann, and H. Bal (2010). “Cuckoo: a Computation Offloading Framework for Smartphones”. *In MobiCASE '10: Proceedings of The Second International Conference on Mobile Computing, Applications, and Services*, pp. 62-81, 2010.

R. Wirfs-Brock,, and B. Wilkerson, (1989). "Object-oriented design: a responsibility-driven approach. In *ACM SIGPLAN Notices* Vol. 24, No. 10, pp. 71-75 . ACM.

S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang (2012). “Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading”. *In IEEE Infocom*. SSN 1536-1268.

Stephen R. Schach (2011), *Object-Oriented and Classical Software Engineering*, 11th Edition, McGraw Hill Companies Inc. ISBN 978-0-07-337618-9.

T. Paradiso, J.A.; Starner (2005). “Energy scavenging for mobile and wireless electronics” .*Pervasive Computing, IEEE*, 4(1):18–27.

T. Ristenpart, G. Maganis, A. Krishnamurthy, and T. Kohno (2008). “Privacy-preserving location tracking of lost or stolen devices: Cryptographic techniques and replacing trusted third parties with DHTs”. *In Proceedings of the USENIX Security Symposium.*

T. Verbelen (2013). “Adaptive Offloading and Configuration of Resource Intensive Mobile Applications” PhD dissertation, Faculty of computer Science, University Kent.

APPENDIX A: SAMPLE SOURCE CODE SNIPPETS

```
/**
*===============================================================
*             KeyService.java
*===============================================================
* This class implement the core security function of RACA framework
* It provides methods for encryption and decryption of file
* Using Advanced Encryption System (AES)on 128 bit.
* The secret key is generated  encoded/decoded to Base64
*
* @authors Alfayo oyugi Adede  Email: alfayaoyugi@googlemail.com
*
* @version $Date: 2014-05-14 13:29:34 +0200  $ $Revision: 1 $
*/
package adede.msc.project.keyservice;

import java.io.File;
import java.security.SecureRandom;
import java.util.HashMap;
import java.util.Map;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.spec.SecretKeySpec;
import adede.msc.project.util.FileUtil;
import android.annotation.SuppressLint;
import android.util.Base64;

public class KeyService {
        public KeyService() {
                super();
        }
        /**
         * @param plainTextFile: unencrypt input file
         * @return: Map <Absolute_Encrypted, SecretKeySpec> used to encrypt
         * @throws Exception
         */
        public Map<String, SecretKeySpec> encrypt(File plainTextFile)
                        throws Exception {
                // Map <Absolute_Encrypted, SecretKeySpec> used to encrypt
                Map<String, SecretKeySpec> encryptResult = new HashMap<String, SecretKeySpec>();
                String fileName = plainTextFile.getName().substring(0, plainTextFile.getName().lastIndexOf("."))+ ".doc";
                File encryptedFile = new File(plainTextFile.getParent()+ File.separator + "Encrypted_" + fileName);
                //Encrypt file, encode file result to base 64 and write it to file.
                SecretKeySpec secretKeySpec = getSecretKeySpec(plainTextFile.getAbsolutePath());
```

```
			FileUtil.writeFile(encrypt(secretKeySpec, FileUtil.readFileDoc(plainTextFile.getAbsolutePath())),
encryptedFile);
			encryptResult.put(encryptedFile.getAbsolutePath(), secretKeySpec);
			return encryptResult;
		}
		/**
		 * @param seed : This used the generate Secure random number
		 * @return  : Is a 128 bit SecretKeySpec based on AES
		 */
		@SuppressLint("TrulyRandom")
		public SecretKeySpec getSecretKeySpec(String seed) {
			// Set up secret key spec for 128-bit AES encryption and decryption
			SecretKeySpec sks = null;
			try {
				SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
				sr.setSeed(seed.getBytes());
				KeyGenerator kg = KeyGenerator.getInstance("AES");
				kg.init(128, sr);
				sks = new SecretKeySpec((kg.generateKey()).getEncoded(), "AES");
			} catch (Exception e) {
				e.printStackTrace();
			}
			return sks;
		}
		/**
		 * This method act as helper method during encryption
		 * @param sks
		 * @param input
		 * @return
		 */
		public String encrypt(SecretKeySpec sks, String input) {
			// Encode the original data with AES
			byte[] encodedBytes = null;
			try {
				Cipher c = Cipher.getInstance("AES");
				c.init(Cipher.ENCRYPT_MODE, sks);
				encodedBytes = c.doFinal(input.getBytes());
			} catch (Exception e) {
				e.printStackTrace();
			}
			return new String(Base64.encodeToString(encodedBytes, Base64.DEFAULT));

		}

	/**
		 * This method act as helper method during decryption
	 * @param inkey
	 * @param encryptedFile
	 * @return
	 * @throws Exception
	 */
	public String decrypt(String inkey, File encryptedFile) throws Exception
	{
		String plainText = null;
		//Construct key
		SecretKeySpec key = new SecretKeySpec(Base64.decode(inkey, Base64.DEFAULT), "AES");
		plainText = decrypt(key, FileUtil.readFile(encryptedFile));
		return plainText;
	}
	/**
	 * Used to decrypt encryted input string that has been encoded to base64 and returns plain text
	 * @param sks : Secret key specification
	 * @param inputBase64Encoded
	 * @return : decrypted plain text
	 */
	public String decrypt(SecretKeySpec sks, String inputBase64Encoded) {
		// decode inputBase64Encoded
		byte[] encodedBytes = Base64.decode(inputBase64Encoded, Base64.DEFAULT);
		// Decode the encoded data with AES
		byte[] decodedBytes = null;
		try {
			Cipher c = Cipher.getInstance("AES");
			c.init(Cipher.DECRYPT_MODE, sks);
			decodedBytes = c.doFinal(encodedBytes);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return new String(decodedBytes);
	}
}
```

```
/**
*==================================================================
*                 Engine.java
*==================================================================
* This is core of the offloading engine.
* It contains all functionalities needed to obtain the values relevant parameters
* that affect offloading process, keep them updated, load and store the persistent ones,
* and decide in agiven situation whether it is worth or not to start an offloading process.
* This file shouldn't be modified by the programmer who wants to use the engine in his application
*
* @authors Alfayo oyugi Adede  Email: alfayaoyugi@googlemail.com
*
* @version $Date: 2014-05-14 13:29:34 +0200  $ $Revision: 1 $
*/
package adede.msc.project.core;

import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.EnumMap;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.apache.http.NameValuePair;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.BasicResponseHandler;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.w3c.dom.CharacterData;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import adede.msc.project.core.Algorithms.AlgName;
import adede.msc.project.entity.ExecutionPerformance;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;
import android.telephony.TelephonyManager;
import android.util.Log;

public class Engine {
		private static final String TAG = "Engine";
		//Constants
		public static final String BASE_SERVER_URL = "http://192.168.0.101/offload/";
		private static final String SERVER_URL = Engine.BASE_SERVER_URL+"run"; //Right now we assume we only have 1 
server (in the future we could search the nearest server)
		private static final String PREFS_NAME = "OffloadingEnginePrefs";
		public static final double SERVER_INST_MS = 6666666; //Calculated through practical values (javap -c), although we 
know the server has 4 cores of 2.5GHz
		public static final double MIN_RELEVANT_TIME = 15.0;
		private final Context appContext; //The context of the main Activity of the Android application using this engine
		private int pingCounter; //Needed to calculate the ping
		private double timePingStart; //Needed to calculate the ping
		private double[] pingsArray; //Needed to calculate the ping
		private double ping; //Represents the time to query and get an answer from the server (actually not done with a real 
ping command over ICMP)
		private double transferredBytesMs; //Indicates the quality of the connection bandwidth
		private double transferredBytesMsUpdatesCounter;
		private boolean connAvailable;
		private boolean serverAvailable;
		private String connType; //Wi-Fi, 3G or Other. Not needed, just to display info.
		//Parameters with information about the last offloading attempt
		private boolean doOffloading; //True if start an offloading process was decided, false otherwise
		private double estAndroidRuntime; //The estimation (in milliseconds) of the runtime in the Android mobile device of the 
potentially offloadable code
		private double estOffloadingTime; //The estimation (in milliseconds) of the offloading process duration
		private double estServerRuntime; //The estimation (in milliseconds) of the runtime in the server of the potentially 
offloadable code
		private double overallTime; //The time (in milliseconds) that took the execution of an algorithm (both if it was executed 
locally or offloaded to the server)
		private double realServerTime; //The time (in milliseconds) that the server needed to execute an algorithm (in case of 
offloading). Not needed, just to display info.
		private double parametersSize; //The sum of the sizes of each of the parameters that were sent to the server
```

```
	private double overhead; //The overhead that produces estimating the execution times of the potentially offloadable 
tasks
	//Persistent parameters
	private class CsrPair {
		public float csrServerDevice; //The computation speed relation between the server and the Android mobile 
device for a particular algorithm
		public float csrUpdatesCounter; //How many times this computation speed relation has been updated for a 
particular algorithm
	}
	SharedPreferences sPrefs; //Provides access to the persistent variables
	Map<AlgName, CsrPair> algCsrs;
	private class NetworkReceiver extends BroadcastReceiver {
		@Override
		public void onReceive(Context context, Intent intent) {
			ConnectivityManager conn =  (ConnectivityManager) 
context.getSystemService(Context.CONNECTIVITY_SERVICE);
			NetworkInfo networkInfo = conn.getActiveNetworkInfo();
			if (networkInfo != null && networkInfo.isConnected()) {
				if (connAvailable == false) calcPingAndBandwidth();
				connAvailable = true;
				int netType = networkInfo.getType();
				int netSubtype = networkInfo.getSubtype();
				if (netType == ConnectivityManager.TYPE_WIFI) connType = "Wi-Fi";
				else {
					if (netType == ConnectivityManager.TYPE_MOBILE  && netSubtype == 
TelephonyManager.NETWORK_TYPE_UMTS) connType = "3G";
					else connType = "Other";
				}
			}
			else {
				connType = "None";
				connAvailable = false;
				serverAvailable = false;
			}
		}
	}
	private NetworkReceiver networkStatusReceiver;
	//To be called onCreate in your Activity
	public Engine(Context theContext) {
		transferredBytesMsUpdatesCounter = 0;
		appContext = theContext;
		sPrefs = appContext.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
		Algorithms.setOffloadingEngine(this);
		//We initialize some variables with default values although they will be properly obtained soon (just in case 
there is an early potentially offloadable algorithm)
		ping = -1;
```

```
                transferredBytesMs = 200.0;
                connAvailable = false;
                serverAvailable = false;
                connType = "Unknown";
                estAndroidRuntime = -1;
                estOffloadingTime = -1;
                estServerRuntime = -1;
                realServerTime = -1;
                overallTime = -1;
                overhead = -1;
                loadPersistentParams(); //Load the stored parameters of the engine
                updateServerStatus();
                keepNetworkInfoUpdated(); //Register a listener to keep updated the network information
                Algorithms.loadAlgCostsDB(appContext);
        }
public void updateServerStatus()
{
        try {
                        if(new ServerStatus().execute(SERVER_URL).get().intValue() == 0)
                        {
                                connAvailable = true;
                                serverAvailable = true;
                                Log.v(TAG, " Server is available==>");
                        }
        } catch (Exception e) {
                        // TODO Auto-generated catch block
                        Log.v(TAG, " Server not availabile==>"+e);
                        e.printStackTrace();
                }
}
private class ServerStatus extends AsyncTask<String, Void, Integer> {
                @Override
                protected Integer doInBackground(String... urlAddress) {
                        try {
                                DefaultHttpClient httpClient = new DefaultHttpClient();
                                HttpGet httpGet = new HttpGet(urlAddress[0]);
                                ResponseHandler<String> resHandler = new BasicResponseHandler();
                                httpClient.execute(httpGet, resHandler);
                                return 0;
                        } catch (Exception e) {
                                Log.v(TAG, " Server not available"+e);
                                e.printStackTrace();
                        }
                        return -1;
        }
}
```

```
	private void loadPersistentParams() {
		algCsrs = new EnumMap<AlgName, CsrPair>(AlgName.class);
		AlgName[] algNamesEnum = AlgName.values();
		for (int i = 0; i < algNamesEnum.length; i++) {
			CsrPair itCsrPair = new CsrPair();
			itCsrPair.csrServerDevice = sPrefs.getFloat(algNamesEnum[i] + "Csr", -1);
			itCsrPair.csrUpdatesCounter = sPrefs.getFloat(algNamesEnum[i] + "Count", 0);
			algCsrs.put(algNamesEnum[i], itCsrPair);
		}
	}
	//To be called onPause in your Activity
	public void savePersistentParams() {
		SharedPreferences.Editor sPrefsEditor = sPrefs.edit();
		Iterator<AlgName> enumKeySet = algCsrs.keySet().iterator();
    while (enumKeySet.hasNext()) {
	AlgName itAlgName = enumKeySet.next();
	CsrPair itCsrPair = algCsrs.get(itAlgName);
	sPrefsEditor.putFloat(itAlgName + "Csr", itCsrPair.csrServerDevice);
		sPrefsEditor.putFloat(itAlgName + "Count", itCsrPair.csrUpdatesCounter);
    }
		sPrefsEditor.commit();
	}
	private void keepNetworkInfoUpdated() {
		networkStatusReceiver = new NetworkReceiver();
		IntentFilter connFilter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
		appContext.registerReceiver(networkStatusReceiver, connFilter);
	}
	//To be called onDestroy in your Activity, undoes the changes made by keepNetworkInfoUpdated() and closes the
DataBaseHelper
	public void unregisterBroadcastReceivers() {
		if (networkStatusReceiver != null) appContext.unregisterReceiver(networkStatusReceiver);
		Algorithms.closeAlgCostsDB();
	}
	/*
	 * Updates the Csr and/or the costs DB when needed.
	 * Needs to read global variables to check the state of the offloading procedure.
	 * There are two systems to calculate the cost estimations:
	 * 1. The developer provides a function to do so
	 * 2. The developer provides a DB with input-cost pairs (can be generated in our server)
	 * Depending on which system has been used for the current case (algName) there will be different updating needs
	 */
	private void updateCostCalcSystems(AlgName algName) {
		if (!Algorithms.isAlgInCostsDB(algName)) {
/*With the first system we only update the Csr when the algorithm has been executed locally and if we already calculated
estServerRuntime; we don't want to calculate the estimated cost of this algorithm only for the purpose of updating the Csr as the
calculations could be a bit expensive (although they shouldn't't). More important, this case would occur when there is no network,
and then any execution, even the heavy ones, would be done locally. We don't want to make the Csr fit with such cases, as
Android gives more priority to heavy executions than to small ones (thus running proportionally faster the heavy ones); this would
lead to a not consistent Csr. In the offloading decision scenario, when there is network connection, the heavy executions would
always be offloaded. We check the time the algorithm took to execute to be greater than 15 ms (smallervalues might be not
accurate enough).*/
if (!doOffloading && estServerRuntime != -1 && overallTime > MIN_RELEVANT_TIME)
        updateCsr(algName, (float) (overallTime/estServerRuntime)); //!doOffloading, so overallTime is an Android runtime
                }
        else {/*With the second system we always update the costs database. Even for heavy executions produced locally
                        because of no network connection.*/
                        if (!doOffloading) Algorithms.updateCostsDB(algName, overallTime, false);
                        else Algorithms.updateCostsDB(algName, realServerTime, true);
                }

        }
        //Updates the Csr for the algorithm algName
        public void updateCsr(AlgName algName, float recentCsr) {
                CsrPair csrPair = algCsrs.get(algName);
                csrPair.csrServerDevice = csrPair.csrServerDevice * csrPair.csrUpdatesCounter /
(csrPair.csrUpdatesCounter+1) + recentCsr / (csrPair.csrUpdatesCounter+1);
                if (csrPair.csrUpdatesCounter < Algorithms.MAX_REPETITIONS) csrPair.csrUpdatesCounter++;
        }
        /*
         * Execute an algorithm in the Android device with a known cost in the server,
         * in order to establish a computation speed relation (how many times faster the server is)
         * This function will be called during the first offloading attempt of an Android application using this engine
         */
        private void calculateRelation() {
                double startAlgorithmTime = ((double) System.nanoTime()) / 1000000.0;
                Algorithms.executeLocally(AlgName.doSomeLoops, Long.valueOf(1000000).toString());
                double timeTaken = ((double) System.nanoTime()) / 1000000.0 - startAlgorithmTime;
                double AndroidInstMs = Algorithms.getCost(AlgName.doSomeLoops, Long.valueOf(1000000).toString()) /
timeTaken;
                float firstTimeCsr = (float) (SERVER_INST_MS / AndroidInstMs);
                Iterator<AlgName> enumKeySet = algCsrs.keySet().iterator();
    while (enumKeySet.hasNext()) {
        AlgName itAlgName = enumKeySet.next();
        CsrPair itCsrPair = algCsrs.get(itAlgName);
        itCsrPair.csrServerDevice = firstTimeCsr;
        itCsrPair.csrUpdatesCounter++;
    }
        }
```

```
/*
 * Decides where to execute the algorithm, locally (no offloading) or on the server (offloading is done).
 * Returns true if the decision is to offload, false otherwise.
 */
public boolean decide(AlgName algName, String... parameters) {
        Log.v(TAG, "Making decision...........");
        estAndroidRuntime = -1;
        estOffloadingTime = -1;
        estServerRuntime = -1;
        if (connAvailable && serverAvailable) {
                Log.v(TAG, "At if (connAvailable && serverAvailable) { block");
//Only the very first time that there is the possibility to do offloading, a initial computation speed relation is calculated
                if (getCsrUpdCountFromAlg(algName) == 0) calculateRelation();
                double startAlgorithmTime = ((double) System.nanoTime()) / 1000000.0;
                this.estServerRuntime = Algorithms.getCost(algName, parameters) / SERVER_INST_MS;
//Estimated server execution time
                this.overhead = ((double) System.nanoTime()) / 1000000.0 - startAlgorithmTime;
                this.estAndroidRuntime = estServerRuntime * getCsrFromAlg(algName);
//Estimated Android execution time
                for (int i = 0; parameters != null && i < parameters.length; i++) parametersSize +=
parameters[i].length();
                this.estOffloadingTime = ping + estServerRuntime + parametersSize/transferredBytesMs;

                Log.v(TAG, " estServerRuntime==>"+estServerRuntime);
                Log.v(TAG, " overhead ===>"+overhead);
                Log.v(TAG, " estAndroidRuntime===>"+estAndroidRuntime);
                Log.v(TAG, " estOffloadingTime===>"+estOffloadingTime);
                Log.v(TAG, " returning ....."+(estOffloadingTime < estAndroidRuntime));
                //Time saving criteria
                return estOffloadingTime < estAndroidRuntime;
        }
        else
        {
                Log.v(TAG, "Failed to execute block if (connAvailable && serverAvailable) { ");
                return false;
        }
}
```

```
	/*
	 * To be called wherever in your Activity, replacing where you had a potentially offloadable part of code.
	 * Decides where to execute the algorithm, locally (no offloading) or on the server (offloading is done), executes it and
retrieves the result.
	 * Returns the result on success (in String form), or the String "Error" on failure.
	 */
	public Map<String, ExecutionPerformance>  execute(boolean forceOffloading, AlgName algName, String...
parameters) {
		Map<String, ExecutionPerformance> resultMap = new HashMap<String, ExecutionPerformance>();
		parametersSize = 0;
		overhead = -1;
		if (forceOffloading) doOffloading = true;
		else doOffloading = decide(algName, parameters);
		String algResult = "";
		realServerTime = -1;
		overallTime = -1;
		double startTime = ((double) System.nanoTime()) / 1000000.0;
		//Execute to forcefully intialize performnce metrics
		//decide(algName, parameters);
		if (doOffloading) { //Do offloading
			int execParamsLength = 1;
			if (parameters != null) execParamsLength += parameters.length;
			String[] execParams = new String[execParamsLength];
			execParams[0] = algName.toString();
			for (int i = 1; i < execParams.length; i++) execParams[i] = parameters[i-1];
			GetServerData getServerData = new GetServerData();
			getServerData.execute(execParams);
			try {
				algResult = getServerData.get(); //This can also return the word "Error"
			} catch (Exception e) {
				Log.v(TAG, " Server execution error#flaging as not available"+e);
				e.printStackTrace();
				serverAvailable = false;
				algResult = "Error";
			}
		if (algResult.equals("Error")) { //Unable to retrieve the data. URL may be invalid or the server may be down.

				serverAvailable = false;
			}
			else {
				try {
			realServerTime = Double.parseDouble(getElementValueFromXML(algResult, "runtime"));
					algResult = getElementValueFromXML(algResult, "result");
				} catch (Exception e) {
					e.printStackTrace();
					Log.v(TAG, " Error retrieveing Server results"+e);
```

```
                                        algResult = "Error";
                                }
                        }
                }

                //In case of a failed offloading attempt, we don't retry offloading, we make the system behave like the
decision would have been to not offload
                //(the main reason of failing is losing the network connection, which takes too long to recover)
                if (algResult.equals("Error")) {
                        Log.v(TAG, " Diabling offloading to false and revering to local execution");
                        doOffloading = false;
                        startTime = ((double) System.nanoTime()) / 1000000.0;
                }

                //Do not offload, execute locally in the Android mobile device
                if (!doOffloading) algResult = Algorithms.executeLocally(algName, parameters);

                //This can be either the Android runtime or the total offloading time
                this.overallTime = ((double) System.nanoTime()) / 1000000.0 - startTime;
                Log.v(TAG, " Overall time ==>"+overallTime);
                if (doOffloading && !algResult.equals("Error")) { //If offloading was done successfully
                        if (parametersSize == 0) { //decide was not called
                                for (int i = 0; parameters != null && i < parameters.length; i++) parametersSize +=
parameters[i].length();
                        }
                        double transferDataTime = overallTime - realServerTime - ping;
                        //If the size of the sent parameters was big enough to be significant and the transferDataTime is
also significant (bigger than 5 milliseconds), update the transferredBytesMs.
                        if (parametersSize > 1024 && transferDataTime >= 5.0) {
                                if (transferredBytesMsUpdatesCounter < 20) transferredBytesMsUpdatesCounter++;
                                transferredBytesMs = transferredBytesMs * transferredBytesMsUpdatesCounter /
(transferredBytesMsUpdatesCounter+1) + (parametersSize / transferDataTime) / (transferredBytesMsUpdatesCounter+1);
                        }
                }

                //Updates the Csr and/or the costs DB when needed
                if (!doOffloading || !algResult.equals("Error")) {
                        UpdateCostCalcSystemsThread updateCostCalcSystemsThread = new
UpdateCostCalcSystemsThread(algName);
                        updateCostCalcSystemsThread.start();
                }
                //Save performance execution
                ExecutionPerformance executionPerformance = new ExecutionPerformance();
                executionPerformance.setAlgorithmName(algName.toString());
                executionPerformance.setDoOffloading(this.doOffloading);
                executionPerformance.setEstAndroidRuntime(this.estAndroidRuntime);
```

```
		executionPerformance.setEstOffloadingTime(this.estOffloadingTime);
		executionPerformance.setEstServerRuntime(this.estServerRuntime);
		executionPerformance.setOverallTime(this.overallTime);
		executionPerformance.setOverhead(this.overhead);
		executionPerformance.setRealServerTime(this.realServerTime);
		resultMap.put(algResult, executionPerformance);
		return resultMap;
}
/*
 * execute can be called only with an algName and its parameters. By default, forceOffloading = false.
 */
public Map<String, ExecutionPerformance> execute(AlgName algName, String... parameters) {
		return execute(false, algName, parameters);
}
private class UpdateCostCalcSystemsThread extends Thread {
		AlgName currentAlgName;
		UpdateCostCalcSystemsThread(AlgName currentAlgName) {
		this.currentAlgName = currentAlgName;
		}
		public void run() {
		updateCostCalcSystems(currentAlgName);
		}
}
private class GetServerData extends AsyncTask<String, Void, String> {
		@Override
		protected String doInBackground(String... execParams) {
				try {
						final HttpParams httpParams = new BasicHttpParams();
						HttpConnectionParams.setConnectionTimeout(httpParams, 60000);
						HttpConnectionParams.setSoTimeout(httpParams, 60000);
						DefaultHttpClient httpClient = new DefaultHttpClient(httpParams);
						HttpPost httpPost = new HttpPost(SERVER_URL);
						List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
						nameValuePairs.add(new BasicNameValuePair("algName", execParams[0]));
						for (int i = 1; i < execParams.length; i++)
						nameValuePairs.add(new BasicNameValuePair("param" + i, execParams[i]));
						UrlEncodedFormEntity entity = new UrlEncodedFormEntity(nameValuePairs);
						httpPost.setEntity(entity);
						ResponseHandler<String> resHandler = new BasicResponseHandler();
						String responseData = httpClient.execute(httpPost, resHandler);
						return responseData;
				} catch (Exception e) {
						e.printStackTrace();
						return "Error";
				}
		}
	}
private void calcPingAndBandwidth() {
		pingCounter = 0;
		pingsArray = new double[10];
		timePingStart = ((double) System.nanoTime()) / 1000000.0;
		//The next will call itself recursively 10 times and calculate the average ping (removing outliers)
		//Once done, it will call calcTransferredBytesPerMs() to calculate the bandwidth quality
		new GetPing().execute(SERVER_URL);
	}

private class GetPing extends AsyncTask<String, Void, Integer> {
		@Override
		protected Integer doInBackground(String... urlAddress) {
			try {
				DefaultHttpClient httpClient = new DefaultHttpClient();
				HttpGet httpGet = new HttpGet(urlAddress[0]);
				ResponseHandler<String> resHandler = new BasicResponseHandler();
				httpClient.execute(httpGet, resHandler);
				return 0;
			} catch (Exception e) {
				e.printStackTrace();
				return -1;
			}
		}
		@Override
		protected void onPostExecute(Integer respCode) {
			if (respCode == -1) { //Ping failed
				serverAvailable = false;
				connAvailable =false;
			}
			else {
				pingsArray[pingCounter] = ((double) System.nanoTime()) / 1000000.0 - timePingStart;
				pingCounter++;
				if (pingCounter < 10) {
					timePingStart = ((double) System.nanoTime()) / 1000000.0;;
					new GetPing().execute(SERVER_URL);
				}
				else if (pingCounter == 10) {
					serverAvailable = true;
					connAvailable =true;
					ping = Engine.calcAverage(pingsArray);
					calcTransferredBytesPerMs();
				}
			}
		}
	}
```

```
private void calcTransferredBytesPerMs() {
		String fileContent = "";
		try {
			InputStream is = appContext.getAssets().open("fileToSend");
			byte[] buffer = new byte[is.available()];
			is.read(buffer);
			is.close();
			fileContent = new String(buffer);
		} catch (IOException e) {
			throw new RuntimeException(e);
		}
		//This should update transferredBytesMs in almost all cases
		//If not, the value assigned in the Engine constructor function (200), will be used until the next update
		execute(true, AlgName.fileAndLoops, Long.toString(0), fileContent);
}

private static String getElementValueFromXML(String xmlString, String tagName)
			throws ParserConfigurationException, SAXException, IOException {
		DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
		DocumentBuilder db = dbf.newDocumentBuilder();
		InputSource is = new InputSource();
		is.setCharacterStream(new StringReader(xmlString));
		Document doc = db.parse(is);
		NodeList summary = doc.getElementsByTagName(tagName);
		Element line = (Element) summary.item(0);
		return getCharacterDataFromElement(line);
}

private static String getCharacterDataFromElement(Element e) {
		Node child = e.getFirstChild();
		if (child instanceof CharacterData) {
			CharacterData cd = (CharacterData) child;
			return cd.getData();
		}
		return "";
}

private static double calcAverage (double[] valuesArray) {

		//Calculate the average
		double valuesSum = 0;
		for (int i = 0; i < valuesArray.length; i++) {
			valuesSum += valuesArray[i];
		}
		double average = valuesSum / ((double) valuesArray.length);
```

```
			//Recalculate the average omitting all values with a high deviation
			double niceValuesSum = 0;
			double niceValuesCount = 0;
			double auxValue, auxAverage;
			for (int i = 0; i < valuesArray.length; i++) {
				auxValue = valuesArray[i];
				if (auxValue < 0) auxValue *= -1;
				auxAverage = average;
				if (auxAverage < 0) auxAverage *= -1;
				if (auxValue <= auxAverage * 2) {
					niceValuesSum += valuesArray[i];
					niceValuesCount++;
				}
			}
			return niceValuesSum/niceValuesCount;
		}
}
/**
=========================================================================
			SyncKeyServiceDatabase.java
=========================================================================
* SyncKeyServiceDatabase submits database records the server for synchronization
*
* @authors Alfayo oyugi Adede  Email: alfayaoyugi@googlemail.com
*
* @version $Date: 2014-05-14 13:29:34 +0200  $ $Revision: 1 $
*
*/
package adede.msc.project.async;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import org.apache.http.NameValuePair;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.BasicResponseHandler;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import adede.msc.project.core.Engine;
import adede.msc.project.util.JavaBeanUtil;
import android.os.AsyncTask;
import android.util.Log;
```

```
public class SyncKeyServiceDatabase<T> extends AsyncTask<T, Void, T>{
        private static final String SERVER_URL = Engine.BASE_SERVER_URL+"SyncKeyServiceRegistry";
        private static final String TAG= "SyncKeyServiceDatabase";
        @SuppressWarnings("unchecked")
        @Override
        protected T doInBackground(T... entityArray) {
                try {
                        final HttpParams httpParams = new BasicHttpParams();
                        T entity = entityArray[0];
                  //Wait max. 60 seconds to establish a TCP connection
                  HttpConnectionParams.setConnectionTimeout(httpParams, 60000);

                  //Wait max. 60 seconds for a subsequent byte of data
                  HttpConnectionParams.setSoTimeout(httpParams, 60000);
                        DefaultHttpClient httpClient = new DefaultHttpClient(httpParams);
                        HttpPost httpPost = new HttpPost(SERVER_URL);
                        List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
                        Map<String, String> attributeMap = JavaBeanUtil.getFieldMap(entity);
                        for(String key : attributeMap.keySet())
                        {
                        nameValuePairs.add(new BasicNameValuePair(key, attributeMap.get(key)));
                        }
                        //To be used Servlet invocation during invocation using Java Reflection API
                        nameValuePairs.add(new BasicNameValuePair("entity", entity.getClass().getCanonicalName()));
                        UrlEncodedFormEntity encodedEntity = new UrlEncodedFormEntity(nameValuePairs);
                        httpPost.setEntity(encodedEntity);
                        ResponseHandler<String> resHandler = new BasicResponseHandler();
                        String responseData = httpClient.execute(httpPost, resHandler);

                        //Convert XML to JAVA Bean
                        return JavaBeanUtil.xmlToObject(responseData, (Class<T>) entity.getClass());
                } catch (Exception exception) {
                        Log.v(TAG,"Error SyncKeyServiceDatabase  "+exception);
                        }
                return null;
            }
        }
```

```
/*
 * ===========================================================================
 *                            DatabaseService.java
 * ===========================================================================
* The DatabaseService an android application Service performing long-running
* synchronization in the background without involving user interface.
* This class polls the database and submit the data to the server for synchronization
* It also update the local database thus synchronizing the two databases
* The class uses a timer to schedule the the synchronization task.
* The task is also executed in async to avoid user interface blocking
* It extends  OrmLiteBaseService<MySQLiteHelper> to support
* managed SQLite database access through ORMLite
*
* @authors Alfayo oyugi Adede  Email: alfayaoyugi@googlemail.com
* @version $Date: 2014-05-14 13:29:34 +0200  $ $Revision: 1 $
*/
package adede.msc.project.async;

import java.util.List;
import java.util.Timer;
import java.util.TimerTask;
import adede.msc.project.entity.AccessControl;
import adede.msc.project.entity.AuditLog;
import adede.msc.project.entity.ExecutionPerformance;
import adede.msc.project.persistence.MySQLiteHelper;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;
import com.j256.ormlite.android.apptools.OrmLiteBaseService;

public class DatabaseService extends OrmLiteBaseService<MySQLiteHelper> {
        private static final int UPDATE_ENTRY = 999;
        private static final long POLL_INTERVAL = 15000;
        private static final String TAG = "DatabaseService";
        public static DatabaseService instance = null;
        private Timer timer;
        private TimerTask task = new TimerTask() {
                @Override
                public void run() {
                        // Query the AccessControl, AuditLog and ExecutionPerformance
                        List<AccessControl> accessControlList = null;
                        List<AuditLog> auditLogList = null;
                        List<ExecutionPerformance> executionPerformanceList = null;
                        try {
                                accessControlList = getHelper().getAccessControlDao().queryForAll();
```

```
                                        auditLogList = getHelper().getAuditLogDao().queryForAll();
                                        executionPerformanceList = getHelper().getExecutionPerformanceDao().queryForAll();
                                } catch (Exception exception) {
                        Log.v(TAG, "Error polling local database##AccessControl##AuditLog#ExecutionPerformance"+ exception);
                                }
                                //Submit AccessControl to the server asynchronous
                                for (AccessControl accessControl : accessControlList) {
                SyncKeyServiceDatabase<AccessControl> syncAccessControl = new SyncKeyServiceDatabase<AccessControl>();
                                        syncAccessControl.execute(new AccessControl[] { accessControl });
                                        try {
                                                // Check if the AccessControl requires an update
                                                AccessControl serverEntry = syncAccessControl.get();
                                                if (serverEntry.getId() == UPDATE_ENTRY) {
                                                        serverEntry.setId(accessControl.getId());
                                                        getHelper().getAccessControlDao().update(serverEntry);
                                                }
                                        } catch (Exception exception) {
                                Log.v(TAG, "Error Synchronizing database##AccessControl "+ exception);
                                        }
                                }
                                //Submit AuditLog to the server asynchronous
                                for (AuditLog auditLog : auditLogList) {
                Log.v(TAG, "Synchronizing  AuditLog");
                SyncKeyServiceDatabase<AuditLog> syncAuditLog = new SyncKeyServiceDatabase<AuditLog>();
                                        syncAuditLog.execute(new AuditLog[] { auditLog });
                                        try {
                                                //ignore since local log need not to be more updated
                                                //getHelper().getAuditLogDao().update(syncAuditLog.get());
                                                } catch (Exception exception) {
                                                Log.v(TAG, "Error Synchronizing database##AuditLog" + exception);

                                        }
                                }

                                //Submit ExecutionPerformance to the server asynchronous
                                for (ExecutionPerformance executionPerformance : executionPerformanceList) {
                                        SyncKeyServiceDatabase<ExecutionPerformance> syncExecutionPerformance= new
        SyncKeyServiceDatabase<ExecutionPerformance>();
                                        syncExecutionPerformance.execute(new ExecutionPerformance[] {
        executionPerformance });
                                        Log.v(TAG, "==========Synchronizing
        ExecutionPerformance====================");
                                        Log.v(TAG, executionPerformance.toString());
                                        Log.v(TAG, "================================");
                                        try {
                                                //ignore since local ExecutionPerformance need not to be more updated
        //getHelper().getExecutionPerformanceDao().update(syncExecutionPerformance.get());

                                } catch (Exception exception) {
                                        Log.v(TAG, "Error Synchronizing database##ExecutionPerformance" +
exception);

                                }
                        }

                }
        };
        @Override
        public void onCreate() {
                timer = new Timer();
                instance = this;
                super.onCreate();
        }
        @Override
         public void onDestroy()  {
           instance = null;
           super.onDestroy();
          }
        @Override
        public int onStartCommand(Intent intent, int flags, int startId) {
                timer.schedule(task, 5000, DatabaseService.POLL_INTERVAL);
                return Service.START_STICKY;
        }
        @Override
        public IBinder onBind(Intent intent) {
                return null;
        }
}
```

APPENDIX B: TABULATION OF DATA

FILE SIZE (Kb)	OFFLOAD_STATUS	EST_ANDROID_TIME	EST_OFFLOADING_TIME	ESTIMATED_SERVER_TIME	OVERALL_TIME	REAL_SERVER_TIME	OVER_HEAD	DECRYPTION_TIME
20	1	0.00028886	8.695004176	4.18E-06	54.59513701	54.59513701	8.911134005	20.41625901
40	1	0.000125871	59.94500733	7.33E-06	104.503583	104.503583	24.71923999	36.62109399
60	1	0.000269582	83.27501604	1.60E-05	53.81384403	53.81384403	31.616211	36.10229599
80	1	0.000472205	150.8500287	2.87E-05	95.72965501	95.72965501	76.965332	44.95239399
100	1	0.000575282	210.3700316	3.16E-05	102.549233	102.549233	81.48193601	87.09717199
20	0	-1	-1	-1	33.325197	-1	-1	0
40	0	-1	-1	-1	146.972658	-1	-1	0
60	0	-1	-1	-1	167.999267	-1	-1	0
80	0	-1	-1	-1	212.127694	-1	-1	0
100	0	-1	-1	-1	309.75343	-1	-1	0
20	0	-1	-1	-1	56.457522	-1	-1	12.634278
40	0	-1	-1	-1	246.704111	-1	-1	103.240971
60	0	-1	-1	-1	233.581548	-1	-1	83.557128
80	0	-1	-1	-1	404.388443	-1	-1	127.258306
100	0	-1	-1	-1	529.693618	-1	-1	162.353521
20	1	7.37E-05	8.695004176	4.18E-06	2.287513018	2.287513018	6.46972701	0
40	1	0.000152716	59.94500733	7.33E-06	7.08398807	7.08398807	24.780275	0
60	1	0.000270235	83.27501604	1.60E-05	12.95967805	12.95967805	33.538819	0
80	1	0.000694281	150.8500287	2.87E-05	20.092219	20.092219	118.804932	0
100	1	0.000900889	210.3700316	3.16E-05	23.54184306	23.54184306	84.411621	0

Table 7 : Performance metrics data

*Source : Author's compilation (**extracted from execution_performance table in raca.db**)*

APPENDIX C: DETAILED CLASS DESIGN

Attached is a detailed class diagrams of:-

i. *Mobile application module*

ii. *Web application module*

APPENDIX D: DETAILED USER INTERFACE DESIGN

i. Mobile phone application User Interface sketches

<u>Login interface</u>

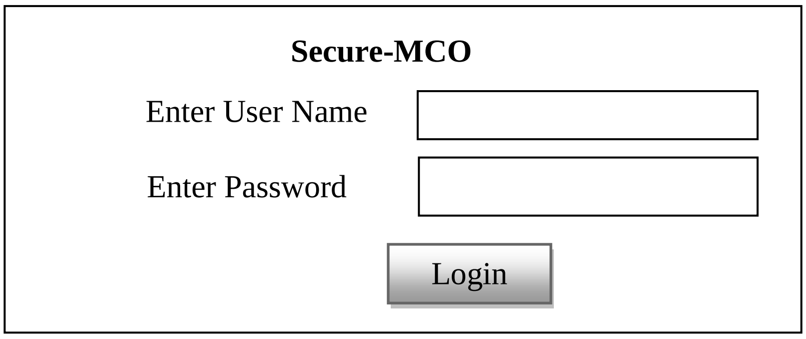


Figure 49 : Mobile phone application login page sketch
Source : Author's compilation

<u>Home page interface</u>

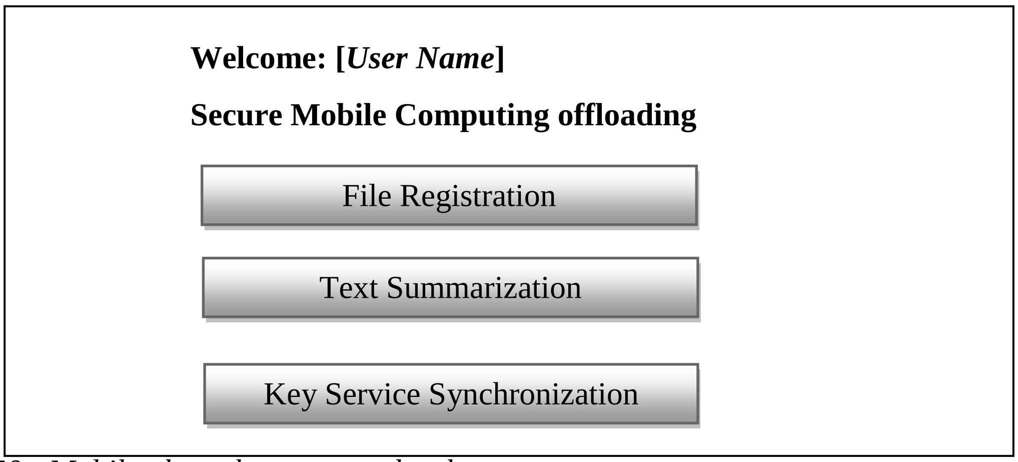

Welcome: [User Name]
Secure Mobile Computing offloading
File Registration
Text Summarization
Key Service Synchronization

Figure 50 : Mobile phone home page sketch
Source : Author's compilation

<u>Select file for encryption</u>

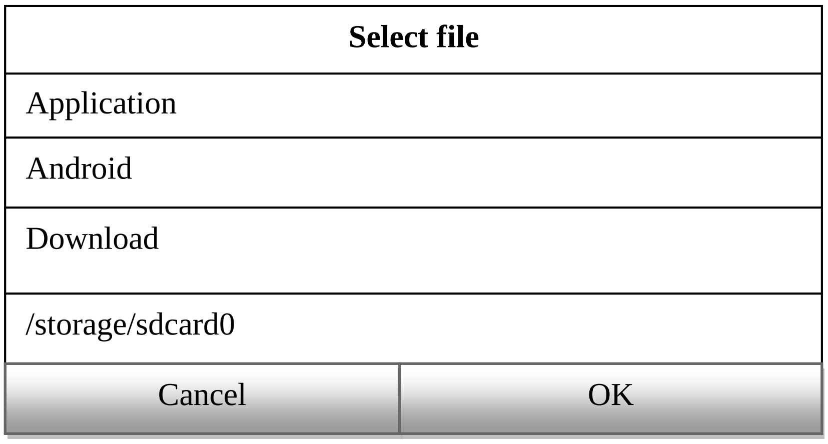
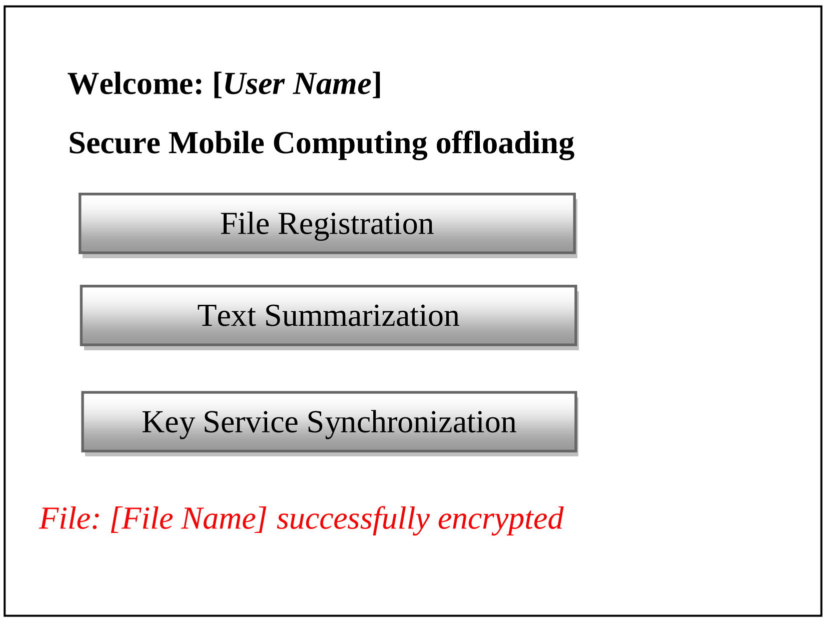


Figure 51 : File selection page sketch
Source : Author's compilation

File encryption result



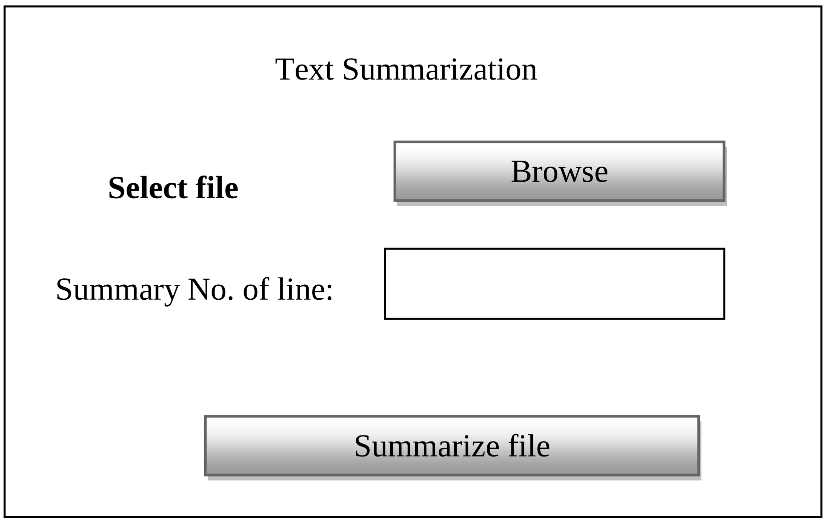
Welcome: [*User Name*]

Secure Mobile Computing offloading

File: [File Name] successfully encrypted

Figure 52 : File encryption result page sketch
Source : Author's compilation

Text summarization



Text Summarization

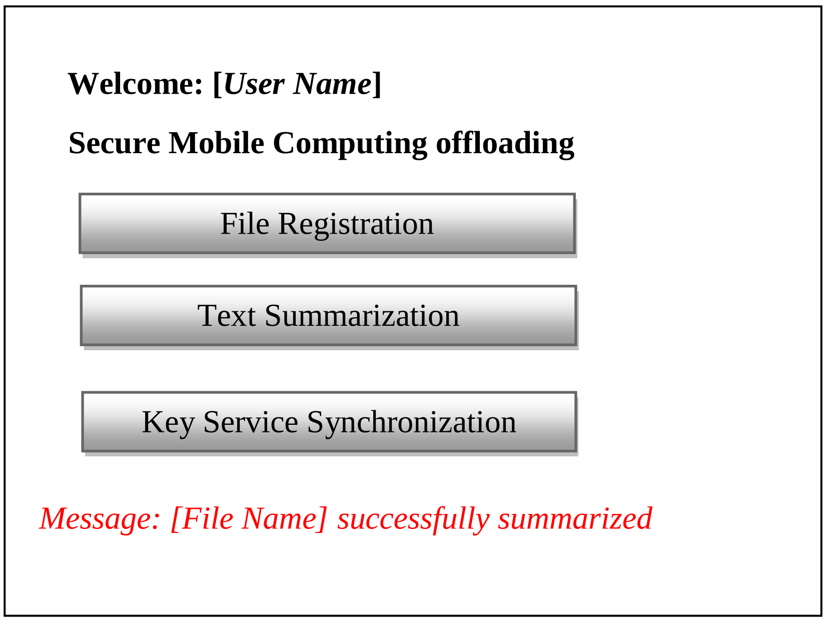
Select file

Summary No. of line:

Figure 53 :Text summarization page sketch

Source : Author's compilation

Text file summarization result


Welcome: [*User Name*]

Secure Mobile Computing offloading

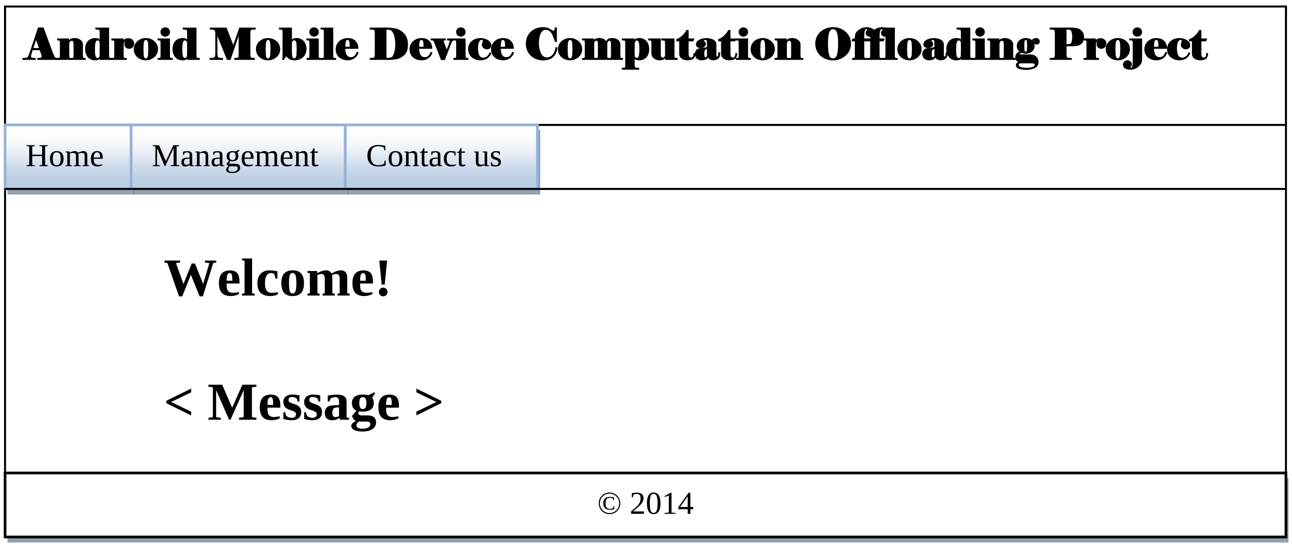
Message: [File Name] successfully summarized

Figure 54 : Text summarization result page sketch

Source : Author's compilation

ii. Web-based application User Interface sketches

Home page before login


Android Mobile Device Computation Offloading Project

Welcome!

< Message >

© 2014

Figure 55: Offloading engine home page sketch

Source : Author's compilation

Login Page

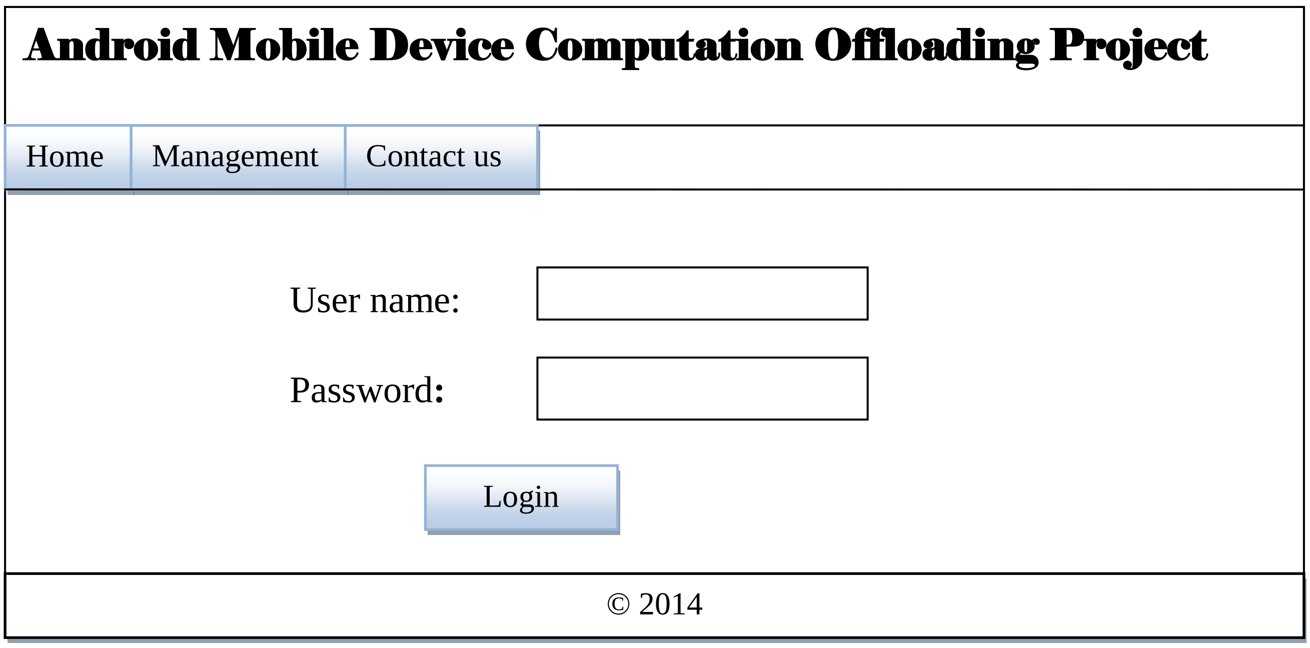


Figure 56 : Web Login page sketch

Source : Author's compilation

Home page after login

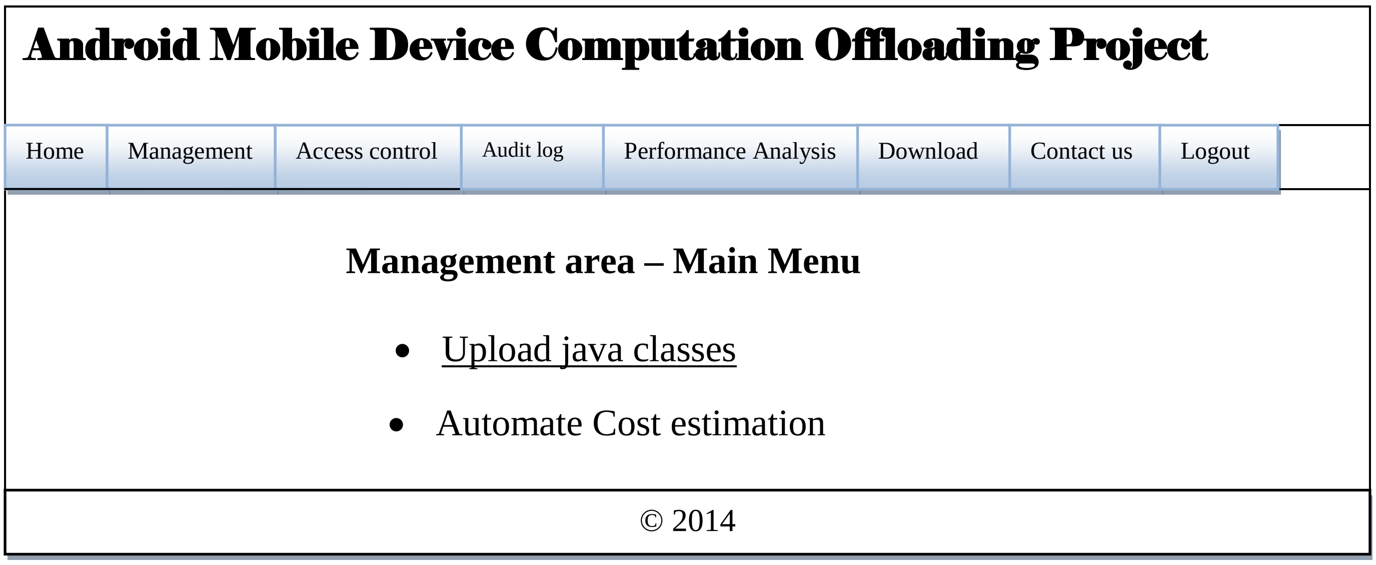
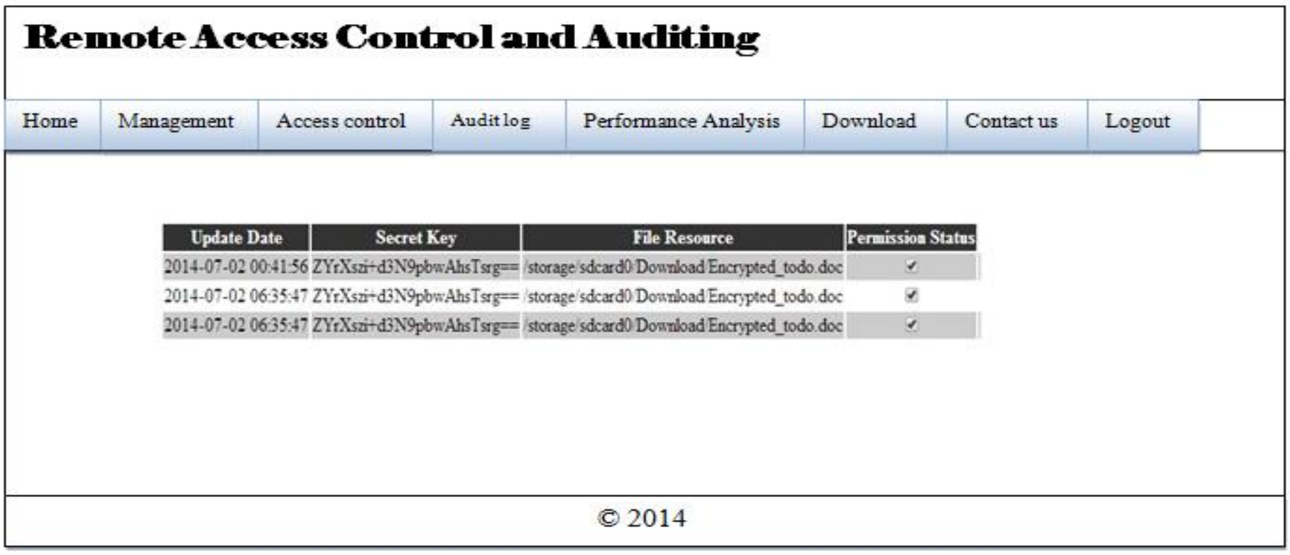


Figure 57 : Home page after login sketch

Source : Author's compilation

Access Control page

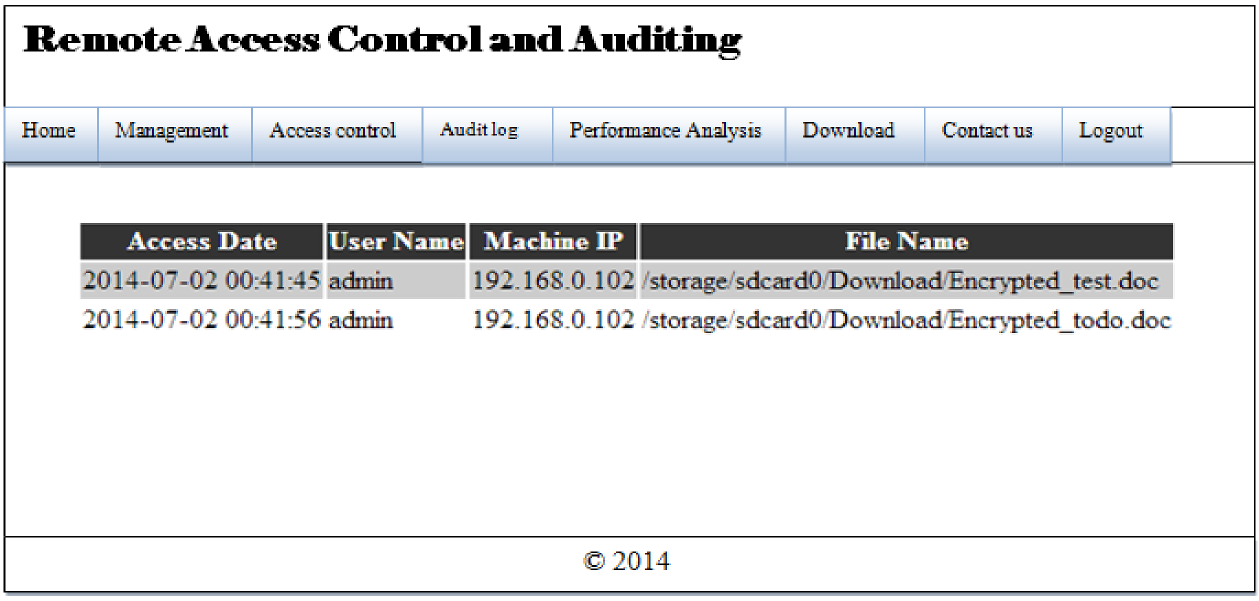


Update Date	Secret Key	File Resource	Permission Status
2014-07-02 00:41:56	ZYrXszi+d3N9pbwAhsTsrg==	/storage/sdcard0/Download/Encrypted_todo.doc	✓
2014-07-02 06:35:47	ZYrXszi+d3N9pbwAhsTsrg==	/storage/sdcard0/Download/Encrypted_todo.doc	✓
2014-07-02 06:35:47	ZYrXszi+d3N9pbwAhsTsrg==	/storage/sdcard0/Download/Encrypted_todo.doc	✓

Figure 58 : Access control page sketch

Source : Author's compilation

Audit log interface



Access Date	User Name	Machine IP	File Name
2014-07-02 00:41:45	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_test.doc
2014-07-02 00:41:56	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_todo.doc

Figure 59 : Audit log page sketch

Source : Author's compilation

Performance metrics analysis-Bar Chart

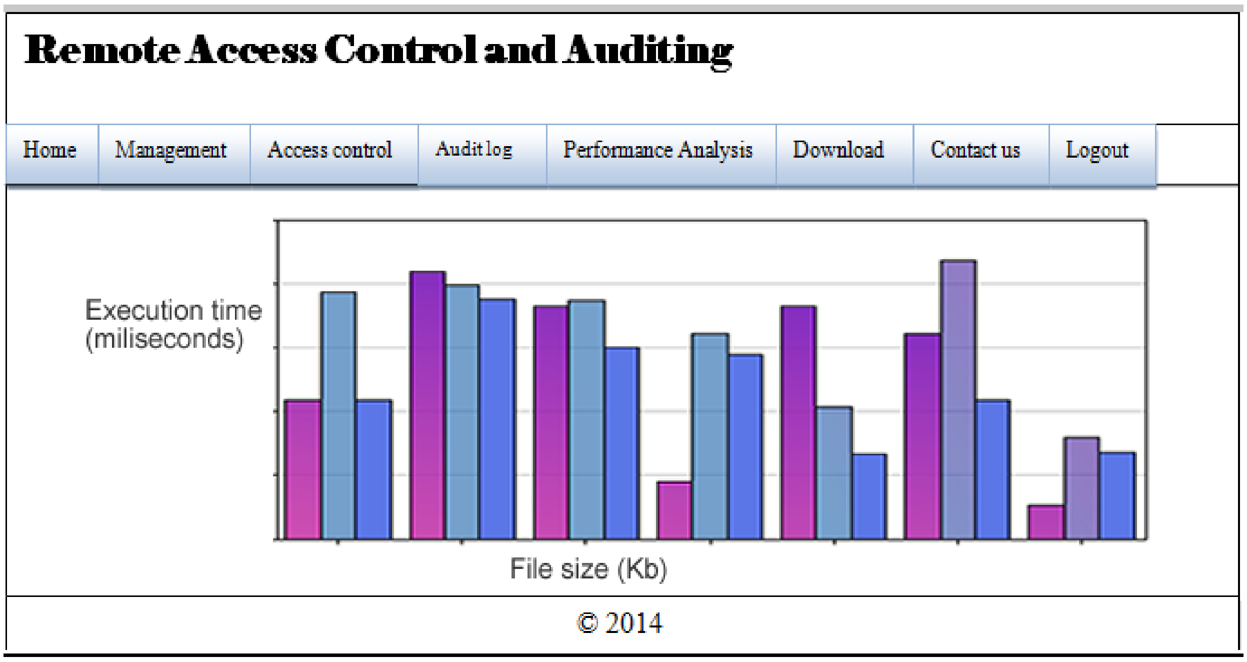


Figure 60 : Performance bar chart sketch

Source : Author's compilation

Performance metrics analysis-line graph

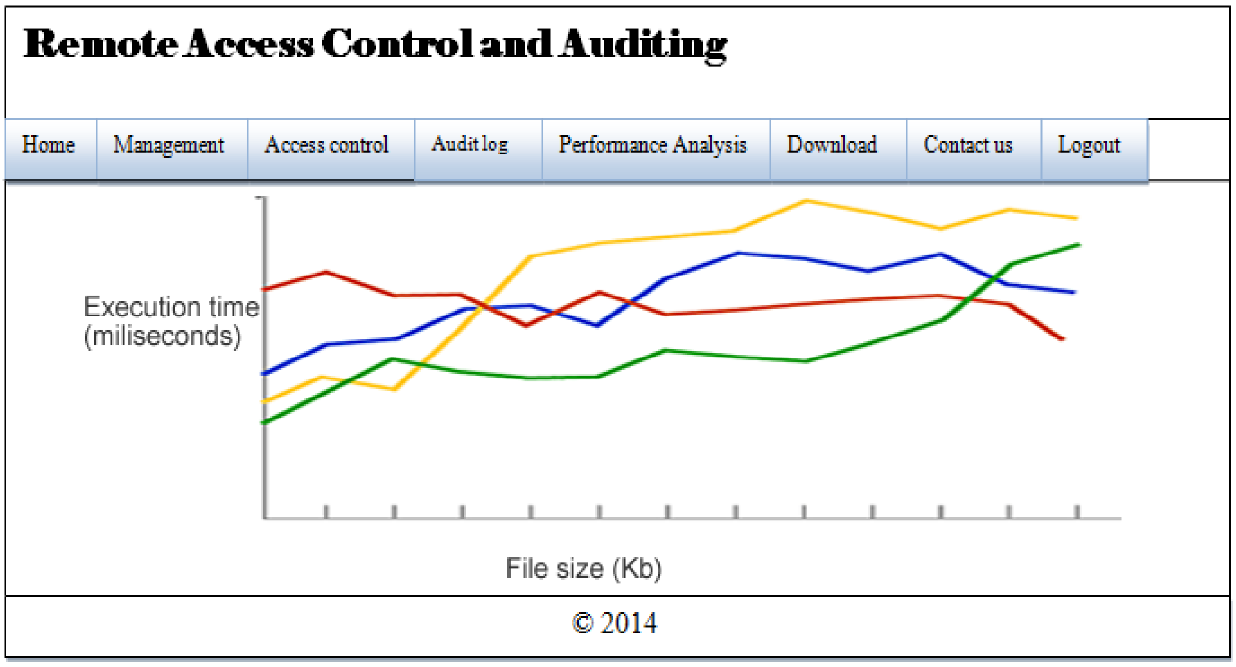


Figure 61: Performance line graph sketch

Source : Author's compilation

APPENDIX E: INSTALLATION GUIDE

The application is distributed through two (2) main packages. For mobile application it is **Secure-MCO.apk** while the server module is distributed as **offload.war.** To install mobile phone module, copy **Secure-MCO.apk** and double click on it to run installation setup.

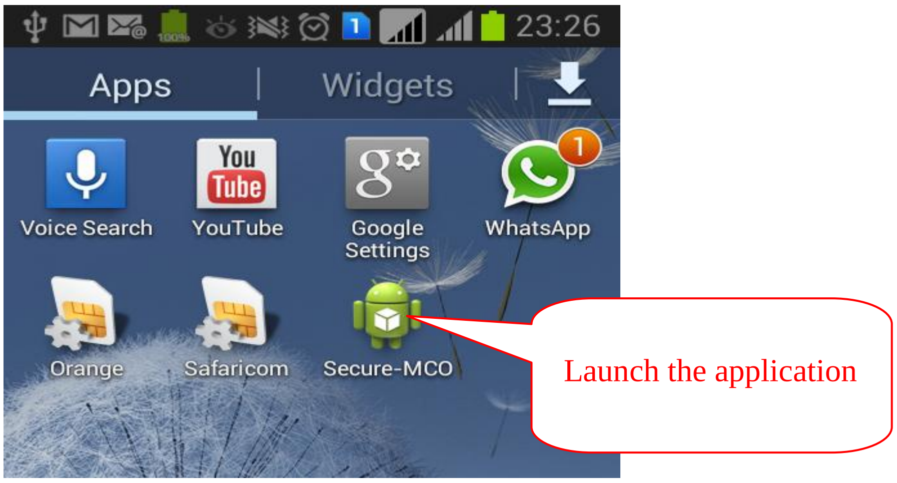
Installation of **offload.war** requires the prior the installation of Apache Tomcat 6.0_20 Servlet/JSP Container. The procedure for tomcat installation can be found online at http://tomcat.apache.org/tomcat-6.0-doc/setup.html. Furthermore, the User must follow the steps outlined at http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html to configure SSL on tomcat to allow the use of HTTPS.

Furthermore, the surrogate machine used to host **offload.war** application be accessible on WLAN with **192.168.0.101** as its IP.

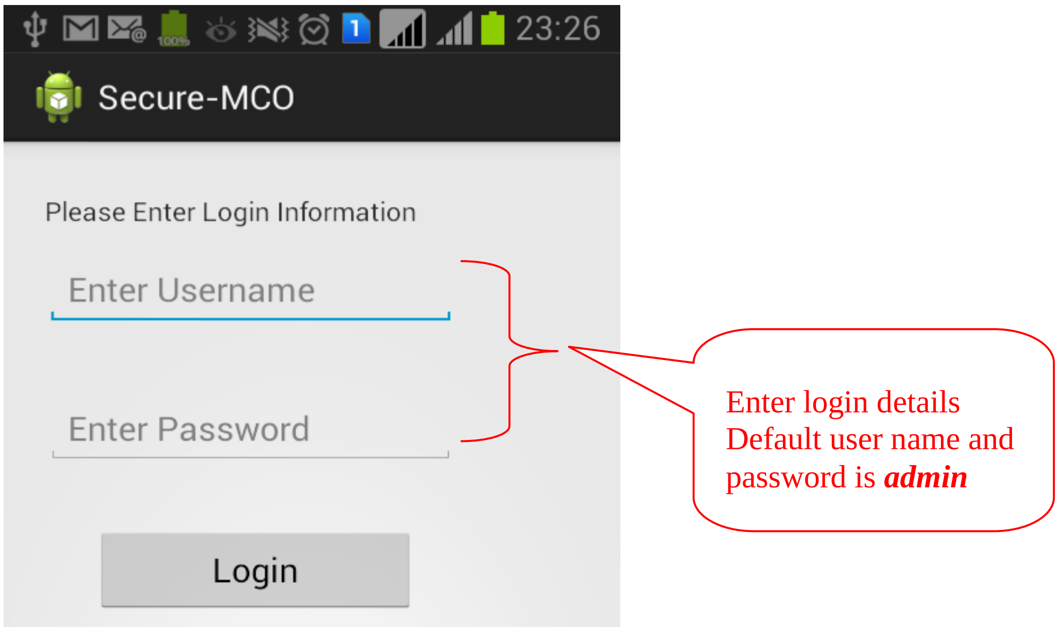
APPENDIX F: USER MANUAL

Mobile phone application

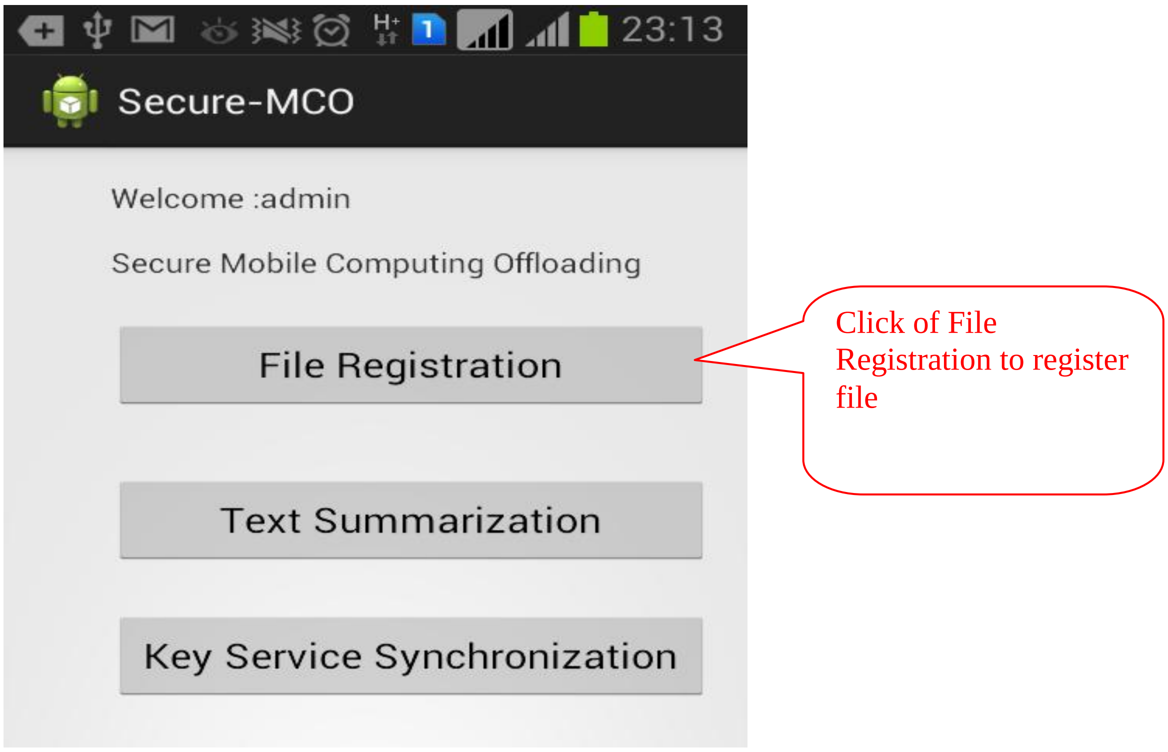
To access the system click on the **Secure-MCO** icon illustrated below



The login interface will be launched.


23:26
Secure-MCO
Please Enter Login Information
Enter Username
Enter Password
Login
Enter login details
Default user name and password is ***admin***

After entering the login details the home page below is displayed. Click on File Registration to encrypt a word file and register it for extrative text summarization from the pop up below.

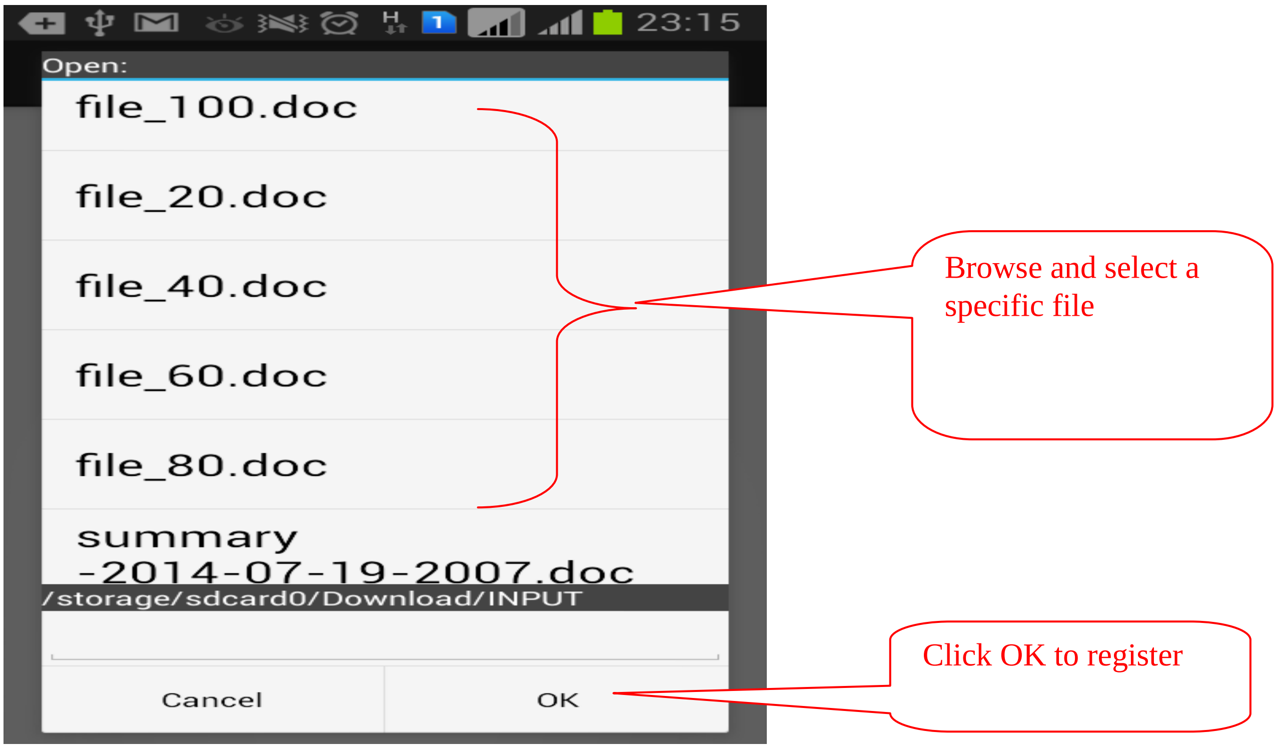


Secure-MCO

Welcome :admin

Secure Mobile Computing Offloading

Click of File Registration to register file



Open:

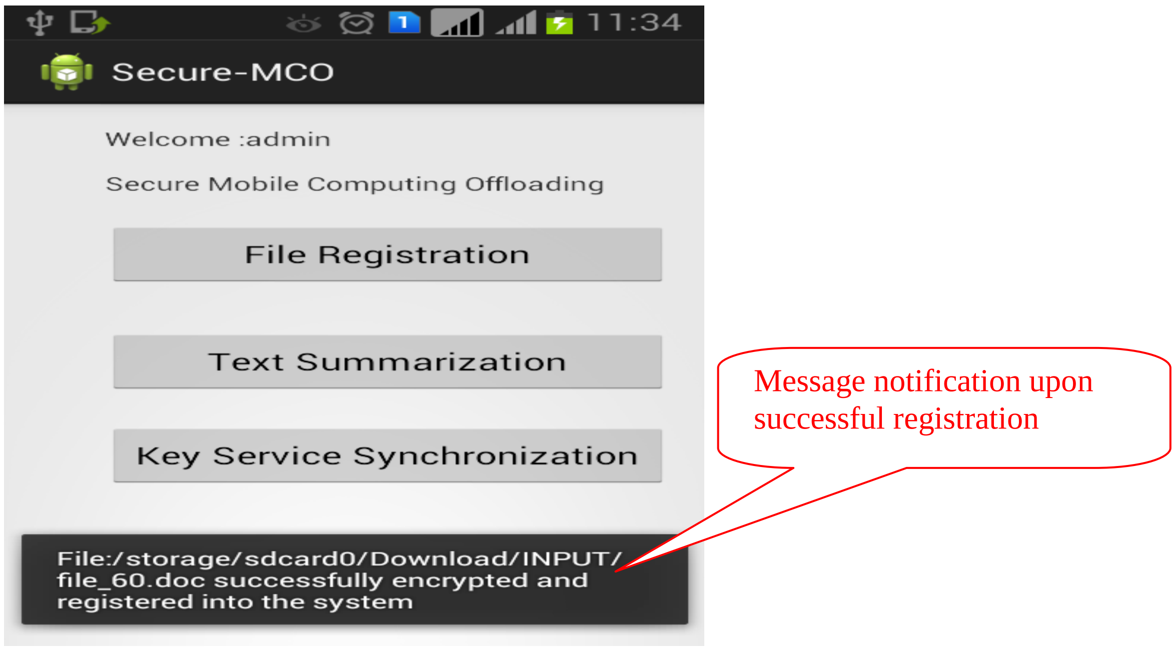
file_100.doc
file_20.doc
file_40.doc
file_60.doc
file_80.doc
summary -2014-07-19-2007.doc

/storage/sdcard0/Download/INPUT

Browse and select a specific file

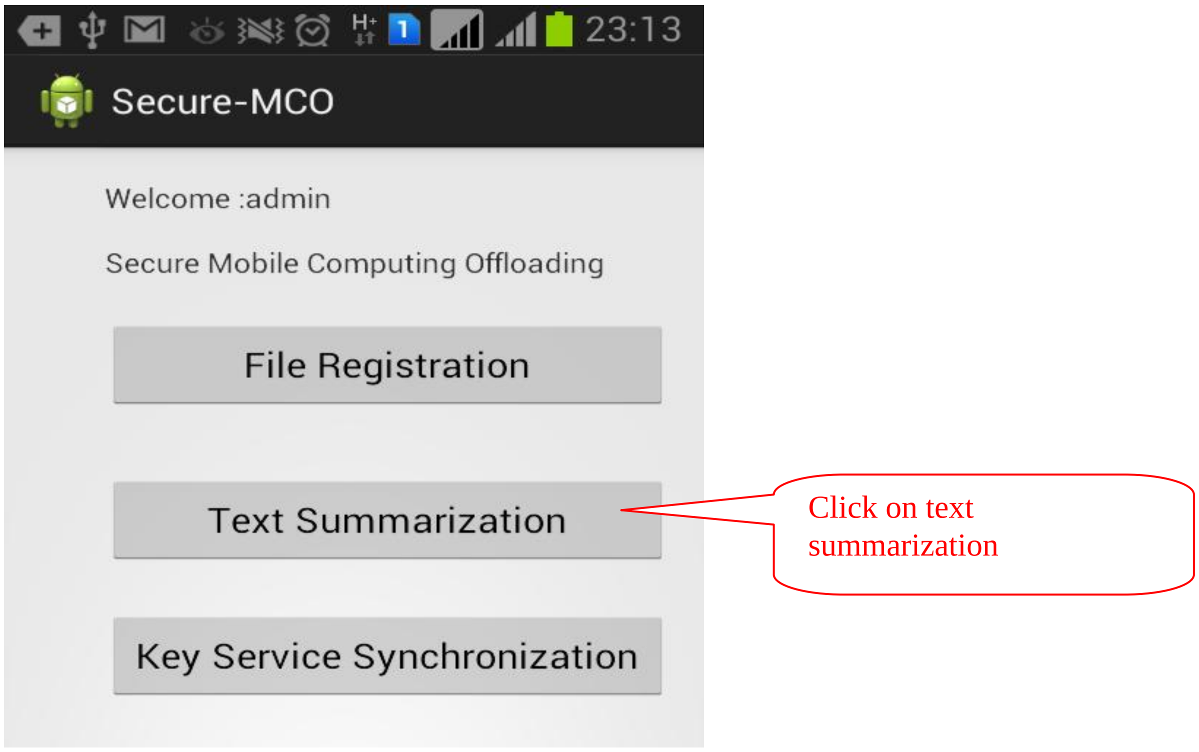
Click OK to register

Upon successful registration of the file the a notification will be displayed as follows.

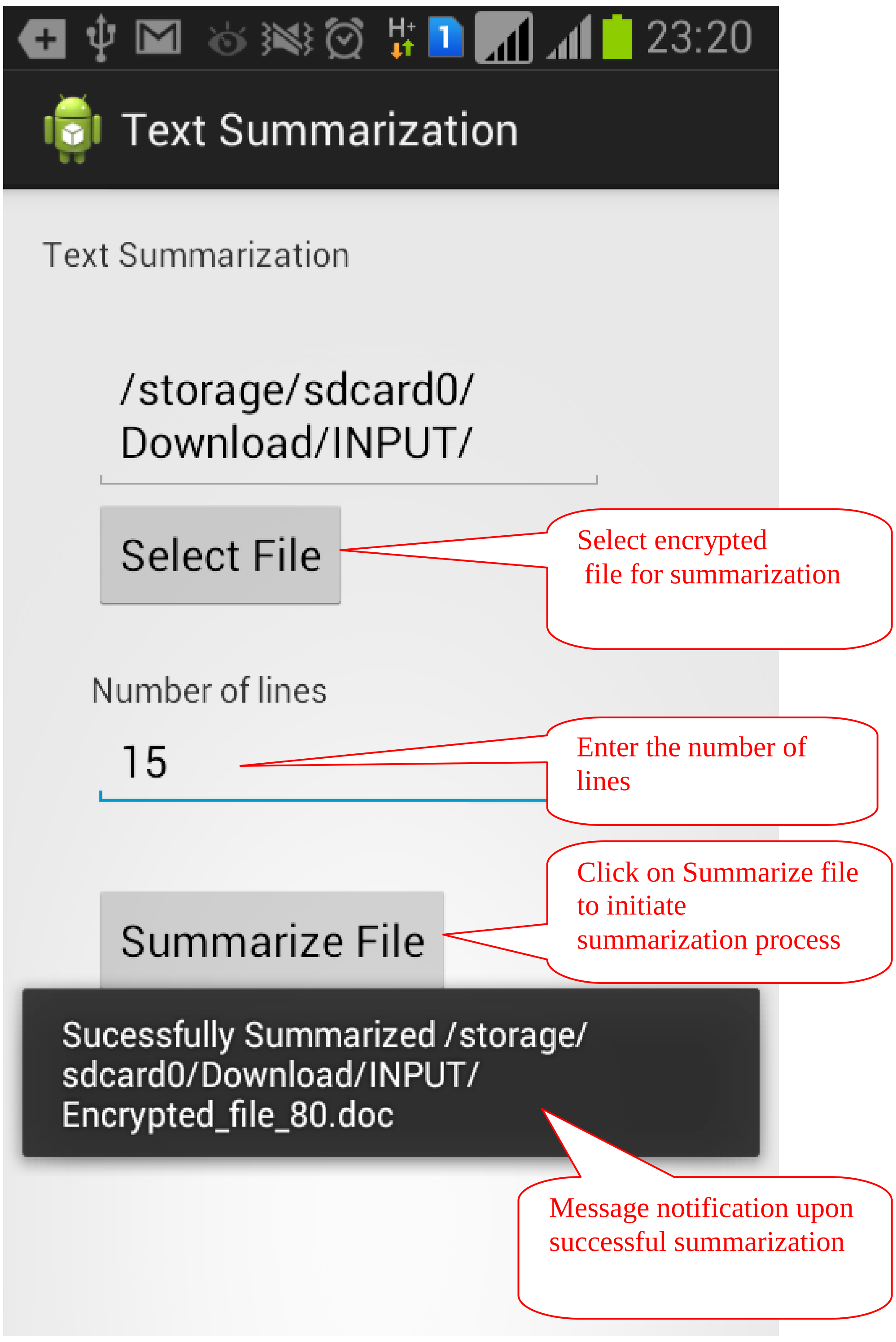

11:34
Secure-MCO
Welcome :admin
Secure Mobile Computing Offloading
File Registration
Text Summarization
Key Service Synchronization
Message notification upon successful registration

File:/storage/sdcard0/Download/INPUT/file_60.doc successfully encrypted and registered into the system

To summarize the text file click on Text summarization button.


23:13
Secure-MCO
Welcome :admin
Secure Mobile Computing Offloading
File Registration
Text Summarization
Key Service Synchronization
Click on text summarization

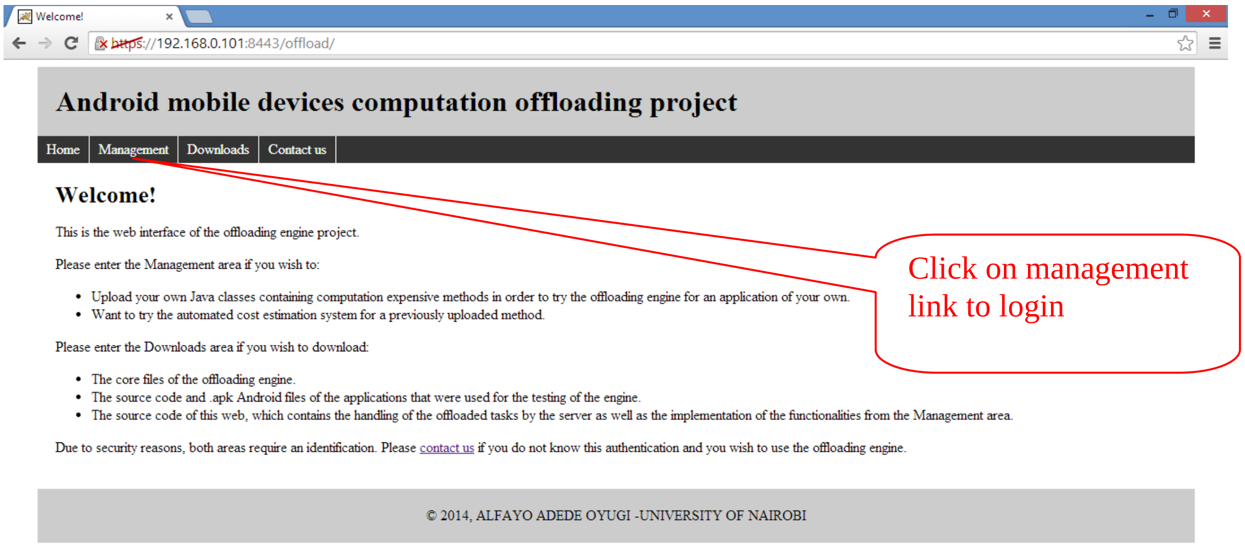
The following page will be displayed


23:20
Text Summarization
Text Summarization
/storage/sdcard0/Download/INPUT/
Select File
Select encrypted file for summarization
Number of lines
15
Enter the number of lines
Summarize File
Click on Summarize file to initiate summarization process
Message notification upon successful summarization

Sucessfully Summarized /storage/sdcard0/Download/INPUT/Encrypted_file_80.doc

Web application module

To access the web application module from the web browser enter following URL: https://192.168.0.101:8443/offload/. The interface below is displayed.



https://192.168.0.101:8443/offload/

Android mobile devices computation offloading project

Home | Management | Downloads | Contact us

Welcome!

This is the web interface of the offloading engine project.

Please enter the Management area if you wish to:

- Upload your own Java classes containing computation expensive methods in order to try the offloading engine for an application of your own.
- Want to try the automated cost estimation system for a previously uploaded method.

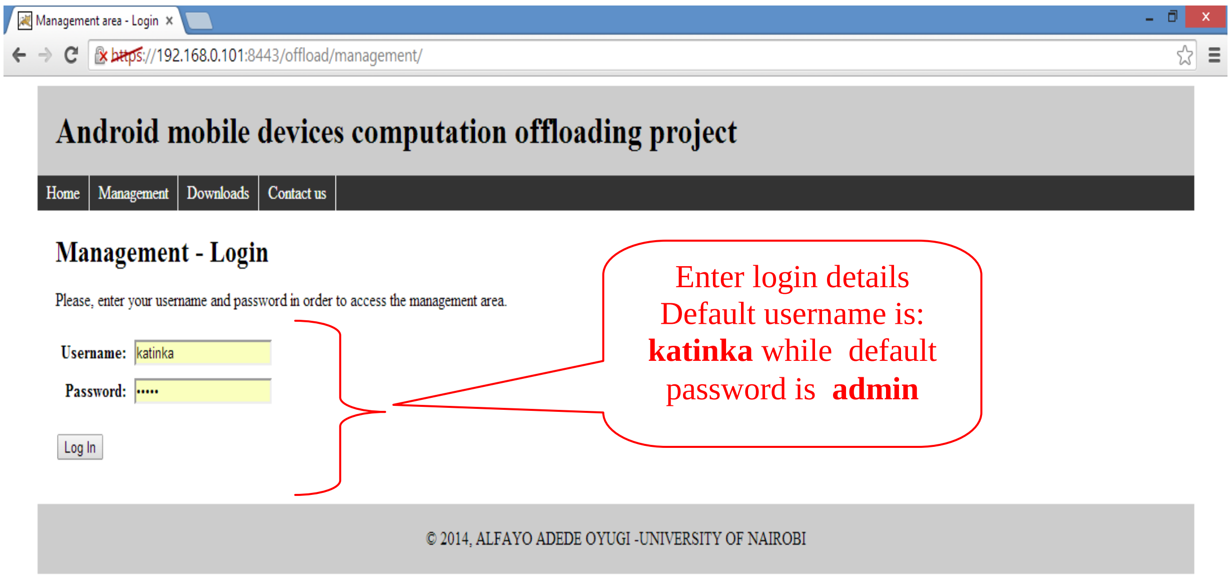
Please enter the Downloads area if you wish to download:

- The core files of the offloading engine.
- The source code and .apk Android files of the applications that were used for the testing of the engine.
- The source code of this web, which contains the handling of the offloaded tasks by the server as well as the implementation of the functionalities from the Management area.

Due to security reasons, both areas require an identification. Please contact us if you do not know this authentication and you wish to use the offloading engine.

© 2014, ALFAYO ADEDE OYUGI -UNIVERSITY OF NAIROBI

Click on Management to login



https://192.168.0.101:8443/offload/management/

Android mobile devices computation offloading project

Home | Management | Downloads | Contact us

Management - Login

Please, enter your username and password in order to access the management area.

Username: katinka

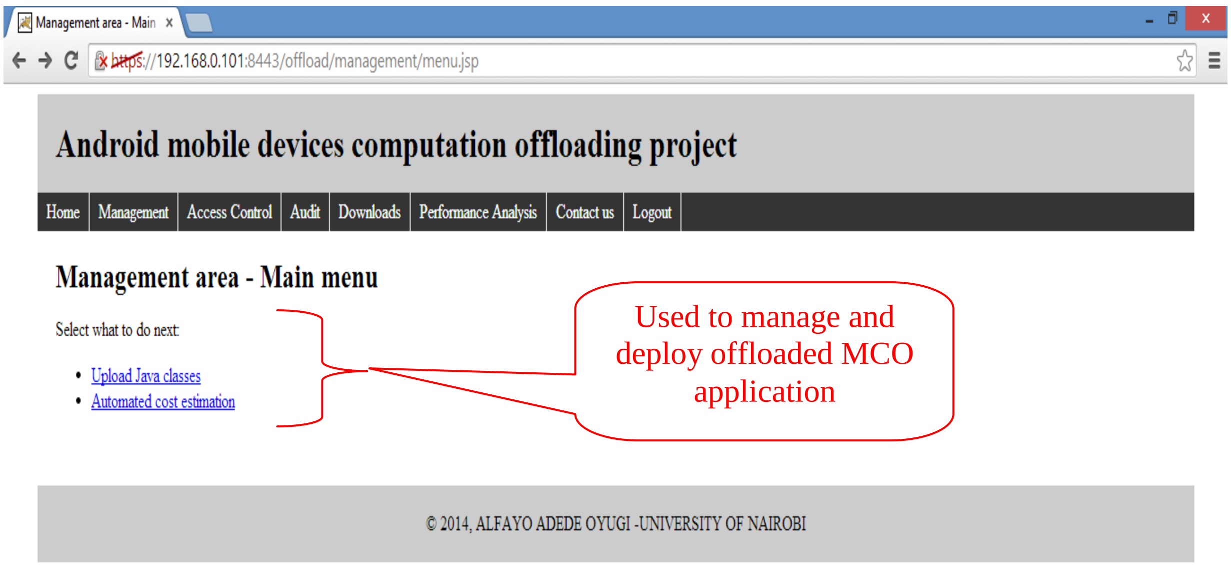
Password: •••••

Log In

© 2014, ALFAYO ADEDE OYUGI -UNIVERSITY OF NAIROBI

Management area - Login
Enter login details Default username is: **katinka** while default password is **admin**

Upon Sucessful login the following page is displayed


Management area - Main
https://192.168.0.101:8443/offload/management/menu.jsp
Home
Management
Access Control
Audit
Downloads
Performance Analysis
Contact us
Logout
Used to manage and deploy offloaded MCO application

Android mobile devices computation offloading project

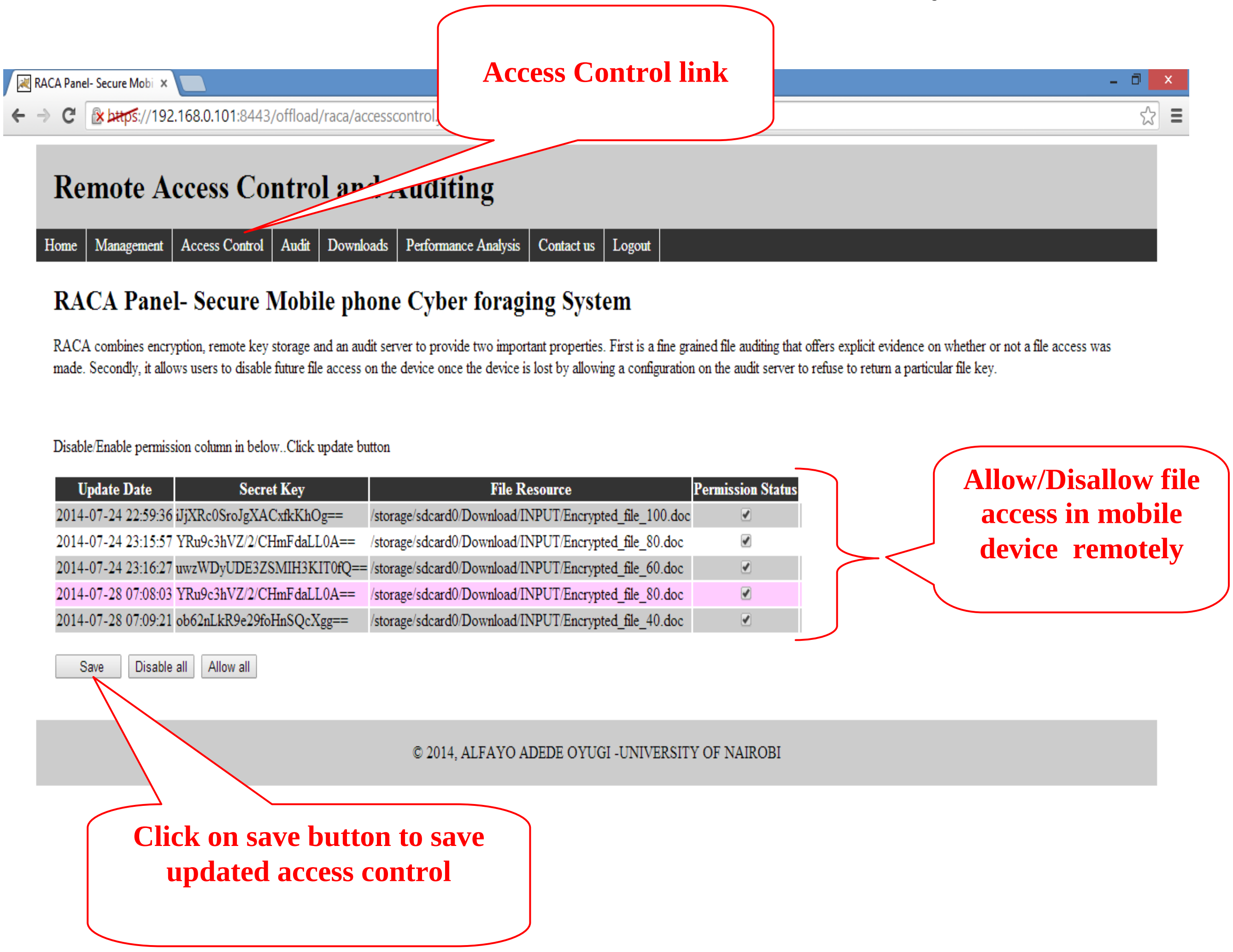
Management area - Main menu

Select what to do next:

- Upload Java classes
- Automated cost estimation

© 2014, ALFAYO ADEDE OYUGI -UNIVERSITY OF NAIROBI

Click on Access Control link to access Remote Access Control functionality


RACA Panel- Secure Mobi
https://192.168.0.101:8443/offload/raca/accesscontrol
Access Control link
Home
Management
Access Control
Audit
Downloads
Performance Analysis
Contact us
Logout
Allow/Disallow file access in mobile device remotely
Save
Disable all
Allow all
Click on save button to save updated access control

Remote Access Control and Auditing

RACA Panel- Secure Mobile phone Cyber foraging System

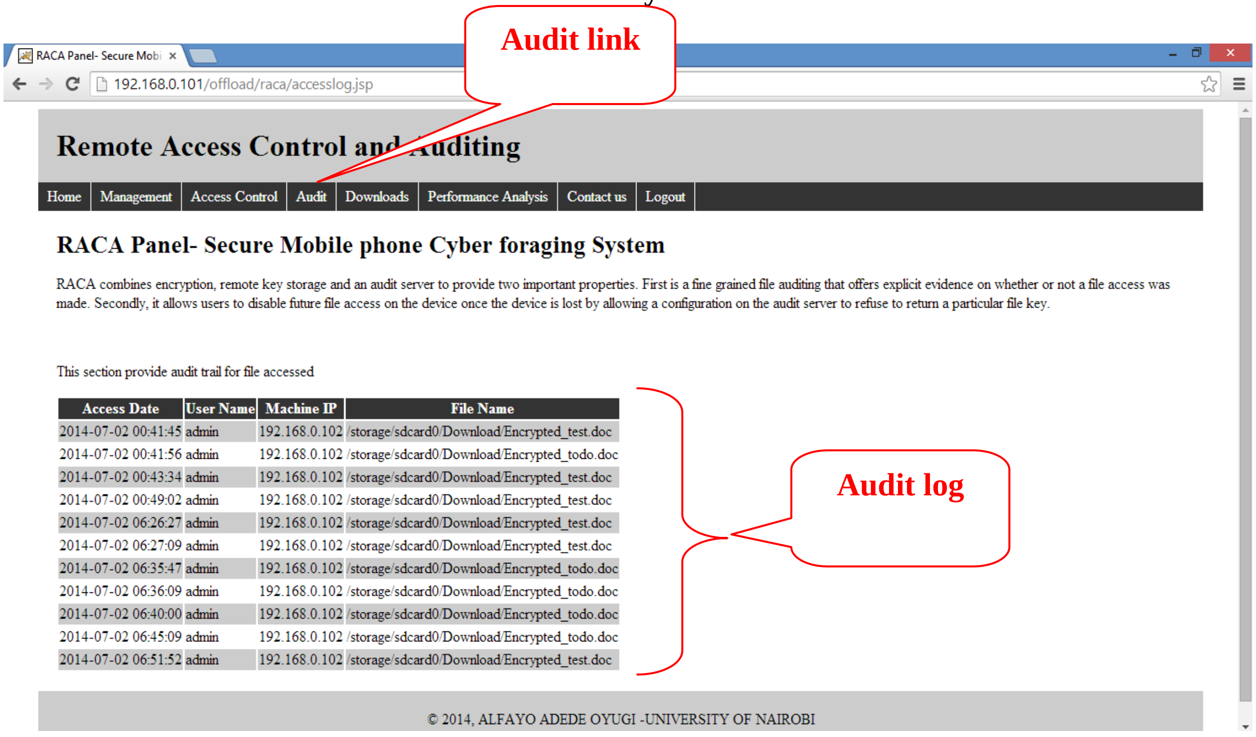
RACA combines encryption, remote key storage and an audit server to provide two important properties. First is a fine grained file auditing that offers explicit evidence on whether or not a file access was made. Secondly, it allows users to disable future file access on the device once the device is lost by allowing a configuration on the audit server to refuse to return a particular file key.

Disable/Enable permission column in below..Click update button

Update Date	Secret Key	File Resource	Permission Status
2014-07-24 22:59:36	iJjXRc0SroJgXACxfkKhOg==	/storage/sdcard0/Download/INPUT/Encrypted_file_100.doc	✓
2014-07-24 23:15:57	YRu9c3hVZ/2/CHmFdaLL0A==	/storage/sdcard0/Download/INPUT/Encrypted_file_80.doc	✓
2014-07-24 23:16:27	uwzWDyUDE3ZSMIH3KIT0fQ==	/storage/sdcard0/Download/INPUT/Encrypted_file_60.doc	✓
2014-07-28 07:08:03	YRu9c3hVZ/2/CHmFdaLL0A==	/storage/sdcard0/Download/INPUT/Encrypted_file_80.doc	✓
2014-07-28 07:09:21	ob62nLkR9e29foHnSQcXgg==	/storage/sdcard0/Download/INPUT/Encrypted_file_40.doc	✓

© 2014, ALFAYO ADEDE OYUGI -UNIVERSITY OF NAIROBI

Click on Audit link to access Audit functionality



Remote Access Control and Auditing

RACA Panel- Secure Mobile phone Cyber foraging System

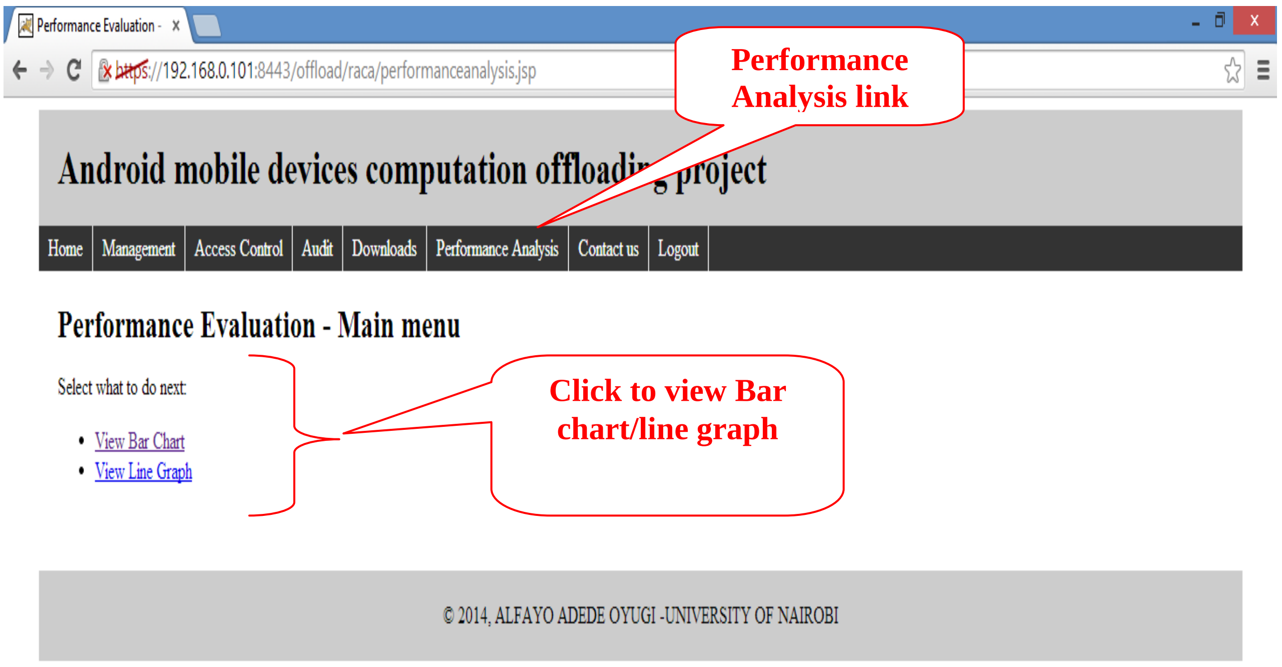
RACA combines encryption, remote key storage and an audit server to provide two important properties. First is a fine grained file auditing that offers explicit evidence on whether or not a file access was made. Secondly, it allows users to disable future file access on the device once the device is lost by allowing a configuration on the audit server to refuse to return a particular file key.

This section provide audit trail for file accessed

Access Date	User Name	Machine IP	File Name
2014-07-02 00:41:45	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_test.doc
2014-07-02 00:41:56	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_todo.doc
2014-07-02 00:43:34	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_test.doc
2014-07-02 00:49:02	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_test.doc
2014-07-02 06:26:27	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_test.doc
2014-07-02 06:27:09	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_test.doc
2014-07-02 06:35:47	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_todo.doc
2014-07-02 06:36:09	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_todo.doc
2014-07-02 06:40:00	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_todo.doc
2014-07-02 06:45:09	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_todo.doc
2014-07-02 06:51:52	admin	192.168.0.102	/storage/sdcard0/Download/Encrypted_test.doc

© 2014, ALFAYO ADEDE OYUGI -UNIVERSITY OF NAIROBI

Click on Performance Analysis to access Execution time evaluation functionality



Android mobile devices computation offloading project

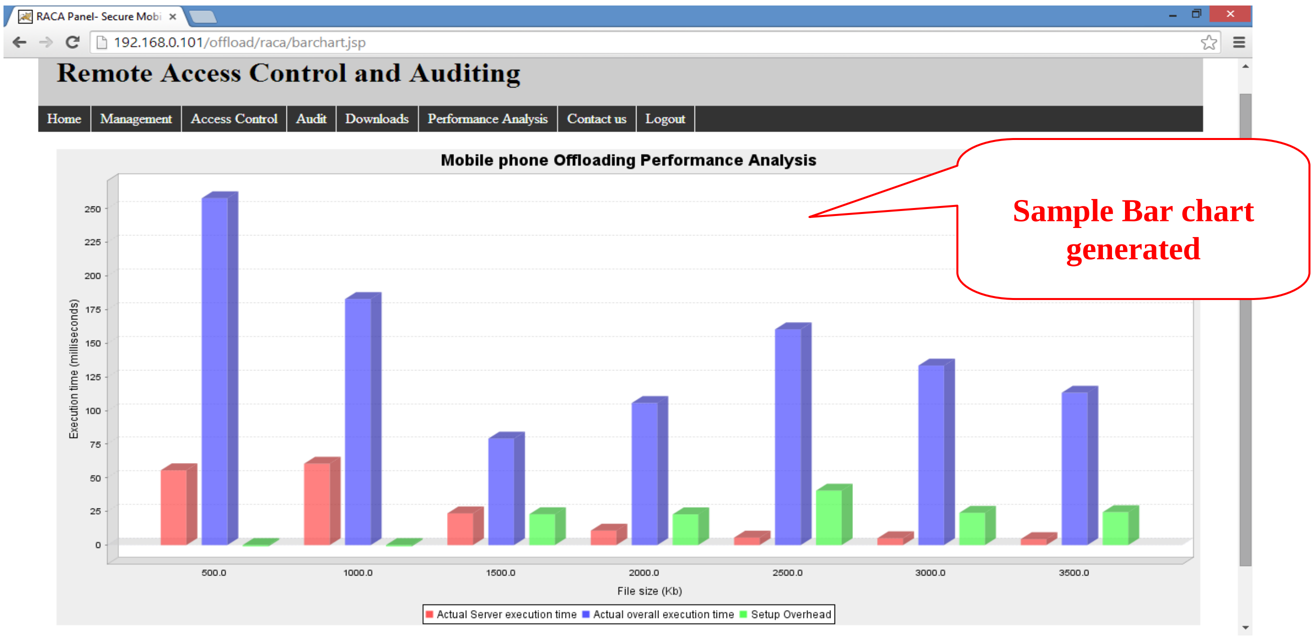
Performance Evaluation - Main menu

Select what to do next:

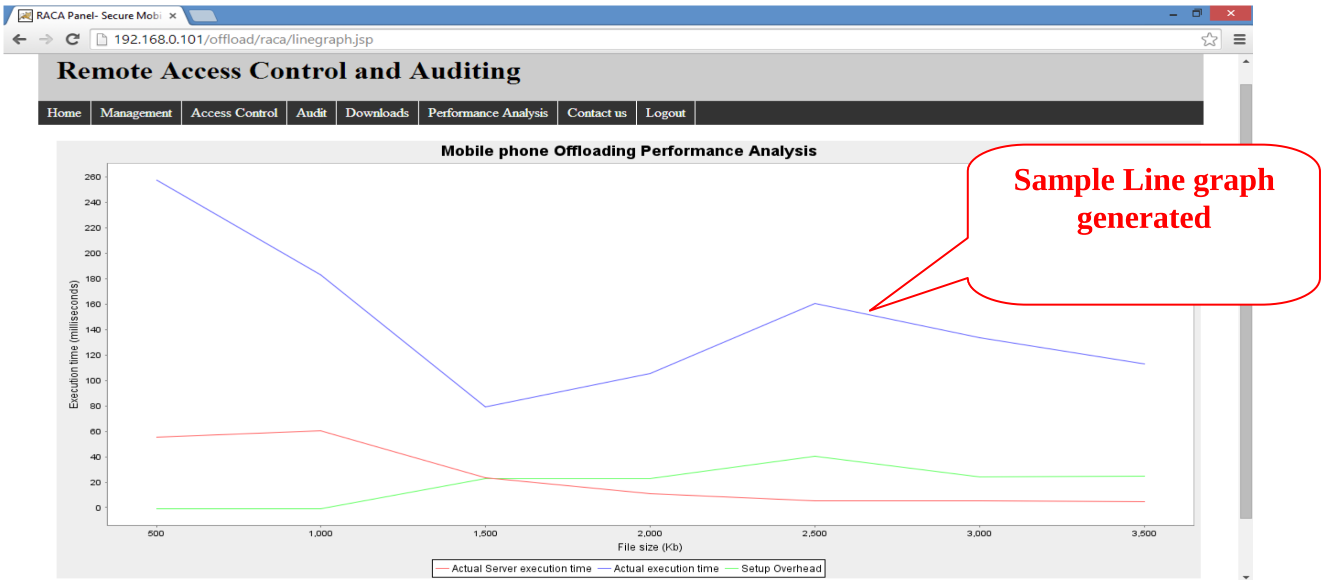
- View Bar Chart
- View Line Graph

© 2014, ALFAYO ADEDE OYUGI -UNIVERSITY OF NAIROBI

Upon clicking View Bar Chart the following chart is displayed


RACA Panel- Secure Mobi
192.168.0.101/offload/raca/barchart.jsp
Remote Access Control and Auditing
Home
Management
Access Control
Audit
Downloads
Performance Analysis
Contact us
Logout
Mobile phone Offloading Performance Analysis
Execution time (milliseconds)
File size (Kb)
Actual Server execution time
Actual overall execution time
Setup Overhead
Sample Bar chart generated

Upon clicking View Line Graph the following graph is displayed


RACA Panel- Secure Mobi
192.168.0.101/offload/raca/linegraph.jsp
Remote Access Control and Auditing
Home
Management
Access Control
Audit
Downloads
Performance Analysis
Contact us
Logout
Mobile phone Offloading Performance Analysis
Execution time (milliseconds)
File size (Kb)
Actual Server execution time
Actual execution time
Setup Overhead
Sample Line graph generated

APPENDIX G: PROJECT SCHEDULE AND RESOURCES REQUIREMENTS

<u>Project schedule</u>

Activities	**YEAR 2014**								**Progress**
	Jan	***Feb***	***March***	***April***	***May***	***June***	***July***	***Aug***	
Study and Review of Literature	■	■							**Done**
Project Proposal		■	■						**Done**
Milestone I: Proposal presentation			■						**Done**
Requirements Specification				■					**Done**
Analysis and Design				■	■				**Done**
Prototype development, Integration and Testing					■	■			**Done**
Milestone II: Progress report presentation						■			**Done**
Evaluation							■		**Done**
Final Report preparation							■	■	**Done**
Milestone III: Final report presentation								■	**Done**

Table 8: Project implementation schedule

Source: Author's compilation

<u>Resource requirement</u>

i. ***One Smart mobile phone device estimated at a cost of KSh 25,000/= with the following minimum technical specification will be used to test the prototype system.***

The Galaxy Grand Duos (GT-19082) on Android v4.1.2 (Jelly Bean) OS, 5-inch capacitive touch screen, with Wi-Fi support, expandable storage capacity of 64 GB and 1.2 GHz dual core processor.

ii. ***Two (2) desktop computers to act as servers estimated at a total cost of KSh 36,000/= with the following minimum technical specification will be used***

Processor: Intel (R) Pentium (R) Dual CPU, 1.86 GHz.

Main memory: 2.0 GB

Hard drive: 80 GB

Network: Wi-Fi support

Operating System: Windows 7.

Java version: 1.6 (free)

HTTP Server: Apache Tomcat 6 (open source)

iii. ***TP-LINK, 3G/4G Wireless N Router TL-MR3220* estimated at a cost of KSh 4,000/= with the following minimum technical specification will be used**

HARDWARE FEATURES	
Interface	USB 2.0 Port for LTE/HSPA+/HSUPA/HSDPA/UMTS/EVDO USB Modem 1 10/100Mbps WAN Port, 4 10/100Mbps LAN Ports, support the auto-Negotiation and auto-MDI/MDIX
Button	WPS/Reset Button Wireless On/Off Switch Power On/Off Button
External Power Supply	9VDC/0.85A
Dimensions (W x D x H)	8*5.4*1.7 in. (204*138*44mm)
Antenna Type	Omni directional, Detachable, Reverse SMA
Antenna Gain	5dBi
WIRELESS FEATURES	
Wireless Standards	IEEE 802.11n*, IEEE 802.11g, IEEE 802.11b
Frequency	2.4-2.4835GHz
EIRP	<20dBm
Wireless Security	Support 64/128 bit WEP, WPA-PSK/WPA2-PSK, Wireless MAC Filtering
Modulation Technology	DBPSK, DQPSK, CCK, OFDM, 16-QAM, 64-QAM
SOFTWARE FEATURES	
Security	NAT Firewall, SPI Firewall, MAC / IP / Packet / Application / URL Filtering, Denial of Service(DoS), SYN Flooding, Ping of Death
Management	Web Based Configuration(HTTP), Web Based Firmware Upgrade

≈≈ The End ≈ ≈