NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CLANDESTINE TRANSMISSIONS AND OPERATIONS OF EMBEDDED SOFTWARE ON CELLULAR MOBILE DEVICES

by

Kalle G. Kangas

September 2011

Thesis Co-Advisors: Craig Martell
Robert Beverly

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2011	Master's Thesis

4. TITLE AND SUBTITLE
Clandestine Transmissions and Operations of Embedded Software on Cellular Mobile Devices

5. FUNDING NUMBERS

6. AUTHOR(S) Kalle G. Kangas

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

In this thesis, we develop a method to offload data in a clandestine fashion from an Android cellular mobile device. Due to the Short Message Service (SMS) message system's reliability and widespread availability, it is ideally suited as the vehicle through which to conduct data transmissions. This thesis found that using a transmission rate of one SMS message every ten seconds, combined with a total file size of 13.53 KB, produced a successful data file delivery rate of 100 percent.

14. SUBJECT TERMS SMS, Android, transmission rates, clandestine, inconspicuous, stealth, covert, message, cellular, mobile, phone

15. NUMBER OF PAGES 127

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

CLANDESTINE TRANSMISSIONS AND OPERATIONS OF EMBEDDED SOFTWARE ON CELLULAR MOBILE DEVICES

Kalle G. Kangas
Captain, United States Marine Corps
B.S., University of California, Los Angeles, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2011

Author: Kalle G. Kangas

Approved by: Craig Martell
Thesis Co-Advisor

Robert Beverly
Thesis Co-Advisor

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, we develop a method to offload data in a clandestine fashion from an Android cellular mobile device. Due to the Short Message Service (SMS) message system's reliability and widespread availability, it is ideally suited as the vehicle through which to conduct data transmissions. This thesis found that using a transmission rate of one SMS message every ten seconds, combined with a total file size of 13.53 KB, produced a successful data file delivery rate of 100 percent.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

3G	3rd Generation
3GPP	3rd Generation Partnership Project
ASCII	American Standard Code for Information Interchange
BCH	Broadcasting Channels
BSC	Base Station Controller
BSS	Base Station Subsystem
BTS	Base Transceiver Station
CCCH	Common Control Channels
CCH	Common Channels
CID	Cell Identifiers
CPU	Central Processor Unit
CSMS	Concatenated Short Message Service
DCCH	Dedicated Control Channels
DCH	Dedicated Channels
EIR	Equipment Identity Register
EMS	Enhanced Message Service
FDMA	Frequency Division Multiple Access
GMT	Greenwich Mean Time
GMSC	Gateway Mobile Switching Center
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HLR	Home Location Register
IED	Information Element Data
IEI	Information Element Identifier
IEIDL	Information Element Identifier Data Length
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
KB	Kilo Byte
LAC	Location Area Code

LAI	Location Area Identifier
MCC	Mobile Country Code
ME	Mobile Equipment
MNC	Mobile Network Code
MS	Message Station
MSC	Message Switching Center
NATO	North Atlantic Treaty Organization
NSS	Network Switching Subsystem
OpenBTS	Open Base Transceiver Station
PBX	Private Branch Exchange
PDU	Protocol Description Unit
PLMN	Public Land Mobile Network
POTS	Plain Old Telephone Service
PSTN	Public Switched Telephone Network
RF	Radio Frequency
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SM-MO	Short Message - Message Origination
SM-MT	Short Message - Message Termination
SMS	Short Message Service
SMSC	Short Message Service Center
SMS-GMSC	SMS-Gateway Message Switching Center
SMS-IWMSC	SMS-Interworking Mobile-Service Switching Center
TCH	Traffic Channels
TDMA	Time Division Multiple Access
TP	Transfer Layer Protocol
TPDU	Transfer Protocol Data Unit
TP-MTI	TP Message Type Indicator
TP-UD	TP User Data
TP-UDH	TP User Data Header
TP-UDHI	TP User Data Header Indicator
TP-UDHL	TP User Data Header Length

UE	User Equipment
UI	User Interface
USRP	Universal Software Radio Peripheral
UTC	Universal Coordinated Time
VLR	Visitor Location Register
VM	Virtual Machine
VoIP	Voice over Internet Protocol

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge the many people who helped make this work possible. I would like to thank my advisors, professors Robert Beverly and Craig Martell. This work would not have been possible without their guidance, insight, and professional expertise.

I would like to thank Charles Prince. His technical expertise and professional acumen were fundamental to the development of many of the technical products utilized in this research.

I would also like to thank my lab mates Captain James Browne, Dylan Freedman, and Marcy Schaeffer. Captain Browne's assistance and motivation were quintessential in setting up the lab equipment. Dylan's technical knowledge was beyond reproach and served as a valuable asset while building many of the products employed by this research. Marcy's critical eye was fundamental during the revising of my thesis.

Lastly and most importantly, I would like to thank my lovely wife, Shawna. This research would not have been possible without her patience, support, and understanding.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Cell phone use has become mainstream in modern society. According to the "2010 Nielsen Media Fact Sheet," there are 223 million mobile phone users over the age of 13 in the United States [1]. Also, there are 302.9 million wireless subscriptions across all United States territories, implying that 96 percent of the population has a mobile device, an increase of 69 percent from 2005 [2]. It is estimated that 26.6 percent of American households have abandoned the traditional land line and depend entirely on wireless technologies for telecommunication [2]. Globally, there are over 5 billion cell phone users [3].

Mobile cellular message services are also increasing in popularity. In 2010, 2.1 trillion Short Message Service (SMS) messages were sent in the United States [2]. Globally, 6.9 trillion SMS messages were sent in 2010, and SMS usage has experienced a 47 percent annual increase [4]. Mobile e-mail-based service usage has also gained in popularity. 480.6 million people used mobile e-mail based services in 2010; mobile e-mail based services usage is expected to quadruple by 2015 [4]. Mobile messaging services will be an integral part of the global communication network in the foreseeable future.

A. EMPLOYING MOBILE DEVICES TO MONITOR PERSONS OF INTEREST

People have become more dependent upon mobile devices to perform both everyday and work related tasks. From children texting friends, to employees coordinating tasks

with other employees or businesses, mobile devices have made these tasks relatively easy and quick. Yet, since mobile devices are employed in a more versatile and dynamic fashion than a personal computer, they are more difficult for parents and supervisors to monitor. All parents should be able to monitor their children's communications in order to ensure they are safe from malicious people and to be able to monitor their children's activities. Likewise, employers and supervisors should be able to monitor their employees' communications on company-owned devices to ensure they are acting in the best interest of the company.

Monitoring communications on a mobile device is a three step process. First, communications should be intercepted and gathered in a near real-time fashion to allow the device owners to respond to events in a timely manner. Second, due to mobile device's central processing unit limitations, the gathered communications must be transferred to a central collection site for monitoring or follow-on processing. Lastly, the communications must be interpreted and processed to determine their intent or subject.

B. RESEARCH QUESTIONS

This thesis addresses the issue of transferring data from an Android-based mobile device in an inconspicuous near real-time fashion. Wi-Fi and data coverage are often limited to urban areas. Furthermore, many mobile devices are not equipped with the ability to use Wi-Fi or other data channels. These considerations make SMS messages the ideal vehicle to conduct the proposed data transfer because SMS messages travel as message traffic in the GSM Control

Channel, a channel that always exists between a mobile device and its associated GSM network.

The adversary-model is a device user who can only monitor the mobile device through their primary senses. This research is not concerned with encrypting the transmitted data or evading passive observers. Additionally, it is assumed that the data transfer is conducted with full cooperation from the GSM service provider, and therefore, security at the various network nodes to include the base stations is a benign issue. Several concerns that must be addressed include: the offloading of data to a remote site, warning messages that might alert the cell phone user to the presence of covert communications (e.g., the "A large number of SMS messages are being sent…" warning), Central Processor Unit (CPU), and other mobile device resource restrictions.

Several assumptions have been made during the course of this research. First, we assume the monitoring and data offloading is performed in cooperation with the mobile device service GSM provider, and that the service provider has allowed the collection of data at the Short Message Service Centre (SMSC). Second, we assume that the monitoring software on the phone was loaded by the service provider or the software was installed by the owner of the device through the Android Application Market. Third, we assume the user will not attempt to re-program the device. Finally, we assume that the owner and user do not have root access and cannot disable any warning messages.

This thesis aims to evaluate the SMS channels' throughput and overhead to offload 1 KB, 10 KB, and 100 KB

data volumes. In doing this, we will evaluate the channel transmission success rates; the time needed for the phone to transmit the data, and the time it takes for the server to receive the data.

C. SIGNIFICANT FINDINGS

This research produced the following significant results:

- Transmitting data files less than or equal to 1 KB in size at a rate of one SMS message every second yielded a successful data file delivery rate of 100 percent. At this transmission rate and data file size, it is recommended that no more than eleven one KB data files be transmitted per hour with at least a 90 seconds transmission pause between each 1 KB data file.
- Due to Short Message Service (SMS) message transmission restriction of no more than 100 SMS messages per hour imposed by the Android operating system, transmitted data file size must be less than 13.53 KB per hour. Furthermore, it is recommended that transmitted data file size be reduced to 11.2 KB per hour to provide the end user with 20 SMS messages per hour.
- Transmitting SMS messages at a rate of one SMS message every ten seconds produces a successful delivery rate of 100 percent for both data files and SMS messages. Furthermore, it produced the lowest per SMS message latency.

D. THESIS STRUCTURE

This thesis is organized as follows:

- Chapter I covers the motivation for conducting this research, the research questions being addressed, and the key findings of the research.
- Chapter II discusses prior work, provides an overview of SMS operation, the GSM network,

Android based mobile devices, and a description of the utilized hardware and software.

- Chapter III describes the modifications to software and hardware necessary to provide a covert communication channel, the Android application to utilize the covert channel, and a description of the transmitted data.
- Chapter IV provides a description of the performance metrics, results, and associated analysis.
- Chapter V provides the conclusions drawn from the results and recommends future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. INTRODUCTION

Establishing a method to covertly transmit data from a personal mobile device is important to extracting data without generating user suspicion. As explained in Chapter I, this secretly transmitted data must leave a mobile device, covertly transition through a Global System for Mobile Communications (GSM) network, and arrive at a server for collection and processing without a user's knowledge. Without a method to covertly transmit data, targeted users could become suspicious of applications operating on their mobile devices without their permission. Repeatedly transmitting on a mobile device can consume much of the cell phone's resources and consequently prohibits user initiated applications from functioning smoothly. This might expose the covert application to the user.

A reliable, ubiquitous, and readily available medium on cellular devices to exploit for transmitting is the Short Message Service (SMS). This chapter provides a functional overview of SMS operation. Next, this chapter will review the GSM network. Then, this chapter will cover important architectural components of an Android-based mobile device. Additionally, this chapter addresses the software and hardware employed. Finally, this chapter describes previous some established techniques using SMS to facilitate covert data transfer.

B. SHORT MESSAGE SERVICE

The Short Message Service may seem unnecessary because other communication mediums are available, but SMS has proved to be reliable following the September 11, 2001, attacks on Washington, D.C., and New York City. Verizon Wireless experienced a 100 percent increase in voice traffic while Cingular Wireless (now AT&T) experienced a 1,000 percent increase in the New York area following the attacks. While normal voice-based telephone and cell phone servers were saturated, the SMS network was still usable because its control channels responsible for its delivery remained open [5].

In the last decade, the Short Message Service has become significant in the global communication market. In 2002, fewer than one billion text messages were sent monthly in the United States. In 2010, 187.7 billion text messages were sent monthly nationwide and over 2.1 trillion annually [2]. Globally, 6.9 trillion SMS message were sent [4]. According to Rear Admiral Gregory J. Smith, Deputy Chief of Staff, Communication NATO International Security Assistance Force and United States Forces Afghanistan, 3G GSM-based cell phones are the second most influential media in Afghanistan, reaching 51 percent of the population, second only to radio [6]. Thus, even in austere locations, SMS messages are a vital and widespread reliable method of communication. This thesis therefore focuses on covert GSM SMS communications.

1. TEXT MESSAGE FORMAT

SMS messages are limited to 140 bytes, allowing for 160 seven-bit ASCII characters, the default seven-bit GSM

alphabet [7]. Routing and metadata requires additional memory space, and header and metadata size varies with service provider and transmission mode [8]. In general, SMS messages are transmitted in one of two modes: text or Protocol Description Unit (PDU).

Abbr.	Reference	P1)	P2)	Description
TP-MTI	TP-Message-Type-Indicator	M	2b	Parameter describing the message type.
TP-RD	TP-Reject-Duplicates	M	b	Parameter indicating whether or not the SC shall accept an SMS-SUBMIT for an SM still held in the SC which has the same TP-MR and the same TP-DA as a previously submitted SM from the same OA
TP-VPF	TP-Validity-Period-Format	M	2b	Parameter indicating whether or not the TP-VP field is present.
TP-RP	TP-Reply-Path	M	b	Parameter indicating the request for Reply Path.
TP-UDHI	TP-User-Data-Header-Indicator	O	b	Parameter indicating that the TP-UD field contains a Header.
TP-SRR	TP-Status-Report-Request	O	b	Parameter indicating if the MS is requesting a status report.
TP-MR	TP-Message-Reference	M	I	Parameter identifying the SMS-SUBMIT.
TP-DA	TP-Destination-Address	M	2–12o	Address of the destination SME.
TP-PID	TP-Protocol-Identifier	M	o	Parameter identifying the above layer protocol, if any.
TP-DCS	TP-Data-Coding-Scheme	M	o	Parameter identifying the coding scheme within the TP-User-Data.
TP-VP	TP-Validity-Period	O	o/7o	Parameter identifying the time from where the message is no longer valid.
TP-UDL	TP-User-Data-Length	M	I	Parameter indicating the length of the TP-User-Data field to follow.
TP-UD	TP-User-Data	O	3)	

1) Provision; Mandatory (M) or Optional (O).

2) Representation; Integer (I), bit (b), 2 bits (2b), Octet (o), 7 octets (7o), 2–12 octets (2–12o).

3) Dependent on the TP-DCS.

Table 1. SMS-SUBMIT Fields from [8]

SMS messages are transmitted primarily using PDU format. A PDU formatted message's content depends on whether the message is in SMS-SUBMIT format or SMS-DELIVER format. Each SMS-SUBMIT and SMS-DELIVER message is preceded with the Short Message Service Center (SMSC) address field,

including the Address Length, the Type of Address, and an Address Field containing the SMSC address [8]. The rest of the SMS-SUBMIT and SMS-DELIVER messages are formatted according to Table 1 and Table 2, respectively.

Abbr.	Reference	P1)	P2)	Description
TP-MTI	TP-Message-Type-Indicator	M	2b	Parameter describing the message type.
TP-MMS	TP-More-Messages-to-Send	M	b	Parameter indicating whether or not the SC shall accept an SMS-SUBMIT for an SM still held in the SC which has the same TP-MR and the same TP-DA as a previously submitted SM from the same OA
TP-LP	TP-Loop-Prevention	M	2b	Parameter indicating whether or not the TP-VP field is present.
TP-RP	TP-Reply-Path	M	b	Parameter indicating the request for Reply Path.
TP-UDHI	TP-User-Data-Header-Indicator	O	b	Parameter indicating that the TP-UD field contains a Header.
TP-SRI	TP-Status-Report-Indication	O	b	Parameter indicating if the MS is requesting a status report.
TP-OA	TP-Originating-Address	M	2–12o	Parameter identifying the SMS-SUBMIT.
TP-PID	TP-Protocol-Identifier	M	o	Parameter identifying the above layer protocol, if any.
TP-DCS	TP-Data-Coding-Scheme	M	o	Parameter identifying the coding scheme within the TP-User-Data.
TP-SCTS	TP-Service-Centre-Time-Stamp	O	o/7o	Parameter identifying the time from where the message is no longer valid.
TP-UDL	TP-User-Data-Length	M	I	Parameter indicating the length of the TP-User-Data field to follow.
TP-UD	TP-User-Data	O	3)	

1) Provision; Mandatory (M) or Optional (O).

2) Representation; Integer (I), bit (b), 2 bits (2b), Octet (o), 7 octets (7o), 2–12 octets (2–12o).

3) Dependent on the TP-DCS.

Table 2. SMS-DELIVER from [8]

2. TEXT MESSGE ROUTING

SMS messages must be routed through a network originating at the Message Station (MS) and through the SMSC, before reaching an SMS capable device. Figure 1 shows a brief depiction of the structure that supports SMS

message routing. Figure 2 provides a generalized description of the message flow for a successfully routed SMS message. A general depiction illustrating the message flow for a successful SMS message routing from a SMSC to a SMS capable user interface is presented in Figure 3.

Routing between SMSCs is not part of the Public Land Mobile Network (PLMN); thus, 3GPP does not control this routing, as SMSC functions and routing are unique to the service provider and highly proprietary. In general, SMSCs function as a relay and "store-and-forward" in the SMS infrastructure.

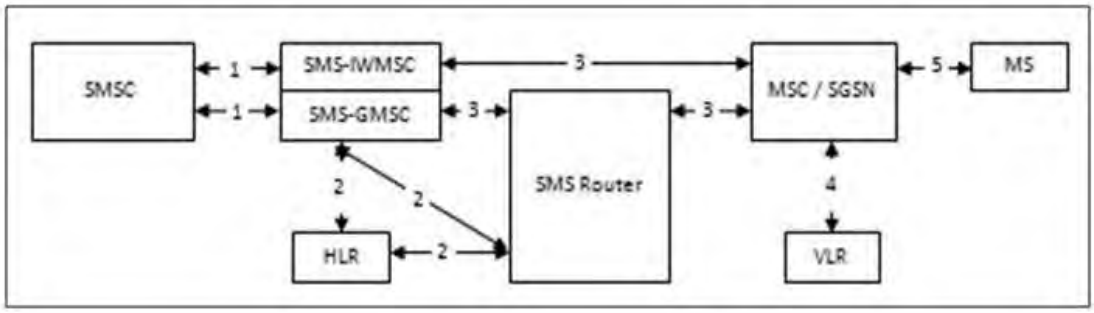

Figure 1. SMS network structure from [8]

SMS is composed of three routing procedures: Short Message–Message Origination (SM–MO), Short Message–Message Termination (SM–MT), and Transfer of Alert [8]. SM-MO and SM-MT will be examined more closely.

SM-MO consists of the operations that are required to transfer a short message from the MS to the SMSC. Additionally, the SM-MO consists of all operations to return a report to the MS containing the result of the attempted message transfer [8].

SM-MT consists of operations that fall into two categories. First, SM-MT consists of all necessary operations to transfer a short message or status report from the SMSC to the MS. Second, the SM-MT consists of all necessary operations to notify the SMSC of the result of the attempted message transfer [8].

a. Short Message - Message Origination

During the Message Origination (MO) phase, the Message-services Switching Center (MSC) has three primary functions. First, it must receive the SM TPDU. Second, it must query the Visitor Location Register (VLR). If no errors have occur, then, the MSC inspects the RP-DA parameter for errors and, if none are found, the TPDU is transferred to the SMS-IWMSC [8].

The SMS-Interworking Mobile-services Switching Centre (SMS-IWMSC) is a component of the MSC that receives SMS messages from within the PLMN and submits them to the SMSC during the Short Message - Message Origination (SM-MO) case. The three main functions of the SMS-IWMSC are to receive the SMS TPDU, to optionally interrogate the Home Location Register (HLR), and, if no errors are found, to forward the TPDU to the SMSC [8].

The HLR database stores user data including subscriber information, billing data, target user availability data, and current location data. See Figure 2 for a depiction of the Message Origination handling.

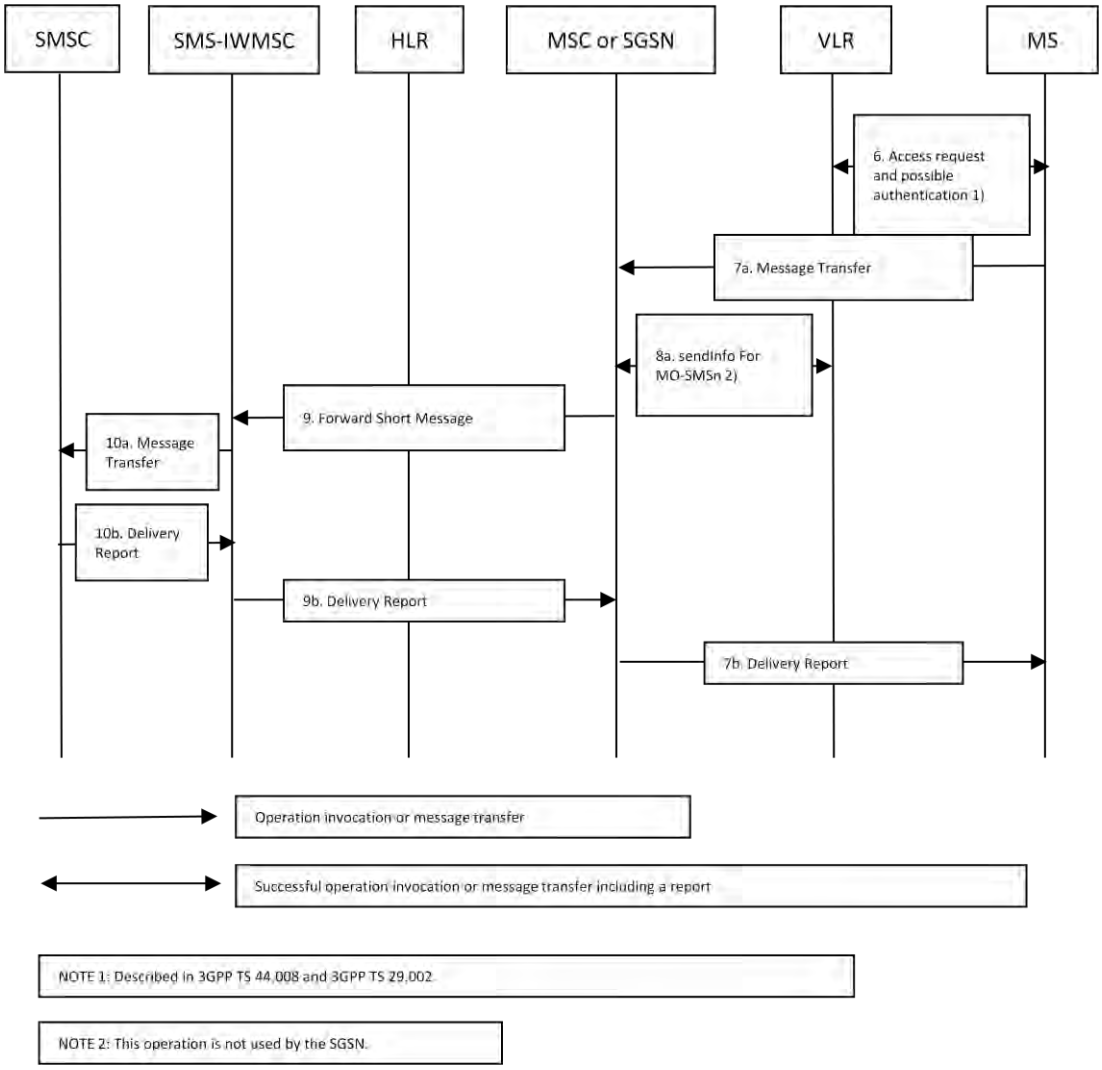

Figure 2. Successful SMS transfer from MS to SMSC–MO Case from [8]

b. Short Message - Message Termination

The SMS-Gateway MSC (SMS-GMSC) receives SMS messages from the SMSC and queries the HLR during the Short Message - Message Termination (SM-MT) case. The SMS-GMSC has four main functions: to receive the SMS TPDU from the SMSC, to inspect the parameters of the TPDU for errors, to query the HLR for data if the parameters do not contain

errors, and, to forward the TPDU to the MSC or the SGSN if no errors are found after the HLR inspection [8].

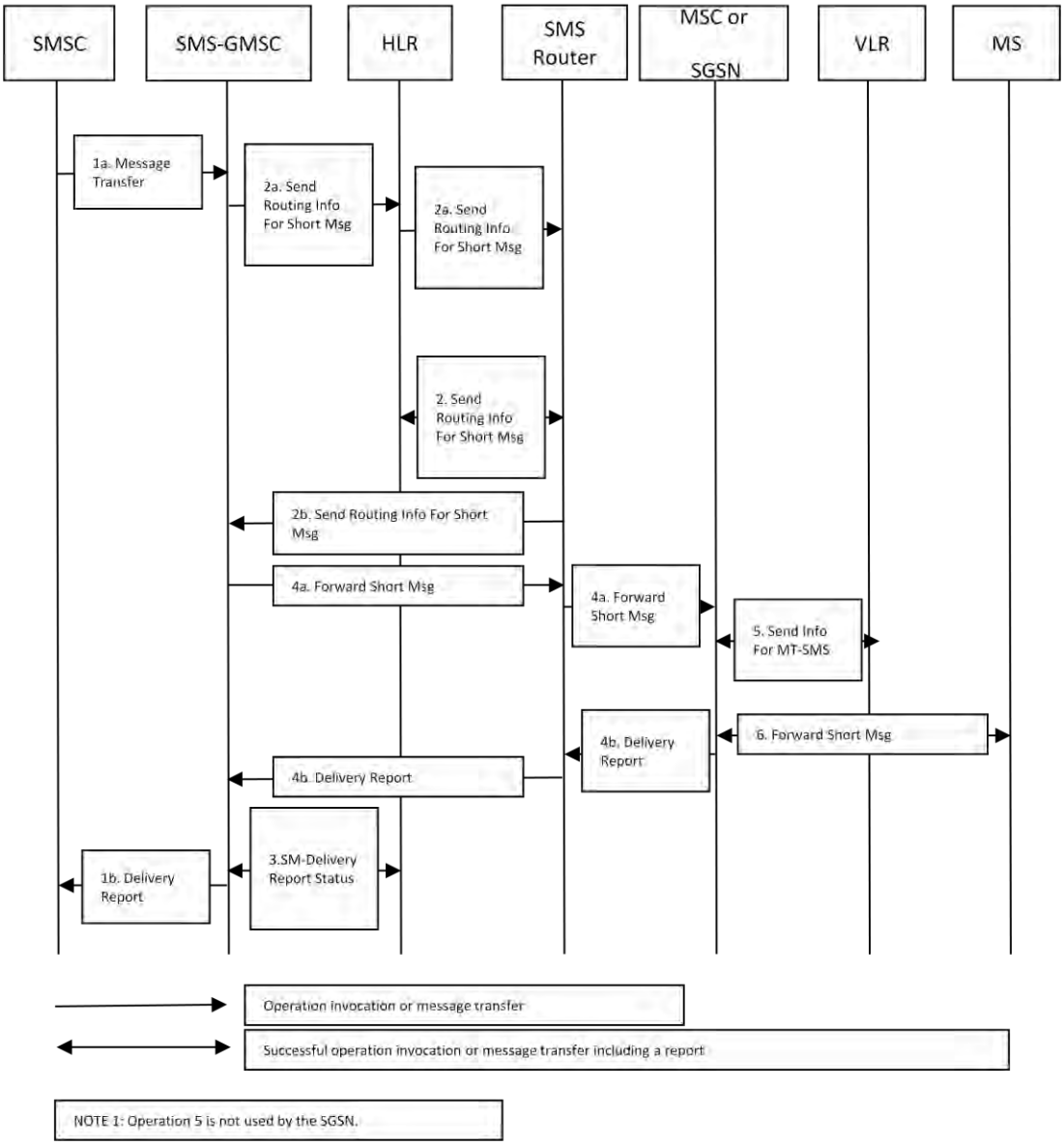

Figure 3. Successful SMS transfer attempt from SMSC to MS-MT Case from [8]

The MSC receives the TPDU from the SMS-GMSC. Additionally, the MSC verifies the receiving SMSC address

and queries the VLR for errors. If no errors are found, the message is forward to the MS [8].

The Serving GPRS Support Node (SGSN) performs packet switching for mobile stations in a specific geographic region like the MSC [8].

The SMS router is optional and is only employed in the Message Terminating (MT) case. The SMS Router interrogates the HLR, and retrieves routing error information [8]. See Figure 3 for a depiction of the Message Origination handling.

C. GLOBAL SYSTEM FOR MOBILE COMMUNICATIONS

1. GSM Network

The GSM network is also part of PLMN. Similar to the SMS network, the GSM network begins with MS. The MS consists of the Mobile Equipment (ME) and contains a Subscriber Identity Module (SIM) card. The SIM card contains: algorithms for authentication, algorithms for generating encryption keys, an International Mobile Subscriber Identity (IMSI), a secret authentication key, and a list of preferred and forbidden PLMNs [9].

The next node in the GSM network is the Base Station Subsystem (BSS). The BSS consists of, one or more towers or Base Transceiver Station (BTS), and the Base Stations Controller (BSC). The BSS establishes and maintains the radio interface between the MS and the central components of the GSM network. The BTS facilitates communication with the MS, while the BCS manages and coordinates within the BSS [9].

The core of the GSM network is the Network Switching Subsystem (NSS). The NSS consists of the Mobile Switching Center (MSC), the Home Location Register (HLR), the Visitor Location Register (VLR), the Equipment Identity Register (EIR), and the Gateway Mobile Switching Center (GMSC). The VLR and HLR are identical to the SMS network components mentioned previously. The MSC is where SMS and voice routing transition to their separate networks [9].

The EIR is a database that divides User Equipment (UE) devices into three lists: white, black, and gray. White-listed UEs are authorized to communicate on the network, while black-listed UEs are prohibited. Gray-listed UEs are tracked for other reasons [9].

The GMSC exchanges with the Public Switched Telephone Network (PSTN). The GSMC references the HLR and routes traffic through the appropriate network nodes to the specific MSC [9].

2. Physical Channels

Throughout the United States, GSM-800, GS-850, GSM-1800, and GSM-1900 systems are utilized [10]. For this research, a USRP1 configured with RFX900 daughter boards was employed. RFX900 daughter boards have a frequency range of 750–1050 MHz [11]. Furthermore, OpenBTS, which facilitates the GSM network infrastructure, was configured to broadcast on GSM-900.

GSM-900 system uses two frequency bands: 890–915 MHz and 935–960 MHz for uplink and downlink respectively [10]. Frequency Division Multiple Access (FDMA) creates 200 kHz frequency spaced carrier frequencies. At least one

carrier frequency is assigned to each Base Station. Next, through Time Division Multiple Access (TDMA), each frequency is divided into eight time slots; creating 1984 half-duplex channels, 283 half-duplex channels per cell. The division of physical channels into seven groups prevents two channels from overlapping when distributed to the cells [9],[12].

3. Logical Channels

Two main classes of Logical channels form the GSM network: Common Channels (CCH) and Dedicated Channels (DCH). CCHs consist of Broadcasting Channels (BCH) and Common Control Channels (CCCH). BCHs transmit information to all the MSs in cell-like frame synchronization signals, frequency synchronization signals, and specific system parameters such as frequency hopping sequence and surrounding cell information. CCCHs primarily send information for accessing management functions, such as paging MSs to inform of incoming traffic, allowing MSs to coordinate for calls or SMS messages, and performing setup requests for calls and messaging services [9].

DCHs consist of Dedicated Control Channels (DCCH) and Traffic Channels (TCH). DCCHs are used for power control, timing advance, location updating, and authentication. Additionally, these channels transmit field strength measurements and data about neighboring cells. TCHs transmit the voice and data traffic, and are either full rate (22.8 KBps) or half rate (11.4 KBps) [9].

D. SOFTWARE AND HARDWARE

This research employed: a UNIX computer, a Universal Software Radio Peripheral (USRP), and an Android Operating System mobile device. Furthermore, the UNIX computer must have OpenBTS, GNU Radio, Asterisk, and Smqueue software installed. Combined, these tools function as the SMSC, GMSC, MSC, and BSS. Figure 4 depicts an overview of the system.

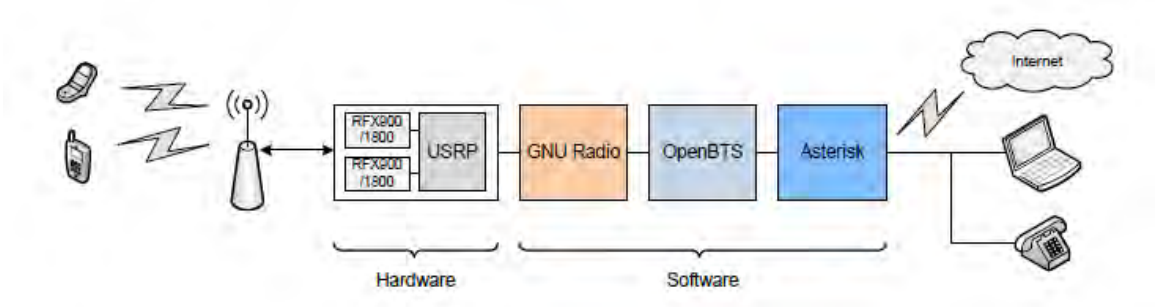

Figure 4. System Overview from [13]

1. Universal Software Radio Peripheral

The Universal Software Radio Peripheral 1 (USRP1) is required to act as the antenna interface for the GSM network and interacts directly with GNU Radio [11]. Varieties for daughter boards are available to create various GSM networks. For this research, a RFX900 daughter board transceiver was used, which has a frequency range of 750–1050 MHz [11].

2. GNU Radio

GNU Radio is a tool that provides the building blocks and signal processing runtime for software radios [14]. GNU Radio interfaces with low-cost external RF radio

equipment such as a USRP, and enables wireless network communications [16]. GNU Radio version 3.2.2 was used in this research.

3. Open Base Transceiver Station

Due to the robust equipment needed to establish a SMS capable GSM network, a more realistic method to establish a GSM network was to utilize an Open Base Transceiver Station (OpenBTS) was utilized instead [14]. OpenBTS performs all the same functions as the GSM network, but with less overhead [14]. OpenBTS version 2.6 was used in this research and comes equipped with Smqueue.

4. Asterisk

Asterisk is a fundamental component to convert a computer into a telephony network. Asterisk provides many services including Internet Protocol (IP) Private Branch Exchange (PBX) and Voice over IP (VoIP) gateways [15]. As an IP PBX, Asterisk acts as a call switch and route manager capable of connecting any Plain Old Telephone Service (POTS), IP, or digital connection, including cell phones [15]. Asterisk version 1:1.6.2.5 was used in this research.

5. Smqueue

Smqueue replaces the SMSC and facilitates a RFC-3428 store-and-forward server to handle SMS with OpenBTS. Smqueue runs in parallel with Asterisk to handle SMS routing. Smqueue uses the same SIP registry as Asterisk,

so they must be installed on the same computer [17]. The Smqueue version used in this research coincides with OpenBTS version 2.6.

E. ANDROID MOBILE DEVICE

This research utilized an HTC Nexus One mobile device with the Android Operation System version 2.2.1 installed.

1. Android Architecture

a. Linux Kernel

Android relies on Linux version 2.6 as its kernel for its core system services. This kernel acts as a layer between the hardware and the remaining software stack. The Linux kernel provides security, memory management, process management, network stack, and driver model services [18].

Figure 5. Android Architecture from [18]

b. Android Runtime

Android is built with core libraries that provide most of the functionality of the Java programming language. Additionally, every Android application runs in its own Dalvik Virtual Machine (VM). Dalvik can run multiple VMs simultaneously. Dalvik relies on the Linux Kernel for lower level resources [18].

c. Libraries, Application Framework, and Applications

Android also has C and C++ libraries with SQLite, Surface Manager, and System C libraries. SQLite is the database manager that is available to all applications [18]. The application framework simplifies the use of components by providing universal services and systems, including content providers, activity manager, and resource manager [18]. Android comes with standard applications to handle e-mail, SMS, telephone, calendars, maps and navigation, and web browsing [18].

2. Activities, Services, Broadcast Receivers, and Intents

a. Activity

Activities are the primary class of object that users interact with on Android mobile devices. An activity possesses the ability to create a User Interface (UI). When an activity is in the foreground it is said to be *active*. An activity is *paused* when is still in view on the UI yet has a non-full screen activity or transparent activity in front of it. A *paused* activity is kept in the memory, but can be killed off by the system in low memory

situations. If an activity is completely blocked by another activity in the UI, it is *stopped*. A *stopped* activity is still kept in memory, but its UI is hidden. Again, just like *paused* activities, a *stopped* activity may be killed off when memory resources are low. An activity can be removed from memory when it is asked to finish. A finished activity must be reloaded into memory if activated later [19].

b. Service

The primary difference between an activity and a service is that activities have short life spans, while Services are meant to be kept running, in some cases, independent of an activity [20]. One concern of a service, just like an activity, is that it runs in the main thread of the hosting process. Yet, services provide a method for applications to tell the system about tasks it wants completed in a background. Additionally, a service allows an application to expose some of its functionality to other applications [21].

c. Broadcast Receivers

Broadcast Receivers are intended to listen for specific Intent to occur on the mobile device. When the specific event Intent occurs, the broadcast receiver is activated. Once the broadcast receiver is complete, it is considered no longer active. Because of its short-lived life cycle, it cannot handle any Asynchronous operations [22].

d. Intents

Intents are messages which activities, services, and broadcast receivers use to communicate with one another. Intents are used to communicate between components of the same or different applications [23].

Typically, Intents can be composed of four different pieces of information. Usually, an Intent contains a combination of a component name, action, data, or category. Occasionally, extras may also be included [23].

Intents come in two varieties: Explicit and Implicit. Explicit Intents are used to communicate with a specific component and are ideal when you know the exact application and component that is suitable for the desired task. Explicit Intents are composed of component and package names. Implicit Intents on the other hand, are best used when you have a task but do not know which service or activity is best to handle it. Implicit Intents contain an action, data, and/or category. Once an Implicit Intent is broadcast, the system message handler searches through the intent filters of all the applications to determine the most suitable component to handle the message [23]. For example, an Implicit Intent is broadcast to handle an inbound SMS message, so the message handler would search for applications that handle SMS messages.

3. AsyncTask

For every process that is launched, the system creates a "main" thread. The main thread is tasked with coordinating and performing all UI tasks and activities of

the application [24]. Additionally, if another application requests a process that is currently in use, the new application is launched in the same thread as the active application [24]. Demands on the main thread can become very intensive if not managed properly and could lead to a system crash.

To mitigate the work load on the main thread, Android has AsyncTask. AsyncTask allows the programmer to send non-UI tasks to a worker or background thread while providing a gateway for any UI activity to be passed back to the main thread [24]. Passing tasks to the background greatly reduces the strain on the main thread, especially if an application does not interact with the UI.

F. PRIOR WORK IN SMS-BASED COVERT CHANNELS

Limited prior research has been conducted to exploit the Short Message Service and its vulnerabilities. The main focus of a secret channel is to transmit data while the user and/or signal analysts are unaware of the transmissions. There are several methods to transmit data through the SMS channel while evading onlookers to include covert transmissions, steganographic hiding of data, and stealth transmissions.

1. Manipulation of PDU Headers

One previously explored covert channel consists of concealing data in the header portion of a PDU formatted Concatenated SMS (CSMS) message. A CSMS is a SMS message that has been separated into segment PDUs because it was too large to be included in the 140 bytes of a single SMS message. An additional header of a CSMS message, called a

TP-User_Data_Header (TP-UDH), is included in the TP-UD to account for segmenting and ordering of the CSMS. The optional TP-UDHI field of the PDU is set to signal that the message portion of the SMS contains additional header information. The TP-UDHI bit is contained in the 7th bit (bit 6) of the first octet of a PDU-formatted SMS message. When set to 0, it indicates that the TP-UD portion only contains message data, while setting it to 1, indicates that there is extra header information in the TP-UD portion [8]. See Table 2 for more information on the PDU bit arrangement.

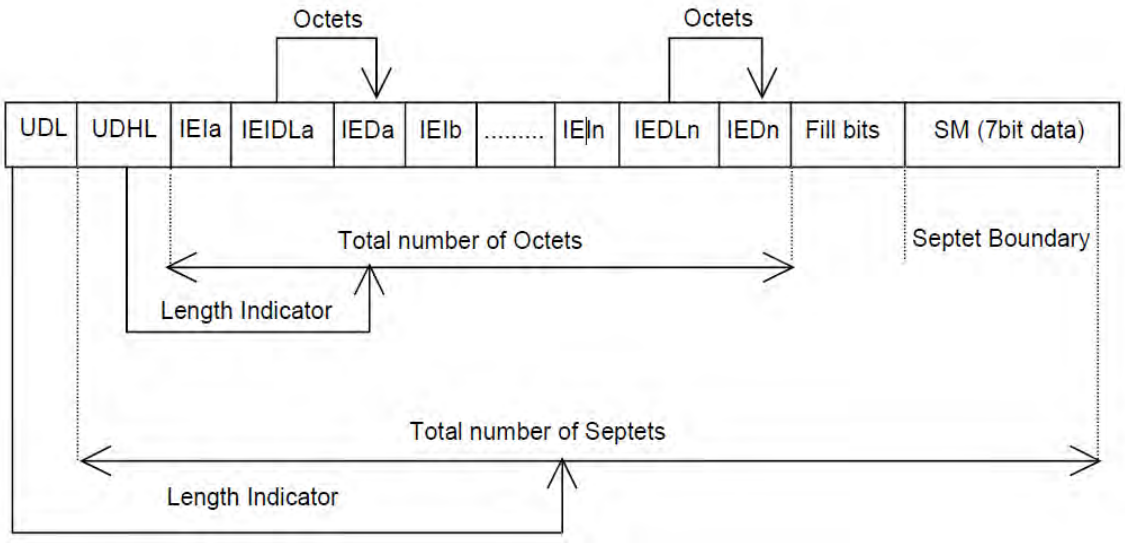

Figure 6. Layout of the TP-User-Data-Length and the TP-User-Data for uncompressed GSM 7-bit SMS message from [8]

The first octet of the TP-UDH indicates the total length of the TP-UDH called the TP-UDH Length (UDHL). For a CSMS, the next octet is the appropriate Information Element Identifier (IEI), 0 for a CSMS. There may be more than one IEI. The following octet is the total length of the Information-Element-Data (IED) referred to as the IEI

Data Length (IEIDL) [8]. Last is the IED which consists of three octets added to identify which CSMS each segment belongs to, the maximum number of segments in the particular CSMS, and the sequence number of the current SMS respectively [8]. Figure 6 provides the layout of a multi-IEI formatted 7-bit SMS message.

One possible covert channel is to set the TP-UDHI bit to 1, signaling that there is additional information header information in the TP-UD portion. Subsequently, the 3 IED octets are available for manipulation. Octets 2 and 3 are each set to 1 indicating the CSMS is composed of 1 segment, which means that the first octet can be set to any value from 0 to 255. This makes for an available one byte of covert information per text message and requires an established connection with the mobile device in order to manipulate the TP-UDHI bits [25].

Similarly, the same octet could be exploited in an actual concatenated message. Since the first octet could be any value between 0 and 255, the value could be purposely set. This method only allows for one byte of information per CSMS and also requires an established connection with the mobile device [25].

2. Manipulation of Enhance SMS messages

Another method to transmit data covertly is to manipulate information in Enhanced Message Service (EMS). Since these types of SMS messages require the use of several applications on the destination mobile device interpret message, each embedded IEI type includes a destination application port number on the destination device and originator application port number from the

sending device. Since the originator application port number is not needed at the destination, it can be manipulated to contain a message. In the case of a CSMS that contains multiple IEIs, the originator application port number can be different in each segment of the CSMS, thus allowing for larger message bandwidth [25]. Again this method would require a direct connection to the mobile device in order to manipulate the bits.

3. Steganographic Approaches to Covert Channels

Steganography is the art and science of hiding secret data or messages in plain sight so that no one other than the sender and the intended recipient are aware of the hidden contents existence. This method varies from the previously mentioned techniques because it conceals the hidden message within text or other media rather than exploit header fields. Though steganographic approaches have been in practice for hundreds of years, the introduction of digital technology has created new opportunities for exploitation. SMS messages are not excluded.

Employing steganographic techniques in text messages encounters two challenges. First, changing bits will result in a different character representation. Second, adding space or characters will increase the size of the original message and be obvious to a trained reader [26].

Three primary methods of manipulating ASCII text are using acronyms, changing the spelling of words, and using synonyms. The use of acronyms invokes switching the original word or phrase with its associated acronym. Changing spelling involves switching two words that mean

the same but are spelled differently in two different yet similar languages. The act of using synonyms is simply the switching two words that have the same meaning [26].

III. TECHNIQUES AND EXPERIMENTAL DESIGN

A. INTRODUCTION

This chapter provides a discussion and analysis of the equipment and software employed and developed in this research. First, is a discussion of some alternative techniques to transmit covertly through SMS messages and an explanation why those techniques were not chosen. Next, is a discussion of the approaches implemented. This is followed by a discussion of the Android application created to exercise the covert channel and validate this research. Then, the modifications made to the Smqueue software to support the research will be discussed. Lastly, the transmitted information will be presented.

B. OTHER CONSIDERATIONS

Several of the covert methods described in Chapter II were considered, but not implemented because of their feasibility, such as using concatenated and alternate types of SMS messages to carry the information.

1. CSMS Header Info

As discussed in Chapter II, one method to covertly transfer data with SMS messages is to invoke a TP-UDH header, and use the first octet (one byte) to conceal a message. However, this method has two weaknesses. Firstly, the bandwidth is extremely low for these purposes, allowing for only one byte per SMS message. Secondly, implementing this channel require physical modification to the mobile device [25]. This channel is more suitable for transmitting covert information through SMS messages to receivers while

evading listeners. In contrast, our goal is to transmit information without the end user's knowledge.

2. Alternate Types of SMS Message

Another possible channel would be to transmit data within SMS message types other than SMS-SUBMIT messages. Using other method types would involve modifying the TP-Message-Type-Indicator bit in the first octet of a PDU formatted SMS message [8]. Figure 7 provides a list of alternative types of SMS messages. This level of SMS message manipulation is outside the scope of an Android application.

Bit 1	Bit 0	Message Type
0	0	SMS-Deliver (in the direction SMSC to MS)
0	0	SMS-Deliver Report (MS to SMSC)
1	0	SMS-Status Report (SMSC to MS)
1	0	SMS-Command (MS to SMSC)
0	0	SMS-Submit (MS to SMSC)
0	1	SMS Submit Report (MS to SMSC)
1	1	Reserved

Figure 7. SMS Message Types from [8]

3. Encryption

Encryption would be ideal if the intent was to broadcast SMS messages and evade interlopers listening to the RF communication channel. For this research, it is assumed that encryption would not be an added benefit since the purpose of this research is to evade the end user. If the end user discovered an unauthorized transmission (encrypted or not), they could prevent further transmissions. Furthermore, encrypting the message would consume more of the CPU and require a larger software footprint. Lastly, if an unencrypted covertly transmitted message is detected its existence will likely be discarded as the original message. An encrypted message certainly will raise suspicion.

C. DESIGN CONSIDERATIONS

The design of the system implemented aims to inconspicuously transmit SMS messages from a mobile device without the end user's knowledge. In achieving this transmission, several considerations were taken. First, the application should not cause the mobile device to present any graphical user interface to the end user. Second, the user should be able to use the mobile device normally and without interruption. Third, the user should not be billed for the extra transmitted SMS messages. Finally, the application should not exceed the one hundred text message per hour limit imposed by the Android Operating System [27].

The 100 SMS per hour should be assumed to be in effect on the mobile device because the only known method to

modify the limit is to have physical possession of the mobile device [27]. This limit was verified by evaluation during the research.

D. SPECIFIC DESIGN OF SYSTEM

Two systems were needed: an Android application capable of covertly transmitting SMS messages and, a network node capable of receiving covert SMS messages. Smqueue was modified to facilitate the second requirement.

1. Android

An Android application must be able to execute five subtasks to covertly transmit information properly. First, the application must be able to evaluate the mobile device's network attachment point. Second, the application must be able to initiate the transmission of data. Third, the application must be able to read in the data to transmit. Fourth, the application must be able to label each SMS message so that they can be collected by the SMS Center (SMSC). Lastly, the application must be able to transmit the SMS messages without the end user's knowledge.

a. Determine Friendly GSM Network

An Android based mobile device has access to a few resources that can identify the device's network. Every BTS cell tower broadcasts a unique Location Area Identifier (LAI). Each LAI consists of the Mobile Country Code (MCC), Mobile Network Code (MNC), and a Location Area Code (LAC). The MCC and MNC are each three digits long while the LAC is a maximum of five digits or sixteen bits. Additionally, each LAC has one or more Cell Identifiers

(CID). A CID is formatted similarly to a LAC [9]. In addition to the LAC and CID, each SMS message contains the address of the SMSC [8]. The LAC, CID, and SMSC addresses will be compared to lists of known safe network nodes to determine if the application is safe to transmit its payload. This method does invoke a self-imposed limitation by forcing the application to function only through known safe nodes. Future research could develop a method to evaluate the GSM network through a series of challenge and responses. The goal of these challenge and responses would be to determine if the current BTS connects to known safe network node, namely the controlled SMSC.

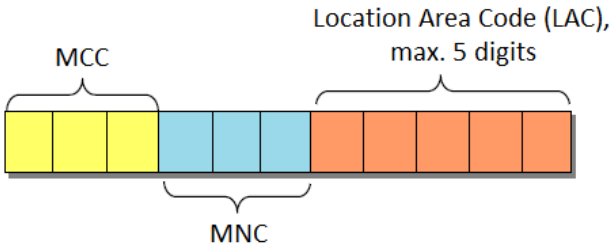

Figure 8. Structure of the LAI from [9]

b. Broadcast Receiver Calls a Service

Once the application verifies the addresses of the pertinent GSM network components, an Intent is broadcast to the Android system manager which then determines which application component should handle the data transmission. The developed application will only produce an Intent that can only be handled by another service within the application.

c. Reading in the Data

Reading in data is straight forward. Suppose an outside application produces and stores data in a file on the Android device. The application would broadcast an Intent to the SMS transmitting application, which would begin reading in the data from a predetermined text file. Future research could explore and develop more clandestine methods for creating and storing the data for transmission. For this research, it is assumed that an application capable of producing data for transmission and the data files already exist on the phone. It is also assumed that the end user does not know the file's location.

d. Modifying the SMS Message

Each transmitted covert SMS message must be marked in a way that does not modify the collected data. Additionally, the marking must consist of characters in 7-bit ASCII characters. For this research, we used a combination of one or more carriage returns "\r," character 0x13. Four carriage returns identify the initial SMS message, three carriage returns identify the last SMS message, and two carriage returns identify the other SMS messages. A printed carriage return is not entirely noticeable, and so is ideal for this application.

e. Timing

The Android Operating System version 2.2.1 alerts the end user if a large volume of SMS traffic is transmitted from the mobile device [27]. Separate evaluations conducted in this research found that the operating system alerts the end user when more than one

hundred SMS messages are transmitted in one hour. This alert can be physically disabled if on the phone, but our adversarial model suggests that this warning would be in effect, so it must be considered in the transmission rates [27].

To prevent the alert, the application must determine the size of the transmitted data and then manage the transmissions accordingly. If the data to be transmitted is less than 13.53 KB, then the entire data file can be transmitted in less than an hour with fewer than one hundred SMS messages.

$$99 \text{ SMS messages} \times 140 \text{ bytes} = 13{,}860 \text{ bytes}$$
$$13{,}860 \text{ bytes} = 13.53 \text{ KB}$$

In fact, when the size is below this threshold, it can be successfully transmitted in less than 20 minutes. Otherwise, when the data size is larger than the 13.53 KB threshold, it must be transmitted at a slower rate to avoid alerting the user. The quickest time period between each transmission for larger data sizes is 36.363 seconds. But transmitting SMS messages with a 36.363 buffer would result in exactly 99 SMS messages being transmitted in an hour, which leave zero tolerance for the end user. It is recommended that a slower transmission rate be utilized to permit the end users to send their own SMS messages without being alerted to the high transmission rate.

2. Smqueue

Smqueue is a store and forward server, used to support the Short Message Service (SMS) with OpenBTS installations. Smqueue is designed to operate alongside Asterisk to

provide SMS routing [28]. Smqueue and OpenBTS act as a SMSC when combined. Because of its centralized location and functionality, the SMSC was chosen as the network node to catch the covert messages.

This research needed a modified Smqueue source code to facilitate the recognition of the covert transmissions. As stated in paragraph 3.d, each SMS message was pre-pended with a sequence of carriage returns. Smqueue was re-coded to recognize these sequences of carriage returns, and reroute the marked, covert SMS messages to a specified text file for storage.

Furthermore, Smqueue needs to change the SMS message's state to prevent further transmission and routing beyond the controlled SMSC. The smqueue.cpp file provides a state machine that controls the routing and processing of all inbound SMS messages. First, to properly process each SMS message, the SMSC must evaluate the validity of each SMS message. This is performed by checking the destination address against known IMSIs and their aliases within the SIP configuration file of Asterisk.

Next, the covert messages must be prevented from entering the SMSC's transmission queue to prevent further unintended transmission. After an analysis of the Smqueue state machine, it was determined that the best time to terminate the message during the REQUEST_MSG_DELIVERY state, completed by modifying the next state to be the DELETE_ME_STATE. A state diagram is provided in Figure 9.

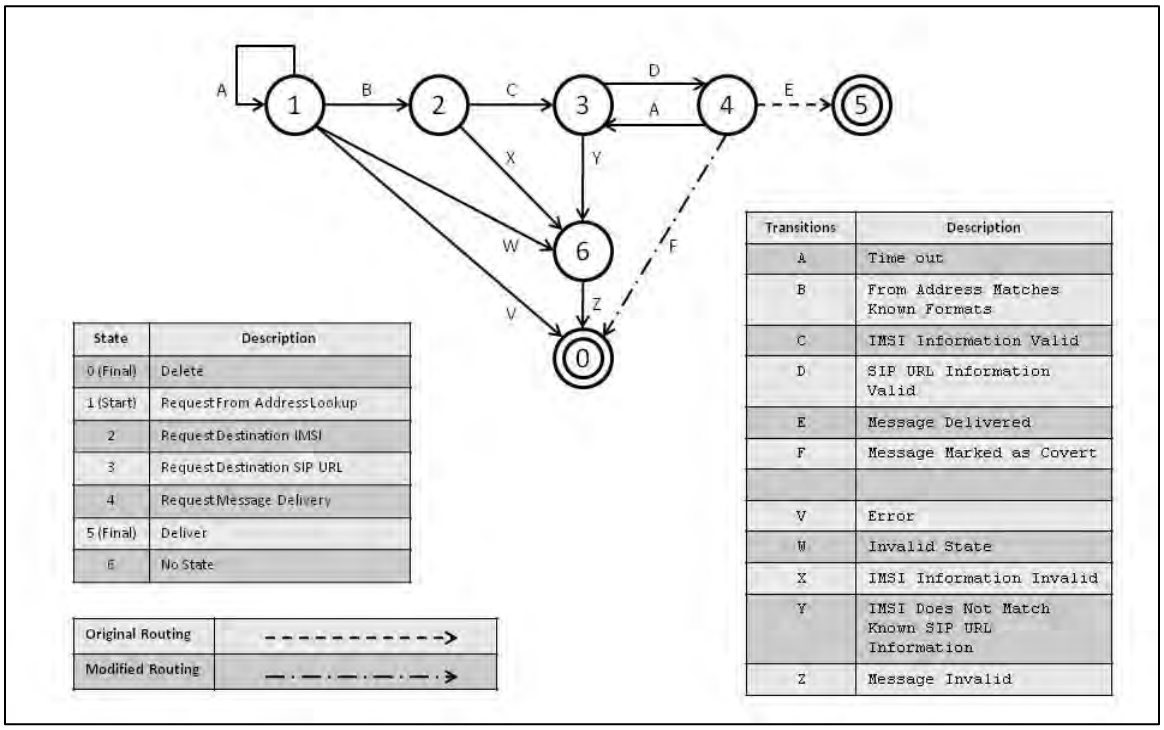

Figure 9. Smqueue SMS processing State Machine

E. TRANSMITTED DATA

The transmitted data file consisted of a portion of a unigram model. Each line in the data file consists of two pieces of information: a key and a count. The key is a word that the author used. The count is the number of times the author used that word or key in their e-mail messages. These keys and counts are used to create authorship prediction models.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND ANALYSIS

A. INTRODUCTION

Experiments were conducted to measure the effectiveness of transmitting three different sized data files: 1 KB, 10 KB, and 100 KB. Transmission rates of one SMS message every one, four, seven, ten, and 36.363 seconds were used. Transmission rates and files sizes were altered throughout the research in an effort to determine the quickest and most reliable method to transmit the data while simultaneously avoiding detection by the end user.

B. LAB SET UP

To ensure all time measurements were accurate, clocks on the mobile device and OpenBTS/Smqueue hosting computer were synchronized and configured to measure from the same epoch time. OpenBTS broadcasts the time and time zone settings read from the hosting operating systems clock. To establish synchronization, a time zone interpreting bug in OpenBTS has to be taken into account. OpenBTS only broadcasts the correct time when the computer's time zone is set to GMT or UTC time. Additionally, the mobile device must be set to synchronize its clock with the broadcasted time from the mobile network.

Once the time and time zone settings have been set and configured correctly, each time measurement method must be checked to ensure it utilizes the same epoch time as its baseline. Ubuntu 10.04 LTS, which is a Unix-based Operating System used in this research, uses an epoch time of 0:00:00 UTC on January 1, 1970 [29]. The Android Operating System

and Java, found on the mobile device, makes use of the same epoch time [30]. C++, the computer language OpenBTS and Smqueue are written in, reads its epoch time from the operating system, in this case is Ubuntu. This means, that as long as the mobile device's and OpenBTS/Smqueue hosting computer's clocks remain synced their time since epoch will be identical.

The lab environment consisted of a closed basement room with no windows. There were no other equipment users. RF interference was negligible. The only broadcasting devices within range of the OpenBTS hosting computer, USRP1, or HTC Nexus One mobile device were the schools wireless network routers, which broadcast on the 2.4 GHz range. During the course of the experiment, the mobile device was kept no further than one meter distance from the USRP1. Additionally, the mobile device was constantly plugged into a power source.

Transmission rates were enforced by invoking prescribed thread pauses between each SMS message transmission. For each combination of file size and transmission rate, a minimum of 20 experiment runs were conducted except in the case of the 100 KB file size, where only 10 experiment runs were conducted. The Tables in this chapter will only present the first ten test iterations for each combination of file size and transmission rate. For complete test results see Appendix C.

C. EXPERIMENT DESCRIPTION

1. Time Measurements

Four time measurements will be taken during each test. First, on the mobile device, time will be measured when the data broadcasting service begins. A second time measurement will be collected when the data broadcast service has completed the transmission of the data file. Third, a time measurement will be collected when each SMS message (140 byte data packet) is transmitted. Lastly, on the OpenBTS/Smqueue computer, a time measurement will be taken every time an SMS message arrives. The first and last SMS messages are uniquely marked and their arrival times will be used to indicate the start and finish times of the data files arrival on the Smqueue computer.

The start and finish times on the mobile device will serve to determine the elapsed transmission time. The start and finish times from Smqueue will determine the elapsed processing and receiving time. The start time on the mobile device and the finish time from Smqueue will be used to determine the Total Elapsed Transmission Time for each data file.

SMS message transmission times from the mobile device and the SMS message arrival times from Smqueue will serve to determine the SMS message latency observed across the network. The transmission times will also verify the set transmission rate while the SMS message arrival times will determine the SMS message arrival rate.

2. Collected Data

The transmitted SMS messages will be collected at the receiving server to verify completeness and proper sequencing. During the research, data files that are not completely received with the correct sequencing must be investigated to determine whether the transmission rate or equipment failure caused the error. If the data is corrupt due to equipment malfunction, that experiment run will be discarded. Otherwise, transmission rates will be adjusted to determine which rate produces the best data file delivery.

D. RESULTS

1. One Kilobyte Data File

The 1 KB data file experiment was conducted with a transmission rate of one SMS message per second. This experiment produced a successful file transmission rate of 100 percent.

When only a single data file was transmitted, the average total elapsed time for a 1 KB data file was 48.843 seconds. When two or more 1 KB data files were transmitted, the total elapsed time increased for each consecutive data file by an average of 42.321 seconds. From Table 3, if three 1 KB data files were transmitted consecutively, the resulting average elapsed times for the second and third data files were 91.274 seconds and 134.020 seconds respectively. This is an important factor to consider when there is more than 1 KB of data to transmit. Table 3 provides the key timing results from the 1 KB data file

test. Figure 10 presents the increase in elapsed time for consecutively transmitted data files.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	sec	sec	sec
0	6.489	41.888	48.843
1	6.376	41.658	91.274
2	6.377	41.894	134.020
3	6.381	41.896	176.817
4	6.386	41.661	219.249
5	6.447	41.696	261.593
6	6.397	41.898	304.247
7	6.390	41.658	346.784
8	6.382	39.777	387.402
9	6.381	39.536	427.836
⋮	⋮	⋮	⋮
Average	6.468	41.597	323.575
Standard Deviation	0.083	0.724	173.382

Table 3. 1 KB data file transmission with one SMS message per second transmission rate.

The SMS message analysis presented some interesting yet unexpected results. Though the SMS messages were transmitted at a rate of one SMS message per second, they arrived at the Smqueue server at an average rate of one SMS message every 6.908 seconds. Evidence during development and testing of the Android code suggests that a transmission queue is utilized within the Android mobile device. Figure 11 presents the increased latency experienced by consecutively transmitted SMS messages.

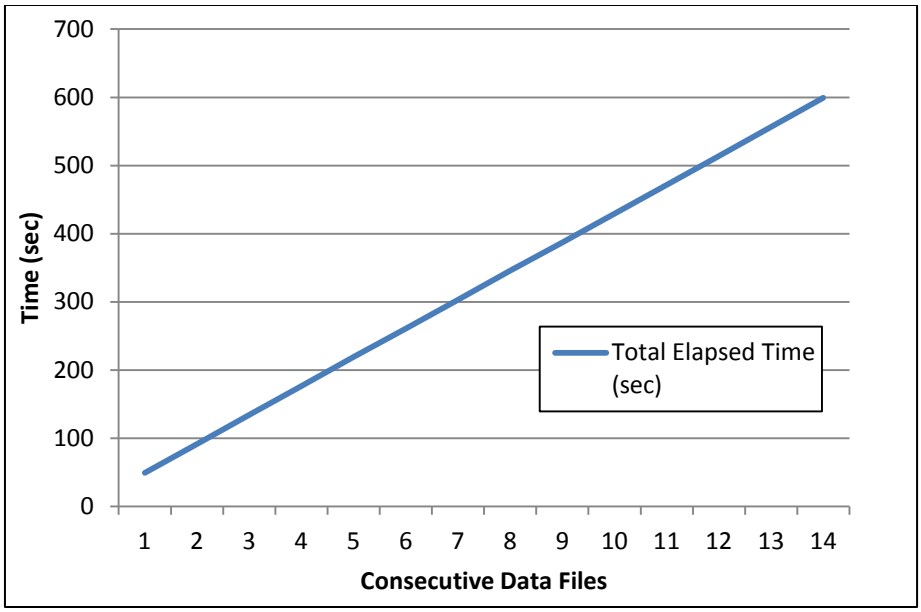

Figure 10. Effects of consecutively transmitted 1 KB data files on data file transmission latency.

2. Ten Kilobyte Data File

The 10 KB data file tests reinforced observations made during the 1 KB data file tests. Three different 10 KB experiments were conducted to observe the effects of transmission rate by setting the transmission rate to either one SMS message transmitted every one, four, seven, or ten seconds.

a. One SMS Message per Second Transmission Rate

During the one SMS message per second tests, a data file delivery success rate of 75.00 percent was observed. Errors in the corrupt transmitted files were the result of either swapped SMS messages, or missing SMS messages.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	sec	sec	sec
0	69.239	471.262	477.626
1	70.894	452.162	474.875
2	70.550	470.553	476.893
3	70.430	470.552	477.393
4	70.427	471.257	478.362
5	70.657	310.249	316.843
6	70.490	471.725	478.343
7	70.315	471.726	478.605
8	70.544	469.373	476.400
9	70.474	467.491	473.436
⋮	⋮	⋮	⋮
Average	69.931	461.861	470.110
Standard Deviation	0.724	36.181	36.185

Table 4. 10 KB data file transmission with one SMS message per second transmission rate.

With a one SMS message per second transmission rate, the average time to transmit a 10 KB data file was 69.931 seconds. The average time to receive a 10 KB data file was 461.861 seconds due to the latency induced by the mobile device's queuing delays. This delay increased the average total elapsed time from transmission to reception for a 10 KB data file to 470.110 seconds (7 minutes and 50.110 seconds). Table 4 provides the key timing results from the 10 KB data file test with a one SMS message per second transmission rate.

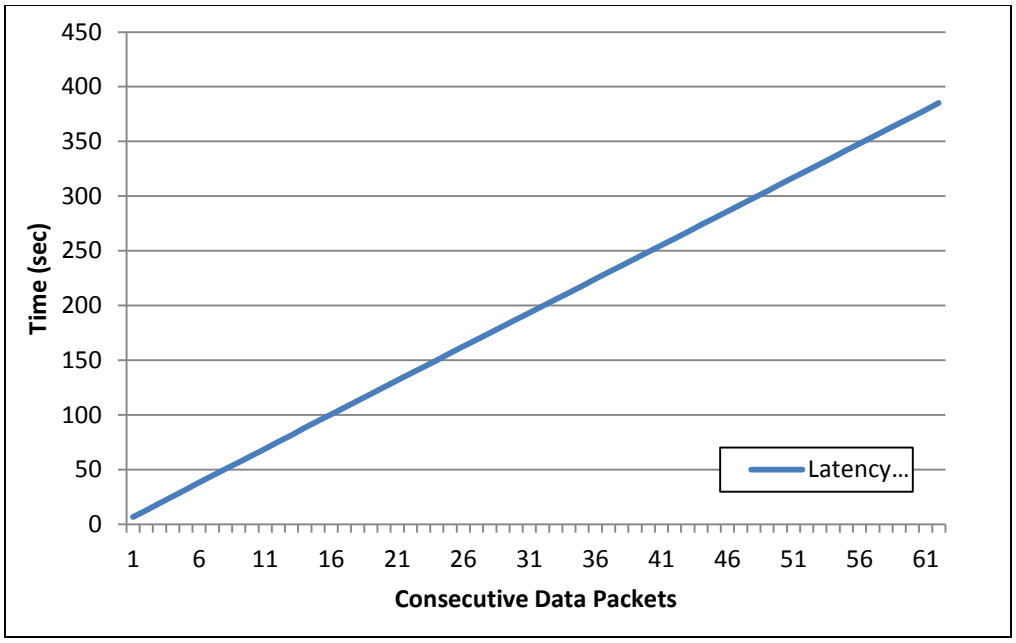

Figure 11. Effects of a one SMS message per second transmission rate on SMS message transmission latency.

The average transmission rate of one SMS message every 1.075 seconds resulted in an arrival rate of one SMS message every 7.256 seconds. As expected, a noticeable increase from the transmission rate. The resulting total elapsed time for a SMS message was 206.443 seconds (3 minutes and 26.443 seconds). Again, Figure 11 presents the increased latency experienced by consecutively transmitted SMS messages with a one SMS message every second transmission rate.

b. One SMS Message per Four Seconds Transmission Rate

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	sec	sec	sec
0	265.880	473.368	479.553
1	265.582	466.787	472.544
2	265.651	470.079	476.063
3	265.620	470.791	477.165
4	265.748	470.786	477.450
5	265.498	469.840	475.093
6	265.771	471.726	478.096
7	265.682	472.432	479.439
8	265.594	476.431	483.095
9	265.921	467.955	476.776
⋮	⋮	⋮	⋮
Average	265.366	471.993	478.508
Standard Deviation	0.606	5.018	4.968

Table 5. 10 KB data file transmission with a SMS message every four seconds transmission rate.

The next 10 KB data file test implemented a transmission rate of one SMS message every four seconds. This transmission rate successfully delivered an intact data file in 90 percent of the tests and delivered the SMS message correctly 99.394 percent.

The one SMS message every four seconds transmission rate produced an average time to transmit a complete data file of 265.366 seconds (4 minutes and 25.37 seconds). The average time to receive the entire data file was 471.993 seconds (7 minutes and 51.99 seconds). The total elapsed time to complete the transmission was 478.508

seconds (7 minutes and 58.51 seconds). Table 5 provides the key timing results from the 10 KB data file test with a one SMS message every four seconds transmission rate.

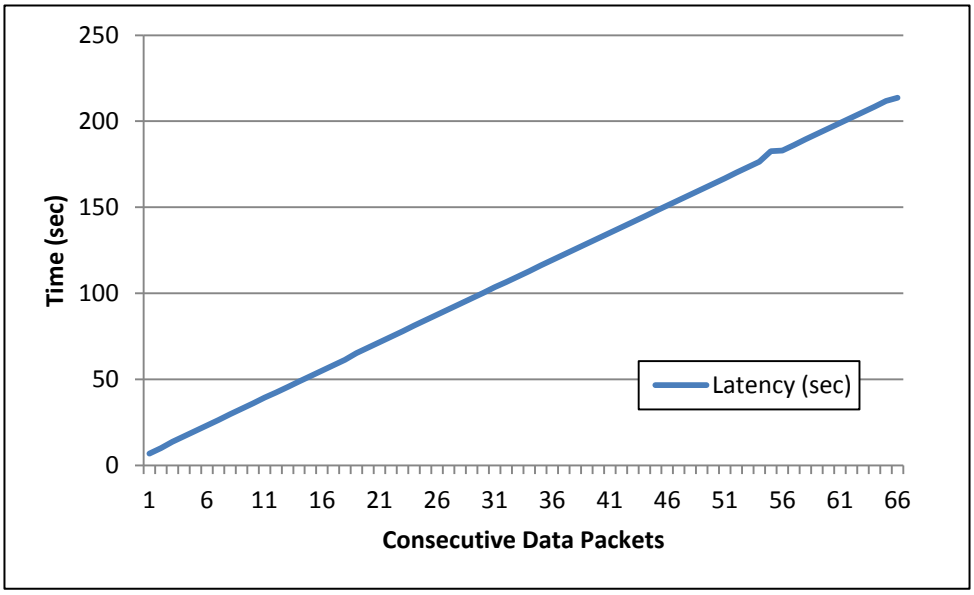

Figure 12. Effects of a one SMS message every four seconds transmission rate on SMS message transmission latency.

With an average transmission time rate of one SMS message every 4.08 seconds, the SMS messages arrived to the Smqueue computer at a rate of one SMS message every 7.306 seconds. Again, these results produce a noticeable increase from the prescribed transmission rate. This transmission rate produces an average elapsed time of 111.242 seconds (1 minute and 51.24 seconds) for a SMS message. Figure 12 presents the increased latency experienced by consecutively transmitted SMS messages with a transmission rate of one SMS message every four seconds.

c. One SMS Message per Seven Seconds Transmission Rate

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	sec	sec	sec
0	459.338	469.375	474.852
1	459.234	478.308	484.830
2	459.195	467.957	474.741
3	459.224	469.371	475.630
4	459.183	470.788	476.790
5	459.166	470.785	477.451
6	459.161	469.135	475.488
7	459.137	468.898	474.884
8	459.210	477.362	483.478
9	459.255	470.296	476.181
⋮	⋮	⋮	⋮
Average	459.196	470.734	476.927
Standard Deviation	0.056	2.665	2.718

Table 6. 10 KB data file transmission with a one SMS message every seven seconds transmission rate.

The next experiment conducted with the 10 KB data file employed a transmission rate of one SMS message every seven seconds. This transmission rate yielded a successful SMS message transmission during 90 percent of the test runs. In the remaining 10 percent of the test runs, errors only resulted from single SMS messages being received out of order. Furthermore, only two of the 1320 SMS messages were received out of order, yielding a successful delivery rate of 99.848 percent.

The average time to transmit the entire 10 KB data file was 459.196 seconds (7 minutes and 39.20

seconds). The average time to receive the whole data file was 470.734 seconds (7 minutes and 50.73 seconds). The total elapsed time was 476.928 seconds (7 minutes and 56.93 seconds). Table 6 provides the key timing results from the 10 KB data file test with a transmission rate of one SMS message every seven seconds.

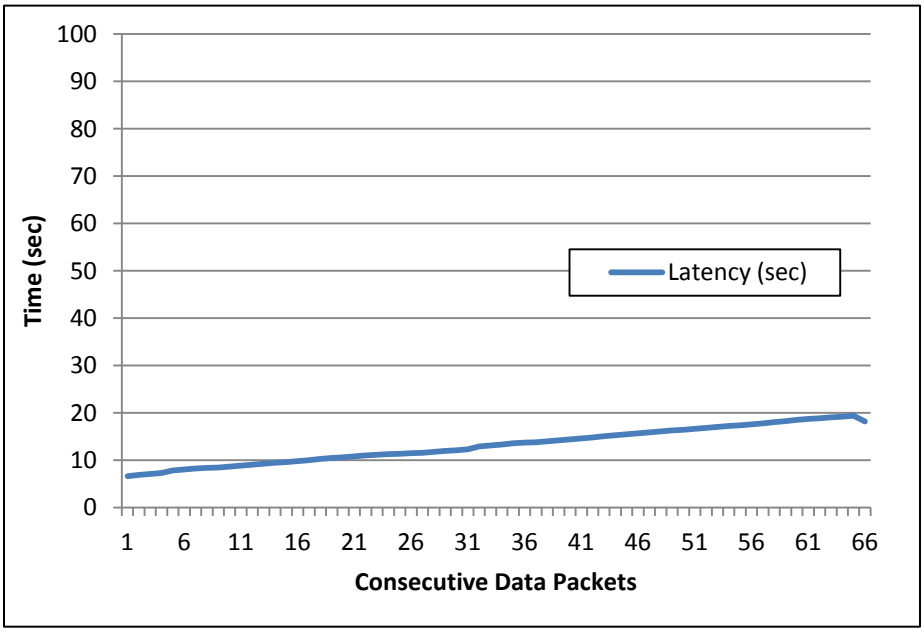

Figure 13. Effects of a one SMS message every seven seconds transmission rate on SMS message transmission latency.

The average transmission rate of one SMS message every 7.059 seconds produced an average arrival rate of one SMS message every 7.236 seconds. The average observed latency of a SMS message was 13.135 seconds. Figure 13 displays the increased latency experienced by consecutively transmitted SMS messages at a rate of one SMS message every seven seconds.

d. One SMS Message per Ten Seconds Transmission Rate

One SMS message every ten seconds was the last transmission rate evaluated with the 10 KB data file. This transmission rate produced a successful SMS message and data file delivery rate of 100 percent.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	sec	sec	sec
0	655.802	651.080	660.692
1	654.365	652.960	659.420
2	654.285	652.959	659.380
3	654.265	652.723	659.199
4	654.255	652.718	659.498
5	654.235	652.958	659.108
6	654.303	652.723	659.209
7	654.283	652.487	658.708
8	654.267	652.955	658.531
9	654.257	652.723	659.831
⋮	⋮	⋮	⋮
Average	655.132	653.307	660.113
Standard Deviation	0.762	1.260	0.900

Table 7. 10 KB data file transmission with a one SMS message every ten seconds transmission rate.

The average time to transmit the 10 KB data file was 655.132 seconds (10 minutes and 55.132 seconds). The average time to receive the data file was 653.307 seconds (10 minutes and 53.307 seconds). The total elapsed time was 660.113 (11 minutes and 0.113 seconds). Table 7

provides the key timing results from the 10 KB data file test with a transmission rate of one SMS message every ten seconds.

The average transmission rate of one SMS message every ten seconds produced an arrival rate of one SMS message every 10.04 seconds. The average observed latency of a SMS message was 6.741 seconds. Figure 14 presents the increased latency experience by consecutively transmitted SMS messages at a rate of one SMS message every ten seconds.

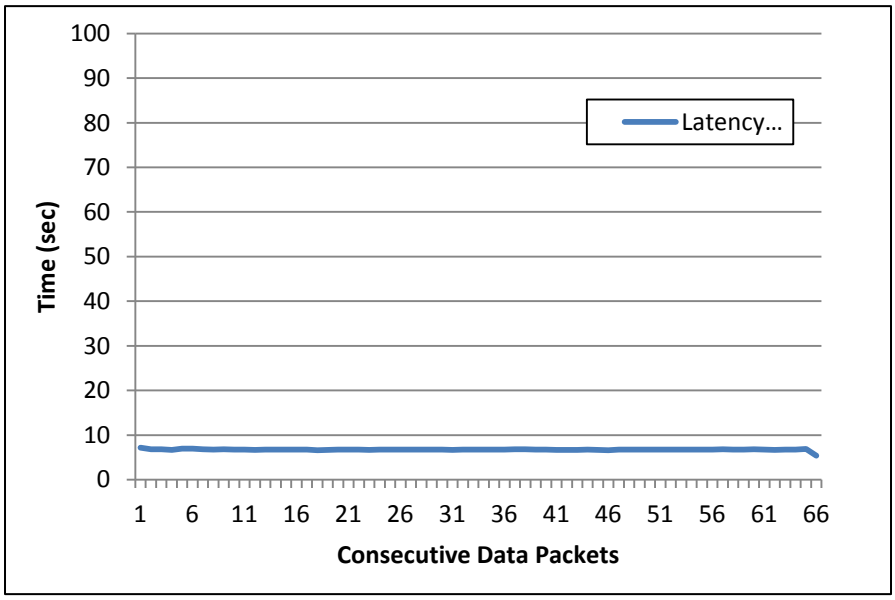

Figure 14. Effects of a one SMS message every ten seconds transmission rate on SMS message transmission latency.

3. One Hundred Kilobyte SMS Message Analysis

The 100 KB data experiment only yielded a data file delivery success rate of 60 percent. The errors consisted of a single missing SMS message in half of the corrupt data

files, an extra SMS message in 25 percent, and one SMS message delivered out of order in the remaining 25 percent. A total of 6490 SMS messages were transmitted with 6486 arriving in tact and in order, yielding a successful SMS message delivery rate of 99.997 percent. The low overall data file delivery rate for a 100 KB data file is a byproduct of one data file being transmitted through 649 SMS messages.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	sec	sec	sec
0	23617.696	23615.000	23621.492
1	23604.152	23601.000	23607.825
2	23606.808	23604.000	23610.388
3	23619.392	23613.000	23623.290
4	23606.831	23604.000	23610.442
5	23606.625	23600.000	23610.526
6	23620.684	23617.902	23624.554
7	23620.002	23617.391	23623.432
8	23604.246	23601.188	23606.931
9	23607.004	23604.153	23610.243
Average	23611.344	23607.763	23614.912
Standard Deviation	7.083	7.195	7.261

Table 8. 100 KB data file transmission with a one SMS message every 36.363 seconds transmission rate.

The 100 KB data file took an average of 23611.34 seconds (6 hours and 33.52 minutes) to transmit and 23607.76 seconds (6 hours and 33.42 minutes) to receive. The average elapsed time to transmit and receive a 100 KB data file was 23614.91 seconds (6 hours and 33.58 minutes).

Table 8 provides the key timing results from the 100 KB data file test with a one SMS message every 36.363 seconds transmission rate.

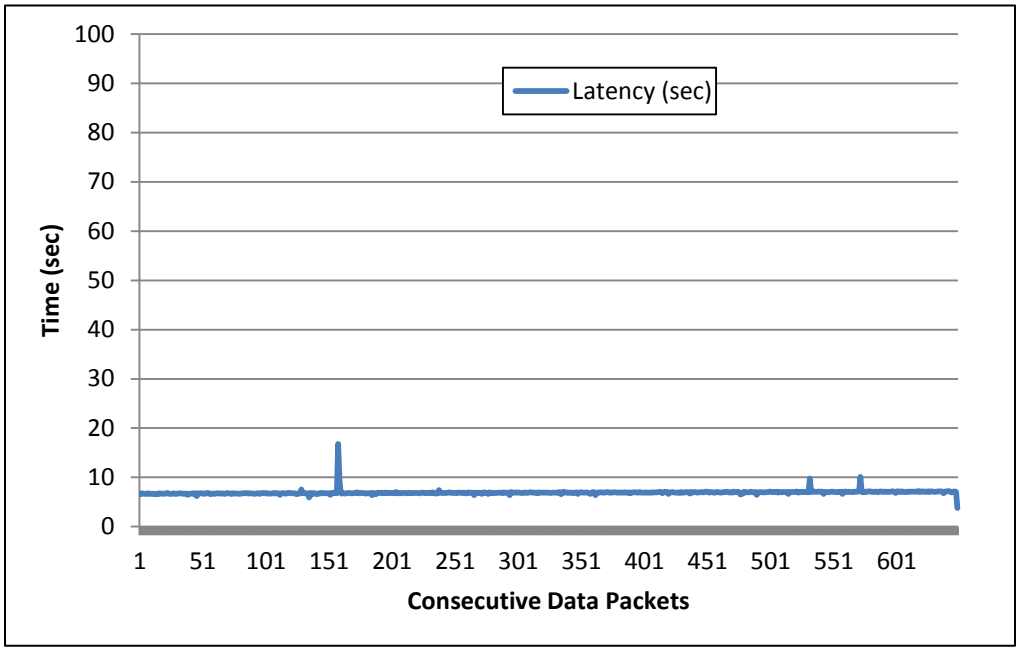

Figure 15. Effects of a one SMS message every 36.363 seconds transmission rate on SMS message transmission latency.

The average time interval between SMS message transmissions was 36.440 seconds. This transmission rate produced an arrival rate of one SMS message every 36.620 seconds. With this transmission rate, a SMS message spent an average of 6.909 seconds in transit. Figure 15 displays the increased latency experienced by consecutively transmitting SMS messages at a rate of one SMS message every with a 36.363 seconds.

E. ANALYSIS

1. Data File Size

Transmitting data files of various sizes produces some interesting results. Initially, it was expected that the 100 KB data file would produce the highest data file delivery success rate because of utilization of the slowest transmission rate, but that was not the case. Due to the higher volume of transmitted SMS messages associated with the 100 KB data file, the probability of successfully transmitting nearly 650 consecutive SMS messages without error is lower than when transmitting lower volumes. Nonetheless, the time interval between transmissions of 36.363 seconds did produce the highest individual SMS message delivery success rate reinforcing the notion that the transmission rate must be slow enough to avoid an overflow condition on the mobile device's transmission queue. Table 9 provides the data file success rates associated with the varying file size experiments.

Under varying transmission rates, the 1 KB and the 10 KB data file tests were able to produce a perfect data file transmission rate. This could be attributed to the relatively lower volumes of transmitted SMS messages compared to the 100 KB data file.

When attempting to covertly transmit data through the SMS channel on an Android based mobile device, the maximum data file size should be no larger than 13.53 KB (ninety-nine 140 byte SMS messages). Utilizing a maximum data file size of no larger than 11.2 KB (80 SMS messages) would allow the user to transmit 20 SMS messages per hour.

Data File Size	Transmission Time Interval	Successful Data File Delivery Rate
KB	seconds	percent
1	1	100.00
10	1	75.00
10	4	90.00
10	7	90.00
10	10	100.00
100	36.363	60.00

Table 9. Successful Data File Delivery Rates

2. Transmission Rate

Varying transmission rates produced expected results. The slower transmission rates of SMS messages produced the highest successful delivery rate for the individual SMS messages. Figure 16 presents the effects of transmission rates on SMS message latency. Table 10 provides the successful SMS message delivery rate associated with transmission rate. Using a one SMS message every second transmission rate for the 10 KB data file produced a successful data file delivery rate of 75.00 percent. This transmission rate also generated a successful SMS message delivery rate of 97.80 percent.

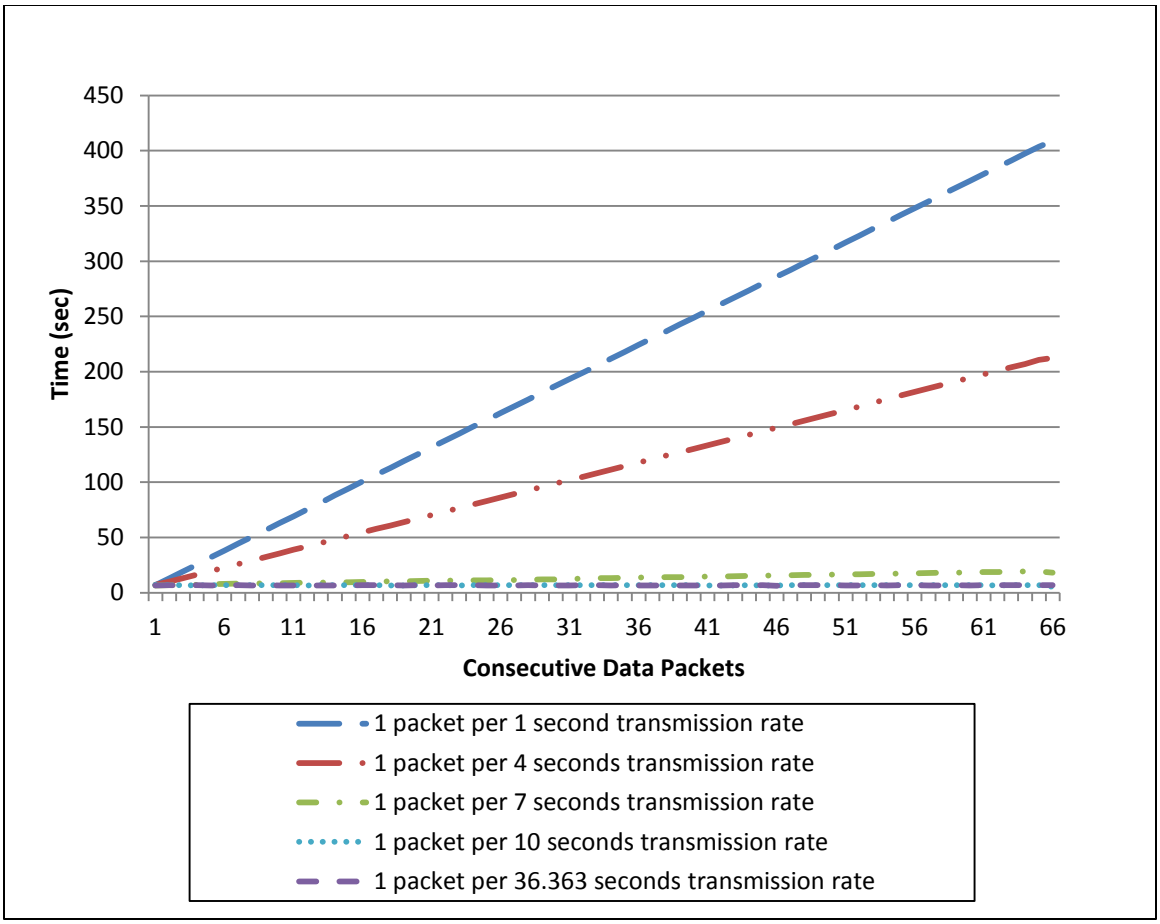

Figure 16. Effects of transmission rate on SMS message latency.

A transmission rate of one data SMS message every second also generated the highest average latency per transmitted SMS message of 206.443 seconds. This value is concerning for three reasons. One, the longer the SMS message is in the mobile device's transmission queue, the more likely it may be incorrectly processed. Two, the longer the SMS message is in this queue, the more likely the queue is going to experience an overflow condition. Lastly, the more congested the transmission queue becomes, the slower the mobile device is going to perform for the end user.

Decreasing the transmission rate to one SMS message every 4 seconds yielded better results. Transmitting one SMS message every four seconds generated a 90 percent data file and 99.39 percent SMS message delivery rate. This transmission rate also reduced the average SMS message latency to 111.243 seconds.

Transmission Time Interval	Successful SMS Message Delivery Rate
seconds	percent
1	97.80
4	99.39
7	99.85
10	100.00
36.363	100.00

Table 10. Successful SMS message Delivery Rates

The transmission rate of one SMS message every seven seconds also produced some promising results. This transmission rate produced a successful data file delivery rate of 90 percent and a successful SMS message delivery rate of 99.848 percent. Additionally, this rate decreased the average SMS message latency to 13.135 seconds.

Using a transmission rate of one SMS message every 10 seconds generated the most promising results. With this transmission rate, a successful delivery rate of 100 percent was achieved for both the SMS messages and data files. The average latency for a SMS message was observed to be 6.741 seconds.

Based on the results presented in this research, it is recommended that a transmission rate of one SMS message every ten seconds be utilized. This transmission rate

yielded a perfect SMS message and data file delivery rate while simultaneously producing the lowest per SMS message latency. The one SMS message every 36.363 seconds transmission rate produced an equal SMS message delivery rate, but it requires more overall elapsed time and therefore not as desirable.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

A. SUMMARY

This research aimed to address one question: Can data be transferred covertly from an Android based mobile device without drawing any suspicion from the end user? The answer to this question is yes, but several considerations must be taken into account in order to reliably conduct the data transmission in a clandestine fashion.

To achieve covert data transmissions, transmission rates and data file size must be taken into account to ensure reliable data delivery and prevent congestion on the mobile device's transmission channel. Throughout the transmission experiments, it was observed that that transmitting data files smaller than 13.53 KB in size yielded the most successful delivery rate. Transmitting data files of 10 KB in size or smaller generated at least an 75.0 percent rate of successful file delivery. Transmitting a 100 KB sized data file only yielded a success rate of 60 percent.

Additionally, it was found that using a transmission rate of one data message every ten seconds produces the best combination of transmission latency, total elapsed time for transmission, and data delivery rate. Using a transmission rate of one SMS message every ten seconds yielded a successful delivery rate of 100 percent for both SMS messages and data files. Furthermore, the transmission rate produced an average SMS message latency of 6.741 seconds. Through analysis of the one SMS message every

36.363 seconds transmission rate, it was observed that SMS message transmission latency would not improve by slower using transmission rates.

Other characteristics which were considered included suppression of the application user interface; thread management to push a majority of the workload to background threads; preventing services, activities, and broadcast receivers not belong to the application from having access to the covert data transmitting service; and broadcasting intents to initiate the covert transmission service.

B. FUTURE WORK

This research suggests a number of avenues for further research in covert application operation on mobile devices.

1. Hiding the Language Model on the Mobile Device

This research presented promising results for transmitting data over a mobile device's SMS channel without the end user's awareness. A covert transmission technique would serve no purpose if the end user could locate and identify data queued to be transmitted on a mobile device. An exploration of the mobile device's file system and an understanding of steganographic concealing techniques could lead to a method to successfully prevent an end user from discovering queued data before transmission.

2. Operating as an Embedded Service

This research produced an Android application capable of transmitting SMS messages through an SMS channel while not disrupting the end user's operational experience. Time

did not permit for an exploration of hiding the application's components. Developing a technique which queries a suspicious user's mobile device for frequently used applications could be developed and the covert SMS transmitting application's components could be embedded within a frequently used application. A possible method to install the covert transmitting application would involve an update of a frequently used application to the suspicious end user. The end user would update their frequently used application while unknowingly also be installing the embedded covert SMS transmitting application.

3. Special Short Message Service (CSMS & MMS)

Transmitting data through single SMS messages presented unexpected challenges on the Android mobile device: namely, the 100 SMS messages per hour limit and subsequent user alerts. An exploration of Special SMS Messages, to include concatenated SMS messages and media message service, could produce a more efficient method to transmit data.

4. Signal Strength effects on Transmission Rates

Signal strength and quality play an important factor in any transmission channels reliability. In this research, the mobile device was positioned within one meter of the mobile network's antenna. An exploration of signal strength and transmission rates could improve delivery rates.

5. Mobile Device Alert Recognition and Suppression

As mention before, Android mobile devices present the end user with an alert whenever SMS transmission rates exceed 100 messages per hour. Developing a method to suppress alerts would aid in enabling transmitting larger data files. Using a ten second pause between transmissions on a 13.53 KB data file only requires 16.33 minutes. Then an hour must expire before more transmissions can commence.

6. Central Processor Unit Consumption and Effects on Transmission Rates

During the course of this research, CPU consumption was not explored. Though the covert SMS transmitting application did not seem to impose any delay on user experience, an understanding of the application's CPU usage could enable transmission rates to be throttled down when the user's applications place high demands on the mobile device's CPU.

7. Energy Consumption and Its Effects on Transmission Rates

This research used a mobile device that was constantly plugged into a power source. Transmitting a 100 KB sized data file with a transmission rate of one SMS message every 36.363 seconds took an average of 6 hours and 33.52 minutes to complete. This duration of operation has the potential to produce a severe drain on a mobile device's energy resources. Enabling the covert SMS transmitting application to check battery power level and subsequently modify transmission rates could prove useful in preventing user suspicion.

C. CONCLUDING REMARKS

This research explored the feasibility of covertly transmitting data through an Android-based mobile device's SMS channel. Various transmission rates and data file sizes were utilized in an effort to determine which combination produced the most reliable delivery rate while simultaneously providing the end user with an uninterrupted user experience. The best results were found when transmission rates were reduced to one every ten seconds. Furthermore, focusing on transmitting data files smaller than 13.53 KB also improved reliability. Practical application of this research includes child monitoring, employee monitoring, law enforcement applications, and intelligence gathering.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: ANDROID APPLICATION

A. MANIFEST.XML

This portion of the Android code establishes access and permissions for the whole application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
      package="netgroup.SMSmanager"
      android:versionCode="1"
      android:versionName="1.0">
      <!--  Started up with this state above -->
      <application
           android:icon="@drawable/icon"
           android:label="@string/app_name">
          <activity
              android:name="."MainActivity"
              android:theme=@android:style/Theme.NoDisplay" >
             <intent-filter>
                  <action android:name="android.intent.action.MAIN" />
             </intent-filter>
          </activity>
          <service
             android:enabled="true"
             android:exported="false"
             android:name="."SMSReceiverService" />
          <service
             android:enabled="true"
             android:exported="false"
             android:name="."SendSmsService" />
          <receiver
             android:name="."SMSReceiver"
             android:enabled="true">
             <intent-filter>
```

```
                <action
                android:name="android.provider.Telephony.SMS_RECEIVED
                " />
            </intent-filter>
        </receiver>
        <receiver
            android:name=."StartupReceiver"
            android:enabled="true">
            <intent-filter>
                <action
                android:name="android.intent.action.BOOT_COMPLETED"
                />
            </intent-filter>
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="4" />
    <uses-permission
    android:name="android.permission.RECEIVE_SMS"></uses-permission>
    <uses-permission
    android:name="android.permission.SEND_SMS"></uses-permission>
    <uses-permission
    android:name="android.permission.WRITE_SMS"></uses-permission>
    <uses-permission
    android:name="android.permission.READ_SMS"></uses-permission>
    <uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-
    permission>
    <uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-
    permission>
    <uses-permission android:name="android.permission.WAKE_LOCK" />

</manifest>
```

B. BROADCAST RECEIVERS

1. SMSReceiver

This portion of the application serves as a Broadcast Receiver listening for inbound SMS messages. Once a SMS message is received, the SMSC address is evaluated. If the

SMSC address matches an SMSC contain in an internal list, an Intent is broadcast to initiate the SMSReceiver Service.

```
package netgroup.SMSmanager;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;

public class SMSReceiver extends BroadcastReceiver {

   @Override
   public void onReceive(Context context, Intent intent) {
      if(SmsLog.DEBUG) SmsLog.v("SMSReceiver: onReceive()");
      Bundle bundle = intent.getExtras();
      Object messages[] = (Object[]) bundle.get("pdus");
      SmsMessage smsMessage[] = new SmsMessage[messages.length];

      for (int n = 0; n < messages.length; n++) {
          smsMessage[n] = SmsMessage.createFromPdu((byte[])messages[n]);
      }
      if (checkSMSC(smsMessage[0].getServiceCenterAddress())){
         try {
            context.startService(new Intent(context,
               SMSReceiverService.class));
         } catch (Exception e) {
            System.out.println("Caught Exception: " +e);
         }
      }
   }  //end onReceive

   boolean checkSMSC(String line){
      boolean check = false; //KGK
      String[] smscNames = {"0122," "0124," "0123"};
      for (int i =0; i < smscNames.length; i++){
```

```
            if(line.equals(smscNames[i])){
               check = true;
               break;
            }
         }
         return check; // KGK
      } //end checkSMSC
}
```

2. StartupReceiver

This broadcast receiver initiates the software upon operating system startup. Once the startup Intent has been broadcast, this broadcast receiver starts the GSM network evaluation service.

```
package netgroup.SMSmanager;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class StartupReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      if(SmsLog.DEBUG) SmsLog.v("StartupReceiver: onReceive()");
      System.out.println("Starting StartupReceiver.onReceive()");
      try {
         context.startService(new Intent(context,
            SMSReceiverService.class));
      } catch (Exception e) {
         System.out.println("Caught Exception: " +e);
      }
   }
}
```

C. SMSRECEIVER SERVICE

This Service is initiated by the applications broadcast receiver. Once it is activated, it evaluates the LAC and CellID to ensure they match values contained within internal lists. If both values match know values an Intent is broadcast to initiate the SMS Sending Service.

```
package netgroup.SMSmanager;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.HandlerThread;
import android.os.IBinder;
import android.os.Process;
import android.telephony.TelephonyManager;
import android.telephony.gsm.GsmCellLocation;

public class SMSReceiverService extends Service {

   @Override
   public void onCreate() {
      if(SmsLog.DEBUG) SmsLog.v("SMSReceiverService:
         onCreate()");
      HandlerThread thread = new HandlerThread(SmsLog.LOGTAG,
         Process.THREAD_PRIORITY_BACKGROUND);
      thread.start();
   } // end onCreate

   @Override
   public void onStart(Intent intent, int startId) {
      if(SmsLog.DEBUG) SmsLog.v("SMSReceiverService: onStart()");
      TelephonyManager tm = (TelephonyManager)
         getSystemService(Context.TELEPHONY_SERVICE);
```

```
   GsmCellLocation location = (GsmCellLocation)
      tm.getCellLocation();

   if (checkLAC(Integer.toString(location.getLac())) &&
     checkCellID(Integer.toString(location.getCid()))){
      try {
         if(SmsLog.DEBUG) SmsLog.v("SMSReceiverService:
            onReceive()");
            startService(new Intent(this,
               SendSmsService.class));
      } catch (Exception e) {
         System.out.println("Caught Exception: " +e);
      }
   }
}  //end onStart

@Override
public void onDestroy() {
   onDestroy();
} //end onDestroy

@Override
public IBinder onBind(Intent intent) {
   return null;
}

boolean checkLAC(String line){
   boolean check = false;
   String[] lacNames = {"1000," "1001," "1002"};
   for (int i =0; i < lacNames.length; i++){
      if(line.equals(lacNames[i])){
         check = true;
         break;
      }
   }
```

```
      return check;
   } // end checkLAC

   boolean checkCellID(String line){
      boolean check = false;
      String[] cellIDNames = {“10,” “11,” “12”};
      for (int i =0; i < cellIDNames.length; i++){
         if(line.equals(cellIDNames[i])){
            check = true;
            break;
         }
      }
      return check;
   }  // end checkCellID
}
```

D. SMS SENDING SERVICE

This service reads in the data, composes the SMS messages, and transmits the SMS messages.

```
package netgroup.SMSmanager;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.app.PendingIntent;
import android.content.Context;
import android.os.AsyncTask;
import android.os.SystemClock;
import android.telephony.SmsManager;
import android.telephony.TelephonyManager;
import android.telephony.gsm.GsmCellLocation;
import java.io.*;

public class SendSmsService extends Service {
```

```
@Override
public IBinder onBind(Intent arg0) {
   return null;
}

String testFileLoc = "/sdcard/tmp/test.txt";
String resultsFileLoc = "/sdcard/tmp/transResults.txt";
String packetTimes = "/sdcard/tmp/transPacketTimes.txt";
File testFile = new File(testFileLoc);
long testFileSize = testFile.length();

/** Called when the service is first created. */
@Override
public void onCreate() {
   super.onCreate();
   if(SmsLog.DEBUG) SmsLog.v("SendSmsService: onCreate()");
}

@Override
public void onStart(Intent intent, int startid) {
   if(SmsLog.DEBUG) SmsLog.v("SendSmsService: onStart()");
   new SendSMSBackGround().execute();
}

@Override
public void onDestroy() {
   super.onDestroy();
}

class SendSMSBackGround extends AsyncTask<Void,Void,Void>{
   @Override
   protected Void doInBackground(Void... unused) {

      String phoneNo = "2103";
      BufferedReader buffRead = null;
```

```
   int numberRuns = 10;
   int start = 0;

   int little_pause_time = 1000;
   int little_end_pause_time = 360000;

   int med_pause_time = 10000;
   int medium_end_pause_time = 3600000;

   int big_pause_time = 36363;

   int pause_time = 0;
   int end_pause_time = 0;

   char secretChar1 = 13;
   char secretChar2 = 13;
   String secretWord = Character.toString(secretChar1) +
      Character.toString(secretChar2);
   GsmCellLocation location = null;
   int cellID = 0;
   int lac = 0;

   try{
      TelephonyManager tm = (TelephonyManager)
         getSystemService(Context.TELEPHONY_SERVICE);
      location = (GsmCellLocation) tm.getCellLocation();
      cellID = location.getCid();
      lac = location.getLac();
   } catch (Exception e) {
      System.out.println("Caught Exception: " +e);
   }
// * Assign as early 1hr sleep period to avoid 100 sms/hour limit
// SystemClock.sleep(medium_end_pause_time);

   while (true) {
      for (int i = start; i < numberRuns; i++){
```

```
int msg_ctr = 0;
String wordstring = secretWord+secretWord;

long startTimeE = System.currentTimeMillis();
long startTime = System.nanoTime();
SmsManager sms = SmsManager.getDefault();

if (testFileSize <= 5000){
   pause_time = little_pause_time;
   end_pause_time = little_end_pause_time;
} else if ((testFileSize <= 11000) && (testFileSize >
   5000)) {
   pause_time = med_pause_time;
   end_pause_time = medium_end_pause_time;
} else {
   pause_time = big_pause_time;
   end_pause_time = big_pause_time;
}
try {
   buffRead =  new BufferedReader(new
      FileReader(testFileLoc));
   BufferedWriter packetResults = new BufferedWriter(new
      FileWriter(packetTimes, true));
   int charac = -1;
   charac = buffRead.read();
   while (charac != -1) {
      if (wordstring.length() >= 159) {
         // add 160th character
         wordstring += Character.toString((char)charac);
         try{
            PendingIntent sent =
               PendingIntent.getBroadcast(getBaseContext
               (), 0, new Intent(), 0);
            sms.sendTextMessage(phoneNo, null,
               wordstring, sent, null);
            msg_ctr += 1;
            packetResults.write("Packet: " + msg_ctr +
               "," Time: " + System.currentTimeMillis() +
               "\n");
```

```
                        } catch (Exception e) {
                            System.out.println("Caught Exception: " +e);
                        }
                        wordstring = secretWord;
                        System.gc()
                        SystemClock.sleep(pause_time);
                    } else {
                        wordstring += Character.toString((char)charac);
                    }
                    charac = buffRead.read();
                }
// Check if last SMS is larger than the max of 160 characters.  If it
is, create 2 messages.
                if (wordstring.length() < 160 ){
                    try {
                        PendingIntent sent =
                            PendingIntent.getBroadcast(getBaseContext(),
                            0, new Intent(), 0);
                        sms.sendTextMessage(phoneNo, null, secretChar1
                            + wordstring, sent, null);
                        msg_ctr += 1;
                        packetResults.write("Packet: " + msg_ctr + ,"
                            Time: " + System.currentTimeMillis() +
                            "\n\n");
                    } catch (Exception e) {
                        System.out.println("Caught Exception: " +e);
                    }
                } else {
                    try {
                        PendingIntent sent =
                            PendingIntent.getBroadcast(getBaseContext(),
                            0, new Intent(), 0);
                        sms.sendTextMessage(phoneNo, null, wordstring,
                            sent, null);
                        msg_ctr += 1;
                        packetResults.write("Packet: " + msg_ctr + ,"
                            Time: " + System.currentTimeMillis() +
                            "\n");
                    } catch (Exception e) {
                        System.out.println("Caught Exception: " +e);
```

```
   }
   // Invoke pause
   SystemClock.sleep(pause_time);
   try {
      PendingIntent sent =
         PendingIntent.getBroadcast(getBaseContext(),
         0, new Intent(), 0);
      sms.sendTextMessage(phoneNo, null, secretChar1
         + secretWord, sent, null); //add \r to make
         \r\r\r for last sms
      msg_ctr += 1;
      packetResults.write("Packet: " + msg_ctr + ,"
         Time: " + System.currentTimeMillis() +
         "\n\n" );
   } catch (Exception e) {
      System.out.println("Caught Exception: " +e);
   }
}
long stopTimeE = System.currentTimeMillis();
long stopTime = System.nanoTime();  // Better for
   Elapsed Time
long elapseTime = stopTime - startTime;
String testResults1 = "Run: " + i +," Test File Size:
   " +testFileSize +" Bytes, Elapsed Time to
   Transmit: " +elapseTime +" ns";
String testResults2 = ," System Start Time: "
   +startTimeE +" ms, System Stop Time: " +stopTimeE
   +" ms\n";
try{
   BufferedWriter results = new BufferedWriter(new
      FileWriter(resultsFileLoc, true));
   results.write(testResults1);
   results.write(testResults2);
   results.close();
} catch (Exception e) {
   System.out.println("Caught Exception: " +e);
}
System.gc();
//Close external files
packetResults.close();
buffRead.close();
```

```
// Invoke a pause after each iteration to avoid 100 sms/hour
                  if (i < (numberRuns-1)){
                     SystemClock.sleep(end_pause_time);
                  }
               } catch (Exception e) {
                  System.out.println("Caught Exception: " +e);
               }
            }
            break;
         }
         return null;
      }
   }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: SMQUEUE.CPP MODIFICATIONS

This code reflects changes made to Smqueue.cpp file. The changes take effect near line 340 of the original code. This code was modified to capture the flagged SMS messages and record pertinent data.

```
case REQUEST_MSG_DELIVERY:
   /* We are trying to deliver to the handset now (or again after
      congestion).  */
   // Only print delivering msg if delivering to non-
   // localhost.
   if (0 != strcmp("127.0.0.1,"qmsg->parsed->req_uri->host)) {
      LOG(INFO) << "Delivering `" << qmsg->qtag << "` from " << qmsg-
         >parsed->from->url->username << " at " << qmsg->parsed-
         >req_uri->host << ":" << qmsg->parsed->req_uri->port << ."";
   }
   sprintf(temp_buff, qmsg->text);
   c = temp_buff;
   for (int i = 0; i<500; i++){
      c++;
      if (*(c-4) == 13 && *(c-3) == 10 && *(c-2) == 13 && *(c-1) == 10
         && *(c-0) == 13 && *(c+1) == 13 && *(c+2) == 13 && *(c+3) ==
         13){
         timeval time1;
         gettimeofday(&time1, NULL);
         ofstream packet_file;
         packet_file.open(packetResultsFile, ios::app);
         packet_file << "First Packet Time: " << time1.tv_sec << " sec,
            " << time1.tv_usec << " us" << "\n";
         packet_file.close();
         ofstream results_file;
         results_file.open(resultsfile, ios::app);
         results_file << "Run Start Time: " << time1.tv_sec << " sec, "
            << time1.tv_usec << " us";
         results_file.close();
         c += 4;
```

```
      ofstream unigram_file;
      unigram_file.open(textoutfile, ios::app);
      while (*c != 0){
         unigram_file << *c;
         c++;
      }
      unigram_file.close();
      sec_msg_flag = 1;
   } else if (*(c-4) == 13 && *(c-3) == 10 && *(c-2) == 13 && *(c-1)
      == 10 && *(c-0) == 13 && *(c+1) == 13 && *(c+2) == 13){
      timeval time1;
      gettimeofday(&time1, NULL);
      ofstream packet_file;
      packet_file.open(packetResultsFile, ios::app);
      packet_file << "End Packet Time: " << time1.tv_sec << " sec, "
         << time1.tv_usec << " us" << "\n\n";
      packet_file.close();
      ofstream results_file;
      results_file.open(resultsfile, ios::app);
      results_file << ," Run End Time: " << time1.tv_sec << " sec, "
         << time1.tv_usec << " us" << "\n";
      results_file.close();
      c += 3;
      ofstream unigram_file;
      unigram_file.open(textoutfile, ios::app);
      while (*c != 0){
         unigram_file << *c;
         c++;
      }
      unigram_file.close();
      sec_msg_flag = 1;
   } else if (*(c-4) == 13 && *(c-3) == 10 && *(c-2) == 13 && *(c-1)
      == 10 && *(c-0) == 13 && *(c+1) == 13){
      timeval time1;
      gettimeofday(&time1, NULL);
      ofstream packet_file;
      packet_file.open(packetResultsFile, ios::app);
```

```
            packet_file << "Mid Packet Time: " << time1.tv_sec << " sec, "
               << time1.tv_usec << " us" << "\n";
            packet_file.close();
            c += 2;
            ofstream unigram_file;
            unigram_file.open(textoutfile, ios::app);
            while (*c != 0){
               unigram_file << *c;
               c++;
            }
            unigram_file.close();
            sec_msg_flag = 1;
         }
         if (*c == 0){
            break;
         }
      }
   // FIXME, if we can't deliver the datagram we
   // just do the same thing regardless of the result.
      if (sec_msg_flag == 1){
         set_state(qmsg, DELETE_ME_STATE);
      } else if (my_network.deliver_msg_datagram(&*qmsg) && sec_msg_flag
         == 0){
         set_state(qmsg, REQUEST_DESTINATION_SIPURL);
      } else {
         set_state(qmsg, REQUEST_DESTINATION_SIPURL);
      }
      break;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: EXPERIMENT RESULTS

A. SMS MESSAGE TRANSMISSION, ARRIVAL, AND ELAPSED TIMES

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	seconds	seconds	seconds
0	6.489	41.888	48.843
1	6.376	41.658	91.274
2	6.377	41.894	134.020
3	6.381	41.896	176.817
4	6.386	41.661	219.249
5	6.447	41.696	261.593
6	6.397	41.898	304.247
7	6.390	41.658	346.784
8	6.382	39.777	387.402
9	6.381	39.536	427.836
10	6.383	42.129	470.813
11	6.399	41.423	512.894
12	6.383	42.136	555.843
13	6.385	41.665	598.393
14	6.609	42.125	49.236
15	6.569	41.437	91.324
16	6.481	41.903	133.881
17	6.513	41.664	176.332
18	6.518	42.134	219.180
19	6.541	39.782	259.716
20	6.536	41.668	302.074
21	6.547	41.898	344.687
22	6.562	41.894	387.273
23	6.458	42.633	430.598
24	6.625	41.194	472.527
25	6.527	41.901	515.056
26	6.575	41.665	557.429
27	6.496	41.898	600.030
Average	6.468	41.597	323.575
Standard Deviation	0.083	0.724	173.382

Table 11. 1 KB data file transmission with one SMS message per second transmission rate.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	seconds	seconds	seconds
0	69.239	471.262	477.626
1	70.894	452.162	474.875
2	70.550	470.553	476.893
3	70.430	470.552	477.393
4	70.427	471.257	478.362
5	70.657	310.249	316.843
6	70.490	471.725	478.343
7	70.315	471.726	478.605
8	70.544	469.373	476.400
9	70.474	467.491	473.436
10	69.185	470.788	477.860
11	70.319	471.025	477.458
12	70.992	456.405	478.833
13	69.204	472.907	479.146
14	69.108	480.679	487.743
15	69.117	471.721	478.883
16	69.171	473.134	480.193
17	69.173	474.529	480.752
18	69.185	469.611	476.272
19	69.145	470.082	476.283
Average	69.931	461.861	470.110
Standard Deviation	0.724	36.181	36.185

Table 12. 10 KB data file transmission with one SMS message per second transmission rate.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	seconds	seconds	seconds
0	265.880	473.368	479.553
1	265.582	466.787	472.544
2	265.651	470.079	476.063
3	265.620	470.791	477.165
4	265.748	470.786	477.450
5	265.498	469.840	475.093
6	265.771	471.726	478.096
7	265.682	472.432	479.439
8	265.594	476.431	483.095
9	265.921	467.955	476.776
10	265.743	468.900	478.103
11	265.396	469.131	475.246
12	265.521	475.963	482.149
13	265.699	467.028	473.192
14	264.169	470.066	476.305
15	264.192	472.907	478.667
16	264.188	472.912	478.375
17	264.195	474.085	480.558
18	264.197	471.021	477.542
19	264.236	468.908	474.589
20	265.768	471.019	480.512
21	265.764	470.079	476.935
22	265.692	483.721	489.323
23	265.623	470.083	476.712
24	265.719	492.205	498.572
25	265.495	468.658	474.910
26	265.556	469.374	475.125
27	265.577	471.949	478.181
28	265.775	470.789	477.373
29	265.522	470.788	477.611
Average	265.366	471.993	478.508
Standard Deviation	0.606	5.018	4.968

Table 13. 10 KB data file transmission with one SMS message per four seconds transmission rate.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	seconds	seconds	seconds
0	459.338	469.375	474.852
1	459.234	478.308	484.830
2	459.195	467.957	474.741
3	459.224	469.371	475.630
4	459.183	470.788	476.790
5	459.166	470.785	477.451
6	459.161	469.135	475.488
7	459.137	468.898	474.884
8	459.210	477.362	483.478
9	459.255	470.296	476.181
10	459.124	467.723	473.828
11	459.138	471.725	477.321
12	459.250	471.252	477.383
13	459.168	471.261	477.623
14	459.197	470.317	476.665
15	459.173	469.373	474.670
16	459.168	469.586	476.666
17	459.125	469.845	476.351
18	459.184	471.491	477.608
19	459.295	469.837	476.110
Average	459.196	470.734	476.927
Standard Deviation	0.056	2.665	2.718

Table 14. 10 KB data file transmission with one SMS message per seven seconds transmission rate.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	seconds	seconds	seconds
0	655.802	651.080	660.692
1	654.365	652.960	659.420
2	654.285	652.959	659.380
3	654.265	652.723	659.199
4	654.255	652.718	659.498
5	654.235	652.958	659.108
6	654.303	652.723	659.209
7	654.283	652.487	658.708
8	654.267	652.955	658.531
9	654.257	652.723	659.831
10	655.997	651.536	660.608
11	655.818	654.859	661.692
12	655.792	654.611	660.592
13	655.673	654.136	660.529
14	655.703	654.371	660.332
15	655.728	654.128	661.098
16	655.651	654.376	661.093
17	655.731	654.374	660.106
18	655.887	650.843	660.990
19	655.879	654.372	660.585
20	655.596	655.546	661.175
Average	655.132	653.307	660.113
Standard Deviation	0.762	1.260	0.900

Table 15. 10 KB data file transmission with one SMS message per ten seconds transmission rate.

Run	Elapsed Transmission Time	Elapsed Arrival Time	Total Elapsed Time
	seconds	seconds	seconds
0	23617.696	23615.000	23621.492
1	23604.152	23601.000	23607.825
2	23606.808	23604.000	23610.388
3	23619.392	23613.000	23623.290
4	23606.831	23604.000	23610.442
5	23606.625	23600.000	23610.526
6	23620.684	23617.902	23624.554
7	23620.002	23617.391	23623.432
8	23604.246	23601.188	23606.931
9	23607.004	23604.153	23610.243
Average	23611.344	23607.763	23614.912
Standard Deviation	7.083	7.195	7.261

Table 16. 100 KB data file transmission with one SMS message per 36.363 seconds transmission rate.

B. CONSECUTIVE 1 KB DATA FILE LATENCY

Data File	Elapsed Time	Total Time to Completion
	seconds	seconds
1	41.888	48.843
2	41.658	91.274
3	41.894	134.020
4	41.896	176.817
5	41.661	219.249
6	41.696	261.593
7	41.898	304.247
8	41.658	346.784
9	39.777	387.402
10	39.536	427.836
11	42.129	470.813
12	41.423	512.894
13	42.136	555.843
14	41.665	598.393
15	42.125	49.236
16	41.437	91.324
17	41.903	133.881
18	41.664	176.332
19	42.134	219.180
20	39.782	259.716
21	41.668	302.074
22	41.898	344.687
23	41.894	387.273
24	42.633	430.598
25	41.194	472.527
26	41.901	515.056
27	41.665	557.429
28	41.898	600.030

Table 17. Consecutively transmitted 1 KB data files and subsequent data file transmission latency.

C. CONSECUTIVE SMS MESSAGE LATENCY

1. 10 KB Data File

SMS Message	1 Second Transmission Interval	4 Second Transmission Interval	7 Second Transmission Interval	10 Second Transmission Interval
	seconds	seconds	seconds	seconds
1	6.756523592	6.9890355	6.5920656	7.199937
2	12.9056462	10.127773	6.8698203	6.8299574
3	19.17350287	13.819317	7.0476688	6.7936945
4	25.35641736	16.98804	7.2454217	6.6773826
5	31.97848342	20.122653	7.7925456	6.9272045
6	38.19575848	23.262346	7.9979637	6.977405
7	44.42599189	26.485035	8.2183315	6.8245941
8	50.59236991	29.791217	8.3724589	6.7726961
9	56.78879258	32.971073	8.4406478	6.8032773
10	62.9183402	36.131643	8.6374741	6.7672036
11	69.06731837	39.403836	8.8269982	6.7472764
12	75.43673574	42.511708	9.0283857	6.6862895
13	81.6153172	45.621177	9.2153324	6.752076
14	88.19419792	48.730927	9.408193	6.7686277
15	94.39581098	51.93359	9.5915337	6.7604242
16	100.5682261	55.111005	9.7888905	6.7686985
17	106.7679665	58.274095	10.003369	6.7470374
18	112.9558856	61.419867	10.21751	6.6307066
19	119.1728635	65.325868	10.428216	6.6426019
20	125.3469918	68.460007	10.603684	6.7588155
21	131.5445834	71.665233	10.792619	6.7487888
22	137.7605355	74.862602	11.00289	6.7628943
23	143.9535247	78.025043	11.107574	6.6370758
24	150.1293199	81.232695	11.228369	6.7470284
25	156.3442166	84.490795	11.328503	6.7593382
26	162.5291148	87.628427	11.431715	6.7600961
27	168.7532499	90.805155	11.495062	6.7486028
28	174.9417896	93.968057	11.697741	6.7648928
29	181.0816249	97.13495	11.947993	6.7104443
30	187.2919518	100.33568	12.044794	6.7354691
31	193.3742897	103.51409	12.253984	6.7044784
32	199.5397033	106.60022	12.862227	6.7294906
33	205.6458818	109.77867	13.105262	6.7146687

34	211.7668003	112.95612	13.317252	6.760063
35	217.97497	116.16561	13.523166	6.7477217
36	224.1996853	119.38908	13.662351	6.7564834
37	230.4378215	122.57831	13.770492	6.7917641
38	236.5980586	125.67396	13.952326	6.777446
39	242.7685478	128.79128	14.15188	6.7550931
40	248.979438	131.99476	14.34788	6.7756127
41	255.1851792	135.19581	14.55855	6.6695659
42	261.3122303	138.34568	14.777567	6.6713299
43	267.4505436	141.45641	15.015041	6.6835791
44	273.6323266	144.6053	15.243444	6.7064276
45	279.8077501	147.75629	15.440827	6.6714326
46	285.9440062	150.93543	15.639037	6.6182458
47	292.1172974	154.09387	15.836104	6.7256932
48	298.3236341	157.29711	16.034247	6.7450058
49	304.5560049	160.53415	16.236957	6.7578178
50	310.7713425	163.65664	16.416842	6.7640239
51	316.9484514	166.88603	16.616681	6.7506946
52	323.1809945	170.09723	16.828234	6.75216
53	329.4275725	173.29309	17.036119	6.7437879
54	335.630339	176.47965	17.210618	6.7464573
55	341.8019659	182.68604	17.366604	6.7667174
56	347.9997646	183.02275	17.538526	6.71406
57	354.2110854	186.21994	17.771615	6.8029797
58	360.3923954	189.42022	18.008919	6.7760156
59	366.5819024	192.62532	18.239442	6.7662497
60	372.7332313	195.82743	18.47109	6.7768741
61	378.9214504	199.059	18.662222	6.7701011
62	385.0789304	202.19492	18.855265	6.6611702
63	391.2691529	205.30501	19.030519	6.7485416
64	397.4970527	208.48043	19.159617	6.7242061
65	403.6589996	211.78903	19.367016	6.8552661
66	408.6684946	213.67478	18.167182	5.3560237

Table 18. Transmission rate and SMS message latency for 10 KB data file.

2. 100 KB Data File

SMS Message	Transmission Latency	SMS Message	Transmission Latency	SMS Message	Transmission Latency
	seconds		seconds		seconds
1	6.572138184	218	6.840211426	435	6.953656982
2	6.800266968	219	6.829297974	436	6.946858521
3	6.677685974	220	6.825277771	437	6.698623962
4	6.729012268	221	6.814336243	438	7.03214502
5	6.651408508	222	6.797484924	439	6.903398804
6	6.638970459	223	6.789701294	440	6.954317017
7	6.743697693	224	6.893293884	441	7.0069245
8	6.671771912	225	6.878906219	442	6.979396423
9	6.721923584	226	6.864518555	443	6.91012854
10	6.650053772	227	6.786130981	444	6.954241943
11	6.638588318	228	6.892037964	445	6.948403015
12	6.681087036	229	6.81464978	446	7.057989929
13	6.60548468	230	6.81224585	447	6.9882901
14	6.654670593	231	6.916709229	448	6.976352783
15	6.585745911	232	6.773205017	449	7.081128906
16	6.7505979	233	6.935471313	450	7.07585791
17	6.736899475	234	6.861900696	451	7.010718567
18	6.714527466	235	6.787793518	452	6.872605469
19	6.65032666	236	6.7291297	453	6.979885254
20	6.700001831	237	6.782458984	454	7.028742737
21	6.685601257	238	7.421860535	455	6.957078491
22	6.732671509	239	6.878459778	456	6.88609552
23	6.847337463	240	6.866861328	457	6.873731018
24	6.717084045	241	6.865719482	458	7.0329245
25	6.647419312	242	6.8038078	459	7.017699524
26	6.633830933	243	6.855914185	460	6.949687988
27	6.680510559	244	6.851758057	461	6.87520575
28	6.794079773	245	6.898850586	462	6.920611755
29	6.613379211	246	6.948926025	463	7.032143494
30	6.648557739	247	6.883539978	464	6.957281799
31	6.694661804	248	6.822418945	465	7.014430481
32	6.746431946	249	6.871132996	466	7.111286743
33	6.739516418	250	6.861237549	467	7.033261475
34	6.672679688	251	6.848391968	468	6.904923706

35	6.722264221
36	6.643756653
37	6.689915527
38	6.678710327
39	6.425911682
40	6.64073877
41	6.69382074
42	6.678599609
43	6.780182983
44	6.649417053
45	6.752060974
46	6.20657074
47	6.774580261
48	6.752336487
49	6.747051941
50	6.794432007
51	6.657476257
52	6.698822876
53	6.743515808
54	6.67240802
55	6.837371033
56	6.712657471
57	6.575328247
58	6.681442322
59	6.73702948
60	6.657530518
61	6.70823877
62	6.752261047
63	6.742935059
64	6.731187988
65	6.721156189
66	6.769710266
67	6.69459314
68	6.619476013
69	6.725910522
70	6.836268433
71	6.745017944
72	6.674516052
73	6.6663125
74	6.777608215

252	6.781267761
253	6.887219299
254	6.871845642
255	6.804374512
256	6.90500177
257	6.8319422
258	6.877900696
259	6.747719727
260	6.790657043
261	6.961964722
262	6.830905823
263	6.880956238
264	6.807287964
265	6.905966309
266	6.361466309
267	6.875995422
268	6.802980225
269	6.799282288
270	6.894505249
271	6.830899048
272	6.635816772
273	6.865200989
274	6.96380658
275	6.831225525
276	6.819977417
277	6.628914795
278	6.974587769
279	6.854707764
280	6.782529541
281	6.890003479
282	6.815637573
283	6.863398804
284	6.847960266
285	6.896683533
286	6.891654236
287	6.818617126
288	6.858810974
289	6.968629944
290	6.833904541
291	6.893251709

469	6.954808044
470	6.943243042
471	6.872333313
472	7.097174011
473	7.022215271
474	7.013994324
475	7.07074823
476	6.937466736
477	6.516742493
478	6.918476868
479	6.732403503
480	7.072549744
481	6.945296448
482	6.991795105
483	6.977875732
484	6.968926758
485	7.067664246
486	7.060791199
487	6.988980957
488	6.984417175
489	7.027823792
490	6.475744751
491	6.995286621
492	6.98547998
493	7.037188721
494	6.963445496
495	6.954703003
496	6.945971741
497	6.989086487
498	7.034553162
499	7.023340759
500	7.070142212
501	7.101825806
502	6.976903259
503	7.019460938
504	7.06086731
505	6.988947266
506	7.09719519
507	6.904472229
508	6.999803101

75	6.704900208
76	6.75670752
77	6.681778564
78	6.673392761
79	6.718929199
80	6.646472168
81	6.687726929
82	6.793643005
83	6.787529541
84	6.769349243
85	6.810815552
86	6.797162537
87	6.727192261
88	6.770602783
89	6.75765625
90	6.679574036
91	6.729421997
92	6.644822754
93	6.701955505
94	6.744011292
95	6.792857544
96	6.667068298
97	6.772735291
98	6.820345215
99	6.755490723
100	6.805944641
101	6.735513428
102	6.777260437
103	6.697939209
104	6.734671753
105	6.734906555
106	6.782435486
107	6.8368078
108	6.703077881
109	6.74838208
110	6.793347229
111	6.791112183
112	6.418854736
113	6.699027954
114	6.866859985

292	6.875968689
293	6.868228516
294	6.378220093
295	7.023363708
296	6.891234192
297	6.871806763
298	6.921897766
299	6.900903442
300	6.950492798
301	6.883291199
302	6.874576721
303	6.856059509
304	6.83963446
305	6.82752948
306	6.940302734
307	6.928859253
308	6.85200531
309	6.899802185
310	6.949490051
311	6.989697937
312	6.902973389
313	6.895612549
314	6.876942871
315	6.863273987
316	6.790574951
317	6.962241821
318	6.89557074
319	6.884835938
320	6.870924255
321	6.897110474
322	6.88960907
323	6.878504272
324	6.928813049
325	6.913831238
326	6.849038513
327	6.839384399
328	6.886565491
329	6.929487976
330	6.914765808
331	6.910238037

509	7.049266968
510	6.976269165
511	7.023679749
512	7.005621277
513	6.996034302
514	7.0408703
515	6.617427734
516	7.016250305
517	6.998845337
518	7.057245056
519	7.042527466
520	7.034555725
521	7.086139282
522	7.073195984
523	6.89137677
524	7.049783508
525	6.977451294
526	7.082291931
527	7.015929199
528	7.125792542
529	6.992020508
530	7.034250977
531	6.96122229
532	9.747829224
533	7.037066467
534	7.139409729
535	7.06213446
536	7.104807251
537	7.031541504
538	7.12489325
539	7.10583197
540	7.090841736
541	7.07161499
542	7.110797913
543	6.625881042
544	7.017212402
545	7.056296021
546	7.037491943
547	6.963676208
548	7.044334778

115	6.794149353
116	6.720927246
117	6.708600281
118	6.803117737
119	6.850507019
120	6.84133197
121	6.76659198
122	6.76107074
123	6.749175781
124	6.736665527
125	6.548922729
126	6.705016113
127	6.746050293
128	6.729084045
129	7.541900391
130	6.821371948
131	6.867210571
132	6.799151245
133	6.726586975
134	6.774743225
135	5.948454712
136	6.822460144
137	6.465629028
138	6.801070251
139	6.845234497
140	6.712421021
141	6.582041931
142	6.732025269
143	6.78102771
144	6.705575806
145	6.809783264
146	6.854947205
147	6.7790495
148	6.77270752
149	6.758365967
150	6.742684265
151	6.795348267
152	6.436488281
153	6.838783203
154	6.887219604

332	6.837132202
333	6.94922821
334	7.054535217
335	6.569038269
336	6.850971252
337	7.073876587
338	6.884840332
339	6.926712158
340	6.976153809
341	6.847335693
342	6.896534363
343	6.882879822
344	6.755768921
345	6.911673523
346	6.910988831
347	6.954419189
348	6.647794983
349	6.872257324
350	6.960218506
351	6.94936499
352	6.887932251
353	6.925963257
354	6.912763245
355	6.962169495
356	6.892471252
357	6.759450287
358	6.626429321
359	6.85898645
360	7.028998047
361	6.949303528
362	6.406641968
363	6.868374268
364	6.854903564
365	6.959467224
366	6.949715515
367	6.995237
368	6.928760498
369	6.915886169
370	6.962832336
371	6.950091736

549	7.086090942
550	7.065920715
551	7.052061462
552	7.030010681
553	7.00848822
554	6.983519714
555	7.084516724
556	7.011729004
557	7.096552979
558	6.608546753
559	7.116925598
560	7.100401855
561	7.080009033
562	6.998428223
563	7.098440247
564	7.025965698
565	7.061941956
566	6.992003967
567	7.096480225
568	7.079769287
569	7.060907227
570	7.152477051
571	7.136145691
572	10.07106372
573	7.050514038
574	6.978740479
575	7.082500549
576	7.060070129
577	7.04422821
578	7.138935242
579	7.147069489
580	7.155203735
581	7.026290039
582	7.067478455
583	7.057462524
584	7.097831726
585	7.125851746
586	6.992788818
587	7.097824524
588	7.136042664

155	6.763863464
156	6.80553717
157	6.80083197
158	16.79456592
159	9.554928284
160	6.885978271
161	6.707490295
162	6.752788025
163	6.800260254
164	6.72727002
165	6.769843994
166	6.814965515
167	6.863526367
168	6.794580078
169	6.785756042
170	6.772206482
171	6.818228455
172	6.802900513
173	6.971443726
174	6.786246277
175	6.828995972
176	6.81230127
177	6.800787292
178	6.853594238
179	6.791422974
180	6.725009705
181	6.776010193
182	6.766627441
183	6.750830078
184	6.802690247
185	6.379976746
186	6.839244812
187	6.771921509
188	6.577743286
189	6.802451721
190	6.904255676
191	6.834410583
192	6.820571777
193	6.867457031
194	6.797307251

372	6.998627197
373	6.922735962
374	6.912237732
375	6.905434204
376	6.955471375
377	6.943298767
378	6.934483643
379	6.916618042
380	6.906206726
381	6.942764526
382	6.87489801
383	6.919087036
384	6.966286743
385	6.902378052
386	6.9498078
387	6.876977051
388	6.854340637
389	6.964323792
390	6.7795448
391	6.884698608
392	6.992959961
393	6.985246094
394	6.976353027
395	6.897915039
396	6.889088745
397	7.004102539
398	6.872027283
399	6.923457153
400	6.913039795
401	6.905910522
402	6.947876282
403	6.877216736
404	6.868565247
405	6.920975708
406	6.912815063
407	6.958616516
408	6.892043579
409	6.935546753
410	6.933366089
411	6.977625183

589	7.12633252
590	7.028627075
591	7.076027771
592	7.126914185
593	7.052507813
594	7.094829041
595	7.020061768
596	7.062861328
597	7.16345282
598	7.152374939
599	7.006863464
600	6.817430542
601	7.166270752
602	7.095976318
603	7.18902948
604	7.061754822
605	7.158040161
606	7.066771729
607	7.048670593
608	7.031955994
609	7.070636536
610	7.052553955
611	7.093690002
612	7.080814453
613	7.117083435
614	7.094693726
615	7.083726501
616	7.117230957
617	7.102535278
618	7.212068787
619	7.023147034
620	7.127153809
621	7.115440247
622	7.153899231
623	7.021637146
624	7.123794434
625	7.166328064
626	7.0335271
627	7.131829346
628	7.114495239

195	6.894182373
196	6.874830994
197	6.799445679
198	6.903605713
199	6.843254822
200	6.828218994
201	6.817017517
202	6.807562317
203	6.798535217
204	7.028161926
205	6.773640686
206	6.757145752
207	6.805164551
208	6.7927323
209	6.838743958
210	6.863819519
211	6.853853027
212	6.789625183
213	6.775369019
214	6.891005798
215	6.764328186
216	6.807770752
217	6.79463855

412	7.036214722
413	7.022923462
414	7.018449219
415	6.764196777
416	6.87319342
417	7.093505737
418	6.958779724
419	7.003850281
420	6.64754303
421	6.931486206
422	6.978584778
423	7.023509766
424	6.948682495
425	6.879527527
426	6.873576782
427	6.923135498
428	6.974678589
429	6.854164551
430	7.012442017
431	6.886016541
432	6.927348694
433	6.980891541
434	7.024640076

629	7.14821875
630	7.069282227
631	7.047091858
632	7.095715149
633	7.077722656
634	7.183902771
635	7.175473022
636	7.160101929
637	7.076103271
638	6.769221741
639	7.102393982
640	7.142695618
641	7.1773302
642	7.217385559
643	7.132365417
644	6.992943176
645	6.989015381
646	7.095635742
647	7.134211548
648	7.054099548
649	3.809190796

Table 19. SMS message latency for 100 KB data file with 36.363 seconds transmission interval.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] The Nielsen Company, "2010 media industry fact sheet," Accessed July 2011, http://blog.nielsen.com/nielsenwire/press/nielsen-fact-sheet-2010.pdf.

[2] CTIA-The Wireless Association, "Wireless quick facts. 2010 year end figures," December 2010, http://www.ctia.org/media/industry info/index.cfm/AID/10323.

[3] www.wikipedia.org, "List of countries by number of mobile phones in use," Last accessed August 2011, http://en.wikipedia.org/wiki/List of countries by number of mobile phones in use.

[4] Portio Research, "Mobile messaging futures - 2011-2015," August 2011, http://www.portioresearch.com/MMF11-15.html.

[5] National Communications System, "SMS over SS7, technical information bulletin 03-2 (NCS TIB 03-2)," December 2003, http://www.ncs.gov/library/tech bulletins/2003/tib 03-2.pdf.

[6] Rear Adm. G. Smith, "Presentation on the communication efforts in afghanistan," given at the Naval Postgraduate School on July 12, 2011.

[7] 3GPP TS 23.038, version 10.0.0, release 10, "Alphabets and language-specific information," March 2011.

[8] 3GPP TS 23.040 version 10.0.0 release 10, "Technical realization of the short message service (SMS)," April 2011.

[9] M. Glendrange, K. Hove, and E. Hvideberg, "Decoding GSM," Norwegian University of Science and Technology, June 2010.

[10] 3GPP TS 05.05 version 8.20.0 release 1999, "Digital cellular telecommunications system (Phase 2+); Radio transmission and reception," November 2005.

[11] Ettus Research, "Universal software radio peripheral," Accessed September 2011, http://www.ettus.com/downloads/ettus_ds_usrp_v7.pdf

[12] T. Turletti, "A brief overview of the GSM radio interface," Telemedia Networks and Systems Group, Laboratory for Computer Science, Massachussets Institute of Technology, March 1996.

[13] A. Loula, "OpenBTS installation and configuration guide v0.1," May 25, 2009, http://gnuradio.org/redmine/attachments/139/OpenBTS_Guide_En_v0.1.pdf.

[14] GNU Radio, "The openBTS wiki subspace," last accessed August 04, 2011, http://gnuradio.org/redmine/projects/gnuradio/wiki/OpenBTS.

[15] Asterisk, "About the asterisk project," last accessed August 15, 2011, http://www.asterisk.org/asterisk.

[16] GNU Radio, "Welcome to GNU radio!" last accessed August 07, 2011, http://gnuradio.org/redmine/projects/gnuradio/wiki.

[17] GNU Radio, "GNU radio!," last accessed August 18, 2011, http://gnuradio.org/redmine/projects/gnuradio/wiki/OpenBTSsmqueue.

[18] Android Developers, "What is android?" last accessed August 05, 2011, http://developer.android.com/guide/basics/what-is-android.html.

[19] Android Developers, "Activity," last accessed August 01, 2011, http://developer.android.com/reference/android/app/Activity.html.

[20] M. Murphy, *The Busy Coder's Guide to Android Development*, CommonsWare, March 2011.

[21] Android Developers, "Service," last accessed August 20, 2011, http://developer.android.com/reference/android/app/Service.html#ServiceLife cycle.

[22] Android Developers, "Broadcastreceiver," last accessed August 20, 2011, http://developer.android.com/reference/android/content/BroadcastReceiver.html.

[23] Android Developers, "Intents and intent filters," last accessed August 08, 2011, http://developer.android.com/guide/topics/intents/intents-filters.html.

[24] Android Developers, "Processes and threads," last accessed August 09, 2011, http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html.

[25] M. Rafique and M. Farooq, "Exposing the CCN (criminal cellular network): SMS vulnerabilities to embed high capacity covert channels," National University of Computer & Emerging Sciences, Pakistan, 2010.

[26] K. Rafat and M. Sher, "Digital steganography for ASCII text documents," International Islamic University, Islamabad, Pakistan, 2009.

[27] xdadevelopers, "Increase the limit on android," June 2010, http://www.xda-developers.com/android/increase-the-sms-limit-on-android/.

[28] GNU Radio, "OpenBTS/smqueue," last accessed August 14, 2011, http://gnuradio.org/redmine/projects/gnuradio/wiki/OpenBTSsmqueue.

[29] ubuntu manuals, "Time - overview of time and timers," last accessed August 07, 2011, http://manpages.ubuntu.com/manpages/jaunty/man7/time.7.html.

[30] Android Developers, "Time," last accessed August 19, 2011, http://developer.android.com/reference/java/sql/Time.html.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

7. Dr. Robert Beverly
   Naval Postgraduate School
   Monterey, California

8. Dr. Craig Martell
   Naval Postgraduate School
   Monterey, California

9. Kalle Kangas
   Naval Postgraduate School
   Monterey, California