New Technique for Cross-Platform Availability Control over Electronic Messages

استخدام طريقة جديدة في عملية التحكم في البريد الالكتروني

By

Ward Ahmed Al-Hussein

Supervisor
Dr. Hebah H. O. Nasereddin

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Master Degree in Computer Information Systems

Faculty of Information Technology

Middle East University

Amman, Jordan

January, 2015

بسم الله الرحمن الرحيم

وَيَسْأَلُونَكَ عَنِ الرُّوحِ قُلِ الرُّوحُ مِنْ أَمْرِ رَبِّي وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

الإسراء, 85

صدق الله العظيم

Authorization Statement

I, Ward Ahmed Al-Hussein Authorize the Middle East University to Supply a Copy of My Thesis to Libraries, Establishments or Individuals.

Signature:

Date: January 25th, 2014

أنا ورد أحمد ال حسين افوض جامعة الشرق الأوسط بتزويد نسخه من رسالتي للمكتبات او المؤسسات او الهيئات او الافراد عند طلبها.

التوقيع:

Thesis Committee Decision

This Thesis "New Technique for Cross-Platform Availability Control over Electronic Messages" was discussed and certified on 25/01/2015

Thesis committee   **signature**

Dr. Hebah Nasereddin   Chairperson & Supervisor

Dr. Maamoun Ahmed   Member

Prof. Ra'ed Abu Za'iter   Member

AKNOWLEDGEMENT

I am forever indebted to my family who supported me during this academic journey and my whole life.

I would like to offer my sincerest gratitude to my supervisor Dr. Hebah H. O. Nasereddin for her useful comments, illuminating views and guidance throughout my master thesis.

I am very grateful to the Information Technology Faculty members at the Middle East University for their insightful conversations and hard questions, thank you for teaching me how to be a dedicated researcher.

Finally, million thanks go to my fellow colleagues for their support and encouragement.

DEDICATION

This thesis is lovingly dedicated to Mema, my grandmother, like a star she has been the guiding light of my life, thank you for all the love you've given me, thank you for making me the man I am today.

To my father, a man of principles and honor, who taught me how to find my way through man's most challenging hardships, thank you for being my role model.

To my mother, for her support, love and encouragement.

To my beloved late grandfather, an exceptionally wise man whom I miss dearly.

To my late friend, Mohammed Al-Zeena, may your soul rest in peace brother.

To FC Bayern Munchen, the best team in the whole world today, and all of days.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AES: Advance Encryption Standard.

API: Application Program Interface.

ARPAN: Advance Research Projects Agency Networks.

ASCII: American Standard Code for Information Interexchange.

COM: Component Object Model.

DNS: Domain Name Server.

Email: Electronic Mail.

ESP: Email Service Provider.

HTTP: Hypertext Transfer Protocol.

IBE: Identity-Based Encryption.

IMAP: Internet Message Access Protocol.

MDA: Mail Delivery Agent.

MHS: Message Handling System.

MIME: Multi-Purpose Internet Mail Extensions.

MPK: Master Public Key.

MSA: Mail Submission Agent.

MTA: Mail Transfer Agent.

MUA: Mail User Agent.

MX: Mail eXchange.

NIST: National Institute of Standards and Technology.

OES: Original Email Size.

PES: Original Email Size.

PGP: Pretty Good Privacy.

POP: Post Office Protocol.

RFC: Request For Comments.

S/MIME: Secure/Multipurpose Internet Mail Extensions.

SecMTP: Secure Mail Transfer Protocol.

SMTP: Simple Mail Transfer Protocol.

SOAP: Simple Object Access Protocol.

SPSS: Statistical Package for the Social Sciences.

SSL: Secure Socket Layer.

T/TCP: Transactional Transmission Control Protocol.

TCP/IP: Transmission Control Protocol/Internet Protocol.

TLS: Transport Layer Security.

TSP: Time Stamping Protocol.

TTP: Trust Third Party.

TTR: Time to Read.

TTS: Trusted Time-Stamping Service.

TTS: Time to Send.

USE: Undo Sent Email.

UUID: Universally Unique Identifier.

New Technique for Cross-Platform Availability Control over Electronic Messages

Prepared By: Ward Ahmed Al-Hussein
Supervised By: Dr. Hebah H. O. Nasereddin

Abstract

Email is considered to be one of the most commonly used services on the internet, whether the recipient is three doors down or miles away, the fact that it has great importance in modern electronic communications is undeniable. In current emailing system the user cannot control the availability of his sent emails after reaching their destination.

In this research, a new technique is proposed to grant email senders control over sent messages even after they arrive at the receiver's mailbox. The method suggests that the email gets encrypted by AES using a randomly generated key which is stored at the sender's server. The sender can control the availability of the key and thus the ability of the receiver to read the email.

The experimental results compared the functions of the proposed technique against the normal process of email processing, and concluded that the process of sending an email in the proposed technique required slightly more time, less than 10% additional time; however, the process of requesting the key and decrypting the email required above 80% additional time, than the normal process. The study also assesses the standard deviations and the p-value of the experiments and discovers the correlation between email size and duration of email processing for the proposed technique.

استخدام طريقة جديدة في عملية التحكم في البريد الالكتروني

اعداد: ورد احمد ال حسين

اشراف: د. هبة ناصر الدين

الخلاصة

يعتبر البريد الإلكتروني أحد أكثر الخدمات استخدامًا على شبكة الإنترنت. في نظام ارسال البريد الإلكتروني الحالي لا يمكن للمستخدم التحكم في الرسائل المرسلة بعد وصولها إلى مستلميها.

في هذه الدراسة، سيتم اقتراح تقنية جديدة لمنح مستخدمي البريد الالكتروني القدرة على التحكم في الرسائل المرسلة حتى بعد وصولها إلى صندوق بريد المستلم. وذلك من خلال تشفير الرسالة الالكترونية بواسطة خوارزمية " معيار التشفير المطور AES"، وباستخدام مفاتيح تشفير عشوائية تكون مخزنة على الخادم الخاص بمرسل الرسالة الالكترونية. يستطيع المرسل التحكم في فترة توفر مفاتيح التشفير الخاصة بالرسائل الالكترونية التي قام بارسالها؛ بالتالي فهو يستطيع التحكم بقدرة المستلم على قراءة هذه الرسائل.

النتائج التجريبية قارنت بين خصائص التقنية المقترحة وخصائص النظام العادي في ارسال البريد الإلكتروني، وأظهرت أنّ عملية ارسال البريد الالكتروني باستخدام التقنية المقترحة، بحاجة إلى أقل من 10٪ كوقت إضافي من الوقت اللازم لارسال الرسالة باستخدام النظام العادي. أمّا عملية اختيار المفاتيح وفك تشفير الرسالة الالكترونية فإنها تحتاج إلى أكثر من80٪ من الوقت اللازم لفتح الرسالة باستخدام النظام العادي. كما قيّمت الدراسة الانحراف المعياري و p-value للنتائج التجريبية، وبينت العلاقة بين حجم البريد الالكتروني والوقت اللازم لتنفيذ التقنية المقترحة.

Chapter One

Introduction

Chapter 1 Introduction

During the last three decades, the communications of our world increased dramatically due to the evolution of computers. In the past computers were used to do simple mathematical equations, today they are one of the most important parts of modern society as they act as communication devices that connect people to each other and to information. One of the most widespread applications of computers nowadays that are used for person to person communication is Electronic Mail (Email).

While the first known wireless transmission dates back to 1895 when Nobel Prize winner of Physics, Guglielmo Marconi's, did the first wireless radio transmission in history, it was until 1971 (Haigh, T. 2012) when Ray Tomlinson sent the first electronic mail from one machine to another over the Advance Research Projects Agency Networks (ARPAN) after several trials and errors. In 2004 electronic messaging was considered to be the number one activity for most users on the Internet (Culotta, A., etal., 2004) both in terms of popularity and amount of traffic generated (Hafsaoui, A., etal., 2010). A study done by A Gartner and Symantec Incorporation stated that up to 75% of a company's intellectual property is stored in emails and other messaging applications, and according to a survey done by Radicati Group in 2014, there were over 4.1 billion email users worldwide, with a total worldwide average of 196.3 billion emails sent and received per day, over 108 billion of which are for business purposes (Radicati, S. 2014).

Today's email system can be divided into two distinct subsystems. The first subsystem is called the Message Handling System (MHS); it is built on a set of servers called Mail Transfer Agents (MTAs) which are responsible for transferring messages from the sender

side to the receiver side. The second subsystem is called the Mail User Agent (MUA), this subsystem handles the processes of receiving, creating and managing emails. MUA interacts with MTAs systems to ensure that messages are being delivered (Partridge, C. 2008).

1.1 Email Processing Agents

Emails are processed through the use of specific programs called Agents; each agent has specific tasks to do in the email processing paradigm. The MUA is the one that user interacts with, it is used to send and read emails, the email is submitted from MUA to Mail Submission Agent (MSA) which temporarily queues outgoing emails (Dye, M., etal., 2007). The MUA is responsible for sending messages and placing received messages into the client mailbox, both of which are distinct processes as shown in Figure (1.1) below.

Figure 1.1. MUA interaction with SMTP/POP3 server (Dye, M., etal., 2007)

The email messages are then sent to the MTA whose function is to transfer the message to the destination. Figure (1.2) on the next page shows how an email message may travel through several MTAs to reach its final destination, to do so; the MTA locates the target host by requesting the Mail eXchange (MX) record from the Domain Name Server (DNS), then it connects to the exchanger server as Simple Mail Transfer Protocol (SMTP) client,

the exchanger server receives the email message from MTA and delivers it to Mail Delivery Agent (MDA) which has the responsibility of delivering the message to the mailbox of the receiver as shown in figure (1.3).
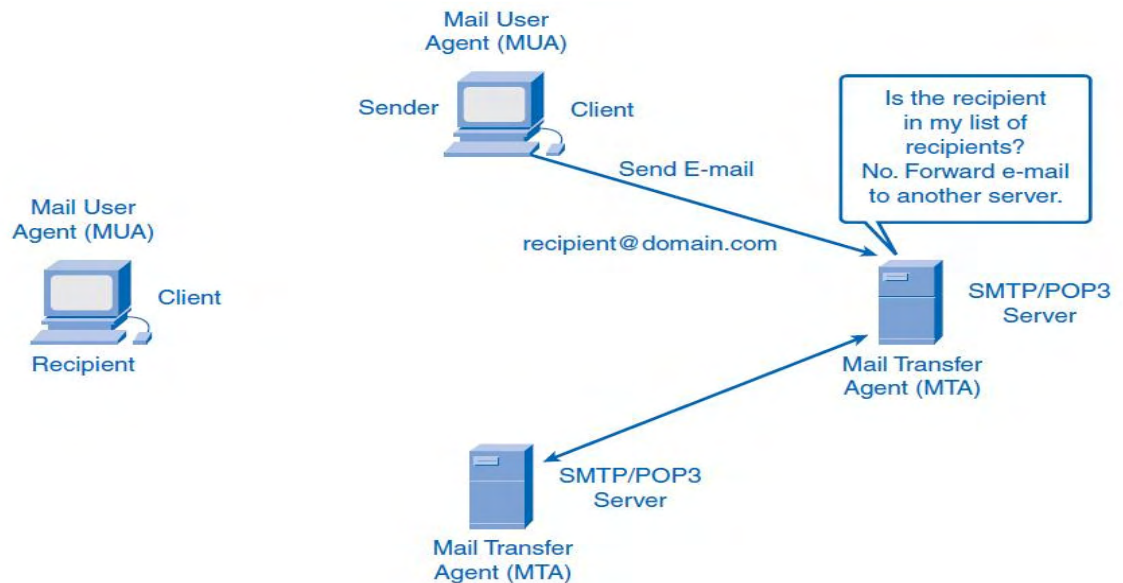
Figure 1.2. MTA interaction with other email agents (Dye, M., etal., 2007)
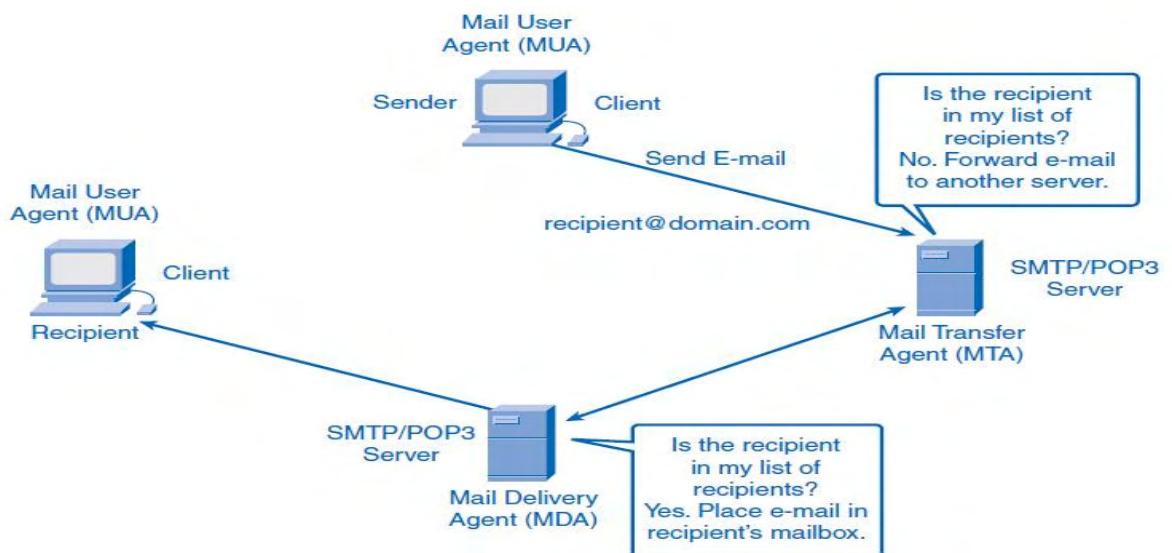
Figure 1.3. MDA interaction with other email agents (Dye, M., etal., 2007)

The two main protocols that are used for fetching emails from MDA are Post Office Protocol (POP3) and Internet Message Access Protocol (IMAP), the receiver uses MUA to fetch the email message to the his mailbox (Crocker, D. 2009).

1.2 Classification of Email Protocols

Most transport email protocols are used to either retrieve emails or to send emails; POP (Myers, J., & Rose, M. 1996) and IMAP (Crispin, M. 2003) are used to retrieve emails. While both of those protocols are used for the same task; i.e. retrieving emails, they differ in the way they access and store emails. POP is the oldest and most recognizable internet protocol; its currently used version is POP3 which uses port 110. In POP3 emails are fetched from the server to get stored locally on the user's computer and the original emails are usually deleted from the server in order to reduce storage load on the server side, this feature allows the users to read downloaded emails without being connected to the server but it also makes the emails remotely inaccessible (Myers, J., & Rose, M. 1996).

IMAP latest version is IMAP4 which uses port 143; IMAP4 keeps the emails stored at the server until the user decides to delete them, this feature makes emails remotely accessible from any machine, the downside to that is an added server's resources consumption, such as storage, and for that particular reason each user gets a limited mailbox size in order to reduce the storage load on the server (Crispin, M. 2003). The MUA/MDA typically retrieves messages from the server by using either IMAP or POP3.

The protocol used to send emails from source to destination in the Transmission Control Protocol/Internet Protocol (TCP/IP) suite, which is a suite of communication protocols that are used to connect hosts to the internet, is referred to as SMTP.

1.3 Simple Mail Transfer Protocol

SMTP is a set of standards and protocols that are used to facilitate the transferring of email messages between servers. SMTP has been around a long time (Request For Comments (RFC) done in 1982) and when SMTP first emerged, security threats and issues such as hacking and spamming were not common so SMTP was considered to be reliable as it had satisfied the users' need of sending and delivering emails and because of that security was a little concern to most people, but with the increased popularity of SMTP, the need for incorporating more security protocols and guidelines into mail servers became a priority, and accordingly, several changes were made to SMTP servers to make those systems more secure while considering the compatibility issues of merging the old systems with the new ones. SMTP was designed to transfer email messages reliability through TCP, whereas the sender (SMTP client) establishes a two-way transmission channel with an SMTP server on port 25. The SMTP client transfers the email messages to one or more SMTP servers which could further act as SMTP clients. If the SMTP server is unable to transfer the message, it sends a failure report to the SMTP client (Klensin, J. 2008). SMTP, similar to Hypertext Transfer Protocol (HTTP), uses Commands/Response interaction, the commands are American Standard Code for Information Interexchange (ASCII) text-based and the responses are Status Codes and Phrases. For instance, when a host initiates a connection to an SMTP server it sends EHLO command, if the EHLO command was not

accepted by the SMTP server it returns failure reply codes such as 501, 502, 503 or 550. The same process of interaction goes for other SMTP commands such as MAIL FROM, RCPT TO, DATA, VRFY...etc. SMTP is usually implemented by MTA and MDA running on mail servers, figure (1.4) below shows a basic model of SMTP.
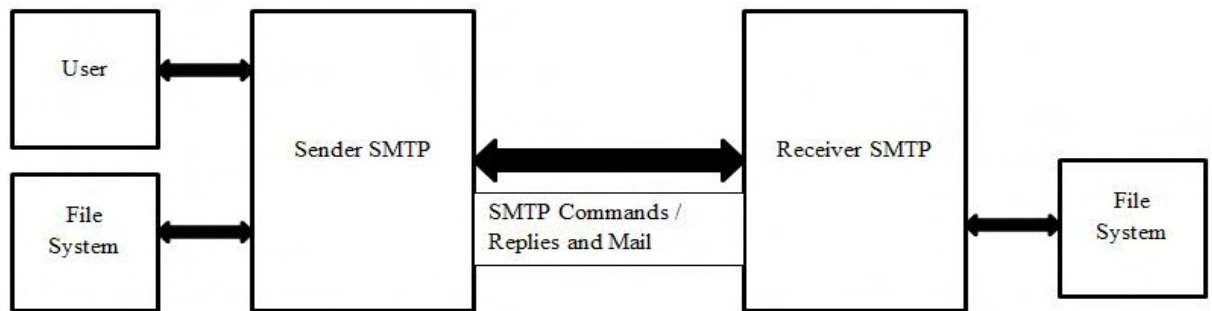
Figure 1.4. SMTP model (Klensin, J. 2008)

1.4 Email Message Components

An email message contains three major sections: Envelope, headers and body. The envelope is used by MHS to direct the routing of email messages; it contains the addresses to whom emails should be delivered to (Crocker, D. 2009). The headers are the most interesting part of the email as they contain information about the sender, receiver and the message. Each header has a name and a value separated by a colon (Resnick, P. 2008) for instance in the header "from: user@emailserver.com":

1. "from" is the header name

2. "user@emailserver.com" is the value

3. ":" is the colon that separates the header from the value.

Other examples of headers: To, From, Subject, Date…etc. and while the envelope part is used by SMTP server, the headers part provide useful and meaningful information to the email reader, thus, an envelope's addresses will most likely be the same as the header's addresses. The body is the main part of an email message containing the actual data such as text, images and videos, it is encoded using ASCII standard and Multi-Purpose Internet Mail Extensions (MIME) standard. The header is separated from the body by a blank line.

1.5 Advance Encryption Standard

Advance Encryption Standard (AES) is a modern standard used for the encryption of digital data; it was proposed by Vincent Rijmen and Joan Daemen and was chosen in 2001 by National Institute of Standards and Technology (NIST) to be the next block cipher standard that is used by the US government in their applications and to replace the old Data Encryption Standard (DES) (Selent, D. 2010). AES is a cipher that encrypts a 128-bit block from plaintext to ciphertext and vice versa. AES supports three key lengths of 128-bit, 192-bit, 256-bit and requires the sender and the receiver to share the same key. AES processes data blocks iterations, which are also called rounds, the number of rounds depends on the cipher length, for 128-bit key length the number of rounds is 10, for 192-bit key length the number of rounds is 12 and for the 256-bit key length the number of rounds is 14 (Heron, S. 2009).

On this study, emails will be encrypted using Advance Encryption Standard; encryption keys used to encrypt those messages will be auto generated using an algorithm presented in Chapter 3 as well as more detailed description of AES and its use in this research.

1.6 Problem Statement

In current emailing system the user cannot control the availability of emails after they have been sent to the destination. This may cause an unsecure mailing environment and in particular causes security penetration and thus confidential information could mistakenly be sent to third parties or incorrect parties without the ability to control them afterwards. In addition, a communication exchange uses the same encryption technique during the whole session; this gives higher probability for security vulnerabilities. Providing more control to the sender on the availability of his emails after being delivered to the receiver is very important to prevent wrong recipients from reading the email or forwarding it to other users. There were few efforts in this area done by some major Corporations like Microsoft, IBM and Google to replace or delete email messages after being sent through the use of several methods and techniques, but those solutions had several limitations that will be discussed in more details in chapter 3.

This study will propose a new model that aims to solve some of the issues mentioned above, the proposed model provides a technique to prevent unwanted recipients from reading electronic messages after receiving them without the need of using platform-dependent services like Microsoft Outlook's Message Recall or acquiring special email client applications. The proposed model should be structured in such a way that is applicable to work on all email servers if it becomes globally adopted in the future. This study focuses on controlling the availability of the emails after being sent to the recipient by encrypting the message before sending it and providing the sender the control on whether or not to allow the recipient to decrypt the message.

1.7 Problem Significance and Motivation

This study aims at providing a suitable solution for mailing systems by implementing an enhanced control over emails via designing a new paradigm in authentication protocols. The new model will allow the sender to control the availability of his messages after reaching the inbox of the receiver on the same email server or on a different email server without third parties being involved.

In order to achieve extra security, email senders must be allowed to gain control over sent messages to ensure that their messages are being viewed by the indented party and not by other parties, this study will present a new model that allows users to prevent unwanted recipients from reading the emails after being delivered to them, such model is expected to provide higher security especially for sensitive organizations that seek mailing systems as a secure and confidential mean of communication. Several scenarios can make use of the proposed method, one of which is the avoidance of sensitive data leakage caused by a company member sending design details to unintended recipient, if the proposed method is integrated, the sender of the message can, at any time, modify the message's availability status so that it can no longer be read by the receiver. Also, when corporations make business contracts, too much data and intellectual property is stored in emails, this causes more vulnerability if these data were used later by a competitor. Another scenario is when an email account that contains sensitive messages is hacked, in such case; the sender of a sensitive message can control its availability to the hacked email and prevent the hacker from reading the message's contents. Also, almost all ads and promotions that are sent to members and subscribers by email have a limited validity period but stay available in mailboxes forever.

1.8 Goals

- Provide control over sent emails and prevent unwanted recipients from viewing emails by encrypting the message before sending it and allowing the sender of the email to control the availability of the encryption key to the recipients.

- Provide enhanced security by encrypting the email message in AES and using a different auto generated 128-bit encryption key for every message, this would make the communication exchange more secure as it uses different message encryption keys for the entire duration of the session.

- The proposed model should be flexible to operate on cross-platform mail servers and not solely on a specific platform-dependent mail server and without the need of relying on online third parties or using special email programs like Microsoft Outlook or IBM Lotus.

1.9 Limitations

- In order for this proposed technique to become globally operational, major Email Service Providers (ESPs) should integrate and adapt the technique.

- An email may contain attachments with images, videos, binary files, etc., that uses MIME protocol over SMTP transmission which is not included in this study.

- The key exchange between the sender side and receiver side cannot be done without using an Application Program Interface (API).

1.10 Thesis Overview

Chapter one discusses the basic principles regarding email protocols in general and AES technique, the problem statement and the goals of this thesis.

Chapter two reviews previous studies and existing techniques related to the topic of the thesis and other related topics.

Chapter three explains the proposed methodology in details.

Chapter four presents the experimental settings and results, and also makes detailed comparisons between the standard technique of sending and retrieving emails and the proposed one.

Chapter five summarizes the major findings of the thesis and proposes future work.

Chapter Two

Literature Surveys

Chapter 2 Literature Surveys

This section describes some of the previous work related to controlling and securing emails, it also includes some of the techniques and protocols have been proposed to improve the way emails are processed.

(Abadi, M., etal., 2002) proposed a protocol used for certified email, the protocol guaranteed the delivery of the message by relying on an online Trust Third Party (TTP). TTP has a public key and a private key for encrypting and decrypting the messages, it also has another public key and another private key for signing messages and verifying those signatures. In this protocol the sender choses one of the four authentication options (BothAuth, TTPAuth, SAuth, and NoAuth) then the message is encrypted using a freshly generated key which is also encrypted using TTP public key, then the encrypted message along with encrypted key are sent to the receiver which sends the encrypted key to TTP. After TTP authenticates the receiver, it sends the key for decrypting the message to the receiver and a receipt to the sender, the receiver then receives the key and decrypts the message. Figure (2.1) on the next page shows the basic sketch of the study's protocol.
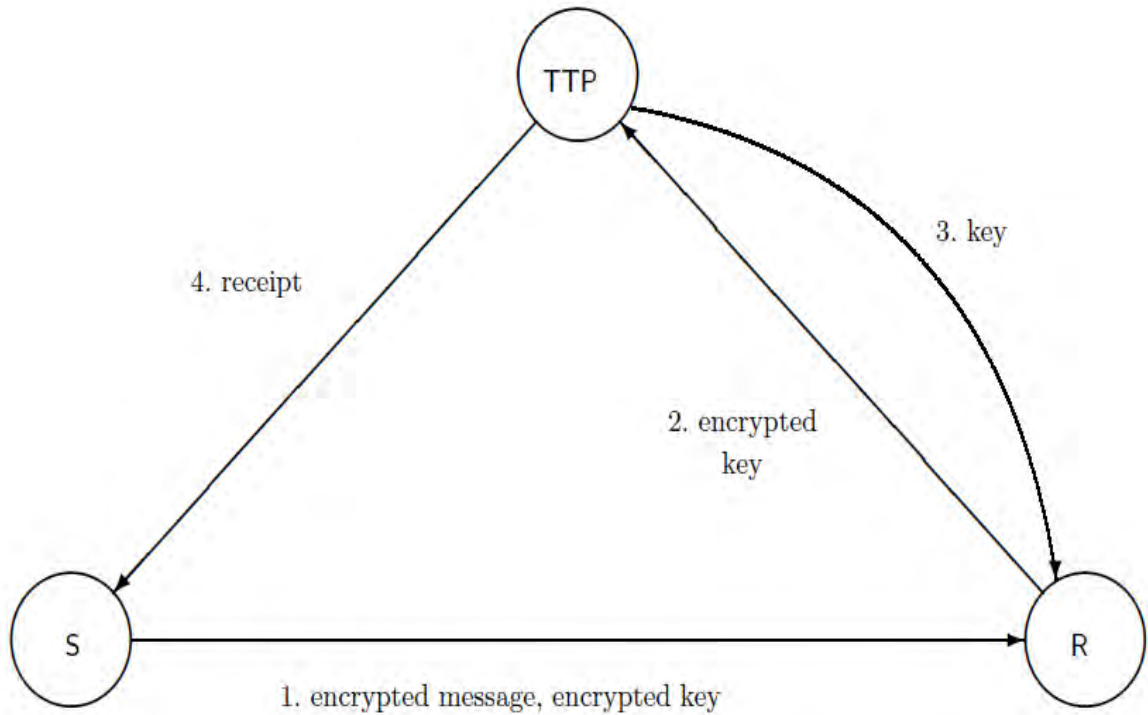
Figure 2.1. Protocol sketch for certified email (Abadi, M., & Blanchet, B. 2002)

In 2005 (Abadi, M., & Blanchet, B. 2005) used an automatic tool for analyzing and verifying the main security properties of the protocol stated above.

(Tanta-ngai, H., etal., 2003) stated SMTP was designed without consideration for security and proposed a new protocol, namely Secure Mail Transfer Protocol (SecMTP), as an extension to SMTP to provide secure email services between users while putting into consideration achieving five major security goals: confidentiality, integrity, authentication, non-repudiation and certification. The authors claimed that the proposed protocol is compatible with SMTP and its service extensions and provides guaranteed security on user-

to-user level as well as user-to-server and server-to-server security while stating some shortcomings like non-repudiation was provided at the server level not at the client level, both the encryption/decryption processes were done at the server, SecMTP's servers bottleneck issues, excessive contention for internal server resources and other shortcomings related to email headers.

(Lux, K. D., etal., 2005) stated that current email protocols have some limitations in security and performance, and discussed the techniques used to mitigate security issues in internet messaging by designing it as a family of web services, called WSEmail, on which the messages are Simple Object Access Protocol (SOAP) messages that use web service security features for message transmissions. In a basic scenario the proposed architecture for sending messages in WSEmail is closely similar to SMTP in which the Sender's MUA makes a call (all of the calls are SOAP calls) on its MTA to send a Message (the Message is contained in the body of SOAP message and it's structured as a collection of XML elements), then the Sender's MTA makes a call to the Receiver's MTA to deliver the email from the Sender Domain to the Receiver Domain. The Receiver's MUA makes a call to its MTA to obtain message headers and then requests the Message if it wishes. There are many variations to this scenario as the authors detailed a design for instant messaging, in such case if the Sender's MUA is outside the Sender's Domain then it calls its MTA to obtain a security token to be recognized by the Receiver's MTA, Once this is obtained, The MUA of the Sender sends the Message authenticated with this credential to the Receiver's MTA and indicates that it should be treated as an instant message by the Receiver's MTA and the

Receiver's MUA. Assume the following, as figure (2.2) below shows the messaging architecture of WSEmail:

- ❖ Sender's MUA = SC1   Sender's Server (Sender's MTA) = SS

- ❖ Sender's MUA outside the Sender's Domain = SC2

- ❖ Message from SC1 = M   Message from SC2 = M2

- ❖ Sender's Domain = SD   Receiver's MUA = RC

- ❖ Token = T   Receiver's Domain = RD

- ❖ Receiver's Server (Receiver's MTA) = RS
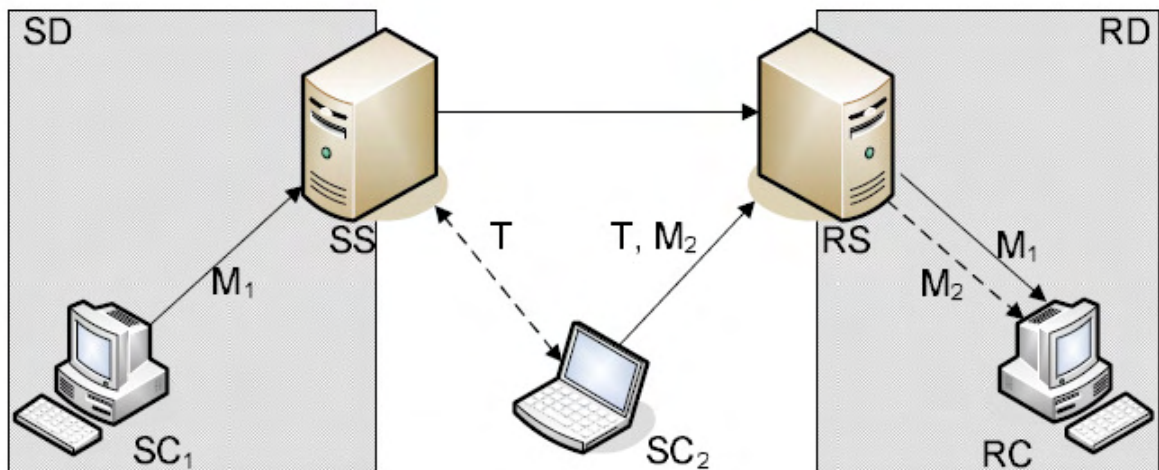
Figure 2.2. WSEmail messaging architecture (Lux, K. D., J May, M., Bhattad, N. L., & Gunter, C. A. 2005)

The authors claimed that relying on web services for internet messaging provides the same functions of ordinary email protocols but with additional security functions and more flexibility. The author then concluded that WSEmail faces major problems with

standardization and interoperability with SMTP, and suggested to mitigate that by writing more plugins to the SMTP.

(Shao, J., etal., 2007) presented a novel approach based on the authorized Diffie-Hellman key agreement protocol to propose a certified email protocol with an offline TTP. Unlike other certified email protocols with an offline TTP, the proposed protocol encrypts the message with a key shared between the sender and the TTP without involving the TTP during the exchange and that's where Diffie-Hellman key agreement came to play, as it was used to achieve the goal of sharing the message encryption between the sender and the TTP, this is in contrast to typical certified email protocols which use TTP's public key to encrypt the message. The authors claimed that their protocol enjoys being fair, optimistic, TTP's stateless and support high performance, the paper compared the proposed protocol with other existing optimistic certified email protocols and concluded that their proposed protocol is the most efficient one in terms of the number of exponentiations and communication data and suggested it would be suitable for applications in a disturbed environment.

(Al Bazar, H., etal., 2008) argued that POP3 protocol has some latency in retrieving email messages from email servers and proposed a new architecture to enhance the current POP3 standard, this was suggested by cancelling the email server greeting message and using the Transactional Transmission Control Protocol (T/TCP), which is a variant of the TCP, and APOP command, which is an alternative for using USER and PASS commands, during the process of creating the TCP connection. Also, a modified pipelining approach in the POP3

procedure is proposed to enhance the email server performance and reduce latency in delivery time.

(Harn, L., & Ren, J. 2008) proposed a new model that aims at providing deniable authentication to protects personal privacy of the message's sender, the authors assumed that each client has two pairs of keys; one pair is used for message encryption while the other is used for message authentication. The message format and delimiters are left open for future integration as they claimed that their design could be easily integrated with current implementations of Pretty Good Privacy (PGP) and Secure/Multipurpose Internet Mail Extensions (S/MIME). The purpose of their design was to enable a specific message receiver to authenticate the message and to allow the message sender to deny generation of the message in order to protect his personal privacy as stated above.

(Banday, M. T. (A) 2011) stated Transport Layer Security (TLS), and its predecessor Secure Socket Layer (SSL) and Secure SMTP are encryption based methods that respectively create encrypted secure channel between the sending and receiving MTA's at sockets and transport layers. They are simple methods to obtain email privacy without user's efforts but Secure SMTP over TLS guards only the path between client and server and not the endpoints that are authenticated by certifying authorities and not the DNS. The author stated that SMTP lacks security features in terms of privacy and authentication of sending party and that emails are being processed through several intermediaries until it reaches the destination which makes them more vulnerable to attackers who can read those emails; the author stated that some Email Service Providers store copies of email messages

even after being deleted by the users from their mailboxes, the author claimed that the current SMTP does not verify email headers which results in giving the opportunity for the senders to lie about their true identity, date and time of messages' creation and other information. The paper also discussed how MTAs could make changes to the messages that could unreliably be attributed back to the sender. The study shed lights on how email messages' headers are being processed by SMTP and argued that the trace information which is inserted at the beginning of the messages is not taken into consideration or being processed correctly by ESPs when reporting spoofed emails.

(Banday, M. T. (B) 2011) claimed that cybercriminals sometimes manipulate date and time of email messages when committing cybercrimes to cover them up and the date header field of an email message can be violated without being detected by email system protocols, a date spoofed email contains a date which is either ahead or before the actual sending date of the email. The author claimed that Timestamping could ascertain the validity of date and time of the email message creation, email message submission and email message transmission of sending sides and receiving side, thus, the author proposed a model that used a third party for Trusted Time-Stamping Service (TTS) for authentication of date in email messages. The proposed model checks date spoofing at sending, forwarding, and receiving servers by using Time Stamping Protocol (TSP) and control date spoofing at every handling node to ensure credibility of dates.

(Sobh, T. S., & Amer, M. I. 2011) aimed to standardize a protocol to encrypt and digitally sign email correspondence, the authors discussed end-to end security, reviewed some

certified email protocols and proposed some improvements to enhance security, the paper demonstrated a prototype of the proposed protocol and the prototype was implemented using Component Object Model (COM) and web services. Figure (2.3) below shows a general block diagram of the proposed model.
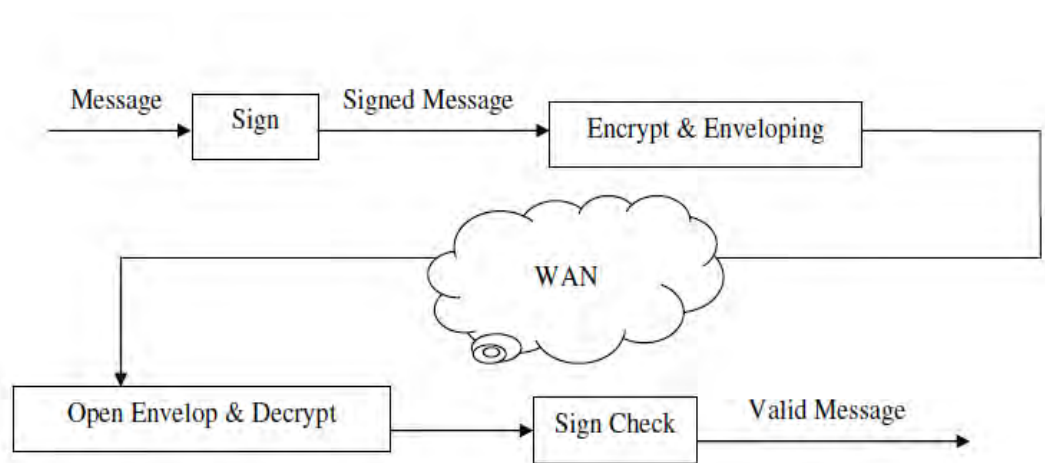
Figure 2.3. General block diagram of the standardized proposed model (Sobh, T. S., & Amer, M. I. 2011)

(Mantoro, T., & Zakariya, A. 2012) proposed a method for securing email communication exchange on mobile devices running on Android platform by using a combination of symmetric encryption (AES 128 bit), asymmetric encryption (RSA 1024 bit) and hash function (SHA-1). The study aimed at meeting the aspects of information security consisting of confidentiality, data integrity, authentication and nonrepudiation between two clients. The proposed model works as follows: both the sender and the receiver have public and private keys and publish their public key on a public-key infrastructure server. The email is encrypted using AES 128 bit by a secret key which is generated randomly in a system using random number generator. The email is hashed using SHA-1 bit to get the message digest of the email before the encryption process. The secret key used for

encrypting the email along with the message digest of the email are encrypted using RSA 1024 and the receiver's public key, then the encrypted secret key along with the encrypted message digest of the email along with the encrypted email are sent to the receiver's server using SMTP. The receiver downloads the email using POP3 and decrypts the encrypted secret key with his private's key and the encrypted message digest of the email with the sender's public key using RSA 1024. The encrypted email is decrypted with the secret key using AES 128, if the receiver produces the same message digest, then the email is verified and there were no alterations or modifications during the transmitting process. The authors also made some experimental tests and stated that their proposed model was successful in meeting those aspects; the authors also suggested making the system interoperable with other platforms on mobile devices and on desktop computers in the future. Figure (2.4) on the next page shows the sending side's processes of securing an email.
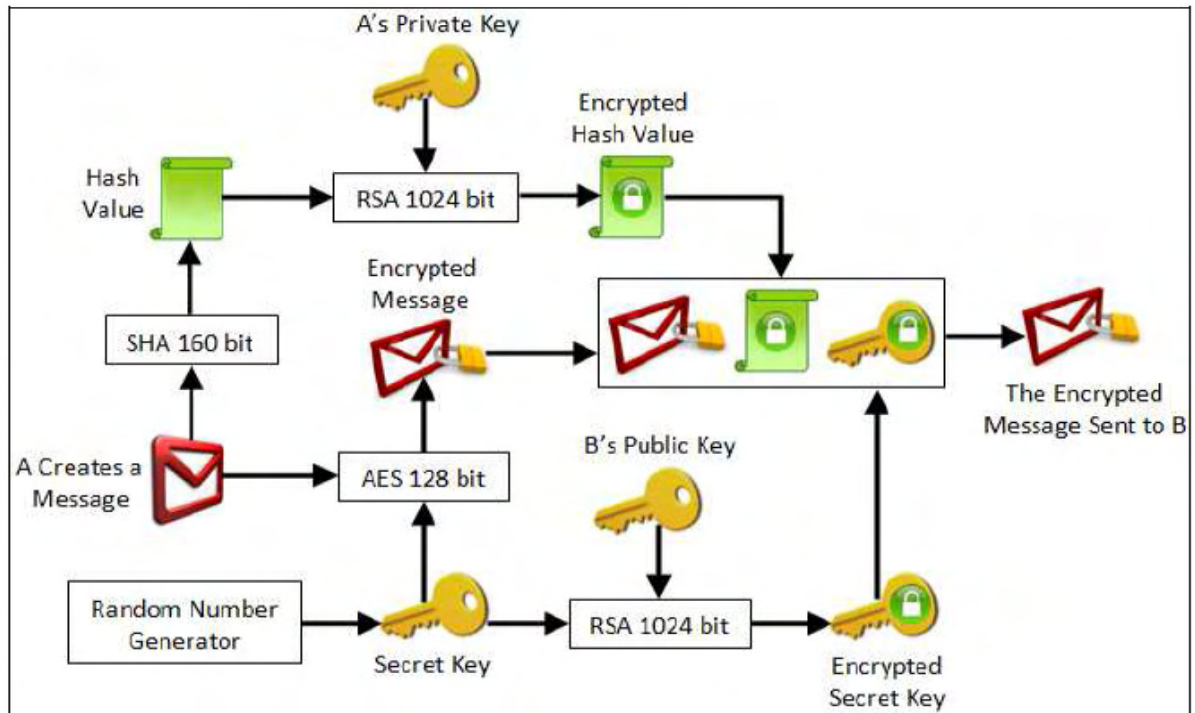
Figure 2.4. The sending side processes for securing an email using hybrid cryptosystem on android-based mobile devices (Mantoro, T., & Zakariya, A. 2012)

The researcher argues that even though the proposed model in this study is impractical to implement nowadays because SHA 160 is no longer approved for most cryptographic uses after 2010, due to its cryptographic weaknesses that were discovered later, the model itself is still very interesting and worthy of studying especially if we consider integrating one of the newer more approved versions of the secure hash algorithms, SHA-2 or SHA-3, into their proposed model instead of the SHA-1.

(Suresh, K. B., & Jagathy, R. V. P. 2012) proposed a secure Email system relied on Identity-Based Encryption (IBE), DNS and Proxy Service. When the sender wishes to send an Email, the proxy server checks if the recipient has the Master Public Key (MPK) in its

DNS, if it does, the proxy server encrypts the email and sends it to the recipient by an SMTP server, then the recipient decrypts the email to read. If the recipient does not has the MPK it cannot handle the decryption process, in such case the sender domain sends a secure web link to the recipient, when the recipients clicks on this link, the sender's proxy server checks its database to verify if the recipient is authorized to view the email, if yes, it decrypts the email message for the recipient. If the recipient is not authorized to view the email, the proxy server sends all of the recipient's details to the sender for approval, if the sender approved, the recipient is be added to the authorization list in the database, the encryption/decryption process is performed by the proxy server, figure (2.5) below shows the process in a simplified way.
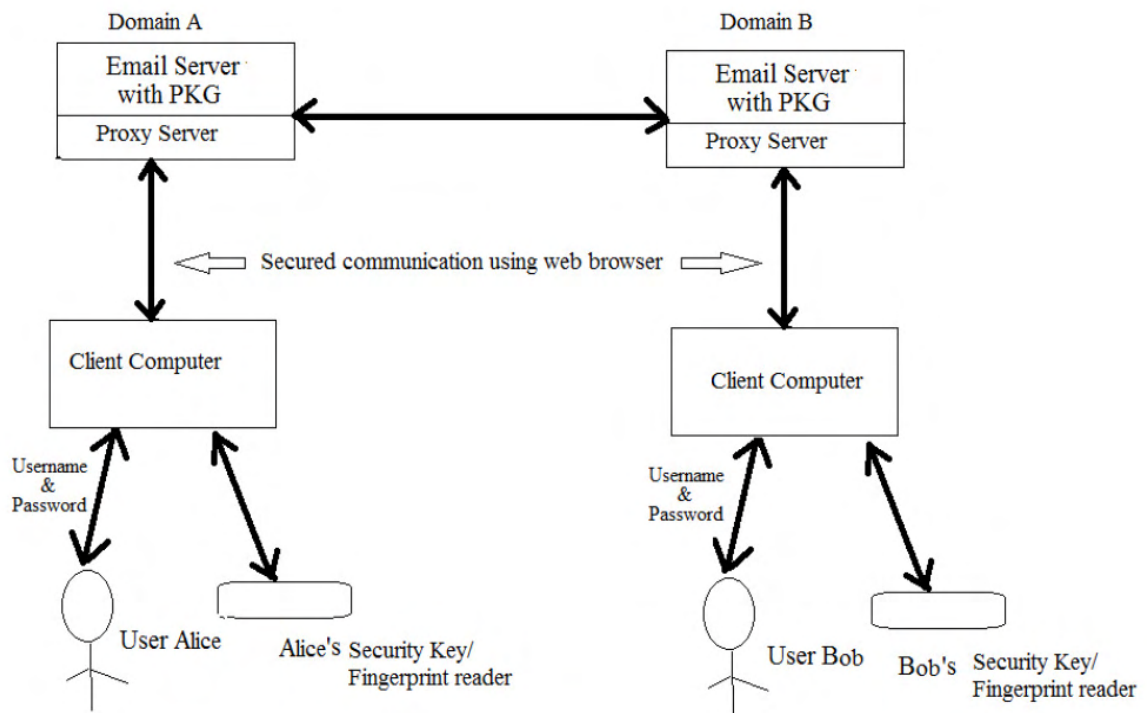
Figure 2.5. Email system architecture based on IBE, DNS and proxy service (Suresh, K. B., & Jagathy, R. V. P. 2012)

(Praveen, B. 2013) proposed a protocol called Undo Sent Email (USE) which aims at retrieving an email after it has been sent to the receiver if the sender manages to pull the email first from the POP3/IMAP server. The proposed protocol has two major shortcomings: if the receiver has read the email first then the email cannot be retrieved successfully, the proposed protocol suffers when the domain names of the sender and the receiver are not the same, meaning, the protocol cannot function between cross-platform email service providers.

There were also some efforts in controlling the availably of sent emails done by leading email corporations on their email services. The researcher lists below some of those techniques along with their limitations:

1. Google Mail allowed its users to use a service called "delay-send"; as the names suggests, this technique delays and schedules the sending of emails for a specific time, during that time the email sender has the opportunity to cancel the sending of the email to prevent the message from being sent. This service had the following limitations:

   • This is not "preventing the receiver from reading the message after it has been sent"- it is simply cancelling the message from getting sent.

   • If the time of the delay ends and the emails were sent to the receiver, the user cannot prevent the receiver from reading them.

2. Microsoft provided their email users with a function called "Message Recall" which is used to bring back a message that was sent earlier, or un-send it. Message Recall has been used to delete unread messages from the recipients' inboxes so they never see it. This function had the following limitations (Microsoft Customer Support. 2012):

- The function would not work if the recipient is not using Outlook.

- The function would not work if the recipient is not logged on to the mail service provider.

- The function would not work if the recipient is using Cached Exchange Mode and is working offline.

- The function would not work if the original message is moved from the recipient Inbox.

- The function would not work if the original message is opened first and marked as read. This can occur when the message is displayed in the Preview Pane or Reading Pane.

3. IBM provided a Message Recall function in their IBM Lotus Notes similar to the one Microsoft has provided in their Outlook but it had the following limitations (Bulloch, S. 2007):

- The Message Recall would only work if the recipient is using Lotus Notes/Domino v8 or later, it doesn't work with IBM Lotus Domino Web Access or other email clients like Outlook or web-based email clients.

- The sender has to keep a copy of the message in his mail file to recall it successfully, this is due to the fact that Message Recall cannot be used until signatures are checked for security purposes, so without the original copy of the email in the sender's mail file, Message Recall cannot function.

One of the problems with Message Recall function is that it can only work on a combination of a specific email client and a specific email server, for example Microsoft Outlook cannot provide this function without relying on Microsoft Exchange Email Server, the same goes for IBM Lotus Notes which needs the Domino Server to work with, for that reason the recipient of the email has to be using for example Microsoft Outlook for this function to work, if the recipient is not using Microsoft Outlook then the sender cannot prevent the recipient from reading the email. This solution might work for users who use Microsoft's Outlook/Exchange Server or IBM's Lotus Notes/Dominos Server but for others who use different or incompatible email clients or web browsers such solutions would not work.

Chapter Three

Proposed Model

Chapter 3 Proposed Model

The proposed model aims at granting control to the sender of an email message over its availability to the receiver, the sender should be able to deter the receiver from reading the message as wished, this can be achieved by encrypting the content of the message using AES algorithm by a secret key before sending it, the secret key is automatically generated by the server using Universally Unique Identifier (UUID) generation algorithm and is stored at the sender's server, this key can be requested by the receiver at the time of opening the message through an API located at the sender's server. The sender's server will provide the secret key located in the server's database needed to decrypt the message for the receiver in order to become readable, however, the availability of the secret key that is needed to decrypt the message can be modified by the sender which in turn controls its availability to the receiver's server and thus to the receiver, the sender can make the key unavailable whenever he wishes to, which is going to deter the receiver from decrypting and reading the message.

The author proposes a customized header called X-Ekey to indicate the location of the API that should be accessed in order to retrieve the secret key needed to decrypt the message; the MUA is going to be responsible for encrypting and submitting the key to the server.

The following steps constitute an overview of the process:

Sender:

1- The user composes a message and requests to send it.

2- The MUA generates a key and encrypts the message using that key.

3- The MUA adds a header, to the email message, indicating the API that will be used to request the key later.

4- The key is then stored at the sender's server's database.

Receiver:

1- After the receiver requests the message, the MUA will look for the API location.

2- The receiver's MUA will post a request to the API along with a message identifier to receive the decryption key.

3- The sender's server will return the requested key needed for decryption if its status is set to available, however if the sender changed the status of the key, the server will send an error code indicating that the key is unavailable.

4- If the key was received successfully, the receiver's MUA will decrypt the message; however, if the MUA received an error code, the message cannot be read because it is encrypted.

3.1 The Proposed Sender Side Model

Figure (3.1) below shows the proposed model from the sender's side perspective.
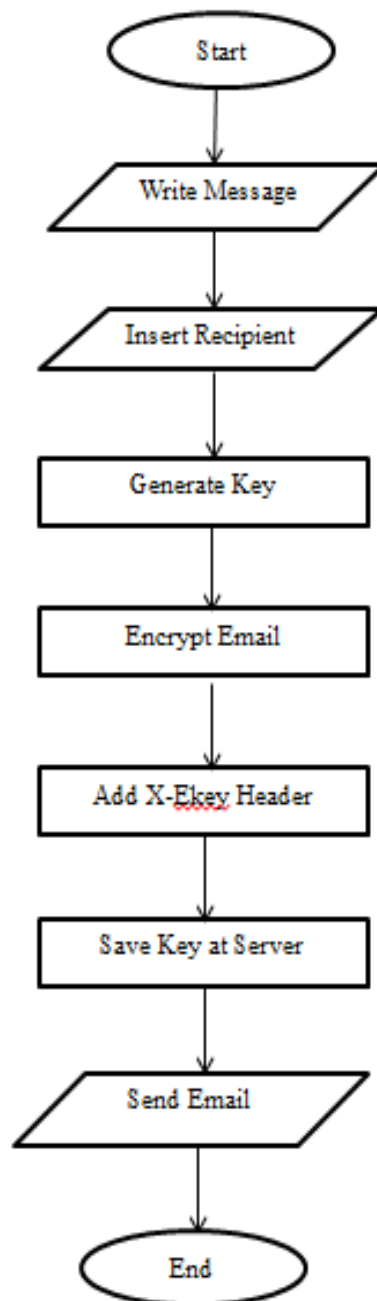
Figure 3.1. The proposed sender side model

The steps of the flowchart are explained in more details below

1. The user writes the message.

2. The user enters the recipient's email address.

3. The server generates a random key for message encryption.

4. The server encrypts the message using the key that was generated in step 3.

5. The server adds a new header "X-Ekey" along with sender's server's address. This header indicates that the message is encrypted and the receiver's server needs to ask the sender's server for the key in order to decrypt it, by doing so, the receiver will only be able to read the message as long as the sender keeps the key available, however, once the key becomes unavailable, the receiver won't be able to read the message, this technique should provide cross platform control over sent messages.

6. When the encryption is done the key is stored at the sender's server.

7. Finally, the message is sent to the receiver.

At the sender's side the composition, addition of headers and encryption of the message is achieved, this is done by the MUA which can be defined as a computer program used to access and manage user's emails; this vital component of the mailing system is responsible of handling:

1- Retrieval of messages from a mailbox.

2- Message composition.

3- Submitting messages to a server.

4- Encryption.

      a. Encryption of mail sessions.

      b. Encryption of the message body.

The work proposed in this study will take place in the message composition and encryption tasks which is explained in more details over the next pages.

3.1.1 Message Composition

Email clients contain user-friendly interfaces that help users compose and view messages. The email clients will format the message headers and body according to RFC 5322 standards; email clients could use MIME standards for non-textual contents and attachments which are not included in this study. Message headers can include the destination fields, To, Cc, and Bcc, and the address from which the message was sent. A new custom header called X-Ekey is being used to provide the location of the secret key and a unique identifier of the message on which the key is associated with, the header line will be combined with the sender's email to remove the possibility of duplicate identifiers, this header is going to be used by the receiver's MUA to access the key and decrypt the email. The custom header can be added to the message SMTP headers by the following code in figure (3.2) on the next page.

```
class MyMailer < ActionMailer::Base
  def welcome
    mail to:    'someone@example.com', # normal mailer stuff
      from:    'you@yourdomain.com',
      subject: 'blah blah'

    headers['X-Ekey'] = "http://www.example.com/keys.php?id=
3F2504E0&sender=you@yourdomain.com"
  end
end
```

Figure 3.2. Adding custom header

The identifier of the email is generated sequentially for each sender and is represented using hexadecimal digits.

3.1.2 Encryption

Along with the standard message encryption standards, the message in the proposed method will also be encrypted using the AES by using auto generated UUID as an encryption key whose length is 128 bits, so that the total number of unique UUIDs is 2^{122} (approximately 5.3×10^{36}), the key is generated using a scheme relying only on random numbers, the algorithm of generation is the same as the UUID Version 4, the algorithm sets the version number (4 bits) along with 2 reserved bits, however, the rest of the bits (122 bits) are generated using pseudorandom data source, V4 UUIDs are formatted as the following scheme: "xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx" where x is any hexadecimal digit and y is either 8, 9, A, or B, however, the dashes in the identifier will be removed before using it as an encryption key, the UUID can be generated using the following PHP code in figure (3.3) on the next page .

```
/**
 * Generates version 4 UUID: random
 */
public static function v4() {
  if (!function_exists('uuid_create'))
    return false;

  $context = $uuid = null;
  uuid_create($context);

  uuid_make($context, UUID_MAKE_V4);
  uuid_export($context, UUID_FMT_STR, $uuid);
  return trim($uuid);
}
```

Figure 3.3. UUID code generation

3.1.2.1 Key Structure & Format

The reason behind using UUID to construct encryption keys is that there is no centralized authority required to control them, and as a result, a generation on demand can be fully automated. The UUID generation algorithm described in this section supports a very high allocation rate (up to 10 million per second per machine) so they can also be used as IDs and identifiers, UUID version 4 is of a fixed length (128 bits) which makes it look relatively short when compared to other alternatives and since it is unique across both space and time with respect to the space of all other UUIDs, this makes it a candidate to be used as encryption keys.

In this section, the structure of the UUID key is discussed in more details. The formal UUID string representation can be defined as the following:

UUID = time-low "-" time-mid "-" time-high-and-version "-" clock-seq-and-reserved clock-seq-low "-" node

An example of UUID is: 859c112c-0d15-4ce6-808a-2d9b7a8b6d65

Table 3.1 below explains the structure of the UUID in more details.

Table 3.1. UUID structure

Field	Length	Data type	Octet	Note
Time-low	4 octets	Unsigned 32 bit integer	0-3	The lower field of the timestamp
Time-mid	2 octets	Unsigned 16 bit integer	4-5	The middle field of the timestamp
Time-high-and-version	2 octets	Unsigned 16 bit integer	6-7	The higher field of the timestamp multiplexed with the UUID version number
clock-seq-and-reserved	1 octet	Unsigned 8 bit integer	8	The higher field of the clock sequence multiplexed with the UUID variant
clock-seq-low	1 octet	Unsigned 8 bit integer	9	The lower field of the clock sequence
Node	6 octets	Unsigned 48 bit integer	10-15	The spatially unique node identifier (For UUID version 4, the node field is a randomly or pseudo-randomly generated 48-bit value)

In this research, the version 4 UUID is being selected for the relative strength found in the Node block, the pseudo-random numbers in the Node block in UUID version 4 can be generated by performing following the steps on next page:

1- Set the two most significant bits of the clock_seq_and_reserved to zero and one, respectively.

2- Set the four most significant bits of the time_high_and_version to the 4-bit version number.

3- Set all the other bits to randomly (or pseudo-randomly) chosen values.

3.1.2.2 Using AES

In this research, AES, which became effective as a federal government standard on May 26, 2002, is selected as the main encryption algorithm to encrypt outgoing messages. The algorithm is based on substitution-permutation network and is a variant of Rijndael algorithm which has a fixed block size of 128 bits.

AES requires low RAM and performs well on different hardware with an accepted level of security; this encryption algorithm is selected to encrypt emails before sending them to their destination.

AES operations are performed in a special finite field and it operates on a 4x4 column-major order matrix of bytes called the state, the length of the key determines the number of rounds (cycles) as shown on the next page.

- 10 cycles of repetition for 128-bit keys.

- 12 cycles of repetition for 192-bit keys.

- 14 cycles of repetition for 256-bit keys.

Since UUID v4, which is used as an encryption key in this research, is 128 bits in length, the AES will repeat the encryption cycle 10 times which should give an acceptable level of security within a reasonable execution time, figure (3.4) below shows the AES flowchart.
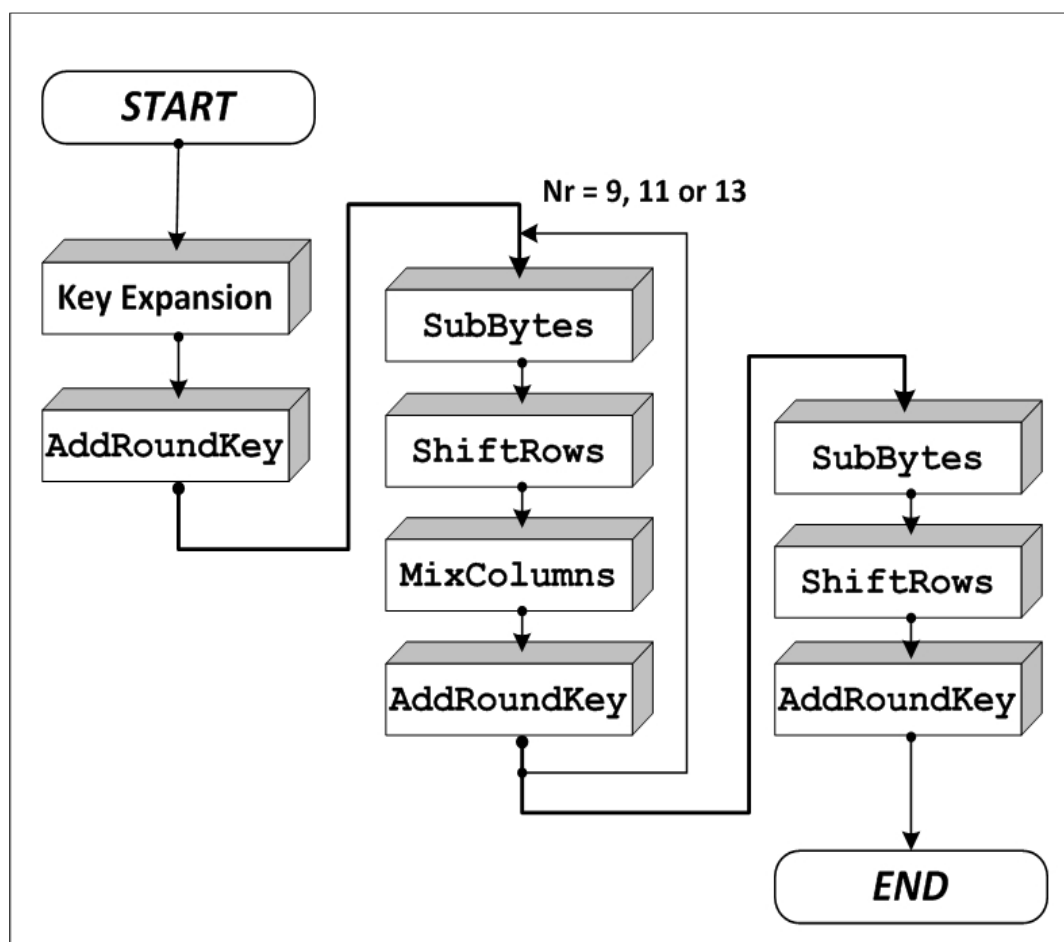
Figure 3.4. AES flowchart

The steps of the flowchart can be described as the following:

1- Expansion of key: round keys are derived from the original key using Rijndael's key schedule to generate different round key of size 128-bit for each round plus one more.

2- Initial Round (Add Round Key): each byte of the state is xored with a block of the round key.

3- Rounds

    a. Sub Bytes: a non-linear substitution step where every byte is replaced with another byte from a lookup table.

    b. Shift Rows: shift the last three rows of the state cyclically a certain number of steps.

    c. Mix Columns: combine the four bytes in each column.

    d. Add Round Key

4- Final Round (no Mix Columns)

    a. Sub Bytes

    b. Shift Rows

    c. Add Round Key.

Figure (3.5) on the next page shows the code for encrypting and decrypting the email messages.

Figure 3.5. Code of encryption and decryption of emails

After composing, generating the key and encrypting the message, the encrypted message is

submitted to the server for transfer and the key is stored in server's database along with flag

indicating its availability status, this key can be accessed through the API provided in the

email header, the server shall send the requested key if the flag is set to available or -1 otherwise.

3.2 The Proposed Receiver Side Model

Figure (3.6) below shows the proposed model from the receiver's side perspective.
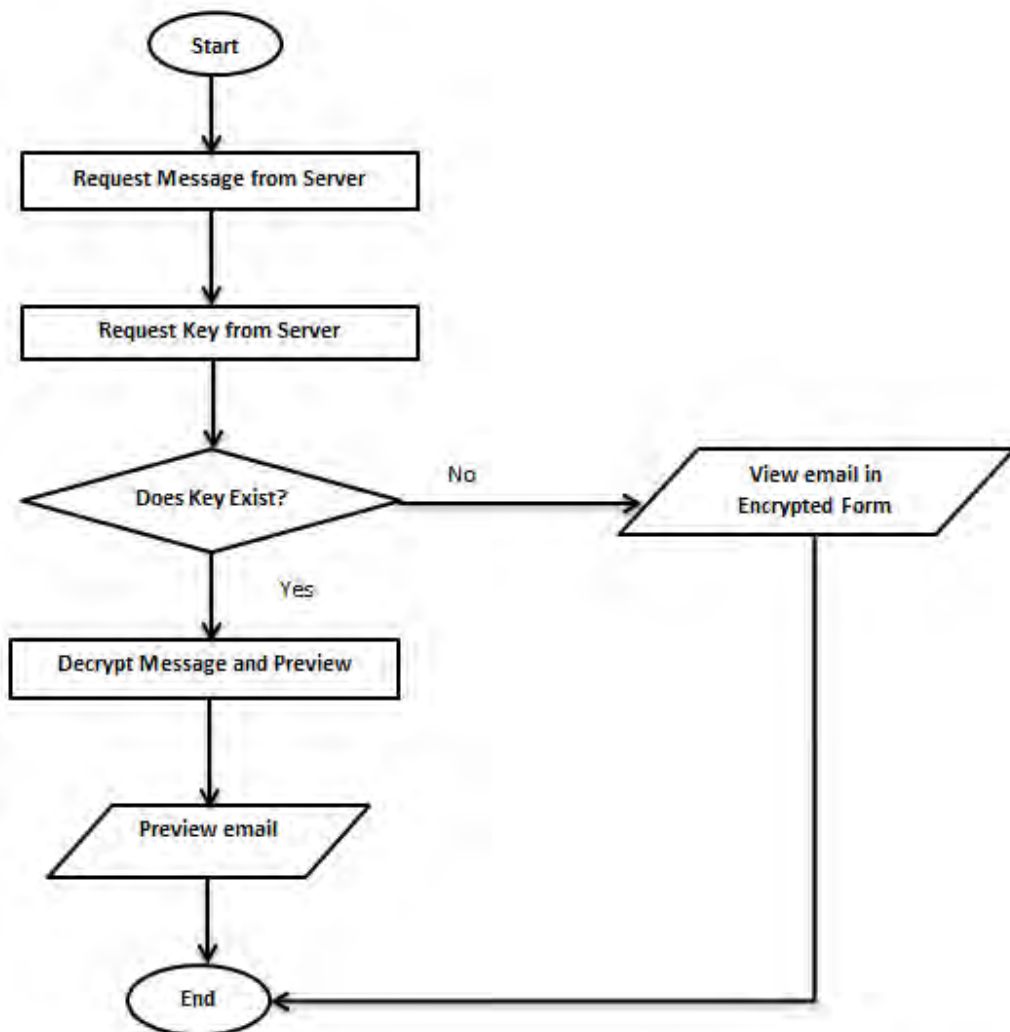
Figure 3.6. The proposed receiver side model

The steps of the proposed receiver side model are explained in more details below:

1. The user requests the message from the email server.

2. The receiver's server reads the X-EKey header of the email and realizes the message is encrypted; the receiver's server locates the API of the sender's server and requests the key for decrypting the message.

3. If the sender didn't suspend the key, the sender's server sends the key to the receiver's server and then the message is decrypted and the server previews the message to user.

4. If the sender has suspended the key then the receiver won't be able to read the message because it is encrypted.

Chapter Four

Experimental Results

Chapter 4 Experimental Results

Several experiments were conducted to evaluate the performance of the proposed technique and its effect on the quality of the email exchange process. Different measures and criteria are being reviewed in this chapter to evaluate the performance of the proposed technique.

4.1 Measures

A series of experiments were conducted to show the effectiveness of the proposed technique. The efficiency of the technique is measured by four parameters which are:

1- Time to Send (TTS) the email:

TTS is the measure of the time required to generate an identifier and encryption key, encrypt the email and send it, TTS is measured in milliseconds by using the following formula:

$$TTS = T_2 - T_1$$

Where:

T_1: The time when send command is received by the server.

T_2: The time when the message took off from the server.

2- Time to Read (TTR) the email:

TTR is the measure of time required to read the email headers, request the decryption key located on a specific server and decrypt the email, TTR is measured in milliseconds by using the formula on the next page:

$$TTR = T_4 - T_3$$

Where:

T_3: The time when email headers are read.

T_4: The time when the decryption of message is completed.

Note that the process of requesting and receiving the key falls within T_1 and T_2.

3- Original Email Size (OES):

OES is a measure of the email body size before the encryption process, it is measured in bytes.

4- Prepared Email Size (PES):

PES is the measure of the email body size after the encryption process, it is measured in bytes.

4.2 Experimental Setup

- **Platform:** The experiments were conducted on a shared host with 2.0 GB of dedicated RAM.

- **Input Email Format:** Experiments were conducted on text based emails only.

- **Message lengths:** There are four different message lengths used in experiments, the messages are generated randomly by typecasting the result of the Random() function in C#, the function takes time as a seed and generates a normally distributed values within the specified range, the results collected from calling the method for x times are then typecasted to chars and appended to a string to simulate the body of a randomly generated message.

    a. The Small Message Experiment is conducted using four different random texts of size 1000 characters.

    b. The Medium Message Experiment is conducted using four different random texts of size 5000 characters.

    c. The Large Message Experiment is conducted using four different random texts of size 10000 characters.

    d. The Very Large Message Experiment is conducted using four different random texts of size 20000 characters.

- **Methods:**

    a. The proposed technique.

    b. The normal process.

4.3 Experiments

In this section, the results of the four experiments are being studied and discussed in details to evaluate the differences in performance.

4.3.1 Small Message Experiment

The first experiment was conducted using four different randomly generated messages of size 1000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email.

Proposed Technique

In this section, the results of the proposed method when sending four different messages of size 1000 characters are being examined.

Table 4.1 shows the examined parameters when sending the first randomly generated message of 1000 characters as an email using the proposed technique.

Table 4.1. Parameters of first 1000 character email using the proposed technique

Original Email Size	10456 bytes
Time to Send	15 msec
Prepared Email Size	14016 bytes
Time to Read	50 msec

Table 4.2 shows the examined parameters when sending the second randomly generated message of 1000 characters as an email using the proposed technique.

Table 4.2. Parameters of second 1000 character email using the proposed technique

Original Email Size	10416 bytes
Time to Send	52 msec
Prepared Email Size	14016 bytes
Time to Read	76 msec

Table 4.3 shows the examined parameters when sending the third randomly generated message of 1000 characters as an email using the proposed technique.

Table 4.3. Parameters of third 1000 character email using the proposed technique

Original Email Size	10616 bytes
Time to Send	11 msec
Prepared Email Size	14336 bytes
Time to Read	45 msec

Table 4.4 shows the examined parameters when sending the fourth randomly generated message of 1000 characters as an email using the proposed technique.

Table 4.4. Parameters of fourth 1000 character email using the proposed technique

Original Email Size	10336 bytes
Time to Send	41 msec
Prepared Email Size	14336 bytes
Time to Read	61 msec

Normal Process

In this section, the results of the normal process when sending four different messages of size 1000 characters are being examined.

Table 4.5 shows the examined parameters when sending the first randomly generated message of 1000 characters as an email using the normal process.

Table 4.5. Parameters of first 1000 character email using the normal process

Original Email Size	10456 bytes
Time to Send	32 msec
Prepared Email Size	10456 bytes
Time to Read	20 msec

Table 4.6 shows the examined parameters when sending the second randomly generated message of 1000 characters as an email using the normal process.

Table 4.6. Parameters of second 1000 character email using the normal process

Original Email Size	10416 bytes
Time to Send	27 msec
Prepared Email Size	10416 bytes
Time to Read	32 msec

Table 4.7 shows the examined parameters when sending the third randomly generated message of 1000 characters as an email using the normal process.

Table 4.7. Parameters of third 1000 character email using the normal process

Original Email Size	10616 bytes
Time to Send	13 msec
Prepared Email Size	10616 bytes
Time to Read	19 msec

Table 4.8 shows the examined parameters when sending the fourth randomly generated message of 1000 characters as an email using the normal process.

Table 4.8. Parameters of fourth 1000 characters email using the normal process

Original Email Size	10336 bytes
Time to Send	14 msec
Prepared Email Size	10336 bytes
Time to Read	20 msec

Summary

The examined results when sending 1000 characters as an email are being averaged out in this section to help show differences in parameters among the tested techniques. Table 4.9 shows the average of values measured for both techniques.

Table 4.9. Average performance for small message length experiment

	Proposed Technique	Normal Process
Original Email Size	10456 bytes	10456 bytes
Time to Send	29.75 msec	21.55 msec
Prepared Email Size	14176 bytes	10456 bytes
Time to Read	58 msec	22.75 msec

4.3.2   Medium Message Experiment

The second experiment was conducted using four different randomly generated messages of size 5000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email.

Proposed Technique

In this section, the results of the proposed method when sending four different messages of size 5000 characters are being examined.

Table 4.10 shows the examined parameters when sending the first randomly generated message of 5000 characters as an email using the proposed technique.

Table 4.10. Parameters of first 5000 character email using the proposed technique

Original Email Size	49616 bytes
Time to Send	64 msec
Prepared Email Size	66240 bytes
Time to Read	68 msec

Table 4.11 shows the examined parameters when sending the second randomly generated message of 5000 characters as an email using the proposed technique.

Table 4.11. Parameters of second 5000 character email using the proposed technique

Original Email Size	49296 bytes
Time to Send	24 msec
Prepared Email Size	65888 bytes
Time to Read	68 msec

Table 4.12 shows the examined parameters when sending the third randomly generated message of 5000 characters as an email using the proposed technique.

Table 4.12. Parameters of third 5000 character email using the proposed technique

Original Email Size	49016 bytes
Time to Send	20 msec
Prepared Email Size	65536 bytes
Time to Read	52 msec

Table 4.13 shows the examined parameters when sending the fourth randomly generated message of 5000 characters as an email using the proposed technique.

Table 4.13. Parameters of fourth 5000 character email using the proposed technique

Original Email Size	49088 bytes
Time to Send	34 msec
Prepared Email Size	65608 bytes
Time to Read	44 msec

Normal Process

In this section, the results of the normal process when sending four different messages of size 5000 characters are being examined.

Table 4.14 shows the examined parameters when sending the first randomly generated message of 5000 characters as an email using the normal process.

Table 4.14. Parameters of first 5000 character email using the normal process

Original Email Size	49616 bytes
Time to Send	44 msec
Prepared Email Size	49616 bytes
Time to Read	38 msec

Table 4.15 shows the examined parameters when sending the second randomly generated message of 5000 characters as an email using the normal process.

Table 4.15. Parameters of second 5000 character email using the normal process

Original Email Size	49296 bytes
Time to Send	24 msec
Prepared Email Size	49296 bytes
Time to Read	32 msec

Table 4.16 shows the examined parameters when sending the third randomly generated message of 5000 characters as an email using the normal process.

Table 4.16. Parameters of third 5000 character email using the normal process

Original Email Size	49016 bytes
Time to Send	26 msec
Prepared Email Size	49016 bytes
Time to Read	28 msec

Table 4.17 shows the examined parameters when sending the fourth randomly generated message of 5000 characters as an email using the normal process.

Table 4.17. Parameters of fourth 5000 characters email using the normal process

Original Email Size	49088 bytes
Time to Send	35 msec
Prepared Email Size	49088 bytes
Time to Read	38 msec

Summary

The examined results when sending 5000 characters as an email are being averaged out in this section to help show differences in parameters among the tested techniques. Table 4.18 shows the average of values measured for both techniques.

Table 4.18. Average performance for medium message experiment

	Proposed Technique	Normal Process
Original Email Size	49254 bytes	49254 bytes
Time to Send	35.5 msec	32.25 msec
Prepared Email Size	65818 bytes	49254 bytes
Time to Read	58 msec	34 msec

4.3.3 Large Message Experiment

The third experiment was conducted using four different randomly generated messages of size 10000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email.

Proposed Technique

In this section, the results of the proposed method when sending four different messages of size 10000 characters are being examined.

Table 4.19 shows the examined parameters when sending the first randomly generated message of 10000 characters as an email using the proposed technique.

Table 4.19. Parameters of first 10000 character email using the proposed technique

Original Email Size	98216 bytes
Time to Send	23 msec
Prepared Email Size	131072 bytes
Time to Read	48 msec

Table 4.20 shows the examined parameters when sending the second randomly generated message of 10000 characters as an email using the proposed technique.

Table 4.20. Parameters of second 10000 character email using the proposed technique

Original Email Size	98656 bytes
Time to Send	43 msec
Prepared Email Size	131776 bytes
Time to Read	47 msec

Table 4.21 shows the examined parameters when sending the third randomly generated message of 10000 characters as an email using the proposed technique.

Table 4.21. Parameters of third 10000 character email using the proposed technique

Original Email Size	97656 bytes
Time to Send	24 msec
Prepared Email Size	130400 bytes
Time to Read	99 msec

Table 4.22 shows the examined parameters when sending the fourth randomly generated message of 10000 characters as an email using the proposed technique.

Table 4.22. Parameters of fourth 10000 character email using the proposed technique

Original Email Size	97688 bytes
Time to Send	13 msec
Prepared Email Size	107872 bytes
Time to Read	43 msec

Normal Process

In this section, the results of the normal process when sending four different messages of size 10000 characters are being examined.

Table 4.23 shows the examined parameters when sending the first randomly generated message of 10000 characters as an email using the normal process.

Table 4.23. Parameters of first 10000 character email using the normal process

Original Email Size	98216 bytes
Time to Send	26 msec
Prepared Email Size	98216 bytes
Time to Read	38 msec

Table 4.24 shows the examined parameters when sending the second randomly generated message of 10000 characters as an email using the normal process.

Table 4.24. Parameters of second 10000 character email using the normal process

Original Email Size	98656 bytes
Time to Send	38 msec
Prepared Email Size	98656 bytes
Time to Read	22 msec

Table 4.25 shows the examined parameters when sending the third randomly generated message of 10000 characters as an email using the normal process.

Table 4.25. Parameters of third 10000 character email using the normal process

Original Email Size	97656 bytes
Time to Send	17 msec
Prepared Email Size	97656 bytes
Time to Read	60 msec

Table 4.26 shows the examined parameters when sending the fourth randomly generated message of 10000 characters as an email using the normal process.

Table 4.26. Parameters of fourth 10000 characters email using the normal process

Original Email Size	97688 bytes
Time to Send	24 msec
Prepared Email Size	97688 bytes
Time to Read	41 msec

Summary

The examined results when sending 10000 characters as an email are being averaged out in this section to help show differences in parameters among the tested techniques. Table 4.27 shows the average of values measured for both techniques.

Table 4.27. Average performance for large message experiment

	Proposed Technique	Normal Process
Original Email Size	98054 bytes	98054 bytes
Time to Send	25.75 msec	26.25 msec
Prepared Email Size	125280 bytes	98054 bytes
Time to Read	59.25 msec	40 msec

4.3.4 Very Large Message Experiment

The fourth experiment was conducted using four different randomly generated messages of size 20000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email.

Proposed Technique

In this section, the results of the proposed method when sending four different messages of size 20000 characters are being examined.

Table 4.28 shows the examined parameters when sending the first randomly generated message of 20000 characters as an email using the proposed technique.

Table 4.28. Parameters of first 20000 character email using the proposed technique

Original Email Size	195976 bytes
Time to Send	16 msec
Prepared Email Size	261472 bytes
Time to Read	42 msec

Table 4.29 shows the examined parameters when sending the second randomly generated message of 20000 characters as an email using the proposed technique.

Table 4.29. Parameters of second 20000 character email using the proposed technique

Original Email Size	195008 bytes
Time to Send	27 msec
Prepared Email Size	260096 bytes
Time to Read	82 msec

Table 4.30 shows the examined parameters when sending the third randomly generated message of 20000 characters as an email using the proposed technique.

Table 4.30. Parameters of third 20000 character email using the proposed technique

Original Email Size	196136 bytes
Time to Send	17 msec
Prepared Email Size	261824 bytes
Time to Read	51 msec

Table 4.31 shows the examined parameters when sending the fourth randomly generated message of 20000 characters as an email using the proposed technique.

Table 4.31. Parameters of fourth 20000 character email using the proposed technique

Original Email Size	196040 bytes
Time to Send	31 msec
Prepared Email Size	214720 bytes
Time to Read	39 msec

Normal Process

In this section, the results of the normal process when sending four different messages of size 20000 characters are being examined.

Table 4.32 shows the examined parameters when sending the first randomly generated message of 20000 characters as an email using the normal process.

Table 4.32. Parameters of first 20000 character email using the normal process

Original Email Size	195976 bytes
Time to Send	22 msec
Prepared Email Size	195976 bytes
Time to Read	26 msec

Table 4.33 shows the examined parameters when sending the second randomly generated message of 20000 characters as an email using the normal process.

Table 4.33. Parameters of second 20000 character email using the normal process

Original Email Size	194816 bytes
Time to Send	21 msec
Prepared Email Size	194816 bytes
Time to Read	33 msec

Table 4.34 shows the examined parameters when sending the third randomly generated message of 20000 characters as an email using the normal process.

Table 4.34. Parameters of third 20000 character email using the normal process

Original Email Size	196136 bytes
Time to Send	35 msec
Prepared Email Size	196136 bytes
Time to Read	24 msec

Table 4.35 shows the examined parameters when sending the fourth randomly generated message of 20000 characters as an email using the normal process.

Table 4.35. Parameters of fourth 20000 characters email using the normal process

Original Email Size	196040 bytes
Time to Send	20 msec
Prepared Email Size	196040 bytes
Time to Read	19 msec

Summary

The examined results when sending 20000 characters as an email are being averaged out in this section to help show differences in parameters among the tested techniques. Table 4.36 shows the average of values measured for both techniques.

Table 4.36. Average performance for very large message experiment

	Proposed Technique	Normal Process
Original Email Size	195790 bytes	195790 bytes
Time to Send	22.75 msec	24.5 msec
Prepared Email Size	249528 bytes	195790 bytes
Time to Read	53.5 msec	25.5 msec

4.4 Results Analysis

The following charts are used to better represent the results of the experiments, figure (4.1) below shows the time to send emails for each experimental method.
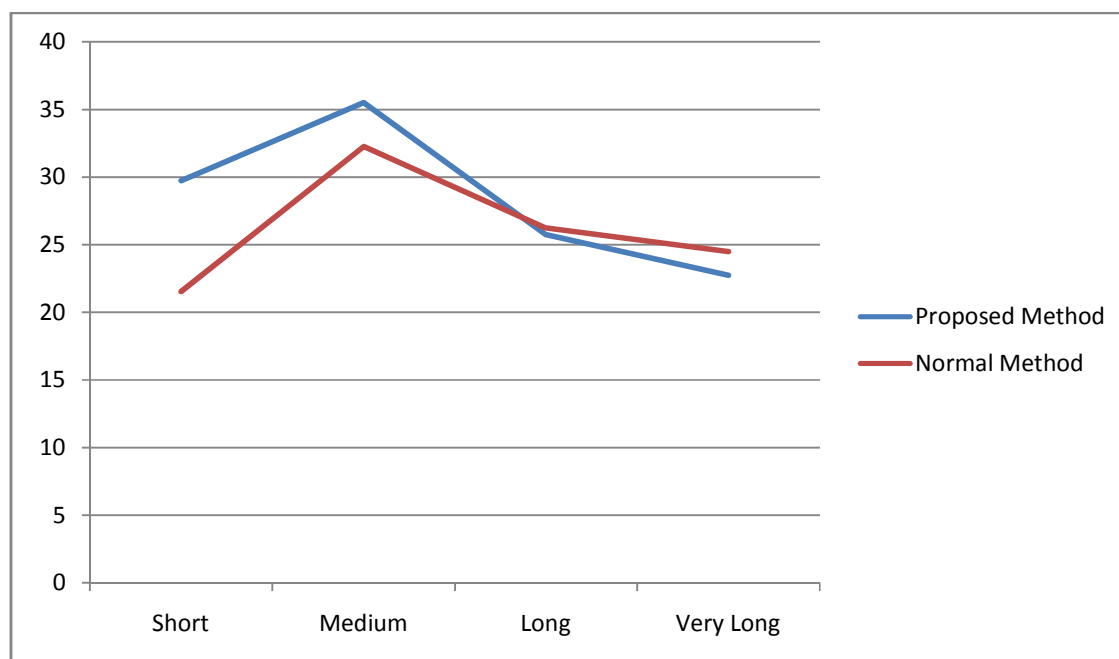
Figure 4.1. Time to Send Chart

The time required to send the email using the normal method was less than the time required for the proposed method, however on very few occasions, there were some irregular spikes in time caused by the traffic variations at different times and low CPU performance due to processes overload on the shared host, which lead to increasing the time required to send an email using the normal method. On average the proposed method required 8.8% more time to send an email than the normal method.

Figure (4.2) below is a chart that shows the size of the prepared email, which is encrypted, when using the proposed method and the normal method in each experimental method.
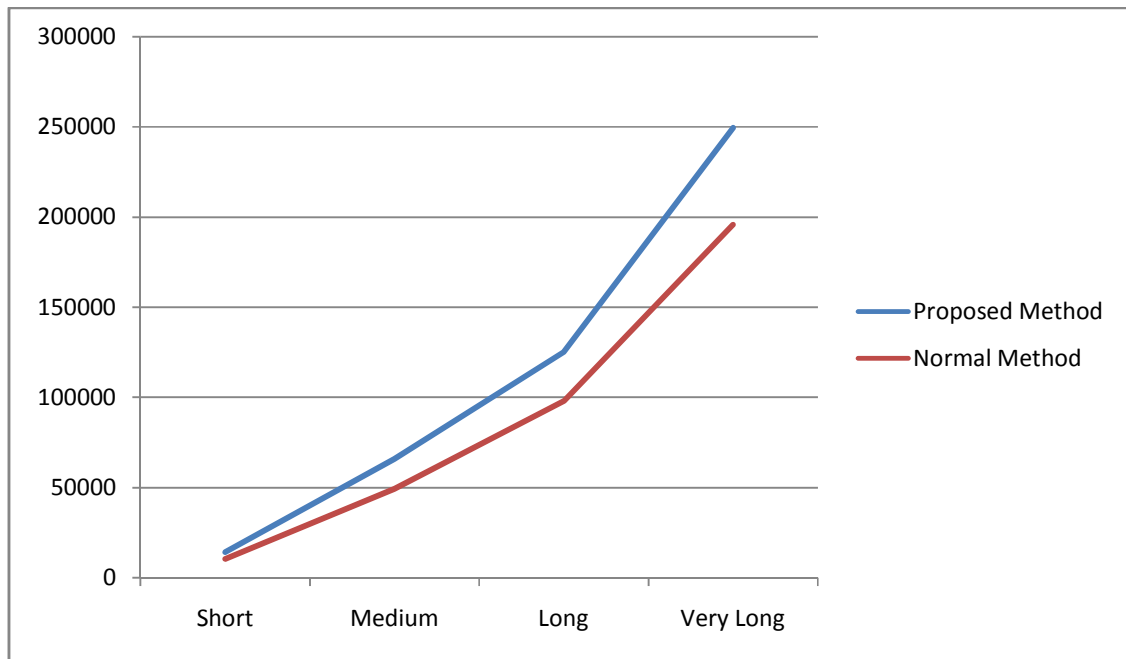
Figure 4.2. Prepared Email Size Chart

The size of the prepared email when using the proposed method was larger than the normal method, the reason behind the increase in size was the extra bits that are injected by AES during the encryption.

Figure (4.3) below shows the chart of the time required to read the email in each experimental method.
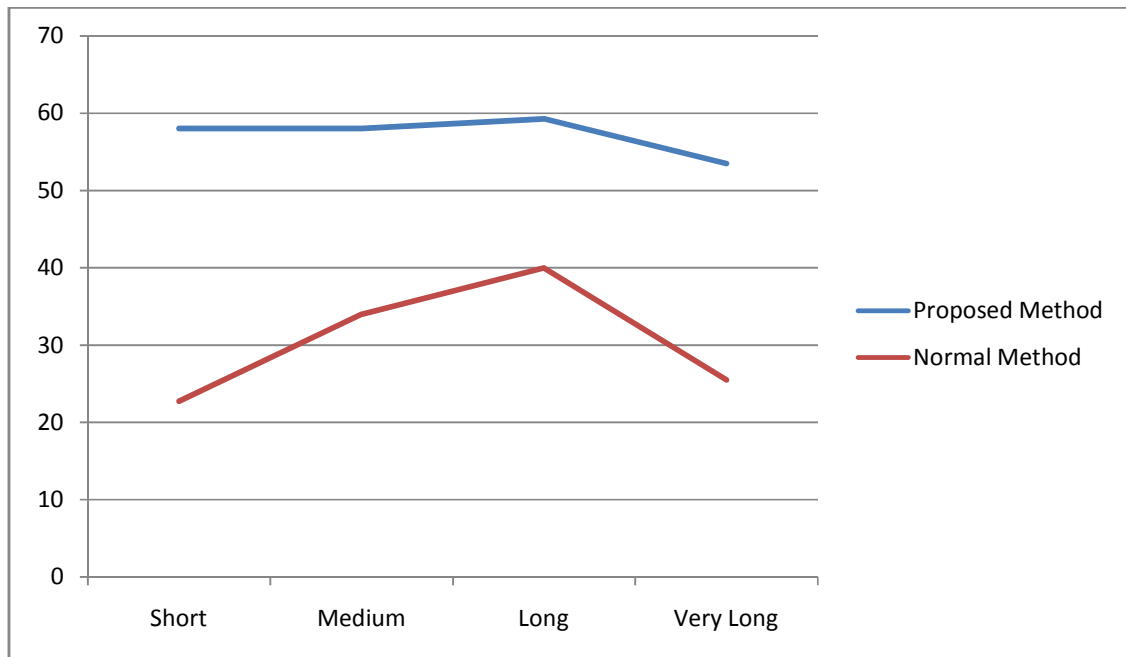
Figure 4.3. Time to Read Chart

The time to read the email using the proposed method was significantly higher than the normal method, the reason behind this increase was the time required for the HTTP request to complete and the key to be received from the sender's server, there were some irregular spikes during the experiments caused by the traffic variations at different times and low CPU performance due to processes overload on the shared host. On average the proposed method required 87.1% more time to complete the task.

4.5 Assumptions

Figures (4.1) and (4.3) have shown the added duration that the proposed method requires for sending and reading emails as opposed to the normal process, but there were other findings that raised some questions and will require further analysis:

1. In figure (4.1), the TTS that was required by the proposed method for messages, whose length was above 10000 characters, was lesser than TTS in the normal process.

2. In figure (4.3), the proposed method required lesser TTR for messages whose length was above 10000 characters as opposed to those whose length was below 10000 characters.

   Assume the following:

   M_1 is a message whose length above 10000 characters

   M_2 is a message whose length is less than 10000 characters.

   Then according to figure (4.1): TTS $_{proposed\ method}$ for M_1 < TTS $_{normal\ process}$ for M_1

   And according to figure (4.3): TTR $_{proposed\ method}$ for M_1 < TTR $_{proposed\ method}$ for M_2

   Are the above assumptions valid?

3. Another question may arise, does the size of the email message affects TTS and TTR in the proposed methodology or not?

4.6 Advanced Analysis

The author needed to conduct more experiments using larger sample size to verify the validity of the statements in the previous page. Did those statements come from results that would happen regularly? Or were they merely coincidental results caused by unfortunate traffic variations and decreased CPU performance that occurred at the time of conducting the previous experiments?

There are four different message lengths that will be used in this extended experimentation:

a. The Extended Small Message Experiment will be conducted using 20 different random texts of size 1000 characters.

b. The Extended Medium Message Experiment will be conducted using 20 different random texts of size 5000 characters.

c. The Extended Large Message Experiment will be conducted using 20 different random texts of size 10000 characters.

d. The Extended Very Large Message Experiment will be conducted using 20 different random texts of size 20000 characters.

The author used IBM Statistical Package for the Social Sciences (SPSS) version 22.0 for measuring T-test, in order to compute the mean and the standard deviation for TTS and TTR in the extended experiments, and compare results of the proposed method against those in the normal process.

4.6.1 Extended Small Message Experiment

This experiment was conducted using 20 different randomly generated messages of size 1000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email. Table 4.37 shows the average of values measured for both techniques.

Table 4.37. Average performance for extended small message experiment

	Proposed Technique	Normal Process
Original Email Size	10490 bytes	10490 bytes
Time to Send	29.30 msec	20 msec
Prepared Email Size	14076 bytes	10490 bytes
Time to Read	57.60 msec	23.60 msec

Table 4.38 below shows the mean, standard deviation and p-value of the proposed method and the normal method in the extended small message experiment.

Table 4.38. Mean, STD, P-Value in extended small message experiment

	Proposed Method Mean ± STD	Normal Process Mean ± STD	p-value
TTS	29.30 ± 7.29	20.00 ± 7.90	0.89
TTR	57.60 ± 11.41	23.60 ± 4.08	0.002

Figure (4.4) and figure (4.5) below show the mean ± standard deviation of TTS and TTR, in extended small message experiment for the proposed method as compared with the normal process, respectively.

Figure 4.4. mean ± STD of TTS in extended small message experiment

Figure 4.5. mean ± STD of TTR in extended small message experiment

From table 4.38 and figure (4.5), there is no significant difference between the proposed method and the normal method in TTS (p = 0. 89).

From table 4.38 and figure (4.6), there is a statistical significant difference between the proposed method and the normal method in TTR (p = 0.002).

4.6.2 Extended Medium Message Experiment

This experiment was conducted using 20 different randomly generated messages of size 5000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email. Table 4.39 shows the average of values measured for both techniques.

Table 4.39. Average performance for extended medium message experiment

	Proposed Technique	Normal Process
Original Email Size	48930 bytes	48930 bytes
Time to Send	36 msec	29.45 msec
Prepared Email Size	65700 bytes	48930 bytes
Time to Read	62.45 msec	31.78 msec

Table 4.40 on the next page shows the mean, standard deviation and p-value of the proposed method and the normal method in the extended medium message experiment.

Table 4.40. Mean, STD, P-Value in extended medium message experiment

	Proposed Method Mean ± STD	Normal Process Mean ± STD	p-value
TTS	36.00 ± 13.34	29.45 ± 7.46	0.36
TTR	62.45 ± 7.64	31.78 ± 5.95	0.00

Figure (4.6) below and figure (4.7) on the page show the mean ± standard deviation of TTS and TTR, in extended medium message experiment for the proposed method as compared with the normal process, respectively.
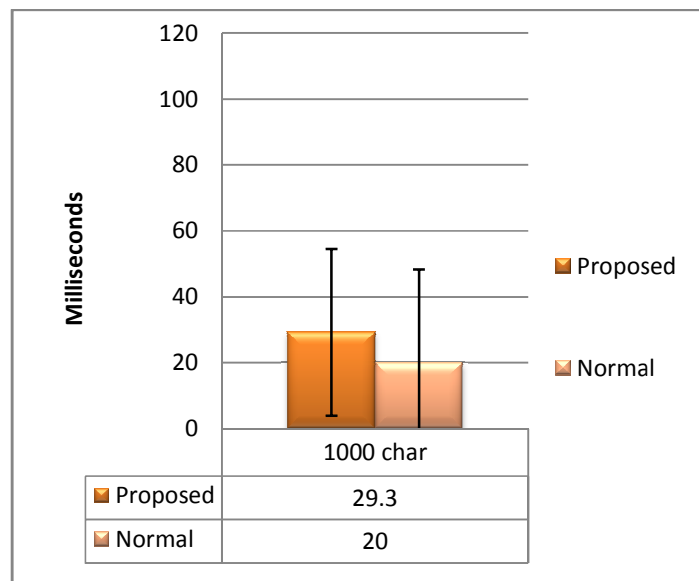

	5000 char
Proposed	36
Normal	29.45

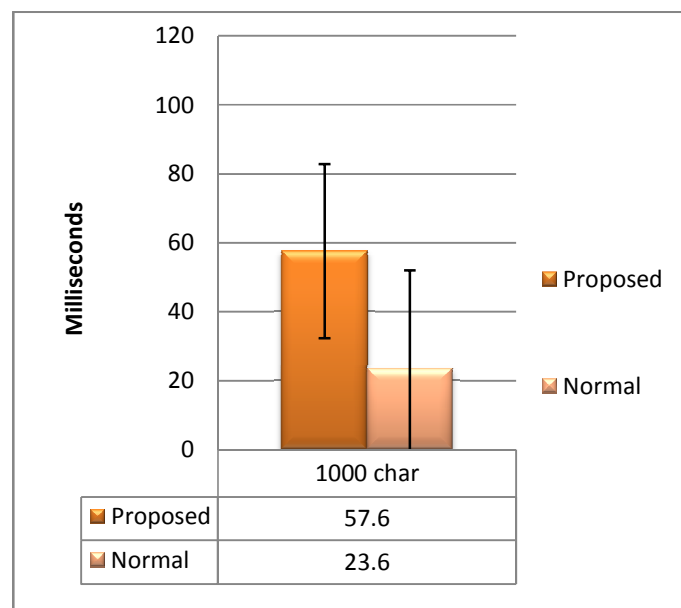Figure 4.6. mean ± STD of TTS in extended medium message experiment

Figure 4.7. mean ± STD of TTR in extended medium message experiment

From table 4.40 and figure (4.6), there is no significant difference between the proposed method and the normal method in TTS (p = 0.36).

From table 4.40 and figure (4.7), there is a statistical significant difference between the proposed method and the normal method in TTR (p = 0).

4.6.3 Extended Large Message Experiment

This experiment was conducted using 20 different randomly generated messages of size 10000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email. Table 4.41 on the next page shows the average of values measured for both techniques.

Table 4.41. Average performance for extended large message experiment

	Proposed Technique	Normal Process
Original Email Size	99004 bytes	99004 bytes
Time to Send	39.50 msec	33.60 msec
Prepared Email Size	125400 bytes	99004 bytes
Time to Read	65 msec	34.60 msec

Table 4.42 below shows the mean, standard deviation and p-value of the proposed method and the normal method in the extended large message experiment.

Table 4.42. Mean, STD, P-Value in extended large message experiment

	Proposed Method Mean ± STD	Normal Process Mean ± STD	p-value
TTS	39.50 ± 9.61	33.60 ± 11.71	0.410
TTR	65.00 ± 12.46	34.60 ± 5.68	0.003

Figure (4.8) and figure (4.9) on the next page show the mean ± standard deviation of TTS and TTR, in extended large message experiment for the proposed method as compared with the normal process, respectively.
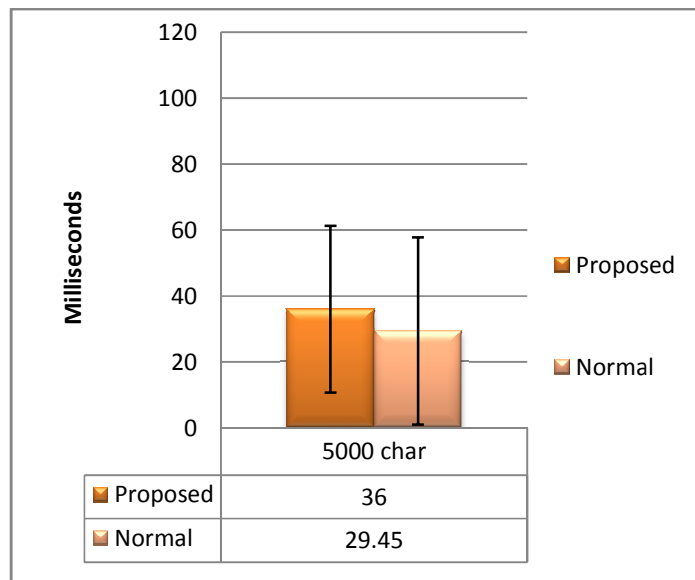
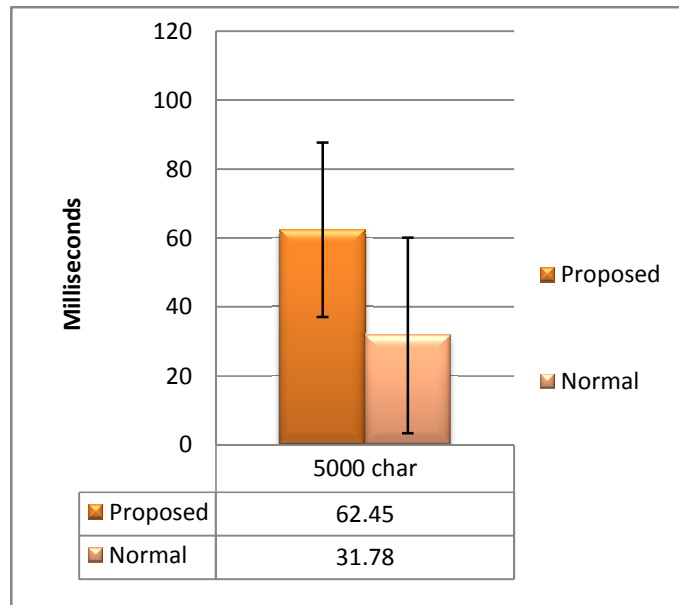Figure 4.8. mean ± STD of TTS in extended large message experiment

Figure 4.9. mean ± STD of TTR in extended large message experiment

From table 4.42 and figure (4.8), there is no significant difference between the proposed method and the normal method in TTS (p = 0.41).

From table 4.42 and figure (4.9), there is a statistical significant difference between the proposed method and the normal method in TTR (p = 0.003).

4.6.4 Extended Very Large Message Experiment

This experiment was conducted using 20 different randomly generated messages of size 20000 characters, the output parameters were calculated automatically for the normal process of sending an email and the proposed technique of sending an email. Table 4.43 shows the average of values measured for both techniques.

Table 4.43. Average performance for extended very large message experiment

	Proposed Technique	Normal Process
Original Email Size	195820 bytes	195820 bytes
Time to Send	43.94 msec	35.44 msec
Prepared Email Size	249601 bytes	195820 bytes
Time to Read	69.04 msec	34.76 msec

Table 4.44 below shows the mean, standard deviation and p-value of the proposed method and the normal method in the extended very large message experiment.

Table 4.44. Mean, STD, P-Value in extended very large message experiment

	Proposed Method Mean ± STD	Normal Process Mean ± STD	p-value
TTS	43.94 ± 5.52	35.44 ± 4.15	0.027
TTR	69.04 ± 7.07	34.76 ± 5.62	0.00

Figure (4.10) below and figure (4.11) on the next page show the mean ± standard deviation of TTS and TTR, in extended very large message experiment for the proposed method as compared with the normal process, respectively.
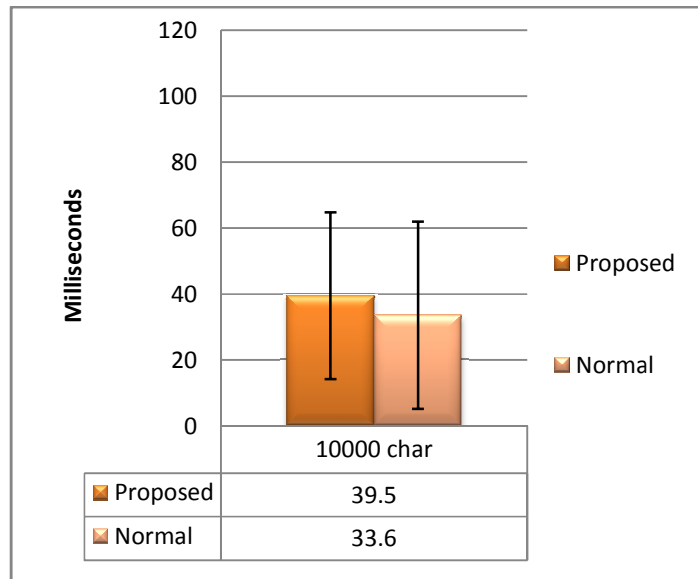

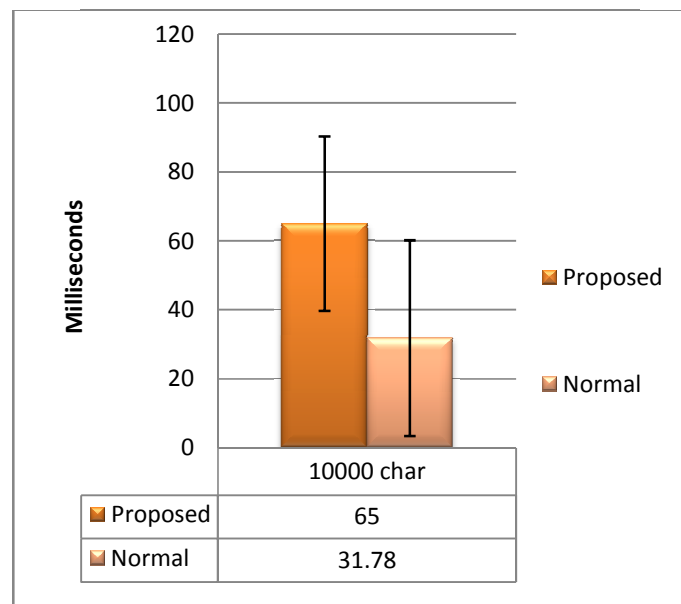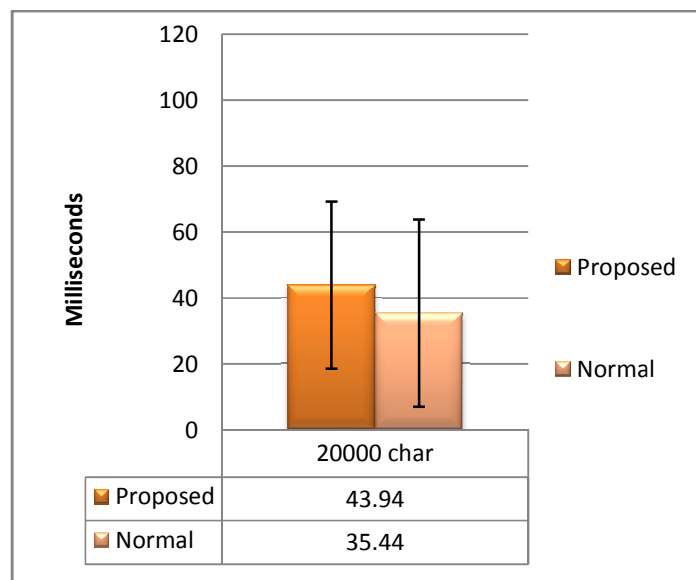Figure 4.10. mean ± STD of TTS in extended very large message experiment
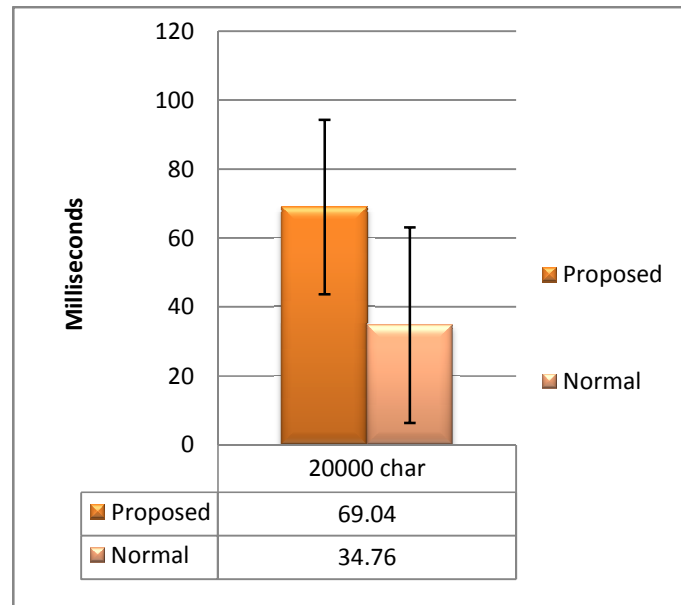
Figure 4.11. mean ± STD of TTR in extended very large message experiment

From table 4.44 and figure (4.10), there is a statistical significant difference between the proposed method and the normal method in TTS (p = 0.027).

From table 4.44 and figure (4.11), there is a statistical significant difference between the proposed method and the normal method in TTR (p = 0).

4.7 Discussion

Based on p-value and mean±STD in tables 4.38, 4.40, 4.42 and 4.40, it is safe to say that the first two assumptions in section 4.5 are not valid:

" $TTS_{\text{proposed method}}$ for M_1 < $TTS_{\text{normal process}}$ for M_1 " is not correct

" $TTR_{\text{proposed method}}$ for M_1 < $TTR_{\text{proposed method}}$ for M_2 " is not correct

To answer the question of the third assumption, the author conducted Pearson correlations using SPSS between TTS and OES, and between TTR and PES, in the proposed technique.

Results have indicated a relationship (Pearson's r = 0.735) between OES and TTS; but there were no statistically significant correlations, which means email size does not affect TTS process at the sender's server.

Results have also indicated a very strong relationship (Pearson's r = 0.967) and a statistically significant correlation between TTR and PES. It means email size has an effect on TTR process at the receiver's server. The larger the email size, the more time the receiver's server requires reading it.

Chapter Five

Conclusions

Chapter 5 Conclusions

5.1 Overview

In this thesis a new technique is presented to provide control over sent emails, the study controlled the availability of email messages by encrypting the body of the message using a randomly generated key, that is created using V4 UUID generation algorithm, before sending it to the receiver. The proposed technique allows the receiver to request the encryption key using data included in a customized email header field that refers to the location of an API on the sender's server, the key is sent to the receiver when requested only if it's set to available.

5.2 Conclusion

In conclusion, the proposed technique was successful in achieving the goals it aimed to deliver. The proposed technique was compared against the standard methodology of email processing, and as expected, there have been delays in TTS and TTR when the proposed technique was applied. On average the receiver's server required 87.1% more time, to read emails, in the proposed method, this is due to the added processes of requesting the encryption key and the decryption of the message. It also took the sender's server 8.8% more time to send emails. There were also noticeable changes in the size of an email after the encryption process, the changes in size varied from one experiment to another. The first 16 experimentations assumed that in the proposed technique, TTS for messages above 10000 characters length required less time than TTS in the normal process, and TTR for messages above 10000 characters length required less time than TTR for messages below

10000 characters length. Those assumptions were based on experiments conducted to compare the proposed technique against the standard technique of email processing using four variant email sizes. 80 more experiments were conducted later in order to verify the validity of the assumptions of the first readings. The mean, standard deviation and the p-value of TTS and TTR for the proposed and the normal method, in those 80 experiments, were calculated and compared. Extended experimentations have shown that the assumptions of the first 16 experiments were incorrect, as some of the readings were affected by traffic variations and decreased CPU performance that occurred at the time of conducting the experiments, which had led to inaccurate readings due to the small sample size. The extended experiments prove, using Pearson coefficients, that there is a relationship between the size of emails and their sending/reading time. Results have also shown that there are statistically significant correlations between TTR and PES.

5.3 Future Work

In this thesis a method is proposed to provide more control over sent emails, this method can be the base for future studies to emerge from, the following list presents some ideas for further studies:

1- Try different encryption techniques in hope to reduce the time overhead.

2- Implement RSA key exchange mechanism to provide robustness against key hijack during the key exchange process.

3- Implement the proposed method on MIME data types and measure its performance.

4- Measure the performance of the proposed method on a dedicated server environment using larger sample size.

References

- Abadi, M., & Blanchet, B. (2005). Computer-Assisted Verification of a Protocol for Certified Email. *Science of Computer Programming*, V58(1), (PP 3-27).

- Abadi, M., Glew, N., Horne, B., & Pinkas, B. (2002). Certified email with a light on-line trusted third party: design and implementation. *In 11th international World Wide Web Conference (WWW'02)*, (PP387-395).

- Al Bazar, H., Sureswaran, R., Abouabdalla, O. (2008). A new approach to enhance e-mail performance through pop3 protocol. *International Conference on Network Applications, Protocols and Services 2008 (Net Apps 2008),* (PP 929-932).

- Banday, M. T. (A) (2011). Effectiveness And Limitations of email Security Protocols. *International Journal of Distributed and Parallel Systems (IJDPS)*, V2(3), (PP 38-49).

- Banday, M. T. (B) (2011). On the Authentication of Date in E-mail using Trusted Time Stamping Service. *IJCA Special Issue on "Network Security and Cryptography" NSC,* (PP 36-42).

- Bulloch, S. (2007). IBM Customer Support, Retrieved from IBM Customer Support Website, URL: https://www.ibm.com/developerworks/lotus/library/notes8-recall/. Last Review Date August, 07, 2007. Accessed Date May 10, 2014.

- Crispin, M. (2003). Internet Message Access Protocol- version4 rev1, RFC 3501.

- Crocker, D. (2009). Internet Mail Architecture. RFC 5589.

- Culotta, A., Bekkerman, R., & McCallum, A. (2004). Extracting Social Networks and Contact Information from Email and the Web. *First Conference on Email and Anti-Spam (CEAS).*

- Dye, M., McDonald, R., & Rufi, A. (2007). Network Fundamentals*, CCNA Exploration Companion Guide*. Cisco press, Chapter 3, (PP 84 - 85).

- Hafsaoui, A., Urvoy-Keller, G., Collange, D., & Siekkinen, M. (2010). A Study of Email Usage and Performance over Cellular Technology. *Communications and Networking (ComNet), 2010 Second International Conference on*, (PP 1-8).

- Haigh, T. (2012). Seven Lessons From Bad History. *Communications of the ACM*, V55(9), (PP 26- 29).

- Harn, L., & Ren, J. (2008). Design of Fully Deniable Authentication Service for E-mail Applications. *IEEE Communications letters,* V12(3), (PP 219-221).

- Heron, S. (2009). Advanced Encryption Standard (AES). *Network Security*, V 2009(12), (PP 8-12).

- Klensin, J. (2008). Simple Mail Transfer Protocol. RFC 5321.

- Lux, K. D., J May, M., Bhattad, N. L., & Gunter, C. A. (2005). WSEmail: Secure Internet Messaging Based on Web Services. *Proceedings of the IEEE International Conference on Web Services* (PP 75-82).

- Mantoro, T., & Zakariya, A. (2012). Securing E-Mail Communication Using Hybrid Cryptosystem on Android-based Mobile Devices. *Telkomnika*, V10(4), (PP 827- 834).

- Microsoft Customer Support (2012). Retrieved from the Microsoft Customer Support Website, URL: http://support.microsoft.com/kb/197094. Article ID 197094. Last Review Date October, 4, 2012. Accessed Date May 10, 2014.

- Myers, J., & Rose, M. (1996). Post Office Protocol-version 3. STD 53, RFC 1939.

- Partridge, C. (2008). The Technical Development of Internet Email. *IEEE Annals of the History of Computing,* V1(2), (PP 3-29).

- Praveen, B. (2013). The Undo Sent E-mail (USE) Protocol. *International Journal of Information and Computation Technology*, *International Research Publications House*, ISSN 0974-2239, V3(11), (PP 1187-1196).

- Radicati, S. (2014). Email Statistics Report 2014-2018. The Radicati Group Inc, URL: http://www.radicati.com/wp/wp-content/uploads/2014/01/Email-Statistics-Report-2014-2018-Executive-Summary.pdf.

- Resnick, P. (2008). Internet Message Format. RFC 5322.

- Selent, D. (2010). Advanced Encryption Standard. *Rivier Academic Journal*, V6(2), (PP 1-14).

- Shao, J., Feng, M., Zhu, B., & Cao, Z. (2007). An efficient certified email protocol. *Information Security, Springer Berlin Heidelberg,* (pp. 145-157).

- Sobh, T. S., & Amer, M. I. (2011). PGP Modification for Securing Digital Envelope Mail Using COM+ and Web Services. *IJ Network Security*, V13(2), 79-91.

- Suresh, K. B., & Jagathy, R. V. P. (2012). A Secure Email System Based on IBE, DNS and Proxy Service. *Journal of Emerging Trends in Computing and Information Sciences,* V3(9), (PP 1271-1276).

- Tanta-ngai, H., Abou-Assaleh, T., Jiampojamarn, S., & Cercone, N. (2003). Secure Mail Transfer Protocol (SecMTP). *In Proceedings of the International Conference on Advances in the Internet, Processing, Systems, and Interdisciplinary Research, IPSI-2003,* Sveti Stefan, Montenegro, Former Yugoslavia.

Appendices

Appendix A: Email System Modules' Source Code Written in PHP

UUID_generator.php

```php
<?php
function generate_UUID() {
  return sprintf('%04x%04x-%04x-%04x-%04x-%04x%04x%04x',

    // 32 bits for "time_low"
    mt_rand(0, 0xffff), mt_rand(0, 0xffff),

    // 16 bits for "time_mid"
    mt_rand(0, 0xffff),

    // 16 bits for "time_hi_and_version",
    // four most significant bits holds version number 4
    mt_rand(0, 0x0fff) | 0x4000,

    // 16 bits, 8 bits for "clk_seq_hi_res",
    // 8 bits for "clk_seq_low",
    // two most significant bits holds zero and one for variant DCE1.1
    mt_rand(0, 0x3fff) | 0x8000,

    // 48 bits for "node"
    mt_rand(0, 0xffff), mt_rand(0, 0xffff), mt_rand(0, 0xffff)
  );
}

function generate_identifier() {
  return uniqid();
}
?>
```

db_connect.php

```php
<?php
/**
 * A class file to connect to database
 */
class DB_CONNECT {

    // constructor
    function __construct() {
        // connecting to database
        $this->connect();
    }

    // destructor
    function __destruct() {
        // closing db connection
        $this->close();
    }

    /**
     * Function to connect with database
     */
    function connect() {
        // import database connection variables
        require_once __DIR__ . '/db_config.php';

        // Connecting to mysql database
        $con = mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD) or die(mysql_error());

        mysql_set_charset('utf8',$con);

        // Selecing database
        $db = mysql_select_db(DB_DATABASE) or die(mysql_error()) or die(mysql_error());

        // returing connection cursor
        return $con;
    }

    /**
     * Function to close db connection
     */
    function close() {
```

```php
    // closing db connection
    //mysql_close();
  }

}
?>
```

db_config.php

```php
<?php
/*
 * All database connection variables
 */
define('DB_SERVER', "50.31.147.17"); // db server
define('DB_DATABASE', "semiapps_webm410"); // database name
define('DB_USER', "semiapps_webm410"); // db user
define('DB_PASSWORD', "N!r0k80!PS"); // db password (mention your db password
here)
?>
```

api.php

```php
<?php
require_once __DIR__.'/db_connect.php';

$db = new DB_CONNECT();

$identifier = $_GET['identifier'];
$from = $_GET['from'];
$to = $_GET['to'];


$query = "SELECT uuid_ FROM secret_keys where identifier_ = '$identifier' AND
from_ = '$from' AND to_ = '$to'";

$result = mysql_query($query);

if($result) {
  $row = mysql_fetch_array($result);
```

```php
    echo $row['uuid_'];
}
?>
```

control.php

```php
<?php
require_once __DIR__.'/db_connect.php';

$db = new DB_CONNECT();

$from_ = $_GET['from'];

$result = mysql_query("SELECT * FROM secret_keys WHERE from_ = '$from_'");

echo "<!DOCTYPE html>
<html>
<body>";

if($result) {
    echo '<table>';
    while($row = mysql_fetch_array($result)) {
        echo '<tr>';

        echo '<td>';
        echo $row['to_'];
        echo '</td>';


        echo '<td>';
        echo "<a
href=http://www.wardmail.info/plugin/delete.php?id=".$row['_id']."&from=$from_>
Delete</a>";
        echo '</td>';

        echo '</tr>';

    }
    echo '</table>';
    echo "</body></html>";
}
?>
```

cryptors.php

```php
<?php
function encrypt($sValue, $sSecretKey)
{
return $sValue;
/*return rtrim(
    base64_encode(
      mcrypt_encrypt(
        MCRYPT_RIJNDAEL_256,
        $sSecretKey, $sValue,
        MCRYPT_MODE_ECB,
        mcrypt_create_iv(
          mcrypt_get_iv_size(
            MCRYPT_RIJNDAEL_256,
            MCRYPT_MODE_ECB
          ),
          MCRYPT_RAND)
        )
      ), "\0"
    );*/
}

function decrypt($sValue, $sSecretKey)
{
return $sValue;
/*return rtrim(
    mcrypt_decrypt(
      MCRYPT_RIJNDAEL_256,
      $sSecretKey,
      base64_decode($sValue),
      MCRYPT_MODE_ECB,
      mcrypt_create_iv(
        mcrypt_get_iv_size(
          MCRYPT_RIJNDAEL_256,
          MCRYPT_MODE_ECB
        ),
        MCRYPT_RAND
      )
    ), "\0"
  );*/
}

?>
```

receiver.php
```php
<?php

require_once __DIR__.'/cryptors.php';
require_once __DIR__.'/db_connect.php';

function decrypt_email($message, $api, $identifier, $from, $to) {


    $_key = fetch_content($api."?identifier=".$identifier."&from=".$from."&to=".$to);

    return decrypt($message, $_key);
}


function fetch_content($url) {

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_AUTOREFERER, TRUE);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_URL, $url);

    $data = curl_exec($ch);
    curl_close($ch);

    return $data;
}

function reading_time($start_time, $finish_time, $identifier) {
    $db = new DB_CONNECT();

    $query = "UPDATE results set r_start_time = '$start_time', r_end_time = '$finish_time' WHERE r_start_time is NULL";

    mysql_query($query);
}
function writing_time($start_time, $finish_time, $identifier, $subject, $size, $originalSize) {
    $db = new DB_CONNECT();
    $date_time = date("F j, Y, g:i a");
    $query = "INSERT INTO results(identifier, w_start_time, w_end_time, email_date, subject, text_size, original_text_size, consistency) VALUES('$identifier','$start_time', '$finish_time', '$date_time', '$subject', '$size', '$originalSize', '%100')";
```

```php
    mysql_query($query);
}
?>
```

sender.php

```php
<?php
require_once __DIR__.'/UUID_generator.php';
require_once __DIR__.'/cryptors.php';
require_once __DIR__.'/db_connect.php';
require_once __DIR__.'/receiver.php';


function sendMail($sTo, $sSubject, $sText, $sFrom) {
    $start_time = msec();

  $identifier = generate_identifier();
  $uuid = generate_UUID();


  $db = new DB_CONNECT();
  $result = mysql_query("INSERT INTO secret_keys(from_, to_, identifier_, uuid_)
VALUES('$sFrom', '$sTo', '$identifier', '$uuid')");

  $originalSize = strlen($sText);

  $sText = encrypt($sText, $uuid);

  if($result) {

    $headers  = 'MIME-Version: 1.0' . "\r\n";
    $headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";

    // Additional headers
    $headers .= 'To: '.$sTo . "\r\n";
    $headers .= 'From: '.$sFrom. "\r\n";

    $headers .= 'X-API: http://www.wardmail.info/plugin/api.php' . "\r\n";
    $headers .= 'X-Identifier: '.$identifier. "\r\n";

    $stop_time = msec();
```

```php
            writing_time($start_time, $stop_time, $identifier, $sSubject, strlen($sText)*8,
$originalSize*8);

    // Mail it
    mail($sTo, $sSubject, $sText, $headers);


    }
}

function msec()
{
    list($usec, $sec) = explode(' ',microtime());
    return intval(($usec+$sec)*1000.0);
}

?>



results.php
<?php
require_once __DIR__.'/db_connect.php';

$db = new DB_CONNECT();
$result = mysql_query("SELECT * FROM results");

if($result) {

    echo '<table>';

    echo '<th>ID</th> <th>Subject</th> <th>Date</th> <th>Net (W)</th> <th>Net
(R)</th> <th>Original text size (Bytes)</th> <th>Text AFTER (Bytes)</th>
<th>Consistency</th>';
    while($row = mysql_fetch_array($result)) {
        echo '<tr>';

        echo '<td style=\'border:solid 1px;\'>';
        echo $row['_id'];
        echo '</td>';

        echo '<td style=\'border:solid 1px;\'>';
        echo $row['subject'];
        echo '</td>';
```

```php
    echo '<td style=\'border:solid 1px;\'>';
    echo $row['email_date'];
    echo '</td>';

    echo '<td style=\'border:solid 1px;\'>';
    echo $row['w_end_time'] - $row['w_start_time'];
    echo '</td>';


    echo '<td style=\'border:solid 1px;\'>';
    echo $row['r_end_time'] - $row['r_start_time'];
    echo '</td>';

    echo '<td style=\'border:solid 1px;\'>';
    echo $row['original_text_size'];
    echo '</td>';


    echo '<td style=\'border:solid 1px;\'>';
    echo $row['text_size'];
    echo '</td>';


    echo '<td style=\'border:solid 1px;\'>';
    echo $row['consistency'];
    echo '</td>';


    echo '</tr>';

  }
  echo '</table>';
}
?>
```

Appendix B: Random Text Generation Source Code Written in C#

```csharp
private void random_text_Click(object sender, EventArgs e)
{
    Random random = new Random();

    int num;
    char let;
    int requested = int.Parse(text_size_to_generate.Text);

    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < requested; i++)
    {
        num = random.Next(0, 26); // Zero to 25
        if (num % 6 == 0)
        {
            sb.Append(' ');
        }
        else
        {
            sb.Append((char)('a' + num));
        }
    }

    secret_text_box.Text = sb.ToString();
}
```

Appendix 3: Publication

International Journal of Computer Technology and Applications (IJCTA), Jan 2015, ISSN: 2229-6093, Volume 6 Issue 1, Pages: 18 - 22

- Author Instructions
- Copyrights Form
- Model Paper

News and Updates

Impact factor: 2.804 NEW

ISO 3297: 2007

ISO International Organization for Standardization

4 **New Technique for Cross Platform Availability Control over Electronic Messages**
-Ward Ahmed,Dr Hebah H. O. Nasereddin
Abstract

In this research, a new technique is proposed to grant email senders control over sent messages even after they arrive at the receiver's mailbox, the method suggests that the email gets encrypted by AES using a randomly generated key, the key is then stored on the sender server and an identifier is added to the header of the email, if the receiver wishes to read the email the receiver's server must request the key from the senders server by sending the identifier, the sender can control the availability of the key and thus the ability of the receiver to read the email. The experimental results suggest that the process of encryption has little impact on the overall time required to send the email, however, the process of requesting the key and decrypting he email required slightly more time than the normal process

18-22 PDF

5 **Saliency of Scene Object**
-Ashwini Nanaware,Harish Barapatre
Abstract

23-26 PDF

6 **Threshold Based Face Detection**
-R.Vinodini,Dr.M.Karnan
Abstract

27-31 PDF

7 **Performance Evolution of Bee Colony Optimization for Fuzzy Clustering**
-R. Prabhu
Abstract

32-34 PDF