**MAASAI MARA UNIVERSITY**

**SCHOOL OF SCIENCE**

**DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES**

**BACHELOR OF SCIENCE COMPUTER SCIENCE**

**TITLE: E-SMS (ENCRYPTED SMS)**

**NAME: ANDREW MAKORI NYACHAKI**

**REG NO: BS02/061/2014**

**COURSE CODE: COM 423**

**COURSE TITLE: COMPUTER SCIENCE PROJECT**

**DATE OF SUBMISSION: 14TH MAY, 2018**

**PROJECT SUPERVISOR: MR. OPE JUSTUS**

The project report to be submitted to the Computer and Information Sciences department of Maasai Mara University in the partial fulfillment for the requirement of Computer Science Project for the award of a Bachelor Degree in Computer Science.

# **DECLARATION**

I hereby declare that this project report is based on the results found by myself while researching, Materials of work found by other researcher are mentioned by reference.

NAME: ANDREW MAKORI NYACHAKI

REG NO.: BS02/061/2014

Signed……………………………. Date………………….

SUPERVISOR DECLARATION.

This project report has been submitted with my approval to the university supervisor.

NAME: MR. OPE JUSTUS

Signed………………… Date…………………

# **ACKNOWLEDGMENTS**

Special thanks to the Almighty God for all His grace through this period, sincere gratitude to my parents Mr. and Mrs. Nyachaki and the special friend for their support, ample time and understanding I also want to express my gratitude to Mr. Abraham Matheka, Mr. Bii Joash and Mr. Ope Justus, my project advisors, without their supervision, help and valuable suggestion it would not be possible for me to complete my project. Also, my gratitude to my fellow classmates and friends for making it a success.

# **DEDICATION**

I dedicate the project to my parents, friends and my esteemed user. I also dedicate it to my supervisors without whose support all would be in vain and the school at large.

# **ABSTRACT**

The emergence of the short messaging service has real transformed communication and information industry. It has transformed the nature of transferring and sharing information. Information is being shared electronically through text message being sent over the network using mobile devices like phones that include smartphones and other handheld devices. This render the messages vulnerable to attacks from crackers and attackers. These are those who intercept the messages while they are on transit and they are able to modify the message before it reaches its destination. These compromises the authenticity, reliability, confidentiality and the security of the message. Which also may change the desired meaning that the intended message was meant to deliver to the receiver.

To curb all these challenges, we need to have a secure channel through which our communication through the SMS secure, or secure the message before it is shared over the network. So, securing the message we have decided to develop an android SMS application system that will encrypt the message using the symmetric AES algorithm. The message will be encrypted into a ciphertext that will be sent to the intended recipient. When the recipient receives the message, he has to decrypt in order to read the content of the message and reply back. The ciphertext will not have meaning to the attacker or interceptor since they are random values and characters therefore hard to change meaning or otherwise it will not be decrypted on the other side if it was modified for the same key.

The E\_Sms application will ensure the privacy of the message in case it was delivered to a wrong recipient by accident, secure from interceptors like crackers, eavesdroppers and hackers. This will make all information shared reliable and confidential maintaining its integrity. The application is an android application that its intended for all android devices supporting android API 21 and above.

Contents

[**DECLARATION** ii](#_Toc513566674)

[**ACKNOWLEDGMENTS** iii](#_Toc513566675)

[**DEDICATION** iv](#_Toc513566676)

[**ABSTRACT** v](#_Toc513566677)

[**LIST OF FIGURES** x](#_Toc513566678)

[**LIST OF ABBREVIATIONS.** xi](#_Toc513566679)

[**CHAPTER ONE** 1](#_Toc513566680)

[**1.0 Introduction** 1](#_Toc513566681)

[**1.1 Background of the study.** 1](#_Toc513566682)

[**1.2 Problem statement.** 2](#_Toc513566683)

[**1.3 Justification** 2](#_Toc513566684)

[**1.4 Scope** 2](#_Toc513566685)

[**1.5 Aim** 3](#_Toc513566686)

[**1.6 Objectives** 3](#_Toc513566687)

[**CHAPTER TWO** 4](#_Toc513566688)

[**LITERATURE REVIEW** 4](#_Toc513566689)

[2.0.1 How SMS works and architecture 4](#_Toc513566690)

[2.0.2 Encryption systems in use today. 5](#_Toc513566691)

[2.1 Encryption key storage and key management 6](#_Toc513566692)

[2.1.0 Types of encryption keys 7](#_Toc513566693)

[2.1.1 How encryption key systems 8](#_Toc513566694)

[2.1.2 Key Management 10](#_Toc513566695)

[2.2 The domains to secure encryption keys 10](#_Toc513566696)

[2.3 Key distribution in symmetric encryption 12](#_Toc513566697)

[2.3.1 Key Distribution and Authentication Protocol 13](#_Toc513566698)

[**CHAPTER THREE** 15](#_Toc513566704)

[**DEVELOPMENT METHODOLOGY** 15](#_Toc513566705)

[3.0.1 Software Processes. 15](#_Toc513566706)

[3.0.2 Process Models 15](#_Toc513566709)

[3.0.3 The Incremental Model. 15](#_Toc513566710)

[3.0.4 Boehm’s Spiral Model. 17](#_Toc513566711)

[3.0.5 Agile Process Model. 18](#_Toc513566712)

[3.0.6 Waterfall Model. 20](#_Toc513566713)

[3.1 Evaluation methodology 22](#_Toc513566714)

[**CHAPTER FOUR** 24](#_Toc513566715)

[**SYSTEM ANALYSIS AND REQUIREMENT**. 24](#_Toc513566716)

[4.0 Current systems. 24](#_Toc513566717)

[4.0.1 Message in a Bottle. 24](#_Toc513566718)

[4.0.1 CryptoSMS 24](#_Toc513566719)

[4.1 The proposed system. 25](#_Toc513566720)

[4.1.1 Conceptual Framework of the proposed system. 25](#_Toc513566721)

[4.1.2 System flowchart 26](#_Toc513566722)

[4.2 Requirement specification. 27](#_Toc513566723)

[4.2.1 Hardware Requirements 27](#_Toc513566724)

[4.2.2 Software Requirements 27](#_Toc513566725)

[4.2.3 Functional Requirements 28](#_Toc513566726)

[4.2.4 Non-Functional Requirement 28](#_Toc513566727)

[**CHAPTER FIVE** 29](#_Toc513566728)

[**SYSTEM DESIGN AND IMPLEMENTATION** 29](#_Toc513566729)

[5.0 Technical Design. 29](#_Toc513566730)

[5.1 Use case diagram 30](#_Toc513566731)

[5.2 Interaction diagram 31](#_Toc513566732)

[5.3 System implementation 32](#_Toc513566733)

[5.3.1 Development environment implementation 32](#_Toc513566734)

[5.3.2 Graphical User Interface and coding Implementation 32](#_Toc513566735)

[5.4 Testing and validation 35](#_Toc513566741)

[5.5 System Changeover 36](#_Toc513566742)

[**CHAPTER SIX** 38](#_Toc513566743)

[**LIMITATION, CONCLUSION AND RECOMMENDATION** 38](#_Toc513566744)

[6.0 Limitation 38](#_Toc513566745)

[6.1 Recommendation 38](#_Toc513566746)

[6.3 Conclusion 38](#_Toc513566747)

[**BIBLIOGRAPHY** 39](#_Toc513566748)

[**APPENDIX** 41](#_Toc513566749)

[Appendix I: User manual 41](#_Toc513566750)

[Appendix II: Sample Codes 43](#_Toc513566751)

[Appendix III: Time and resource scheduling. 45](#_Toc513566752)

[Budget 45](#_Toc513566753)

[Time Scheduling 46](#_Toc513566754)

# **LIST OF FIGURES**

[Figure 1: Symmetric key flowchart 16](#_Toc510430166)

[Figure 2: Asymmetric key flowchart 17](#_Toc510430167)

[Figure 3: Incremental model diagram 27](file:///D:\love\Com423Report.docx#_Toc510430168)

[Figure 4: Spiral model diagram 28](#_Toc510430169)

[Figure 5: Waterfall model diagram 31](#_Toc510430170)

[Figure 6: Conceptual framework of the proposed system 35](file:///D:\love\Com423Report.docx#_Toc510430171)

[Figure 7: System flowchart of the proposed system 36](file:///D:\love\Com423Report.docx#_Toc510430172)

[Figure 8: Technical design 39](file:///D:\love\Com423Report.docx#_Toc510430173)

[Figure 9: User case diagram 40](file:///D:\love\Com423Report.docx#_Toc510430174)

[Figure 10: Interaction diagram 41](file:///D:\love\Com423Report.docx#_Toc510430175)

[Figure 11: Coding and interface coding menu in Android Studio IDE 42](#_Toc510430176)

[Figure 12: MainActivity.java code implementation 43](#_Toc510430177)

[Figure 13: Manifest permissions 43](#_Toc510430178)

[Figure 14: Broadcast receiver functions 44](#_Toc510430179)

[Figure 15: Display message code implementation 44](#_Toc510430180)

[Figure 16: User interface for sending sms 52](#_Toc510430181)

[Figure 17: Receiver interface 53](#_Toc510430182)

**LIST OF TABLES**

[Table 1: Budget 55](#_Toc513033234)

[Table 2: Time schedule 56](#_Toc513033235)

# **LIST OF ABBREVIATIONS.**

AES- Advanced Encryption Standard

API- Application Programming Interface

CA- Certification Authority

CSP- Content Security Policy

DEA- Data Encryption Algorithm

DEK- Decryption Encryption Key

DES - Data Encryption Standard

FIPS- Federal Information Processing Standard

GCHQ- Government Communications Headquarters

GCM- Google Cloud Messaging

GHz- Giga Hertz

GSM- Global System for Mobile Communication

HSM- Hardware Security Module.

IDE- Integrated Development Environment

JDK- Java Development Kit

KDC- Key Distribution Center

KEK- Key Encryption Key

KMAPI- Key Management Application Program Interface

KMS- Key Management System

MB- Mega Bytes

NIST- National Institute of Standards and Technology

PKI- Public Key Infrastructure

RAM- Random Access Memory

RIF- Real Message Index File

SDK- Software Development Kit

SMS-Short Message Service

SMSC- Short Message Service Center

TLS- Transport Layer Security

VA- Validation Authority

VPN- Virtual Private Network

XML- eXtensible Markup Language

# **CHAPTER ONE**

# **1.0 Introduction**

Short Message services is a text messaging service component of phone web or mobile communication systems, using standardized communications protocols that allow the exchange of short text messages between fixed line or mobile. Users can send or receive text messages from a single user or several persons. Personal messages, email notifications, information services and many more. With this usable application SMS is now more and more common among mobile phone users. However, the security issue of the SMS remains an open challenge.

SMS is now a common communication tool. The security protection of SMS messages is not yet sophisticated and difficult to implement in practice. The confidentiality and integrity mechanisms are only specified as optional security measures that can be made available but they are not mandatory requirement for SMS system implementation. We are going to use symmetric cryptography for SMS transfer security using the advanced encryption standard algorithm.

# **1.1 Background of the study.**

These days most of the data is being exchanged electronically. The privacy of the data therefore, becomes a requirement. This is in order to maintain the authenticity. Legibility and reliability of the data being shared electronically. Encryption restricts unintended recipients from viewing the data that is confidential without authorization from the two parties sharing the information.

Encryption is the transformation of plaintext that is the data that can be read or the original message into ciphertext. The cipher needs decoding to be viewed. This can only be done with the only person who has the decryption key which is the secret key only known by the two parties sharing information. Plaintext is transformed to ciphertext via the encryption method or procedure.

Currently the SMS is widely used. This is simply because it is fast and convenient way of communication. It has no limitation to its users whatsoever. However, there is no security for messages that are transmitted via messaging, email over the handheld phones by the parties, this renders the process of communication via text messages insecure and inefficient.

Being no guaranteed security for the messages it leads to directly or indirectly to some problems where confidential information is accessed by unauthorized individuals like passwords and passcodes. Also, when one sends wrong message to a wrong individual via the mobile number accidentally. This can be complicated if the phone is lost or stolen.

By developing and implementing this messaging application, security and privacy on confidentiality of data will be sorted out. Also, the system will not only be available for public use but also for other sectors and industries like the banking sector and commercial industries in interacting with their customers. The application will be a user-friendly application since it will handle many users and guarantee their communication through short messages security and privacy.

# **1.2 Problem statement.**

The existing messaging platform allows texts to be sent as plaintext which renders the SMS to be easily intercepted and be replayed. These days where the number of users of this service has increased requires more security or security to be beefed up when using the services. To avoid this kind of problems we need to develop an encrypted android messaging application to encrypt the message making them secure for transmission over the network

# **1.3 Justification**

With the increasing number of mobile users, has triggered the increase in number of the messages being sent daily. The proposed messaging application will ensure or enhance privacy of the users. It will provide a secure messaging channel. Only the users with the right and correct keys are the only ones who will be able to read the messages that have been sent over or through this channel. This is simply because the messages remain encrypted throughout transmission and therefore, need decryption once received at the receiving end. This ensures privacy, security, authenticity, reliability and confidentiality of the message.

# **1.4 Scope**

The secure E-SMS application version 1.0 only runs in android phones. The application does not run on neither java enabled phones, Symbian phones, i-phones, and blackberries. However further studies can feature on how to implement on another platform running different operating systems apart from android operating system.

# **1.5 Aim**

To come up with a secure messaging application for android devices that will be used to encrypt a message and ensure end to end secure communication.

# **1.6 Objectives**

1. To develop a secure SMS application.
2. To ensure high confidentiality and improved security of message information.

# **CHAPTER TWO**

# **LITERATURE REVIEW**

Emergence of mobile technology services in the recent years extensively transformed mobile communication and information sharing. Among the most popular services is the Short Message services (SMS) in which billions of SMS are sent daily across the world (Baron & Harris, 2006). Short messages service has not only been used for individual conversations but in corporate (mobile banking), social and the political world. However, Studies have shown that the SMS channel is vulnerable for the man in the middle attacks and other hacking attacks. With billions of text messages sent in plain text, the integrity of the text messages and the privacy of the users is bound to be breached especially now with the availability of many software crackers available for free in the internet. This has led to studies to be carried out so as to improve the short messages services platform. This has recommended the use of encryption in this platform such that messages can be encrypted before transit over the network. This aims at understanding the vulnerabilities and the threats that exist within the system. Also, I discuss the aspects and trends of the SMS technology.

### 2.0.1 How SMS works and architecture

The Short Message Service (SMS) is a GSM technology that allows exchange of text messages up to 160 characters among mobile phones through the short message service Center (SMSC) of the particular network operator. The relative ease of the use of SMS makes it the most wanted means of communication among mobile users.

SMS uses the GSM special signaling channel instead of voice channel and is therefore a very cheap and reliable media channel. The SMSC receives the message from the mobile device and routes it to the destination device. The operation of SMSC is offered as a service by many service providers. The SMS processing computers usually run on corporate servers that are connected to the SMS network through specialized routers and gateways connected to the SMS centers of the mobile operators. These servers are assigned short numbers usually 3 to 6 digits long. These numbers are operator specific.

A base transceiver stations facilitates wireless communication between user equipment and a network. User Equipment are devices like mobile devices. Then the message is received by the mobile switching center which is the primary service delivery node for GCM/CDMA, responsible for routing voice calls as well as messages and other services like FAX. When SMS is transmitted from a cell phone, the message will be received by mobile carrier’s SMS Center (SMSC), do destination finding, and then send it to destination devices (cellphone). SMSC is SMS service center which is installed on mobile carrier core networks. Beside as SMS forwarding, SMSC also acts as temporary storage for SMS messages. So, if the destination cell phone is not active, SMS will store the message and then deliver it after the destination cell phone is active. As additional, SMSC also notify the sender whether the SMS delivering is success or not. However, SMSC cannot store the SMS message forever since the storage capacity is limited.

### 2.0.2 Encryption systems in use today.

Since early 50s, there has been a growing interest in *securing sensitive digital data*, for the purpose of storing it safely, as well as transmitting it securely over unsecured communication channels (El-Khalil & Keromytis, 2007). Bruce (Bruce,1996) demonstrated that ‗the initial attempts at modern data security came in the form of the Data Encryption Algorithm (DEA), developed in the 1970's and adopted (with some modification) by the United States Government as the Data Encryption Standard (DES).

(M & Devrim, 2010) in Cyprus evaluated encryption and decryption time for three algorithms RSA, Elliptic-curve and EIGamal to which plain text of different sizes is provided based on results one is chosen for further encryption. Their performance evaluation in securing SMS shows that key generation, encryption and decryption time increases with an increase in the key size: Large key size algorithms are not suitable for SMS encryption due to small memory and low computational power of mobile phones. However, the study focused on evaluating how the keys should be generated and not a secure means of sharing the keys.

(Rupa & Avadhani, 2009) in India proposed a cost-effective scheme which uses a concept called Cheating Text. The original message is embedded in a meaningful text cheating text. Here, index table called (Real Message Index File) RIF file is hashed and sent to the receiver along with the cheating text in which the original message is embedded. Authentication is achieved by verifying the hash value of the plain text (Albuja & Carrera). They introduced an SMS security framework, which allows programmers and users to exchange confidential, non-reputable and digitally signed text messages. This framework can fit in many development scenarios, such as commercial transactions or bureaucratic delegations. In addition, the proposed framework is highly flexible and efficient, since programmers can choose among several encryption algorithms according to the computational power and battery usage of each mobile device.

## 2.1 Encryption key storage and key management

Encryption key management is administering the full lifecycle of cryptographic keys and protecting them from loss or misuse. The lifecycle includes: generating, using, storing, archiving, and deleting of keys. Protection of the encryption keys includes limiting access to the keys physically, logically, and through user/role access (Townsendsecurity, Townsendsecurity, 2017). The proper management of cryptographic keys is essential to the effective use of cryptography for security. Keys are analogous to the combination of a safe. If a safe combination is known to an adversary, the strongest safe provides no security against penetration. Similarly, poor key management may easily compromise strong algorithms. (NIST Recommendation for Key Management (Townsendsecurity, Townsendsecurity, 2017). The protection of encryption keys, also called encryption key management, is critical to successful encryption. In fact, it is so crucial that most industry compliance regulations require or strongly recommend the use of an encryption key manager. When you encrypt sensitive data, a key is used to “lock” the data on encryption and also created to “unlock” the data on decryption by authorized users. If that key is stored on the same server as your encrypted data, then any hacker or malicious intruder will be able to decrypt and access plaintext data resulting in a data breach. In order to prevent this, you must store encryption keys in a separate location away from the encrypted data in a hardware security module (HSM), virtual appliance, or cloud key manager—dedicated key servers that store and manage encryption keys for data in databases, virtual systems, or the cloud (Pimpale, Rayarikar, & Upadhyay, October 2011)

Not protecting your encryption keys is a lot like leaving your house in the morning and taping your key to the front door. It would be easy to find your key there, but you’re practically inviting an unwanted intruder. Unauthorized access to encryption keys. In order to have effective encryption you must securely separate the data being encrypted from the keys performing that data encryption. An encryption key manager enables a secure channel between the encryption keys and wherever that data may reside. Technology has evolved to enable stronger management so that companies no longer need to leave their encryption keys vulnerable to attackers. If you do experience a breach of encrypted sensitive data, and you have secured your encryption keys away from the compromised data, many compliance regulations will consider that data “safe,” and you may be able to avoid breach notification (Pimpale, Rayarikar, & Upadhyay, October 2011).

### 2.1.0 Types of encryption keys

1. Symmetric keys (Data at Rest)- In symmetric key cryptography, the same encryption key is used to both encrypt and decrypt the data. This means of encryption is used primarily to protect data at rest. Example encrypt sensitive data stored in a database and decrypt it to plaintext when needed by an authorized user.

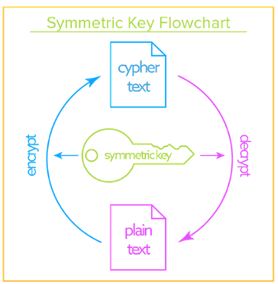


Figure 1: Symmetric key flowchart

1. Asymmetric keys (Data in Motion)- Asymmetric keys, on the other hand, are a pair of keys for the encryption and decryption of the data. Both keys are related to each other and created at the same time. They are referred to as a *public* and a *private* key:

*Public key* is primarily used to encrypt the data and can be freely given as it will be used to encrypt data, not decrypt it.

*Private key* is used to decrypt the data that it’s counterpart, the public key, has encrypted. This key must be safeguarded as it is the only key that can decrypt the encrypted data.

Asymmetric keys are used to secure data-in-motion. An example might be a virtual private network (VPN) connection. With a VPN: an AES symmetric session key is used to encrypt the data that is a public key is used to encrypt the session key and once the encrypted data is received, the private key is used to decrypt the session key so that is can be used to decrypt the data.

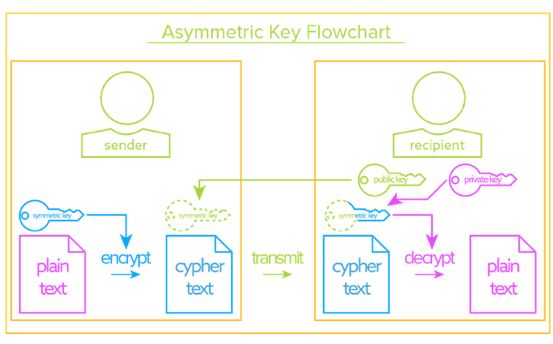


Figure 2: Asymmetric key flowchart

### 2.1.1 How encryption key systems

#### Symmetric Key Systems

First, let’s define a few terms:

1. ***Data encryption key (DEK):*** is an encryption key whose function it is to encrypt and decrypt the data.
2. ***Key encryption key (KEK*):** is an encryption key whose function it is to encrypt and decrypt the DEK.
3. ***Key management application program interface (KM API):*** is an application interface that is designed to securely retrieve and pass along encryption keys from a key management server to the client requesting the keys.
4. ***Certificate* *Authority (CA):*** is an entity that creates public and private keys, creates certificates, verifies certificates and performs other PKI functions.
5. ***Transport layer security (TLS):*** is a cryptographic protocol that provides security, through mutual authentication, for data-in-motion over a computer network.
6. ***Key Management System (KMS):*** is the system that houses the key management software

How it works when an authorized user accesses encrypted data.

1. A user requests to access encrypted data.
2. The database, application, file system, or storage then sends a DEK retrieval request to the client (KM API).
3. Next, the client (KM API) and KM verify each other’s certificates:
   1. The client (KM API) sends a certificate to the KM for verification.
   2. The KM then checks the certificate against the CA for authentication.
   3. Once the client (KM API) certificate has been verified, the KM then sends its certificate to the KM API for authentication and acceptance.
4. Once the certificates have been accepted, a secure TLS connection is established between the client (KM API) and the KM.
5. The KM then decrypts the requested DEK with the KEK
6. The KM sends the DEK to the client (KM API) over the encrypted TLS session.
7. The KM API then sends the DEK to the database, application, file system, or storage.
8. The database (may) cache the DEK in temporary secure memory.
9. The database, application, file system, or storage then sends the plaintext information to the user.

*Asymmetric key system*

1. The Sender and Recipient verify each other’s certificates:
   1. The sender sends a certificate to the recipient for verification.
   2. The recipient then checks the certificate against their Certificate Authority (CA) or an external Validation Authority (VA) for authentication.
   3. Once the sender’s certificate has been verified, the recipient then sends their certificate to the sender for authentication and acceptance.
2. Once the sender and recipient have mutual acceptance:
   1. The sender requests the recipient’s public key.
   2. The recipient sends their public key to the sender.
3. The sender creates an ephemeral symmetric key and encrypts the file to be sent. (an ephemeral symmetric key is a symmetric encryption key used only for one session)
4. The sender encrypts the symmetric key with the public key.
5. The sender then sends the encrypted data with the encrypted symmetric key.
6. The recipient receives the packet and decrypts the symmetric key with the private key.
7. The recipient decrypts the data with the symmetric key.

### 2.1.2 Key Management

When protecting sensitive Server data with en­cryption, these core principles of key manage­ment are essential:

* Encryption keys are stored away from the data they protect, usually on specially designed security devices or dedicated virtual services.
* Encryption keys are managed by individuals who do not have access to the data stored in the Server database (Separation of Duties).
* Encryption key management requires more than one security administrator to authenticate before performing any critical work on keys (Dual control).
* Key retrieval requests from users and applications are authenticated using industry standard methods.

## 2.2 The domains to secure encryption keys

1. Physical security- In NIST’s Special Publication 800-14, they offer this definition of physical security: “Physical and environmental security controls” should be “implemented to protect the facility housing system resources, the system resources themselves, and the facilities used to support their operation.” This may include things like; physical access controls, ports: FIPS 140-2 notes that in the case sending plaintext encryption keys, physical ports should be dedicated for only that purpose, make sure all physical environments housing the system have adequate, and current, fire suppression systems and many more.
2. Logical Access Security- involves separating the different cryptographic components housing the keys from the rest of the larger network. It includes three main items to be considered that is the interface that is the “logical interface(s) used for the input and output of plaintext cryptographic keys, cryptographic key components, authentication data, and CSPs may be shared physically and logically with other ports and interfaces of the cryptographic module.” And “the logical interfaces used for the input and output of plaintext cryptographic key components, authentication data, and CSPs shall be logically separated from all other interfaces using a trusted path.”, the DEK(s) should be logically separated from the data that is encrypted. This effectively keeps the DEK(s) from being used to decrypt the data in case unauthorized users gain access to the sensitive material, and finally, within the encryption key manager, the KEK(s) should be logically separated from the DEK(s). This ensures that though the database DEKs be compromised, they will be rendered unusable because the KEK is in a logically separate location from the DEKs.
3. User/Role Access- The core concept promulgated by NIST is the concept of least privilege: where you restrict “the access privileges of authorized personnel (e.g., program execution privileges, file modification privileges) to the minimum necessary to perform their jobs.”
4. Certification- Using NIST validated AES encryption and FIPS 140-2 compliant key management is critical to ensuring that your security solution will stand up to scrutiny in the event of a data breach. These certifications are difficult to acquire and are only given to encryption and key management systems that have been heavily tested against government standards. Using trusted third-party systems is typically the easiest way to acquire and implement this technology.
5. NIST Validated AES Encryption- The National Institute of Standards and Technology (NIST) established AES as the highest standard for encryption in 2001. AES supports nine modes of encryption, and NIST defines three key sizes for encryption: 128-bit, 192- bit, and 256-bit keys. Any encryption that you use to protect data at rest should be AES standard encryption.
6. FIPS 140-2 Compliant Key Management- The highest standard for encryption key

management is the Federal Information Processing Standard (FIPS) 140-2 issued by NIST. A key management hardware security module (HSM) with NIST FIPS 140-2 compliance will offer the highest level of security for any company.

## 2.3 Key distribution in symmetric encryption

Traditionally, symmetric encryption suffered one enormous shortcoming – it was necessary for either the sender or the recipient to create a key and then send it to the other party. While the key was in transit, it could be stolen or copied by a third party who would then be able to decrypt any ciphertexts encrypted with that key. Another problem is that a large number of key pairs are needed between communicating parties. This quickly becomes difficult to manage the more there are. This can be calculated as n(n-1)/2 where n is the number of communicating parties. For example, if ten parties want to communicate with each other securely they would need 45 different key pairs: 10(10-1)/2 = 45. This would increase to 4,950 if there were 100 communicating parties! (University, 2015)

The *key distribution problem,* affected anyone wishing to use encryption until the 1970s when a method of distributing keys without actually sending the keys themselves was developed independently by GCHQ in the United Kingdom and Whitfield Diffie and Martin Hellman in the United States. The British discovery was kept secret for many years, so today the solution is known as the Diffie–Hellman key exchange method (University, 2015)

Symmetric encryption methods have the advantage that encryption and decryption is extremely fast, thus ideal for transmitting large amounts of secure data over the network.

The major problem in symmetric key cryptography is that of the key distribution because the key must be shared secretly. Keys can be distributed by any one of the following ways:

|  |
| --- |
| * Sender can select the key and physically deliver it to receiver. |
| * A trusted third party can select the key and physically deliver it to the sender and the receiver. |
| * If sender and receiver have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key. |
| * If sender and receiver each has an encrypted connection to a third party, then the third party can deliver a key on the encrypted links to sender and receiver.   For the first and second option the key needs to be delivered manually for end to end encryption to take place, in the third option, there is a possibility of either link encryption or end to end encryption. However, if the attacker manages to have access to the key then all the subsequent keys are revealed to him or her even if frequent changes are made rendering it insecure. And finally, in the fourth option two kinds of keys are used   1. Session keys- when two systems wish to communicate, they establish a logical connection. For the session all user data is encrypted with a onetime key which expires at the end of the session period and the key is destroyed upon expiry of the session. 2. Permanent keys- it is used between entities for the purpose of distributing session keys. |

In the fourth option, it requires the Key Distribution Center (KDC) that determines which systems are allowed to communication with each other. A KDC provides a onetime session key when two systems establish a connection. If host A wishes to establish a connection with host B it sends a connection on request to the KDC and the communication between host S and the KDC is encrypted, using a master key only shared by A is used to encrypt the communication. If the KDC approves the communication, the KDC encrypts the session keys using the master key shared only by A and it share the session key with A and delivers it to A, and again it encrypts the session key and shares it and delivers it to B and they make their communication securely through the network.

### 2.3.1 Key Distribution and Authentication Protocol

## Authentication protocols are all about distribution and management of secret keys. Key distribution in a distributed environment is an implementation of distributed authentication protocols. Based on this idea many key distribution and authentication protocols have been proposed. Generally, all protocols assume that some secret information is held initially by each principal. Authentication is achieved by one principal demonstrating the other that it holds that secret information. All protocols assume that system environment is very insecure and is open for attack. So, any message received by a principal must have its origin authenticity, integrity and freshness verified. To achieve these goals, most protocols need to rely on an authentication server and this server should have the following features

## Should deliver good quality session keys and distribute them to the requesting principal securely.

## Maintain a table containing a name and a secret key for each principle. The secret key is used only to authenticate client processes to the authentication server and to transmit messages securely between client processes and the authentication server.

## Key distribution and authentication Protocols are divided into two categories to verify the authentication of a message. First category uses nonce and challenge/ response handshake to verify freshness, example is Needham-Schroeder Protocol. Second category uses timestamps and assumes that all machines in distributed system are clock-synchronized

|  |
| --- |
| *Needham-Schroeder Key Distribution Protocol* |
| It is a secret-key protocol based on nonce session keys i.e., number used once session keys and also provides a solution to authentication and key distribution based on an authentication server. This protocol is based on the generation and transmission of ticket by the authentication server. A ticket is an encrypted message containing a secret key for use in communication between A and B.  *Kerberos Protocol*  Kerberos protocol, each session key generated by the Kerberos KDC server will have time stamp associated with it, so that after some time it is going to expire. The server can easily identify the authorized client by checking the validity of the session key. |

# **CHAPTER THREE**

# **DEVELOPMENT METHODOLOGY**

### 3.0.1 Software Processes.

## Software processes are either classified as plan driven or agile processes. Plan driven processes are processes where all process activities are planned in advance and progress is measured against this plan. In Agile process planning is incremental and it is easier to change the process to reflect changing customer requirements. Boehm and Turner (2003) discuss each process is suitable for different types of software.

## Software processes can be improved by process standardization where the diversity of software processes across an organization is reduced, leading to improved communication and a reduction in training time and makes automated process support more economical. Also, standardization is the first step in introducing new software engineering methods and techniques and good software engineering processes.

### 3.0.2 Process Models

A process model is a simplified representation of a software process. Each model represents a process from a different perspective and thus, only provides partial information about that process. Process models are not definitive descriptions of software process. Rather they are abstractions of the process that can be used to explain different approaches of software development. The following models have been discussed and carefully analyzed with their advantages and disadvantages before evaluating and choosing on one.

### 3.0.3 The Incremental Model.

Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed. Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities.

It is part of either the agile or plan driven development approach. In a plan-driven approach, the system increments are identified in advance; if an agile approach is adopted, the early increments are identified but the development of later increments depends on progress and customer priorities. It focuses on the way we solve problems. It is easier and cheap to develop and make changes to a software incrementally.

Each increment or version of the system incorporates some of the functionality that is needed by the user. The early increments of the system include the most important or most urgently required functionality. Therefore, the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments.

Benefits of incremental model over the waterfall model:

1. The cost of accommodating changing customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.

2. It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented. Customers find it difficult to judge progress from software design documents.

3. More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain.

Problems with this model include:

1. The process is not visible. Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
2. System structure tends to degrade as new increments are added. Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly. Below is the diagrammatic representation of the incremental model.

Outline

Description.

Specification

Development

Validation

Initial Version

Intermediate

Version

Final Version

Figure : Incremental model diagram

### 3.0.4 Boehm’s Spiral Model.

A risk-driven software process framework (the spiral model) was proposed by Boehm (1988). The software process is represented as a spiral, rather than a sequence of activities with some backtracking from one activity to another. Each loop in the spiral represents a phase of the software process. Thus, the innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design, and so on. It combines change avoidance with change tolerance.

It assumes changes are as a result of project risks and includes explicit risk management activities to reduce these risks.

Each loop in the spiral is split into four sectors that include:

1. *Objective setting:* Specific objectives for that phase of the project are defined. Constraints on the process and the product are identified and a detailed management plan is drawn up. Project risks are identified. Alternative strategies, depending on these risks, might be planned also.
2. *Risk assessment and reduction:* For each of the identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk.
3. *Development and validation: A* development model for the system is chosen. For example, throwaway prototyping may be the best development approach if user interface risks are dominant. If safety risks are the main consideration, development based on formal transformations may be the most appropriate process, and so on. If the main identified risk is sub-system integration, the waterfall model may be the best development model to use.
4. *Planning:* The project is reviewed and a decision made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

*Advantages.*

1. High amount of risk analysis.
2. Good for large and mission-critical projects.
3. Software is produced early in the software life cycle.

*Disadvantages.*

1. Can be a costly model to use.
2. Risk analysis requires highly specific expertise.
3. Project’s success is highly dependent on the risk analysis phase.
4. Doesn’t work well for smaller projects

Below is the diagrammatic representation of the spiral model.



Figure 4: Spiral model diagram

### 3.0.5 Agile Process Model.

Agile methods are incremental development methods in which the increments are small and, typically, new releases of the system are created and made available to customers every two or three weeks. They involve customers or users in the development process to get rapid feedback on changing requirements. They minimize documentation by using informal communications rather than formal meetings with written documents.

Agile methods universally rely on an incremental approach to software specification, development, and delivery. They are best suited to application development where the system requirements usually change rapidly during the development process. They are intended to deliver working software quickly to customers, who can then propose new and changed requirements to be included in later iterations of the system. They aim to cut down on process bureaucracy by avoiding work that has dubious long-term value and eliminating documentation that will probably never be used.

*Principles of agile development models.*

1. Incremental delivery of software
2. Continuous collaboration with customer
3. Embrace change
4. Value participants and their interaction
5. Simplicity in code.

*Advantages.*

1. Lightweight methods suit small-medium size projects

2. Produces good team cohesion

3. Emphasizes final product

4. Iterative

5. Test-based approach to requirements and quality assurance

*Disadvantages.*

1. Difficult to scale up to large projects where documentation is essential

2. Needs experience and skill if not to degenerate into code-and-fix

3. Programming pairs is costly.

4. Test case construction is a difficult and specialized skill.

### 3.0.6 Waterfall Model.

The first published model of the process development process and was derived from a more general system engineering processes (Royce, 1970). It is a plan driven process in principle, where all process activities are planned before starting working on them.

The waterfall model takes the fundamental process activities of specification, development validation, and evolution and represents them as a separate process phases such as requirement specification, software design, implementation, testing and many more.

The main stages in the waterfall model are:

*Requirement Analysis and definition:* the system’s services constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

*System and Software Design:* it allocates requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

*Implementation and Unit Testing:* software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets the specifications.

*Integration and System Testing:* individual components of the system are integrated and tested as a complete system to ensure that software requirements have been met.

*Operation and Maintenance:* the system is installed and put into practical use. Maintenance involves correcting errors that were not discovered in early stages of development, improving the implementation of the system units and enhancing the system’s services as new requirements are discovered.

*Advantages of the waterfall model.*

1. Each stage has well defined deliverable or milestone.
2. It is simple to use and understand.
3. In this model phases are processed and completed one at a time. Phases do not overlap.
4. Waterfall model works well for smaller projects where requirements are very well understood

*Disadvantages.*

1. You cannot go back a step; if the design phase has gone wrong, things can get very complicated in the implementation phase.
2. High amounts of risk and uncertainty.
3. Not a good model for complex and object-oriented projects.

Below is the diagrammatic representation of the waterfall model stages in their sequence.



Figure 5: Waterfall model diagram

Due to the above advantages I decided to use the waterfall model

*Requirement Analysis and definition:* I will collect information required by the system from the target users.

*System and Software Design:* We will use the android studio IDE, android SDK and the Java JDK for the software development and design process

*Implementation and Unit Testing:* after user acceptance of the system, I will deploy the system from theory to practical to the users. I will correct the errors that may arise during implementation phase.

*Integration and System Testing:* All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

*Operation and Maintenance:* after deployment the system is monitored on how it works and some maintenance is performed in order to realize effectiveness and reliability. Example of maintenance to perform is perfective and corrective maintenance. Also, I intend to train a few users on how to use the application on how to create, encrypt, send decrypt and receive text messages.

## 3.1 Evaluation methodology

A method describes the means of accomplishing a task. For instance, developing a software. It specifies what decision are to be made how to make them and in what order they should be made. Example top down design is a method that describes one approach for designing a software, in contrast a methodology is a collection of complementary methods and a set of rules for applying them. To be more specific it organizes and structure the task comprising the effort to achieving a global objective, establishing the relationships among the tasks within a framework of the global objective and prescribes an order in which certain classes of decisions are made, and way of making those decisions leading to the desired objectives.

In the project I will use usability walkthroughs whereby its objective is either the system design, a paper prototype, a built-in prototype or a completed system. Also, it is a predictive evaluation technique. Here the main feature is a set of instructions to check step by step the artifact under development. For each step the participants note the problems with the task or screen design. In case of a paper prototype or with the interface of a built prototype. After the data collection the problems are discussed and improvements of the system are proposed. The task sequence of a walkthrough must be fixed, since otherwise the simulation is not meaningful, for evaluation purposes.

Walkthroughs are well suited to detect those usability problems caused by discrepancy of system behavior and users habits or expectations. Also, they aid in ascertaining whether the user interface is perceived to be adequate.

However, they are limited because they are expensive since a whole user interface cannot be taken for evaluation to realize a reasonable cost benefit relation and therefore only selected features are considered. Also, the explorative behavior can only be simulated in a limited way; therefore, unexpected errors or misfits cannot be detected.

Cognitive walkthrough is a method for evaluating user interfaces by analyzing the mental processes required of a user. Its scope is the ease of learning in particular of learning by exploration and can be carried out by an individual expert or by a group of peers.

The cognitive walkthrough is guided by the following questions:

* Will the user try to achieve the right effect?
* Will the user notice whether the correct action is available?
* Will the user associate the correct action with the effect to be achieved?
* If the correct action is performed, will the user see that progress being made toward a solution of the task?

Input session consists of a detailed design description of the interface such as a working prototype. Additional input maybe a task scenario, precise description of prospective users, the context of the system and correct sequence of actions that a user should successfully perform to complete designed task.

The advantage of cognitive walkthrough is that it can be applied in early stages of development and that not only usability weaknesses of the system are reported but other problems as well. However, its restriction is that the simulated sequence of actions has to be correct which demands high qualification of analyst.

It has also been criticized that the focus of analysis of the cognitive walkthrough sacrifices other important usability information such as overall consistency.

# **CHAPTER FOUR**

# **SYSTEM ANALYSIS AND REQUIREMENT**.

## 4.0 Current systems.

### 4.0.1 Message in a Bottle.

Message in a Bottle is a commercial application, that allows to protect your SMS communications from intrusions and unauthorized access. The application is implemented in Java, runs on a large variety of mobile phones and can be downloaded free of charge by a user. After downloading the user can purchase the credits that are used to send the message securely. Its implementation is closed source, but from its manual the following observations were made.

Message in a Bottle allows users to encrypt messages that are sent through SMS, using asymmetric elliptic curve cryptography. The public/private key pair of a user is established during installation of the software. Furthermore, messages can be signed in order to achieve integrity and accountability. Both the sending and receiving parties must install the software to enable secure messaging. Therefore, it addresses compromised security due to lost or stolen mobile station and interceptions and modification of messages on transit.

### 4.0.1 CryptoSMS

Is an open source application that runs on a variety of devices and developed using the Java language. Can be downloaded and be used free of charge by any user whose phone supports Java Application Programming Interfaces.

It allows users to encrypt their messages using asymmetric elliptic curve cryptography. The private and public keys are established during installation of the software. Also, both the receiver and the sender must install the software for them to experience secure messaging. It addresses the following problems that is the compromised security due to lost or stolen mobile station and interceptions of messages on transit. (Jovana, Boban, Ivan, & Ivana, 2016)

However, it does not address the modification of messages that are on transmission, thus the integrity of the message is not guaranteed. Another weak point is related to the first threat addressed that is stolen or lost mobile device. To enhance the user experience, the passphrase of a user is stored within the phone for a certain time period after the user has entered it. However, the private key of the user, as well as all messages the user has sent or received are stored in the memory of the phone, encrypted using a symmetric algorithm with the passphrase as a key. This expose both the private key as well as all messages during the time period in which the passphrase is stored within the phone.

## 4.1 The proposed system.

The proposed system that is the E\_Sms mobile application will be used to send the messages securely through the GSM network. The message will be encrypted using the Advanced Encryption Standard block algorithm, the user will have to key in the phone number of the receiver, his or her key and the message before sending the message. It uses the symmetric key encryption, where by the same key that is used in encryption is the same key for decryption. The keys are neither stored anywhere, thus they are only with the users. Also, the message cannot be modified while on transmission since upon reception it won’t be decrypted it will be corrupted. This ensures integrity of the message and security incase of the loss of the mobile device. The message is stored in the form of ciphertext locally and temporarily upon reception in the shared preferences and to read it you have to decrypt it first therefore, ensuring security of the message is confidential and private.

### 4.1.1 Conceptual Framework of the proposed system.

SENDER RECEIVER

Message plaintext

Received Ciphertext

AES block cipher encryption algorithm

Ciphertext

AES block cipher decryption algorithm

Message plaintext

GSM

Figure : Conceptual framework of the proposed system

### 4.1.2 System flowchart

INPUT: Key

INPUT:

Phone number, Key, Message

OUTPUT:

Phone number, Message

Key length must be 16 characters long. Input and try again!

Input the phone number for the receiver.

Encrypt

Send

Decrypt

Receive

Is Key length ==16?

Is Phone No. !=null?

Is Key length ==16?

Input the phone number for the receiver.

No

Yes

No

No

No

Yes

Figure : System flowchart of the proposed system

## 4.2 Requirement specification.

### 4.2.1 Hardware Requirements

Core i3 processor 1GHz

Hard disk 20Gb free

Memory (RAM) 4gb minimum.

Android phone with Android 4.4 or higher operating system.

### 4.2.2 Software Requirements

1. Windows 7 operating system (ultimate/enterprise)

It the operating system for the personal computer.

1. Android studio IDE version 2.2.3 and above.

An easy to use Integrated Development Environment (IDE) that brings together Java and the Android SDK to make it simple to write Android apps

1. Android SDK

Incorporated in android studio and provides access to android libraries that allows for the development of android applications. It includes sample projects with source code, development tools, an emulator that are necessary in developing android applications. Applications are written using Java programming language. Applications run on the Dalvik a custom virtual machine designed for embedded use running on top of a Linux kernel. Also, it enables applications compatibility through an application programming interface with android platforms.

1. Java JDK

Lays the foundation for the android SDK. That is for the easy installation of the Android SDK and proper running of the tools of the SDK.

### 4.2.3 Functional Requirements

*Plaintext* – it is the data that is to be protected by the system through encryption. It includes alphabetic letters, numeric characters and the bytes. It is in the form that no any effort made to make the data protected that is they are plain characters.

*Ciphertext-* Is the scrambled version of the plaintext produced by the encryption

algorithm using a specific encryption key. The ciphertext is not guarded. It flows on public channel. It can be intercepted or compromised by anyone who has access to the communication channel. It may be characters from the alphabet, numbers or bytes.

*Encryption/Decryption key-* A parameter that will control the execution of the encryption or decryption algorithm. The encryption key is the value known to the sender that he or she inputs it along with the plaintext in order to computer the ciphertext. The decryption key is related with the ciphertext that is known to the receiver and he or she inputs it along with the ciphertext to compute the plaintext.

### 4.2.4 Non-Functional Requirement

*Confidentiality requirement-* it means that the message sent over a channel cannot be read by anybody. This includes that nobody can know the content of the message unless he or she has read it and he should not be able to read the contents of the message.

*Integrity requirement*- ensures that the message as received by the receiver has not changed during the transmission process. If it has been modified then the message won’t be decrypted it displays an error message that it is corrupt thus it won’t be trusted since its’ integrity has been infringed. Also, it may be obtained through using a hash function or adding a signature to the encrypted message before the transmission begin.

*Authentication requirement*- it involves identifying the sender of the message, it is important since it is achieved through encrypting the messages before transmission and you receive the phone number of the sender already in your contact list where you can verify from. Also, since we use symmetric encryption it is easy to achieve authentication as long as the key remains secret

# **CHAPTER FIVE**

# **SYSTEM DESIGN AND IMPLEMENTATION**

## 5.0 Technical Design.

The application’s user interface has the sender part and the receiver part. On the sender part we have the phone number field where we type in the phone number of the receiver, the key field where we type in the encryption key that is 16 characters long and the message field where we type in our message to be encrypted and be sent. We also, have the send button when clicked it calls the sending message method which gets the text from the message field then the message is encrypted and the ciphertext and receiver’s number are sent through the GSM network to the receiving end and the cancel button that exits the application. On the receiver part we have two text views one where the sender’s number is displayed and the other displays the ciphertext received, also we have the key field here the receiver types in the decryption key which must be sixteen characters long and when done he or she clicks the submit button which decrypts the message and displays the message with replacing the ciphertext that is received. The key must be the same as the one used during encryption.

Message

Encrypt

Secret Key

Ciphertext

Secret Key

Decrypt

Encrypted Message

Decrypted Message

Figure : Technical design

## 5.1 Use case diagram

Once, the message is received the Android applications shows a notification and the user loads the application that displays the receiver activity showing the received message and he or she enters the decryption key and decryption begins once it is submitted.

Mobile operator server

Application module

Application user

Figure : User case diagram

## 5.2 Interaction diagram

The user when interacting with the system as the first level, the user composes the message, he or she then adds the encryption key to encrypt the message, he then send the encrypted message. On reception the receiver inputs the decryption key which is the same as that used in the encryption and he decrypts the message to read and reply to it.

Users Compose Message Receive

compose message

Add encryption key

Encrypt message

Add decryption key

Figure : Interaction diagram

Decrypt message

Send message

## 5.3 System implementation

### 5.3.1 Development environment implementation

The application was developed using android studio integrated development environment version 2.2.3. The android SDK with the libraries required to build applications for android, java JDK that provides the foundation for the Android SDK. Java language was used with the XML. This version 1.0 of the application was installed, debugged and tested in different mobile devices that were supporting android operating system. The application was installed in the Lenovo a2020, and Itel it1508 mobile devices and results were collected and analyzed. The application was transferred to the devices in debug mode via the universal serial bus cable. Also, the emulators that is the virtual devices were also used to test the application.

### 5.3.2 Graphical User Interface and coding Implementation

In the E\_Sms application, we have developed the interface for the sender and the interface for the receiver. The interface for the sender is the default activity that is launched once the application is started and no message has been received yet, otherwise the receiver interface is launched showing the message received last on your mobile device.

In the android studio we started a new project, gave the project name then chose an empty activity and made it the main activity. This activity was created in the res/layout it created as activity\_main.xml file. This was used to implement the sender interface.

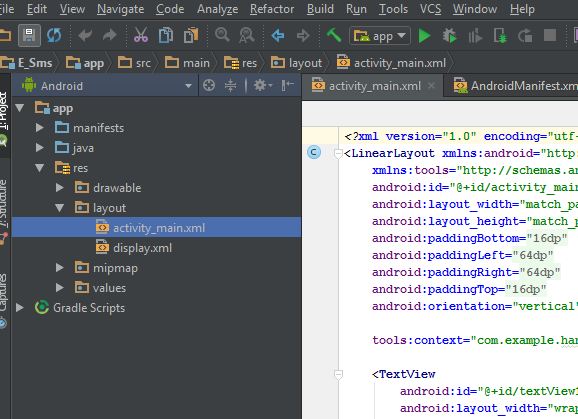


Figure 11: Coding and interface coding menu in Android Studio IDE

# Also, we create the display activity on the same folder as the main activity, that will be used on the receivers’ end for receiving the message. We get the decrypted message by entering the decryption key and decrypt the ciphertext to read the message.

To get data from the main activity we created the MainActivity.java class, where we are accessing the receiver’s phone number, value of the encryption key, the message to be sent, the send button and the cancel button.



Figure 12: MainActivity.java code implementation

# The other two classes that is the Display.java class is used to get data received from the broadcast and the Receiver.java for broadcasting the message over the network class has also been implemented. To send an SMS message for android application special permissions are required to send a message, receive and read the message. The permissions have also been declared in the AndroidManifest.xml file.



Figure 13: Manifest permissions

# The function required to grant the special permissions is:

Bundle bundle = intent.getExtras();

The other functions that will be needed to broadcast and receive the message are as shown below.

# 

Figure 14: Broadcast receiver functions

# Also, to display the message from the broadcast using the Display.java class some functions are also implemented as show in the code snapshot below.



Figure 15: Display message code implementation

The received cipher is temporarily held in the SharedPreferences where it is retrieved from for decryption, also the sender’s number is held there before being set for the display.

## 5.4 Testing and validation

Testing is the process through which and an application software developed is tested it functionality, consistency and usability. This can be automated or manual. The application can be preinstalled or it be installed from the play store for android operating system.

*Usability testing*- was carried out to determine whether the system does what it was expected to do. It was found from the users who were my fellow classmates to have achieved the main objective of having a secure messaging application. So, according to them it was reliable easy, and simple to use. Thus, the usability testing was highly achieved.

*Compatibility testing*- it was also carried out on different android platforms that had a minimum API 21 and above. It was found to be compatible with most of the devices, browsers, screens and most of the android operating system versions. Only, for those few devices where the debugging mode did not work posed the challenge.

*Interface testing*- the buttons were tested whether they performed the intended function and were found to do so. The edit text field also did the appropriate task that they were intended for thus the interface was okay.

*Memory testing-* the application after installation on most of the mobile devices used around 5.0 MB of memory space that is relatively low thus saving on the memory space. Also, it did not require a database to store the ciphertexts or messages thus required a little amount of memory space. It only holds the messages temporarily for sending and receiving for encryption and decryption purposes.

*Service testing*- was offline that it sent the text message without requiring an internet connection. It only took the SMS bundles to send a text message.

*Validation testing*-the application was checked on its validations, that it was found it would not send a message when the key length was not 16 characters long, no message typed in and when there was no number the message would be sent to.

*Security testing*- the application was found to encrypt the message and only the ciphertext would be sent through the GSM network. Also, it would not be able to decrypt the ciphertext if it was intercepted and modified. These, ensured integrity and privacy of the message received.

## 5.5 System Changeover

Changeover is the process through which one shifts from using the current system to using the proposed system. It involves

* 1. Direct changeover- the user stops using the current system all at once and from that point he or she completely begins to use the proposed system. However, this may pose a challenge if the proposed system does not meet the desired goals and objectives and it is not reliable and efficient which will bring integrity and functionality challenges.
  2. Phased changeover- here the user shifts from the current system phase by phase that is he or she uses the system as per the module. This helps the user to determine the flaws of the system at an earlier stage that are in the system, determine whether the system has met its objectives and goals, and whether the modules implemented have achieved their functionality and they produce the required output. However, this mode is expensive to use and implement in complex systems, though reliable.
  3. Parallel changeover- here the systems are run concurrently. The current system and the proposed systems are run side by side, this helps to determine the level of functionality of the proposed system compared to the current system, whether it has achieved its objectives and goals, compatibility of the proposed system, effectiveness can be easily compared and efficiency of the proposed system also can be compared to the current system. This gave us, an incentive to implement this mode in changing to the proposed system before we fully implemented the proposed system. This was because comparisons would be easily drawn and the level of functionality would be easily determined. However, it was very expensive in running both systems, since they both took memory space of the device which would be doubled, and when sending double SMS bundle was to be used.

Advantages

* Its main advantage is that comparisons can be clearly depicted out when running the two systems concurrently.
* It helps determine the level of functionality of the proposed system comparing with the current system hence, easy to see the flaws and limitations of the proposed system.

Disadvantages

* It is an expensive mode of changeover, since it runs both systems they more of resources like the devices’ memory space and users’ SMS bundles.

# **CHAPTER SIX**

# **LIMITATION, CONCLUSION AND RECOMMENDATION**

## 6.0 Limitation

* Identifying the right design patterns for my system software and establishing an actual design review, quality evaluation criteria and design management has been a bit challenge to me in trying to come up with an impressive interface design.
* Time factor has also been a challenge since my project demanded a period not less than six months and only we are required to do it in three months which was a limited time.
* Internet hiccups at times during power failure within the university becomes a problem since most of the time I relied on getting some concepts from internet.

## 6.1 Recommendation

After, the system has been developed we recommend that it should be implemented and all the security tests carried upon it. This is to identify the loopholes that might have not been identified at any phase before becoming fully functional. Also, it can a further study can be made the system to improve the system since this is a limited study like the system there is no way we can share the key unless we meet physically, therefore it can be improved to share and manage the keys online. This may also include public key cryptography. The system can also be improved to securely sharing the multimedia messages and other graphics through encrypting them and sending them as a file securely through the network and finally a database for the system that is a server-based database where the ciphertext can be stored and be retrieved from for decryption once we have managed the keys securely.

## 6.3 Conclusion

The proposed E\_Sms android application will solve the problem of security in sending text messages securely and privately over the network. It is easy and simple to use, cheap to maintain and it requires little resources like the memory resource. It is fast and efficient to the users. Also, the application is intended for open source thus readily available to the users and can be used freely.

# **BIBLIOGRAPHY**

Albuja, J. P., & Carrera, E. V. (n.d.). Trusted SMS communication on mobile devices.

Baron, S. P., & Harris, K. (2006). Beyond technology acceptance: understanding consumer practice. International Journal of Service Industry Management. *Vol. 17 No. 2, 2006*, 111-135.

Bell, D. (2000). *Software Engineering* (3rd ed.). London: Addisson Wesley.

Castillo, R. E., Cayabyab, G. T., & III, B. T. (2015, January-February). SECURITY ASSURANCE FRAMEWORK FOR SMS USING CASCADED ENCRYPTION ALGORITHM. *International Journal of Latest Research in Science and Technology, 4*( 1), 47-55. Retrieved 2 2018, from http://www.mnkjournals.com/ijlrst.htm

El-Khalil, & Keromytis, A. D. (2007). , Hydan: Hiding Information in Program Binaries, the 9th International Conference on Information and Communications Security (ICICS 2007).

G. Racherla, & Saha, D. (2000). , “Security and Privacy Issues in Wireless and Mobile Computing”, Proceedings of 2000 IEEE International Conference on Personal Wireless Communications., (pp. 509-513).

Goslin, J., B. J., Steele, G., & Bracha, G. (2005). *The Java™ Language Specification* (3rd ed.). California, USA: Sun Microsystems, Inc.

Helander, M., Landauer, T., & (Eds)., P. P. (n.d.). *Handbook of Human Computer Interaction. (* (2nd ed.). Amsterdam: Elsevier.

International Journal of Innovative Research in Computer and Communication Engineering. (2015). *Vol 3*.

Jovana, Đ., Boban, M., Ivan, T., & Ivana, O. (2016, October 15). CryptoSMS ANDROID APPLICATION. *The Eighth International Conference on Business Information Security (BISEC-2016)*, 1-5.

Knudsen, J. B. (1998). *Java Cryptography* (1st ed.). 101 Morris Street, USA: O'Reilly & Associates, Inc.

Lee, W. M. (2012). *Beginning Android™ 4 Application Development.* Indianapolis, Indiana, USA: John Wiley & Sons, Inc.

M, A., & Devrim, S. (2010). SMS Security: An Asymmetric Encryption Approach, IEEE International Conference on Wireless Mode. 448-452.

Madhwani, M. (2012). 13. Cryptography On Android Message Application using Look Up Table and Dynamic Key (Cama)”. *IOSR Journal of Computer Engineering (IOSRJCE), Volume 6,*(Issue 2 (Sep-Oct.2012)), 54-59.

Pimpale, P., Rayarikar, R., & Upadhyay, S. (October 2011). “Modifications to AES Algorithm for Complex Encryption”. *IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.10*.

Rasmita, S., & Shajid, A. (2017, September ). A Secure Framework for Messaging on Android Devices with Honey Encryption. *International Journal Of Engineering And Computer Science, Volume 6* (Issue 9 ), 22480-22485. Retrieved March 2018, from www.ijecs.in

Rupa, C., & Avadhani, P. (2009). Message Encryption Scheme Using Cheating Text. *IEEE International Conference on Information Technology*, 470-474.

Scacchi, W. (2001). Process Models in Software Engineering. *Encyclopaedia of Software Engineering*. (J. Marciniak, Ed.) John Wiley and Sons.

Scacchi, W. (2001). Process Models in Software Engineering. This is an excellent overview of a wide range of software engineering process models that have been proposed. . *Encyclopaedia of Software Engineering*. (J. J. Marciniak, Ed.) John Wiley and Sons. Retrieved 2018, from http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf.

Stallings, W. (2005). *Cryptography and Network Security Principles and Practices.* USA: Prentice Hall.

Townsendsecurity. (2015). *The Encryption Guide. Townsend security, the security company*.

Townsendsecurity. (2017, 10 14). Retrieved from Townsendsecurity: https://info.townsendsecurity.com/definitive-guide-to-encryption-key-management-fundamentals#Types-of-Encryption-Keys-and-How-They-Work

Townsendsecurity. (2017, 10 14). Retrieved from Townsensecurity: https://info.townsendsecurity.com/sql-server-encryption-key-management-definitive-guide

University, T. O. (2015). Retrieved from open.edu: http://www.open.edu/openlearn/ocw/mod/oucontent/view.php?id=48322&section=1.3

Walt, S., Software, R. I., California, U. o., & Irvine. (2001, 5). Retrieved from UCI.EDU: http://www.ics.uci.edu/~wscacchi/Papers/SE-Encyc/Process-Models-SE-Encyc.pdf.

Whatson, C., J., R., Lewis, C., & Polson, P. (1994). The Cognitive Walkthrough Method: A Practitioner’s Guide. In J.Nielson, & R. M. (Eds), *Usability Inspection Methods.* New York.: Wiley.

Yashaswini, J. (2015). Key Distribution for Symmetric Key Cryptography: A Review. *International Journal of Innovative Research in Computer, 3*(5, May 2015), 1-5.

# **APPENDIX**

## Appendix I: User manual

1. Sending message
2. Go to the menu, and scroll to the installed application E\_Sms
3. Tap to open.
4. The sender activity is launched, if there was a message received it launches the receiver activity displaying the message that was receive. Uses your devices back button and go back to the senders’ activity.
5. Compose you message in the senders’ activity and input the phone number of the receiver and a sixteen-character long encryption key.
6. After you are done click send to send the message. It will display the message sent and delivered successfully reports respectively if the details entered are correct. Otherwise either the fields are blank report or your key length is less than or more than sixteen characters long. You are asked to reenter them and try again.

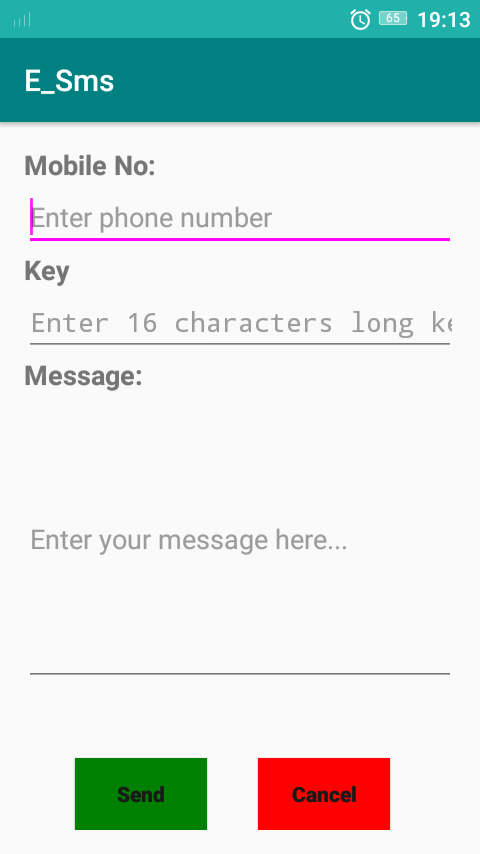


Figure 16: User interface for sending sms

1. Receiving message
2. A notification is displayed on the notification bar when the message is received if the E\_Sms application is installed in your device.
3. Go to your application and start it, it launches the receiver activity displaying the senders’ number and the ciphertext received.
4. Enter the decryption key in the key field, the key must be the same as the one used for encryption. Click submit to decrypt and read the message. The decrypted message will replace the ciphertext.
5. After reading go back to the sender activity and reply your message from there.

*NOTE: The message will only be decrypted if the ciphertext was not intercepted and modified and the key is correct. Otherwise, message is corrupted toast report is displayed.*

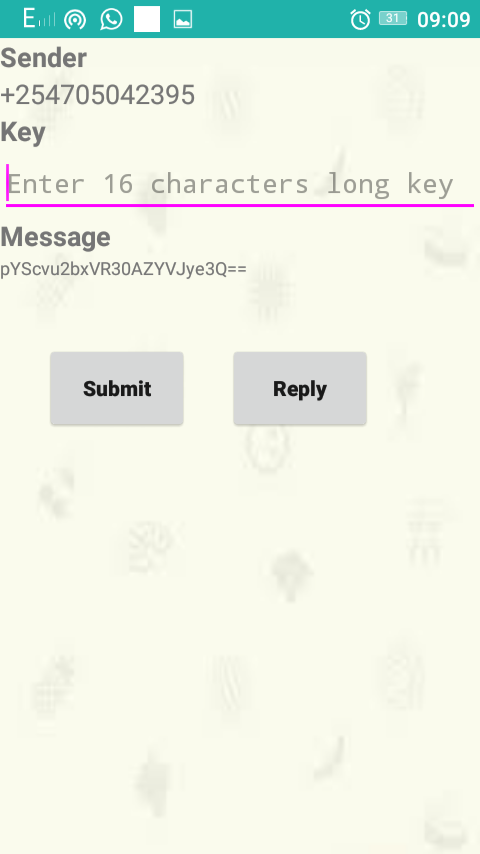


Figure : Receiver interface

## 

## Appendix II: Sample Codes

**Decryption code**

private String decrypt(String outputString, String password) throws Exception {

SecretKeySpec key= MainActivity.*generateKey*(password);  
 Cipher c= Cipher.*getInstance*(MainActivity.*AES*);

c.init(Cipher.*DECRYPT\_MODE*, key);  
 byte[] decodedVal= Base64.*decode*(outputString, Base64.*DEFAULT*);  
 byte[] decVal= c.doFinal(decodedVal);  
 String decryptedVal = new String(decVal);  
 return decryptedVal;  
}

**Code for Encryption**

private String encrypt(String Text, String password) throws Exception {  
 SecretKeySpec key= *generateKey*(password);  
 Cipher c= Cipher.*getInstance*(*AES*);  
 c.init(Cipher.*ENCRYPT\_MODE*, key);  
 byte[] encryptText = c.doFinal(Text.getBytes());  
 String encryptedText = Base64.*encodeToString*(encryptText, Base64.*DEFAULT*);  
 return encryptedText;  
}

**Code for generating key**

public static SecretKeySpec generateKey(String password) throws Exception {  
 final MessageDigest digest = MessageDigest.*getInstance*("SHA-256");  
 byte[] bytes = password.getBytes("UTF-8");  
 digest.update(bytes, 0, bytes.length);  
 byte[] key=digest.digest();  
 SecretKeySpec secretKeySpec= new SecretKeySpec(key, "AES");  
 return secretKeySpec;  
}

**Code for sending the message**

private void sendingMessage(String phoneNo, String sms){  
  
 if(phoneNo.length()>0 && sms.length() !=0){  
  
 try {  
 SmsManager smsManager = SmsManager.*getDefault*();  
 smsManager.sendTextMessage(phoneNo, null, sms, null, null);  
 Toast.*makeText*(getApplicationContext(), "Message sent successfully",  
 Toast.*LENGTH\_LONG*).show();  
 } catch (Exception e) {  
 Toast.*makeText*(getApplicationContext(),  
 "Message did not send. Try again!",  
 Toast.*LENGTH\_LONG*).show();  
 e.printStackTrace();  
 }  
  
 }else {  
 Toast.*makeText*(getApplication(), "Enter the phone number or key, or message",  
 Toast.*LENGTH\_LONG*).show();  
 }  
}

**Send button**

send.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
  
 *//get data from the fields* String phoneNo = no.getText().toString();  
 String sms = msg.getText().toString();  
 String sKey= key.getText().toString();  
 String outputString = "";  
  
 *//test the fields are null* if (sKey.length()>0 && sKey.length()==16 && msg.length() != 0){  
 try {  
 *//encrypt* outputString = encrypt(sms, sKey);  
 *//send* sendingMessage(phoneNo, outputString);  
  
 } catch (Exception e) {  
  
 Toast.*makeText*(getApplicationContext(), "Enter all the required fields, they are required.",  
 Toast.*LENGTH\_LONG*).show();  
 e.printStackTrace();  
 }  
 }  
 else Toast.*makeText*(getApplicationContext(), "Enter the text or a 16 length key and try again!",  
 Toast.*LENGTH\_LONG*).show();  
  
  
 }});

**Receiver activity**

public void onReceive(Context context, Intent intent) {  
 *//---get the SMS message passed in---* Bundle bundle = intent.getExtras();  
 SmsMessage[] msgs = null;  
 String str = "", code = null, from = null;  
  
 if (bundle != null) {  
 *//---retrieve the SMS message received---* Object[] pdus = (Object[]) bundle.get("pdus");  
 msgs = new SmsMessage[pdus.length];  
 for (int i = 0; i < msgs.length; i++) {  
 msgs[i] = SmsMessage.*createFromPdu*((byte[]) pdus[i]);  
 str += "SMS from " + msgs[i].getOriginatingAddress();  
 from = msgs[i].getOriginatingAddress();  
 str += " :";  
 str += msgs[i].getMessageBody().toString();  
 str += "n";  
 code = msgs[i].getMessageBody().toString();  
 }

**Submit button**

submit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
  
 String secretKeyString = key.getText().toString();  
 cipher.setText(cipher.getText().toString() + "\n");  
  
 if (secretKeyString.length() == 16) {  
 try {  
 cipher.setText(decrypt(code, secretKeyString));  
 } catch (Exception e) {  
 Toast.*makeText*(getApplicationContext(), "Wrong key entered, Enter the correct key and try agein",  
 Toast.*LENGTH\_LONG*).show();  
 e.printStackTrace();  
 }  
 }  
 else Toast.*makeText*(getApplicationContext(), "Key length must be 16 characters long",  
 Toast.*LENGTH\_LONG*).show();  
  
  
 }  
 });  
  
}

**Cancel button**

cancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 finish();  
 System.*exit*(0);  
 }  
});

## Appendix III: Time and resource scheduling.

### Budget

|  |  |
| --- | --- |
| Item | prize |
| Toshiba laptop core-i3 500gb HDD, 4gb RAM | KShs. 54,000 |
| 2 Android Smartphones | KShs. 18,000 |
| Total | KShs. 72,000 |

Table 1: Budget

### Time Scheduling

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | FROM | TO | DEC | | JAN | | | | FEB | | | MARCH | | | APRIL | | MAY | |
| TASK |  |  |  | |  | | | |  | | |  | | |  | |  | |
| SYSTEM PLANNING AND SELECTION | 15/12/17 | 02/01/18 |  |  |  |  | | |  | | |  | | |  | |  | |
| PROPOSAL WRITTING | 03/01/18 | 08/01/18 |  | |  |  | |  | |  | | |  | |  | | |  | |
| SYSTEMS ANALYSIS AND DESIGN | 09/01/18 | 25/02/18 |  | |  | |  | |  | |  |  | | |  | |  | |
| SYSTEMS IMPLEMENTATION | 26/02/18 | 11/03/18 |  | |  | | | |  | |  |  | |  |  | |  | |
| DOCUMENTATION | 12/03/18 | 20/04/18 |  | |  | | | |  | | |  | |  |  |  |  | |
| DELIVERY | 22/04/18 | 10/05/18 |  | |  | | | |  | | |  | | |  |  |  | |

Table 2: Time schedule