Data Integrity and Confidentiality in Mobile Phone Based Cyber Foraging System

Alfayo O. Adede[1*], William Okelo-Odongo[2], Christine K. Mulunda[3]

[1]University of Nairobi
School of Computing and Informatics,
Nairobi, Kenya
[*]Email: alfayaoyugi [AT] gmail.com

[2]University of Nairobi
School of Computing and Informatics,
Nairobi, Kenya

[3]University of Nairobi
School of Computing and Informatics,
Nairobi, Kenya

Abstract— **Following the emergence of cyber foraging systems (CFS); resource constrained mobile devices are able to offload resource intensive work to computationally powerful surrogate computers accessible on Wireless Local Area Network(WLAN). However, users in possession of confidential data remain hesitant in entrusting data to CFS due to fear of losing control over it. This is exacerbated by rudimental enforcement of confidentiality, integrity and availability computer security attributes in such systems. We examine the challenge of data integrity and confidentiality arising from using mobile phone based CFS. We implement and integrate data integrity and confidentiality enforcing mechanism based on Remote Access Control and Auditing (RACA) framework into an open source mobile phone based CFS prototype using use case approach. Experimental method is applied to measure and evaluate execution time overhead cost attributed to RACA integration. Our experimental result demonstrates that RACA integration not only enhances data integrity and confidentiality but also pose an insignificantly compromise on task offloading execution time. However, to further enhance confidentiality and availability attributes; we recommend use of SSL protocol for data transmission and deployment of surrogate computers through defensive responses to Denial of Service (DoS) attacks.**

Keywords--- Cyber foraging systems, data integrity, data confidentiality

I. INTRODUCTION

Satyanarayanan (2001) defined cyber foraging system as opportunistic use of available computing resources by computationally deprived devices such as mobile phone devices that offload resource intensive work to computationally powerful surrogate computers within WLAN. However, when no surrogates is available the device should solve tasks own its own albeit with degraded execution runtime.

Mobile phoned based CFS envisioned a scenario in which when a mobile phone device enters a neighbourhood, 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 neighbourhood. When mobile phone device leaves the neighbourhood, its surrogate bindings are broken, and any data staged or cached on its behalf are discarded.

A. Motivation for Cyber foraging

Mobile computing devices, such as smart phone, are not only ubiquitous but also quickly advancing to provide full-fledged personal computing due to rapid improvement in their display size, network connectivity and input methods as predicted by Barton et al. (2006). However, relatively powerful, mobile phone 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. 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) as illustrated in figure 1.

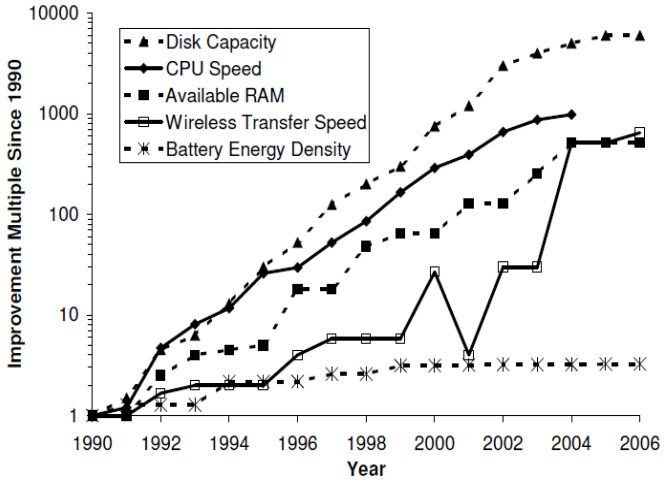


Fig.1: Mobile devices processing trend
Source: Adapted from Paradiso and Starner (2005)

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 CFS that has gained popularity in recent years is mobile cloud computing. However, its failure to address monetary cost, network latency, internet bandwidth and energy efficiency challenges continues to inspire interest in CFS as a possible alternative. Notwithstanding the aforementioned challenges; Satyanarayanan et al.(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.

B. Data integrity and confidentiality

Avizienis et al. (2004) defined computer security as a composite confidentiality, integrity and availability (also called CIA) attributes. A system is considered secure when all of these three attributes is maximized. This paper focuses on implementing data integrity and confidentiality attributes in mobile phone based cyber foraging system. The availability attribute is outside the scope of this study.

C. Problem statement

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.

D. Proposed solution

To address the challenge of data integrity and confidentiality we implement 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. Figure 2 below illustrate security threats facing cyber foraging architecture, figure 3 illustrates an integration of RACA framework in a typical cyber foraging while figure 4 depict a conceptual model of RACA framework.

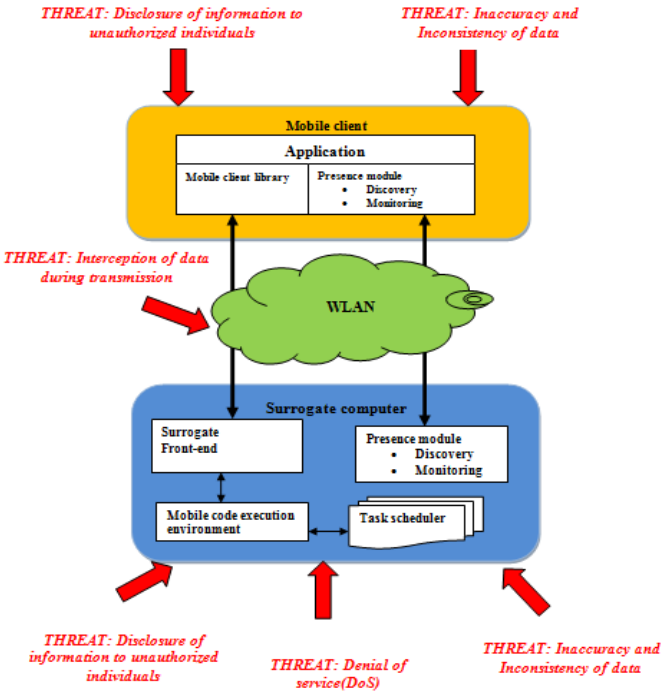


Fig. 2: Security threats facing cyber foraging
Source: Author's compilation

On the mobile client device, each file **F** has a unique identifier ($\mathbf{ID_F}$) and the file's data is encrypted with a unique symmetric key, $\mathbf{K_F}$. 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 ($\mathbf{M_F}$) information includes a file's path, the process that created it, and the file's extended attributes.

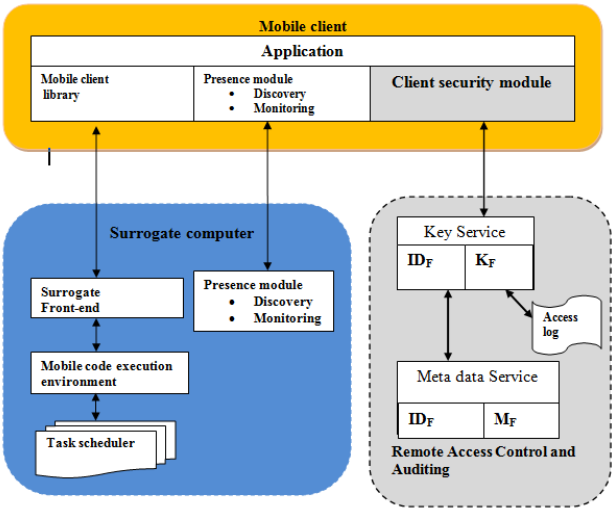


Fig. 3: RACA framework integration

Source: Author's compilation

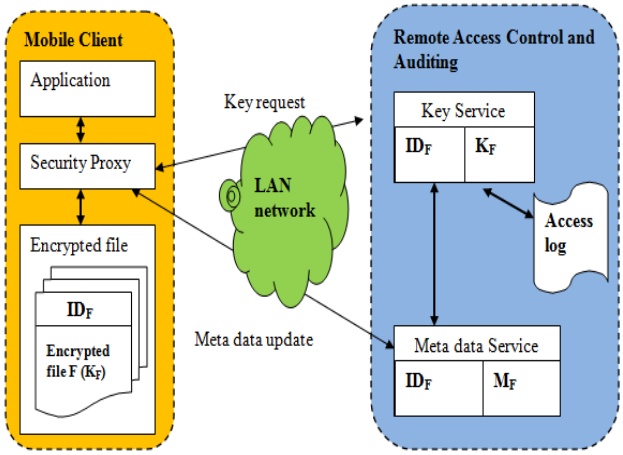


Fig. 4 : RACA framework
Source: Adapted from Geambasu et al. (2011)

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.

Apart from Geambasu et al. (2011), Rahim and Saravanan (2013) also applied RACA framework 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.

II. LITERATURE REVIEW

According to Kumar et al. (2012), studies on cyber foraging systems has evolved considerably and can be traced from the year 1996. 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. As from 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. Though the trend indicates considerable efforts have been 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.

A. 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.

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 research objectives.

B. Remote Access Control and Auditing (RACA)

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. 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.

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.

Distributed /Network file systems (NFS) allow a server to share directories and files with clients over a network as if they are stored locally. Their notable benefit 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. 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.

III. METHODOLOGY

This study 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.

The first task was fulfilled 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. (1992). Finally, to evaluate the resulting execution overhead cost attributed to the integration of RACA framework an experimental methodis used with extractive text summarization application adopted as a test case scenario.

A. RACA framework implementation

Implementation of RACA framework to mitigate on security threats facing mobile phone based cyber foraging highlighted in figure 2, 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.

Step 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 5 illustrates these steps.

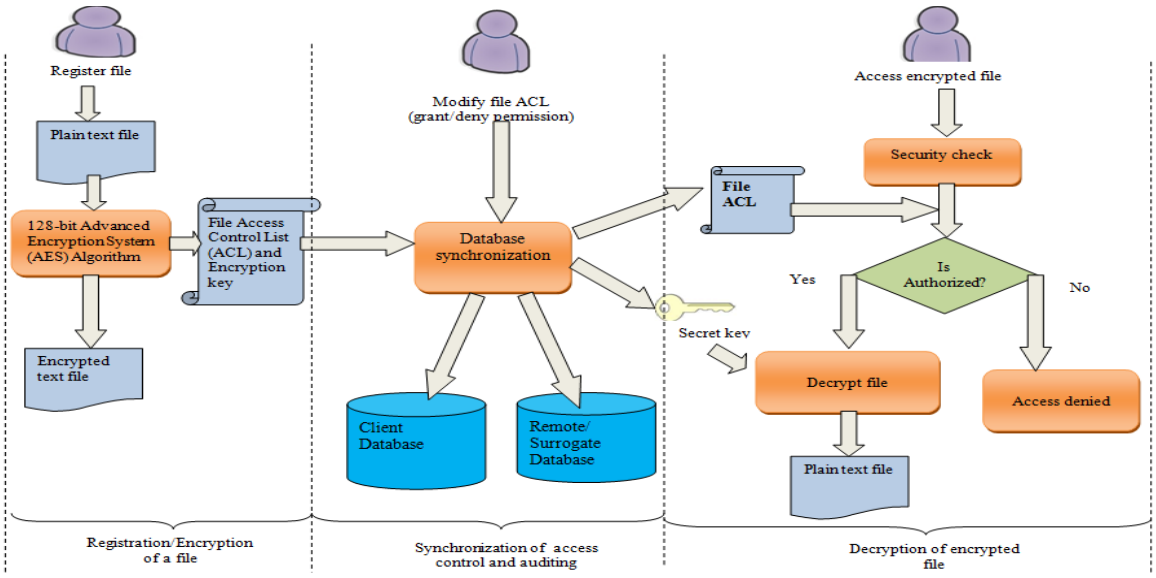


Fig. 5 : RACA process
Source: Author's compilation

It is worth pointing out that RACA framework presented in figure 2 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.

B. *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 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.

C. *Analysis and interpretation results*

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.

IV. PROTOTYPE SYSTEM IMPLEMENTATION

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. Application Architecture as illustrated in figure 6 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 Architectureillustrates 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.

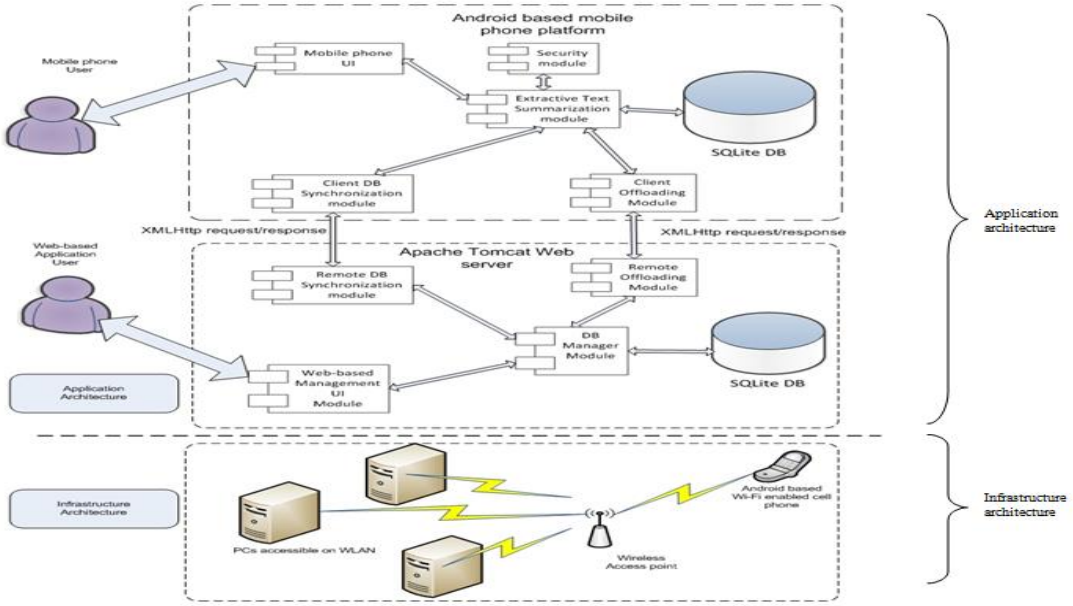


Fig. 6 : Overall system design
Source: Author's compilation

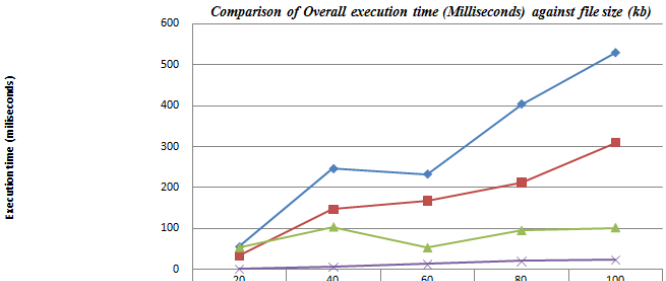
Mobile phone application was implemented on Android platform v4.1.2 (Jelly bean) while the server side was implemented on Java EE6 web platform (based on Servlet 2.5 API). Java programming language was used to implement virtually the whole prototype except user interface of Web-management module that relied on JSP, JSTL, JavaScript and HTML for dynamic web pages implementation.

V. RESULTS AND EVALUATION

Whereas main outcome of the study was an implementation of **Secure-MCO prototype system**, experimental results of execution time conducted is tabulated in table 1 while figure 7 and 8 illustrate bar chart and corresponding line graph.

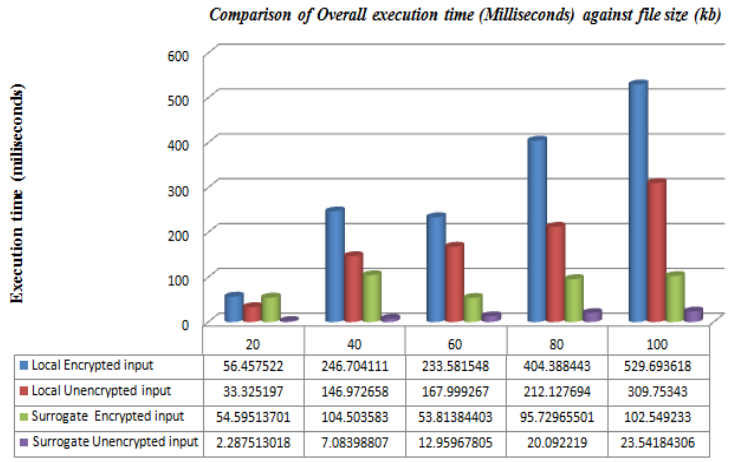
Overall execution time (Milliseconds)				
Input file size (Kb)	**Local/Mobile phone based execution**		**Offloaded/Surrogate computer execution**	
	Encrypted	**Unencrypted**	**Encrypted**	**Unencrypted**
20	56.4575220048	33.3251969963	54.5951370	2.287513018
40	246.704110994	146.9726580083	104.503583	7.08398807
60	233.581547990	167.9992669969	53.8138440	12.95967805
80	404.388442993	212.1276939958	95.7296550	20.092219
100	529.6936179	309.7534299939	102.549233	23.54184306

Table1: Overall execution time (milliseconds)
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

Fig. 7 : 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

Fig. 8 : Overall execution time bar chart
Source: Author's compilation

Using extractive text summarization as a use case scenario in the prototype system, the results indicates that execution time overhead arising from the integration is minimal hence integration of RACA into MCO is deemed feasible.

A. 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.

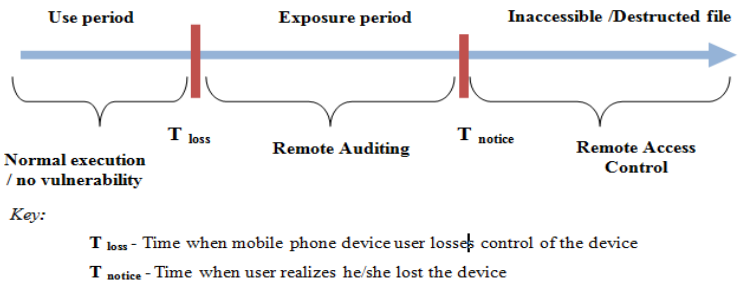


Fig. 9 : Timeline of theft of mobile phone device

Source: Author's compilation

Illustrated above in figure 9 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 $\mathbf{T}_{\mathbf{loss}}$ and $\mathbf{T}_{\mathbf{notice}}$ 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.

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.

B. Recommendations

Our solution enforces confidentiality and integrity attributes using RACA framework. However, the framework exhibits several in adequacies such as lack of transport security during data 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.

VI. CONCLUSION

RACA framework provides users with evidence on weather sensitive data/file was accessed or not. If file was accessed, it gives the provide users with an audit log indicating the same. It also allows users to disable file access on lost devices. This goals were achieved through the 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 (1) Application of RACA framework in mobile phone based cyber foraging to enhance data integrity and confidentiality and (2) Evaluation of execution time overhead cost attributed to integration of RACA mechanism in mobile phone based cyber foraging.

REFERENCES

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

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

[3] Casey E., & Stellatos G. J.. "The impact of full disk encryption on digital forensics" . *ACM SIGOPS Operating Systems Review*, 2008, *42*(3), 93-98.

[4] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra and P. Bahl. "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, 2010, 49-62.

[5] H. Y. Chen, Y. H. Lin, and C. M. Cheng. "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.

[6] I. Giurgiu, O. Riva, D. Juric, I. Krivulev and G. Alonso. "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, 2009, 83-102.

[7] J. Flinn, D. Narayanan and M. Satyanarayanan. "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, 2001, 61-63.

[8] J. J. Barton, S. Zhai, and S. B. Cousins. "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.

[9] K. Kumar, J. Liu, Y. Lu and B. Bhargava "A Survey of Computation Offloading for Mobile Systems", *Springer Science Business Media, LLC*, 2012

[10] M. Griera. "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, 2013.

[11] M. Satyanarayanan. "Pervasive computing: vision and challenges. *Personal Communications IEEE", 2001, Vol.* 8(4):10–17.

[12] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. "The case for VM based cloudlets in mobile computing". *IEEE Pervasive Computing*, 2009, 8:14–23,

[13] M.S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao and X. Chen. "COMET: Code Offload by Migrating Execution Transparently". *In Proceedings of OSDI*, 2012

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

[15] Mads D. K.. "Empowering Mobile Devices Through Cyber Foraging: The Development of Scavenger, an Open, Mobile Cyber Foraging System" Ph.D dissertation, Faculty of Science, Aarhus University, 2010.

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

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

[18] R. K. Balan, M. Satyanarayanan, S.Y Park and T. Okoshi. "Tactics-based remote execution for mobile computing". *In Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys 2003).* ACM, New York, NY, USA, 2003, 273-286.

[19] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. "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.

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

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

[22] T. Verbelen. "Adaptive Offloading and Configuration of Resource Intensive Mobile Applications" PhD dissertation, Faculty of computer Science, University Kent, 2013.