APPLICATION OF CRYPTOGRAPHY IN DATA SECURITY (E-MAIL SECURITY AS A TEST CASE)

BY

OZUMBA ADORA CHINELO
PGD/MCS/05/06/1188

DEPARTMENT OF MATHEMATICS/COMPUTER SCIENCE
FEDERAL UNIVERSITY OF TECHNOLOGY
MINNA, NIGER STATE

FEBRUARY, 2008

APPLICATION OF CRYPTOGRAPHY IN DATA SECURITY (E-MAIL SECURITY AS A TEST CASE)

BY

OZUMBA ADORA CHINELO
PGD/MCS/05/06/1188

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS/COMPUTER SCIENCE IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A POST-GRADUATE DIPLOMA IN COMPUTER SCIENCE, FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGER, STATE.

FEBRUARY, 2008

CERTIFICATION

This project titled "Application of Cryptography in Data Security (E-mail security as a test case)",by Ozumba Adora Chinelo,meets the regulations governing the award of Post-graduate Diploma in Computer Science of Federal University of Technology, Minna, Niger State.

____________________	________________
Prince R. O. Badmus	Date
Supervisor	
____________________	________________
Dr. N. I. Akinwande	Date
Head of Department	

DEDICATION

This project is dedicated to my lord Jesus Christ and my family for their love and support in all my endeavours.

ACKNOWLEDGEMENT

My sincere gratitude goes to the Almighty God who has called me specially and also given me the grace to go through this programme in line with His divine will and purpose for my life. My special thanks go to my project supervisor, Prince R.O. Badmus for his concern and guidance, Dr.N.I.Akinwande (the Head of Maths/Computer Science Department) and Mr. Jiya Mohammed for their support. I also acknowledge the rest of the members of staff of Maths/Computer Science Department whose impartation of knowledge helped me in carrying out this project; and my classmates for their team spirit.

Finally, my appreciation goes to Okel Consulting, Abuja, for their assistance.

ABSTRACT

Over the centuries, security of information transmission has been of paramount importance such that security measures on information were used by leaders like Julius Caesar (with his generals) and also during world war II. With increasing sophistication in technology, digital communication, electronic data exchange, the use of computers and crime rate, information security has become a crucial issue in industry, business and administration. This project attempts to look in to the issue of cryptography providing essential techniques for securing information and protecting data.

TABLE OF CONTENTS

LIST OF FIGURES PAGE

CHAPTER ONE

GENERAL INTRODUCTION

1.1 INTRODUCTION

Cryptography is the science of writing in secret code and is an ancient art. The first documented use of cryptography in writing dates back to around 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is no surprise, then, that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about *any* network, particularly the Internet.

Within the context of any application-to-application communication, there are some specific security requirements, including:

- *Authentication:* The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak).
- *Privacy/confidentiality:* Ensuring that no one can read the message except the intended receiver.

- *Integrity:* Assuring the receiver that the received message has not been altered in any way from the original.
- *Non-repudiation:* A mechanism to prove that the sender really sent the message.

Cryptography, then, not only protects data can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or from theft or alteration, but asymmetric) cryptography, and hash functions. In all cases, the initial unencrypted data is referred to as *plaintext*. It is encrypted into *ciphertext*, which will in turn (usually) be decrypted into usable plaintext.

This project is aimed at implementing a cryptographic algorithm – the RSA Algorithm, for securing e-mail.

1.2 OBJECTIVES OF THE STUDY

1. To emphasize on some salient areas of cryptography
2. To show the role of electronic encryption in E-mail Security; and
3. To implement the RSA algorithm in order to secure E-mail.

1.3 SCOPE OF STUDY

This study takes an overview of cryptography or exposition on it, in a simplified form. Also the study is largely on encryption, E-mail, security risks with E-mail and the workings of electronic encryption. On the whole the study centers around securing the internet e-mail.*1.4 METHOD OF DATA COLLECTION*

The method of data gathering for this study was through interviews with organizations with bias on this study and from relevant books and the internet.

1.5 DEFINITION OF TERMS

Supposing some one wants to send a letter to another person, and wants to be sure that no one else can read the message, there is the possibility that some one else may open the letter. In cryptographic terminology, the message contained in the letter is called **PLAINTEXT**. Encoding the contents of the letter in such a way that it is hidden from outsiders is called **ENCRYPTION**. The encrypted message is called **CIPHERTEXT**. The process of retrieving the plain text from cipher text is called **DECRYPTION**.

- **CRYPTANALYSIS**: it is the art of breaking ciphers
- **CRYPTOGRAPHERS**: people who do cryptography
- **CRYPTOANALYSTS**: practitioners of cryptanalysis.
- **CRYPTOLOGY**: the branch of mathematics that studies the mathematical foundations of cryptographic methods.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

Cryptography was derived from the Greek word 'krypto' that is 'hidden' and 'grafo' that is 'to write' or 'to speak' and means 'the study of hiding information' Liddell et al(1978).It is the process or skill of communicating in or deciphering or analyzing secret codes, writings, ciphers and cryptograms. In it one writes in codes or ciphers. It involves encoding(encryption) which is the activity of converting data or information into code; recoding which is converting from one code to another; decipherment(decoding, decryption)-the activity of making clear or converting from code into plain text . A secret key or password is required for decryption. Encryption attempts to secure secrecy in communications such as those of spies, military leaders, diplomats and religious applications. Examples abound of the use of cryptography in earlier centuries. Notable among them are :(i)The use of encryption by Julius Caesar to communicate with his generals;(ii)The use of the German Lorenz cipher machine for the encryption of very high level general staff messages ,during world war II.

Leon (1497) invented the poly-alphabetic cipher. His innovation was to use different ciphers (i.e. substitution alphabets) for various parts of a message.

The enigma machine used in several variants by the German military between the late 1920's and the end of World War II, implemented a complex electromechanical poly-alphabetic cipher to protect sensitive communications. Breaking the Enigma cipher at the Biuro Szyfrow and the subsequent large scale decryption of Enigma traffic at Bletchley Park, was an important factor contributing to the Allied victory in world war II.Babbage(1857) showed that poly-alphabetic ciphers of this type remained partially vulnerable to frequency analysis techniques. Bribery, espionage and burglary were used to discover a ciphers algorithm. Such ciphers could be broken. Secrecy of the key is therefore required for a good cipher to maintain confidentiality under attack.

Before the modern era encryption was solely concerned with confidentiality – i.e. conversion of messages from a comprehensible form to an incomprehensible one and back again at the other end, rendering it unreadable by interceptors or eavesdroppers without secret knowledge of the key needed for the decryption of that message.

Essentially prior to the 20th century, cryptography was chiefly concerned with linguistic patterns , Diffie(1975).In modern times, cryptography is considered to be a branch of Mathematics and Computer Science and affiliated closely with Information theory, Computer Security and Engineering. Cryptography is used in applications present in technologically advanced countries in some areas like the security of ATM cards, computer passwords and electronic commerce. The field has expanded beyond confidentiality concerns to include techniques for message integrity checking, sender-receiver identity authentication, digital signatures, and interactive proofs and secure computation amongst others.

In earlier times it required transposition ciphers ,for example 'help me' becomes 'ehpl em 'in a trivially simple re-arrangement scheme and substitution ciphers which systematically replace letters or groups of letters ,for example, 'fly at once' becomes 'gmz bu podf' by replacing each letter with the one following in the alphabet. The development of digital computers and electronics after world war II made possible much more complex ciphers. Also, computers allowed for the encryption of any kind of data that is represented by computers in any binary format, unlike classical ciphers which only encrypted written language texts, dissolving the utility of a linguistic approach to cryptanalysis in many cases.

Since the mid 1970's, with the public specification of Data Encryption Standard(DES) in the United States, the Diffie-Hellman paper and the public release of RSA algorithm, cryptography has become a widely used tool in communications, computer network and computer security generally.

2.2 CRYPTOGRAPHY OVERVIEW

Modern cryptography addresses a wide range of problems. But the most basic problem remains the classical one of ensuring security of communication across an insecure medium. To describe it, the first two members of a cast of characters is hereby introduced: the sender, *S*, and the receiver, *R*. The sender and receiver want to communicate with each other. Imagine our two parties are provided with a dedicated, untappable, impenetrable pipe or tube into which the sender can whisper a message and the receiver will hear it. Nobody else can look inside the pipe or change what's there. This pipe provides the perfect medium, available only to the sender and receiver, as though they were alone in the world. It is an "ideal" communication channel from the security point of view. Unfortunately, in real life, there are no ideal channels connecting the pairs of parties that might like to communicate with each other. Usually such parties are communicating over some public network like the Internet.

The most basic goal of cryptography is to provide such parties with a means to imbue their communications with security properties akin to those provided by the ideal channel. At this point let the third member of the cast be introduced. This is the *adversary*, denoted *A*. An adversary models the source of all possible threats. Imagine the adversary as having access to the network and wanting to compromise the security of the party's communications in some way.

Not all aspects of an ideal channel can be emulated. Instead, cryptographers distill a few central security goals and try to achieve them. The first such goal is *privacy*. Providing privacy means hiding the content of a transmission from the adversary. The second goal is *authenticity* or *integrity*. We want the receiver, upon receiving a communication pertaining to be from the sender, to have a way of assuring it that it really did originate with the sender, and was not sent by the adversary, or modified en route from the sender to the receiver.

In order to achieve security goals such as privacy or authenticity, cryptography supplies the sender and receiver with a *protocol(* here, just a collection of programs) one for each party involved. In this case, there would be some program for the sender to run, and another for the receiver to run. The sender's program says how to package or encapsulate the data for transmission. The receiver's program says how to decapsulate the received package to recover the data, together possibly with associated information saying whether or not it should be regarded as authentic. Both programs are a function of some *cryptographic keys.*

2.2.1 TRUST MODEL

It is not hard to be convinced that in order to communicate securely, there must be something that a party knows, or can do, that the adversary does not know, or cannot do. There has to be some "asymmetry" between the situation in which the parties find themselves and the situation in which the adversary finds itself. The *trust model* specifies who, initially, has what keys. There are two central trust models: the symmetric (or shared-key) trust model and the asymmetric (or public-key) trust model. These models and the cryptographic problems they give rise to, will be looked at below.

2.2.2 THE SYMMETRIC SETTING

In practice, the simplest and also most common setting is that the sender and receiver share a *key* that the adversary does not know. This is called the *symmetric setting* or symmetric trust model. The encapsulation and decapsulation procedures above would both depend on this same shared key. The shared key is usually a uniformly distributed random string having some number of bits, k. Recall that a *string* is just a sequence of bits. The sender and receiver must somehow use the key K to overcome the presence of the adversary.

One might ask how the symmetric setting is realized. Meaning, how does a sender and receiver initially come into possession of a key unknown to the adversary? The symmetric model is not concerned with how the parties got the key, but with how to use it. In cryptography it is assumed that the secret key is kept securely by the party using it. If it is kept on a computer, it is assumed that the adversary cannot penetrate these machines and recover the key. Ensuring that this assumption is true is the domain of computer systems security.

It is time to take a closer look at some specific problems in the symmetric setting.

2.2.3 SYMMETRIC ENCRYPTION SCHEMES

A protocol used to provide privacy in the symmetric setting is called a *symmetric encryption scheme*. When such a scheme is specified as Π, three algorithms should also be specified, so that the scheme is a triple of algorithms, Π = (*K, E, D*). The encapsulation algorithm discussed above is, in this context, called an *encryption* algorithm, and is the algorithm *E*. The message *M* that the sender wishes to transmit is usually referred to as a *plaintext*. The sender *encrypts* the plaintext under the shared key *K* by applying *E* to *K* and *M* to obtain a *ciphertext C*. The ciphertext is transmitted to the receiver. The above-mentioned decapsulation procedure, in this context, is called a *decryption* algorithm, and is the algorithm *D*. The receiver applies *D* to *K* and *C*. The decryption process might be unsuccessful, indicated by its returning a special symbol □, but, if successful, it ought to return the message that was originally encrypted. The first algorithm in Π is the *key generation* algorithm which specifies the manner in which the key is to be chosen. In most cases this algorithm simply returns a random string of length, the key length.

The encryption algorithm *E* may be randomized, or it might keep some state around. A picture for symmetric encryption can be found in Figure 2.1 overieaf.

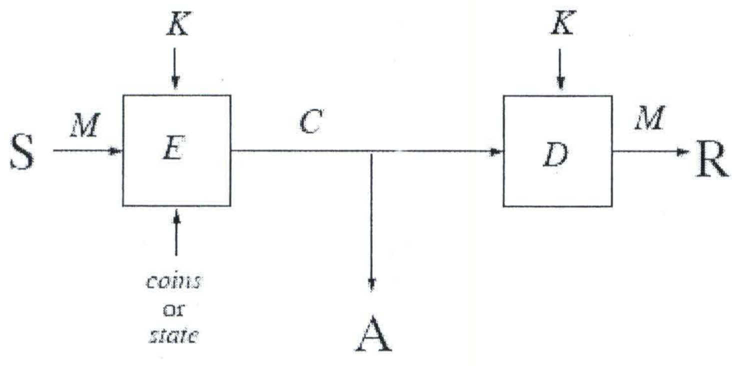


Figure 2.1: Symmetric encryption. The sender and the receiver share a secret key, *K*. The adversary lacks this key. The message *M* is the plaintext; the message *C* is the ciphertext.

The encryption scheme does not tell the adversary what to do. It does not say how the key, once generated, winds its way into the hands of the two parties. And it does not say how messages are transmitted. It only says how keys are generated and how the data is processed.

2.2.4 PRIVACY

The goal of a symmetric encryption scheme is that an adversary who obtains the ciphertext should be unable to learn anything about the plaintext.One thing encryption does not do is hide the length of a plaintext string. This is usually recoverable from the length of the ciphertext string.

As an example of the issues involved in defining privacy, let a question be asked about whether it is impossible for the adversary to figure out *M* given *C*. This however, cannot be true, because the adversary could just guess *M*, by outputting a random sequence of |*M*| bits. (As indicated above, the length of the plaintext is usually computable from the length of the ciphertext.) He would be right with probability 2^{-n}. Not bad, if, say $n = 1$! Does that make the scheme bad? No. But it shows that security is a probabilistic thing. The scheme is not secure or insecure; there is just some probability of breaking it.

Another issue is a priori knowledge. Before M is transmitted, the adversary might know something about it, for example, that M is either 0 or 1. Why? This is because she knows Alice and Bob are talking about buying or selling a fixed stock, and this is just a buy or sell message. Now, she can always get the message right with probability 1/2. How is this factored in? So far one might imagine that an adversary attacking the privacy of an encryption scheme is passive, merely obtaining and examining ciphertexts. In fact, this might not be the case at all.

2.2.5 MESSAGE AUTHENTICITY

In the message-authentication problem the receiver gets some message which is claimed to have originated with a particular sender. The channel on which this message flows is insecure. Thus the receiver R wants to distinguish the case in which the message really did originate with the claimed sender S from the case in which the message originated with some impostor, A. In such a case, the design of an encapsulation mechanism with the property that un-authentic transmissions lead to the decapsulation algorithm outputting the special symbol will be considered.

The most common tool for solving the message-authentication problem in the symmetric setting is a *message authentication scheme*, also called a *message authentication code* (MAC). Such a scheme is specified by a triple of algorithms, Π = (*K*, *T* , *V*). When the sender wants to send a message *M* to the receiver, he computes a "tag," *σ*, by applying *T* to the shared key *K* and the message *M*, and then transmits the pair (*M*,*σ*). The encapsulation procedure referred to above thus consists of taking *M* and returning this pair. The tag is also called a MAC. The computation of the MAC might be probabilistic, just as with encryption or it may well be deterministic. The receiver, on receipt of *M* and *σ*, uses the key *K* to check if the tag is OK by applying the *verification algorithm V* to *K*,*M* and *σ*. If this algorithm returns 1, he accepts *M* as authentic; otherwise, he regards *M* as a forgery. An appropriate reaction might range from ignoring the bogus message to tearing down the connection to alerting a responsible party about the possible mischief. See Figure 2.2 below.

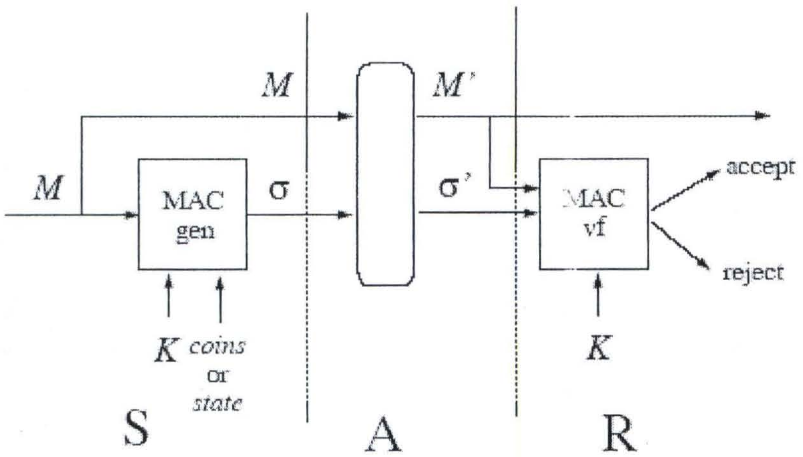


Figure 2.2: A message authentication code. The tag *σ* accompanies the message *M*. The receiver *R* uses it to decide if the message really did originate with the sender *S* with whom he shares the key *K*.

2.2.6 THE ASYMMETRIC SETTING

A shared key K between the sender and the receiver is not the only way to create the information asymmetry that is needed between the parties and the adversary. In the *asymmetric setting*, also called the *public-key setting*, a party possesses a *pair* of keys—a *public key*, *pk*, and an associated *secret key, sk*. A party's public key is made publicly known and bound to its identity. For example, a party's public key might be published in a phone book. The problems that arise are the same as before, but the difference in the setting leads to the development of different kinds of tools.

2.2.7 ASYMMETRIC ENCRYPTION METHOD

The sender is assumed to be able to obtain an authentic copy *pkR* of the receiver's public key. (The adversary is assumed to know *pkR* too.) To send a secret message M to the receiver the sender computes a ciphertext $C \leftarrow \varepsilon_{pkR}$ (M) and sends C to the receiver. When the receiver receives a ciphertext C he computes $M \leftarrow D_{skR}(C)$.

2.2.8 THE ASYMMETRIC ENCRYPTION

Scheme $\Pi = (K, E, D)$ is specified by the algorithms for key generation, encryption and decryption.

For a picture of encryption in the public-key setting, see Fig. 2.3 overleaf.

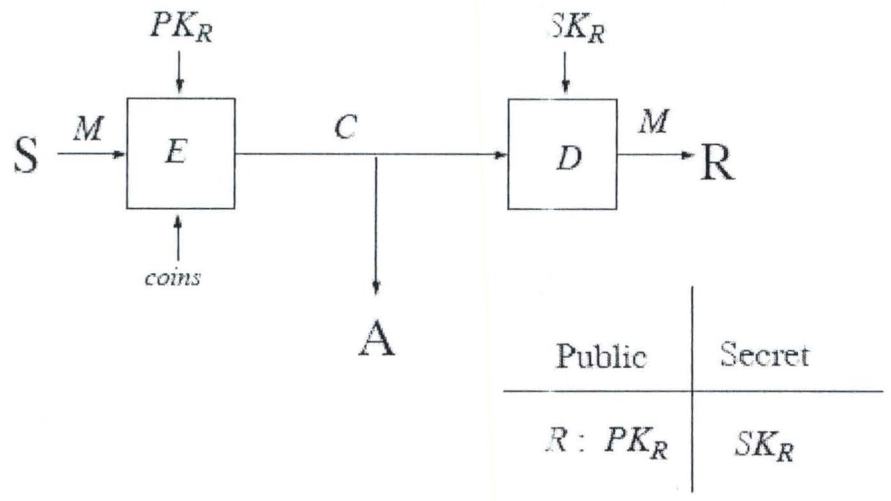


Figure 2.3: Asymmetric encryption. The receiver *R* has a public key, *pkR*, which the sender knows belongs to *R*. The receiver also has a corresponding secret key, *skR*.

The idea of public-key cryptography, and the fact that we can actually realize this goal, is remarkable. Somebody has never met the receiver before he can send him a secret message by

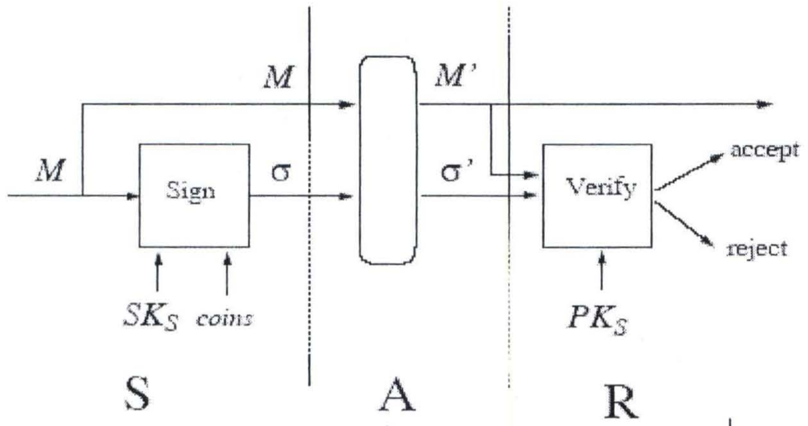
looking up some information in a phone book and then using this information to help him garble up the message he wants to send. The intended receiver will be able to understand the content of

the message, but nobody else will. The idea of public-key cryptography is due to Whitfield Diffie and Martin Hellman and was published in 1976.

2.2.9 DIGITAL SIGNATURES

The tool for solving the message-authentication problem in the asymmetric setting is a *digital signature*. Here the sender has a public key *pkS* and a corresponding secret key *skS*. The receiver is assumed to know the key *pkS* and that it belongs to party *S*. The adversary is assumed to know *pkS* too. When the sender wants to send a message *M,* he attaches to it some extra bits, σ, which is called a *signature* for the message and is computed as a function of *M* and *skS* by applying to them a *signing* algorithm Sign. The receiver, on receipt of *M* and σ, checks if it is OK using the public key of the sender, *pkS*, by applying a *verification* algorithm *V*. If this algorithm accepts, the receiver regards *M* as authentic; otherwise, he regards *M* as an attempted forgery.

The digital signature scheme Π = (*K*, Sign, *V*) is specified by the algorithms for key generation, signing and verifying. A picture is given in Fig. 2.4.



Public	Secret
$S : PK_S$	SK_S

Figure 2.4: A digital signature scheme. The signature σ accompanies the message *M*. The receiver. *R* uses it to decide if the message really did originate with the sender *S* with has public key *pkS*.

One difference between a MAC and a digital signature concerns what is called *non-repudiation*.

With a MAC anyone who can verify a tagged message can also produce one, and so a tagged message would seem to be of little use in proving authenticity in a court of law. But with a digitally-signed message the *only* party who should be able to produce a message that is verified under public key *pkS* is the party *S* himself. Thus if the signature scheme is good, party *S* cannot just maintain that the receiver, or the one presenting the evidence, concocted it. If signature σ authenticates *M* with respect to public key *pkS*, then it is only *S* that should have been able to devise σ. The sender cannot refute that. Probably the sender *S* can claim that the key *skS* was stolen from her. Perhaps this, if true, might still be construed the sender's fault.

CHAPTER THREE

ANALYSIS OF CASE STUDY

3.1 INTRODUCTION

Having treated cryptography generally in the previous chapters, attention is now being specifically shifted to the internet e-mail of which this topic of data ecurity revolves round.

.2 HOW E-MAIL WORKS

.2.1 SENDING AN E-MAIL MESSAGE

Sending an email message is like sending a letter. When someone sends a letter, he drops it off at the local post office. The local post office looks at the address and figures out which regional post office the letter should go to. The regional post office then looks at the address and figures out which local post office is closest to the recipient. Finally, the recipient's local post office delivers the letter to its recipient. Computers are like "post offices", and the "Simple Mail Transport Protocol" (SMTP) is the "procedure" which a post office uses to figure out where to send the letter next. Any program that sends an e-mail message uses SMTP to deliver that message to the next "post office" for relaying to its final destination.

Most people send mails in two ways - with a web-based interface like Yahoo! or Hotmail, or with an "email client" program like Outlook or Eudora.

When someone sends a message with an e-mail program on his personal computer (or his cell phone or PDA), he has to specify a server so that the e-mail program knows where to send the message. This server is like the local post office. The e-mail program talks directly to the server using the computer protocol (language) known as SMTP. This is like dropping off a letter at the local post office.

When one uses WebMail ,the personal computer uses an internet connection to communicate with a web server. The "language" that the internet connection uses is HTTP that is "Hypertext Transfer Protocol". When message is sent with WebMail the web server contacts its SMTP server and sends the message to it.

3.2.2 DELIVERY OF E-MAIL FROM A SENDER'S SMTP SERVER TO THE RECIPIENT'S SMTP SERVER:

When an SMTP Server receives an e-mail message, it first checks if it has an inbox for the message recipient. If it does not it must "relay" that e-mail message to another SMTP Server closer to the recipient. This is analogous to how the local post office forwards a letter to a regional post office. This process is known as "e-mail relaying".

How does your SMTP Server know where to relay the message to? If the recipient's e-mail address is "bob@odun.net", then the recipient's domain name is "odun.net". Part of the "DNS settings" for the recipient's domain includes an ordered list of SMTP Servers that expect to receive e-mail for this recipient. The highest priority SMTP Server listed is the recipient's actual SMTP Server; the others are "backup SMTP Servers". These backup servers merely queue e-mail for later delivery to the recipient's actual SMTP Server.

There are many scenarios that govern the path an e-mail message may take from the sender's to the recipient's SMTP Server. Some of these include:

1. The sender's server successfully contacts the recipient's server and sends the e-mail message directly.
2. The sender's server cannot contact the recipient's actual SMTP Server (maybe the recipient's server is busy, down, or has some other connection problem). In this case the sender's server tries to contact and deliver the message to the recipient's first backup server.
3. The sender's server cannot contact the recipient's actual SMTP Server or its first backup server. In this case the sender's server tries to contact and deliver the message to the recipient's second backup server.
4. The sender's server can not contact any of the recipient's servers. In this case ,it will queue the message and try to send it later. It will keep

retrying periodically for several days until it succeeds in sending or gives up.

Any message delivered to the backup servers goes through the same process of trying to contact the recipient's actual SMTP Server, or a higher priority backup server. Backup servers may also queue e-mails so as to send them later (Note that a recipient may have zero or more backup servers, not necessarily two as in this example).

Once the e-mail message arrives at the recipient's SMTP Server and is delivered to the recipient's e-mail box, the recipient may pick up the message and read it whenever he chooses .

Each server that receives a message adds its "Received" stamp to the message. This stamp identifies what server received the message, at what time, and *from what other server*. This information allows the recipient to see a message's entire journey.

What should be clear from the above is that:

- All e-mail servers communicate with each other using SMTP
- It is never known how long it will take an e-mail message to get from sender to recipient because it is not known how busy the servers are, how much traffic there is on the Internet, what machines are down for maintenance, etc.

- The messages may sit in queues on any number of servers for any amount of time. Some of these servers may belong to third parties (i.e. may not be under the purview of either the sender or the recipient).
- The recipients can determine the Internet address and name of the computer from which the messages are being sent.

3.2.3 RETRIEVING E-MAIL FROM AN SMTP SERVER

When one receives an e-mail message, it sits in a file in his SMTP Server. If he wishes to view this e-mail message, he must access this file. Any computer wishing to access this file must speak one of the languages the SMTP Server does. With some exceptions, there are really only 2 languages that e-mail computers understand (for e-mail retrieval, as opposed to e-mail sending, for which they use SMTP), one is called the "Internet Message Access Protocol" (IMAP) and the other is called the "Post Office Protocol" (POP).

As a recipient, one can generally retrieve his e-mail by either using a web-based interface known as "WebMail", or via an "e-mail client" program, such as Microsoft Outlook or Eudora, running on his personal computer. The e-mail client programs will talk directly to his e-mail server and speak IMAP or POP. With WebMail, his computer will talk to a WebMail server using a web connection (speaking HTTP); the WebMail server will, in turn, talk to his e-mail server using POP or IMAP.

3.2.4 SECURITY THREATS TO E-MAIL COMMUNICATIONS

This section describes many of the common security problems involved in communications and e-mail in particular.

Eavesdropping: The Internet is a big place with a lot of people on it. It is very easy for someone who has access to the computers or networks through which one's information is traveling to capture this information and read it. Just like someone in the next room listening in on someone's telephone conversation, people using computers "near" the path one's e-mail takes through the Internet can potentially read and copy such messages!

Identity Theft: If someone can obtain the username and password that someone uses to access his e-mail servers, he can read his e-mail and send false e-mail messages to the person. Very often, these credentials can be obtained by eavesdropping on SMTP, POP, IMAP, or WebMail connections, by reading e-mail messages in which one includes this information, or through other means.

Invasion of Privacy: If one is very concerned about his privacy, then he should consider the possibility of "unprotected backups". He may also be concerned about letting his recipients know the IP address of your computer. This information may be used to tell in what city he is located or even to find out what his address is in some cases! This is not an issue with

WebMail, POP, or IMAP, but is an issue when sending e-mail, securely or insecurely, from any e-mail client over SMTP.

Message Modification: Anyone who has system administrator permission on any of the SMTP Servers that one's message visits, cannot only read that message, but can delete or change the message before it continues to its destination. The recipient has no way of telling if the e-mail message that he received has been altered! If the message was merely deleted they wouldn't even know it had been sent.

False Messages: It is very easy to construct messages that appear to be sent by someone else. Many viruses take advantage of this situation to propagate themselves. In general, there is no way to be sure that the apparent sender of a message is the true sender - the sender's name could have been easily fabricated.

Message Replay: Just as a message can be modified, messages can be saved, modified, and re-sent later! Someone could receive a valid original message, but then receive subsequent fake messages that appear to be valid.

Unprotected Backups: Messages are stored in plain text on all SMTP Servers. Thus, backups of these servers' disks contain plain text copies of one's messages. As backups can be kept for years and can be read by anyone with access to them, these messages could still be exposed in insecure places even after one thinks that all copies have been "deleted".

Repudiation: Because normal e-mail messages can be forged, there is no way for one to prove that someone sent him a particular message. This means that even if someone DID send a message to him, the sender can successfully deny it. This has implications with regards to using e-mail for contracts, business communications, electronic commerce, etc.

CHAPTER FOUR

ANALYSIS OF RSA ALGORITHM AND IMPLEMENTATION

4.1 INTRODUCTION

The RSA is a public key cryptographic algorithm that is used to help ensure data communication security. It is simply based on two main cryptographic processes. First, using a public key it converts an input data called the plaintext into an unrecognizable encrypted output called cipher text (encryption process), such that it is impossible to recover the original plaintext without the encryption password in a reasonable amount of time. Secondly, using a private key, the RSA then converts the unrecognizable data back to its original form. Today it is used in web browsers, e-mail programs, mobile phones and virtual private networks.

The challenge of RSA is to develop an algorithm in which it is impossible to determine the private key. This algorithm is based on one-way function. As the name implies, the function is only one-way i.e. given some input values it is relatively easy to compute the result. However, it is extremely difficult, nearly impossible to determine the input values given the result .In the mathematical terms, given *x,* computing *f (x)* is relatively easy, but given *f (x),* computing *x* is extremely difficult. The one-way function used by RSA is the multiplication of two very large prime numbers. It is relatively easy to multiply them but extremely difficult, rather impossible and time consuming to factorize them.

have no prime factors in common. Note that e does not necessarily have to be prime. The value of e is used along with the value n to represent the public key used for encryption.

6. Calculate the unique value d (to be used during decryption) that satisfies the requirement that, if d · e is divided by φ, then the remainder of the division is 1. The mathematical notation for this is $d \cdot e = 1 (\mathrm{mod}\ \varphi)$. In mathematical jargon, it is said that d is the multiplicative inverse of e modulo φ. The value of d is to be kept secret. If one knows the value of φ, the value of d can be easily obtained from e using a technique known as the Euclidean algorithm. If one knows n (which is public), but not p or q (which have been destroyed), then the value of φ is very hard to determine. The secret value of d together with the value n represents the private key.

Once someone has generated a public/private key pair, he can encrypt a message with the public key with the following steps.

1. Take a positive integer m to represent a piece of plaintext message. In order for the algebra to work properly, the value of m must be less than the modulus n, which was originally computed as p · q. Long messages must therefore be broken into small enough pieces that each piece can be uniquely represented by an integer of this bit size, and each piece is then individually encrypted.

2. Calculate the ciphertext c using the public key containing e and n. This is calculated using the equation c = me (mod n).

Finally, the decryption procedure with the private key can be performed using the following steps.

1. Calculate the original plaintext message from the ciphertext using the private key containing d and n. This is calculated using the equation m = cd (mod n).
2. Compare this value of m with the original m, and you should see that they are equal, since decryption is the inverse operation to encryption.

4.2 A Miniature RSA Example

Here is an example of RSA that is almost simple enough to do with pencil and paper. The bit size of the numbers used in this example is ridiculously small (32-bit integers) and offers no real security whatsoever, but at a conceptual level, this example provides a complete picture of what actually happens in the RSA algorithm. The advantage of studying this tiny paper and pencil example is that with these very small bit sizes, the underlying concepts are much more tangible and easily visualized. After all, not too many people can do 1024-bit arithmetic in their head! Even working with such tiny 32-bit numbers, the exponentiation step of the algorithm will easily overflow this 32-bit capacity if one is not careful about how he implements it.

Following the conceptual steps outlined above, start off by choosing two unequal prime numbers p and q. Since very small values are intentionally chosen, prevent subsequent calculations from overflowing the 32-bit integer arithmetic. This also allows one to follow along using the Calculator program provided with Windows to verify the arithmetic.

1. Assume that the random values for the primes p and q have been chosen as
   - $p = 47$
   - $q = 73$
2. Then the product n of these two primes is calculated:
   - $n = p \cdot q = 3431$
3. The Euler quotient φ for these two primes is found easily using the following formula:
   - $\Phi(n) = (p - 1) \cdot (q - 1) = 3312$
4. Now that we have n and φ, we should discard p and q, and destroy any trace of their existence.
5. Next, we randomly select a number e that is greater than 1, less than n, and relatively prime to phi. Of course, there is more than one choice possible here, and any candidate value you choose may be tested using the Euclidian method. Assume that we choose the following value for e:
   - $e = 425$
6. Then the modular inverse of e is calculated to be the following:

  - d = 1769

7. Keep d private and make e and n public.

Now that one has his private key information d and his public key information e and n, he can proceed with encrypting and decrypting data. As one would probably imagine, this data must be represented numerically to allow the necessary calculations to be performed. In a real-life scenario, the plaintext is typically a hash value or a symmetric key, but it could actually be just about any type of data that one could imagine. Whatever form this data takes, it will have to be somehow represented as a sequence of integer numbers, each with a size that will be limited by the key size that one is using. One does not concern himself here with the details of encoding and chunking of the data, but instead one focuses on the conceptual aspects of RSA. For this reason, this example simply considers a scenario in which the plaintext data is one simple, small integer value.

1. Assume that one has plaintext data represented by the following simple number:
   - plaintext = 707
2. The encrypted data is computed by c = me (mod n) as follows:
   - ciphertext = 707^425(mod 3431) = 2142
3. The ciphertext value cannot be easily reverted back to the original plaintext without knowing d (or, equivalently, knowing the values of p

and q). With larger bit sizes, this task grows exponentially in difficulty. If, however, one is privy to the secret information that d = 1769, then the plaintext is easily retrieved using m = c d(mod n) as follows:

- plaintext = 2142^1769(mod 3431) = 707

If one compiles the following code, he will verify that the results shown above are correct. While one looks at this code, keep in mind that a realistic RSA implementation uses a much larger modulus than n = 3431, and a realistic message typically contains too many bits to be represented by a tiny number such as m = 707.

Public Key(Example 2)

Another example of how the public key for every user can be obtained is shown below:

1. Locate two adequate prime numbers *p* and *q* at random. (For example 256-bit prime numbers).

2. Multiply *p* and *q*, which comes up with *n*.

3. Locate one integer *e*, which can satisfy GCD[*e*,Φ(*n*)]=1. Φ(*n*) represents Euler's quotient, which means this integer is smaller than *n* and a prime number against *n*.When *n*=*pq*, Φ(n)=(*p*-1)(*q*-1).

4. Calculate *d*, which satisfies *ed* •1 mod Φ(*n*).

5. (*e*, *n*) is the public key and (*d*1 ,*n*) is the private key.

RSA ALGORITHM CRYPTOGRAPHY:

A. KEY GENERATION:

1. Choose two primes p and q

2. Calculate $n = p*q$.

3. Calculate $\Phi(n) = \Phi(p \times q) = (p-1)(q-1)$.

4. Select one integer e, which satisfies $\gcd(\Phi(n),e) = 1$. $1 < e < \Phi(n)$.

5. As $ed = 1 \bmod \Phi(n)$, utilize Euclid's Algorithm to get $d=e^{-1} \bmod \Phi(n)$.

6. A pair of public keys is obtained, which is $\{e, n\}$.

7. A pair of secret keys is obtained, which is $\{d, n\}$.

B. ENCRYPTING:

1. Plain text M and $M < n$.

2. Encoded text $C = M^e \pmod{n}$

C. DECRYPTING:

1. Cryptographic text is C.

2. Plain text can be obtained through decoding, which gets $M=C^d \pmod{n}$.

RSA ALGORITHM CRYPTOGRAPHY EXAMPLE 3:

PUBLIC KEY

1. There are two prime numbers p and q and $p=7$, $q=17$. Then, $n=p*q=7*17=119$.

2. $\Phi(n) = \Phi(pq) = (p-1)(q-1) = 96$ according to Euler's algorithm.

3. Locate one integer e, which is a prime number against $\Phi(n)$ and is smaller than

$\Phi(n)$. We choose $e = 5$ in this example.

4. As $ed = 1 \bmod 96$, we know that $d = e^{-1} \bmod 96 = 77$ in compliance with Euclid's algorithm ($77*5 = 385 = 4*96 +1$).

5. In this formula, both transmitter and receiver have to know n, but only the former knows about e and the latter knows about d. Thus, we realize that the public key is $\{e, n\}$ and the secret key is $\{d, n\}$.

RSA ALGORITHM CRYPTOGRAPHY EXAMPLE 2:

PUBLIC KEY:

1. Presume two prime numbers p and q and $p=7$, $q=17$. Then, $n = pq = 7 *17 = 119$.

2. $\Phi(n) =\Phi(pq) = (p-1)(q-1) = 96$ according to Euler's algorithm (quotient).

3. Locate one integer e, which is a prime number against $\Phi(n)$ and is smaller than $\Phi(n)$. We choose $e = 5$ in this example.

4. As $ed = 1 \bmod 96$, we know that $d = e^{-1} \bmod 96 = 77$ in compliance with Euclid's algorithm ($77*5 = 385 = 4*96 +1$).

5. Finally, we know that the public key = $\{e, n\} = \{5, 119\}$ and the secret key = $\{d,n\} = \{77, 119\}$.

ENCRYPTING:

1. The public key is $\{e, n\} = \{5, 119\}$. We choose the plain text $M = 20$, then $C = M^e (\text{mod } n) = 20^5 \text{ mod } 119 = 3200000/119 = 26890...90$ (remainder). Therefore, cryptographic text is 90.

DECRYPTING:

2. The secret key is $\{d, n\} = \{77, 119\}$. Then $M = C^d \text{ (mod } n) = 90^{77} \text{ mod } 119$ $=(8^{38}*90) \text{ mod } 119 = (8^{36}*8*8*90) \text{ mod } 119 = (36^{12}*48\ ) \text{ mod } 119 =$ (106^6*48)

$\text{mod } 119 = (50^3*48) \text{ mod } 119 = 20$ (remainder).

4.3 CHOICE OF RSA PARAMETER

The RSA system is the first system of placing security upon factor algorithm. We know that if n can be broken down into factors with the public key $\{e, n\}$, then $\Phi(n) =(p-1)(q-1)$ cannot be hidden, which makes the decoding key d become a secret no more and the whole RSA system becomes insecure. As a result, it is critical to choose the public key n while applying the RSA system. Once the public key n is chosen, nobody is able to get p and q from n. We are going to discuss the choices of strong prime numbers as well as things to be noticed while selecting parameters e and d.

4.4 THINGS TO BE NOTICED WHILE CHOOSING N

Among all cryptosystems that base their security on factor algorithm, prime factors of *N*, *p* and *q*, have to be selected appropriately to prove that it is impossible to divide factors.

A. *p* and *q* have to be strong prime numbers.

If prime number *p* can satisfy the following requirements, then this prime number is called a strong prime number:

1. Two large prime numbers p1 and p2; p1/p-1 and p2/p+1

2. Four large prime numbers r1, s1, r2 and s2; r1/p1-1, r2/p2-1,s1/ p1+1, and s2/ p2+1.

The figure below shows strong primes. Prime numbers such as r1, s1, r2 and s2 are termed as level-3 primes and p1 and p2 are called level-2 primes. As for p, it is named as level-1prime. Obviously, general prime numbers belong to level-3 primes; whereas, strong primes belong to level-1 category.

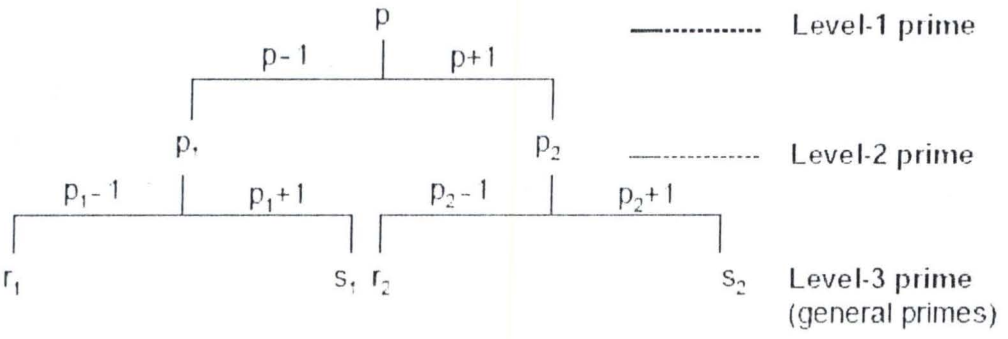


Figure 4.1 Levels of prime numbers

N comes from multiplication of strong primes *p* and *q*, whose factor algorithm becomes a more difficult math's problem. We can generate random primes with fixed length/capacity first, then locate a level-2 prime either +1 or -1.Finally; a strong prime can be generated by means of level-2 prime.

If *r* and *s* are odd primes, then prime *p* can satisfy the formula of $p1=1 \pmod{2r} = -1 \pmod{2s}$ and *p* can be transformed into $p=2ss^{-1} + 2krs$. $ss^{-1} = 1 \bmod r$, $1 \le s^{-1} < r$.

B. The difference between *p* and *q* has to be great (more than several bits). When the difference between *p* and *q* is small, we can predict the average value of *p* and *q*, $p+q/2 = (N)^{1/2}$, under the circumstance of $N=pq$.

The following formula results:

$(p+q/2)^2 - N = (p-q/2)^2$ (4.1)

If N=164009, we predict $p+q/2 = (164009)^{1/2} = 405$ (4.2)

Through Eq. (4.1):

$(405)^2 - 164009 = (p-q/2)^2$,then $p-q/2 = 4$ (4.3)

We already know $p+q/2 = 405$ (4.4)

According to formulae (4.2) and (4.4), $p-q=8$ and $p+q=810$.

Then, $p=409$ and $q=401$.

As a result, the difference between p and q has to be great (above several bits), which makes it uneasy to decode.

C. *p* and *q* should be great enough that it becomes impossible to divide factor *N*.

It is evident that if factor *N* can be divided, then RSA can be decoded. Thus, length/capacities of *p* and *q* should be large enough that dividing factor *N* becomes impossible. As dividing factors is the basic problem in cryptography, algorithm of dividing factors has made a lot of progress for the past decade. We list the evolving results of factor algorithm, those so-called compounds difficult to divide.

4.5 THINGS TO BE NOTICED WHILE CHOOSING E AND D

After the RSA Cryptosystem is proposed, it is associated that if d or e is small, then logarithm of encoding or decoding can be highly accelerated. However, insecurity has been found out after scholars conducted researches.

Knuth (1982), a famous scholar, suggested all users of RSA Cryptosystems should use $e=3$ to serve as a public key. One reason is that it is faster and the other one is that it can reduce the file length/capacity of public key. Whereas, in 1986, Hastad claimed that if *e* was too small, then the RSA system was defective and could be cracked. As for the consideration of the secret key *d*, **Wiener (1990)** proposed that the RSA cryptosystem with smaller *d* could be cracked with successive fraction algorithm. As long as the secret key is within 1/4 length/capacity of *n* bit, Wiener could crack down the complication of the RSA cryptosystem through multiple exponents. As a result, the length/capacity of the secret key d has to be paid attention to while programming. That is to say that it has to be within 1/4 length/capacity of modulus n to be prevented from cracking by Wiener's method.

CHAPTER FIVE

SUMMARY, FINDINGS AND CONCLUSION

5.1 SUMMARY

The RSA algorithm is indeed a milestone in cryptography. With the growing threat to data security, this algorithm couldn't be more useful. One hopes to see more of encryption algorithms stronger than the RSA's that will greatly stem the tide of data insecurity and on the whole e-mail insecurity.

5.2 FINDINGS

The RSA keys are very difficult to calculate. The implementation of RSA algorithm may require special purpose hardware.

5.3 CONCLUSION

One may ask, how secure is RSA? Some one could break RSA by finding a way to calculate the private key from the public key. The security of the RSA rests in the sheer mathematical difficulty in doing that. The only feasible way to calculate the private key is to know the prime factor in the public key; to be accurate, the two prime factors in its modulus. If you know what these prime numbers are, then it's possible for your software to calculate the private key. Indeed, that's what RSA does when it generates a person's private key in the first place. It picks two large prime numbers at random and multiplies those together. That gives the public key modulus. Using the two prime numbers and the exponent picked, RSA then works out the private key. It is a formidable calculation but possible. Without the prime numbers, it can be hopeless.

RSA is every where. It is useful as a secure electronic envelope for small messages.RSA's customers include: Apple Computers, NOVEL, Lotus, AT&T, etc. they have built RSA into their operating systems, networks, E-mail applications, and electronic commerce system. The system is also used in smart cards and in some browsers.

Users of e-mails can now breathe a sigh of relief, knowing that their mails can go to their destinations without harm.

REFERENCES

1. ANSI X. (1998) *Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (DSA).* Appendix A, American National Standards Institute.
2. Aumann, Y.et al (2002) *Everlasting Security in the bounded storage model.* IEEE Transactions on Information Theory 48(6):1668-1680.
3. Bamford, J. (2001) *Body of Secrets : Anatomy of the Ultra-Secret National Security Agency from the Cold War Through the Dawn of a New Century.* New York: Doubleday.
4. Barr, T.H. (2002) *Invitation to Cryptology.* Upper Saddle River (NJ): Prentice Hall.
5. Bauer, F.L. (2002) *Decrypted Secrets: Methods and Maxims of Cryptology,* 2nd ed. New York: Springer Verlag.
6. Boneh D. (2001) *Simplified OAEP for the RSA and Rabin functions. Advances in Cryptology* - CRYPTO. Unpublished Lecture Notes in Computer Science.
7. Burton Kalinski (1993) *Some Examples of the PKCS Standards.* RSA Laboratories. <ftp://ftp.rsasecurity.com/pub/pkcs/ascii/examples.asc.
8. Clifford C. (1973) *A Note on Non-Secret Encryption.* CESG Research Report. <http://www.cesg.gov.uk/site/publications/media/notense.pdf.

9. Denning, D.E.(1982) *Cryptography and Data Security.* Reading (MA): Addison-Wesley.

10. Diffie, W. et al. (1998) S. *Privacy on Line.* Boston: MIT Press.

11. Diffie, W. et al. (1976) *New Directions in Cryptography.* IFEE Transactions on Information Theory, Vol IT-22,pp.644-654.

12. Goldreich, O. (2004) *Foundations of Cryptography.* Volume II Basic Applications. Cambridge University Press.

13. Liddel et al (1984) *Cryptography.* Greek-English Lexicon, Oxford University Press.

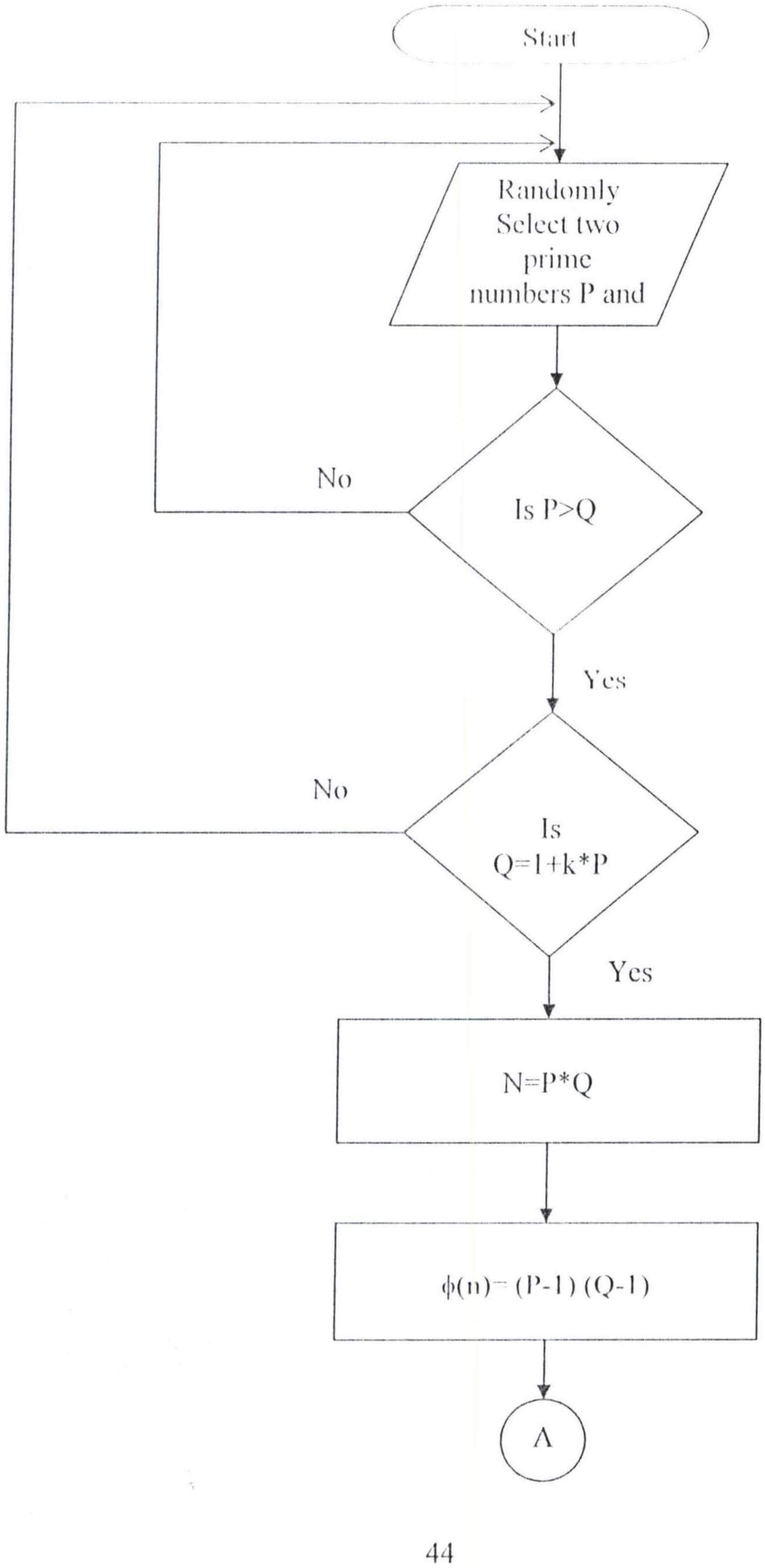
14. Leon, B.A. (1497) *Cryptography goes Public.* Foreign Affairs 141,pp.153.

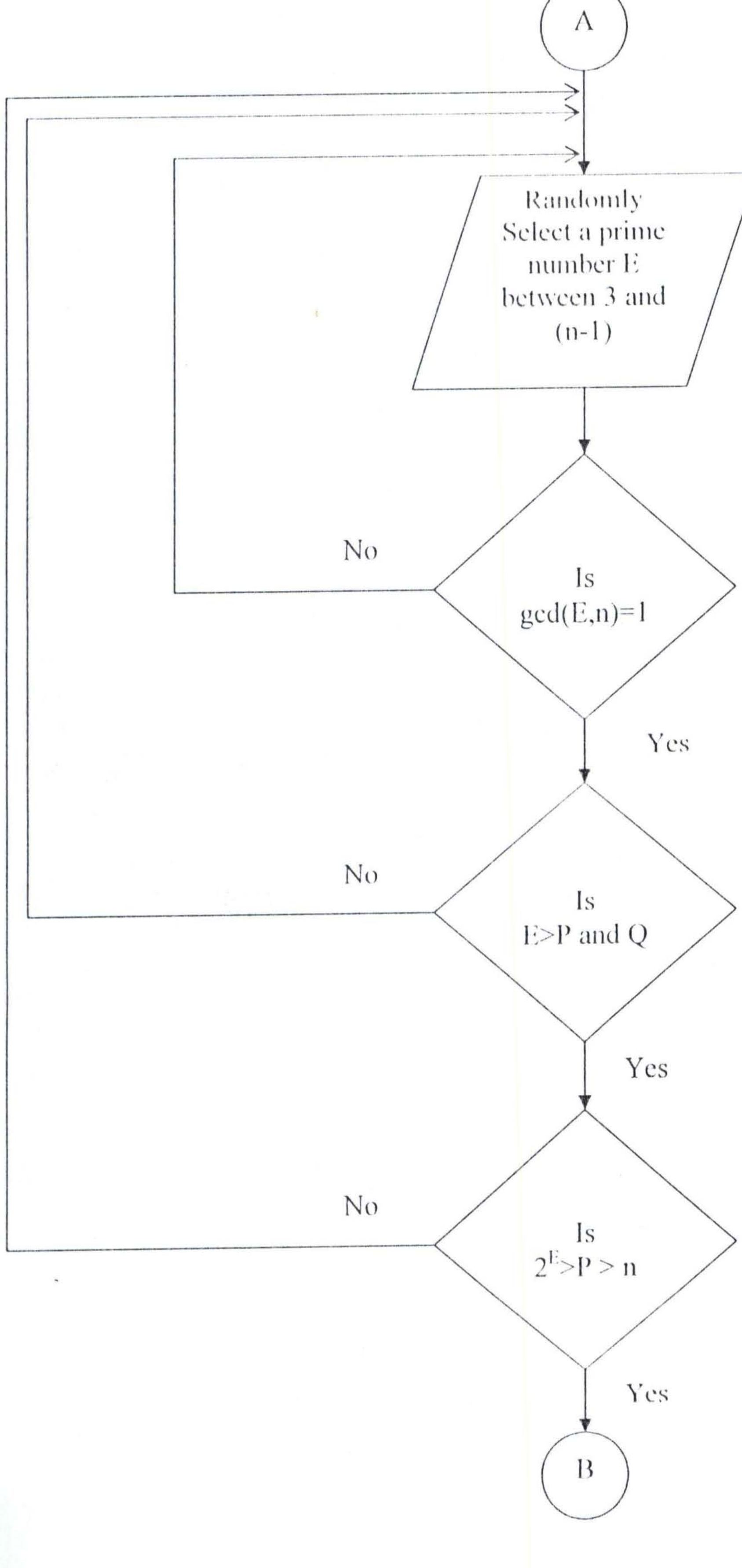
15. Rivest, A. et al(1978) *A Method for Obtaining Digital Signatures and Public-key Cryptosystems.* Communications of the ACM, 21 (2), pp. 120-126.

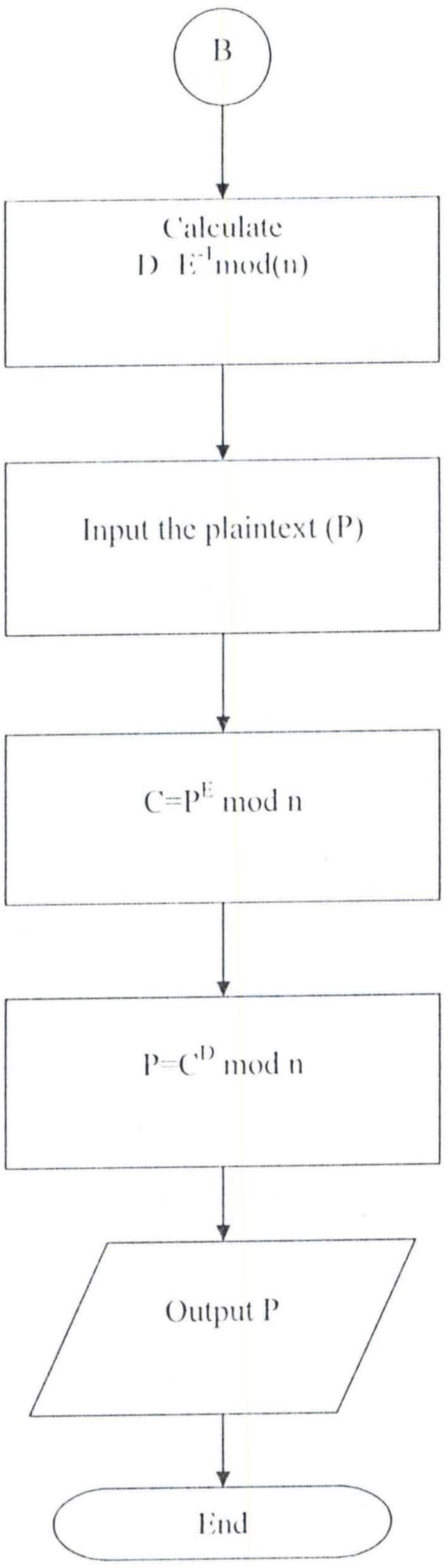
16. *The Puzzle Palace (1983) Inside the National Security Agency, America's most secret intelligence organization.* New York: Penguin Books.

17. RSA Laboratories. PKCS #1 v2.1 *RSA Encryption Standard.* June 2002, <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf.

APPENDIX A
FLOWCHART




A
Randomly Select a prime number E between 3 and (n-1)
No
Is gcd(E,n)=1
Yes
No
Is E>P and Q
Yes
No
Is 2^E>P > n
Yes
B


B
Calculate
D=E^{-1}mod(n)
Input the plaintext (P)
C=P^{E} mod n
P=C^{D} mod n
Output P
End

APPENDIX B

PROGRAM LISTING (INPUTS AND OUTPUTS)

CODE LISTINGS

```
Dim fnt, fn, lin As String
Dim dsize As Double
Dim i As Integer
Dim pdata(), pout, filepout As String
Dim adata(100000) As Integer
Dim out(100000) As Double
Dim linefeed As String
Dim tryI, tryJ, jval As Long
Dim i1, a1 As Long
Dim GetRndA, GetRndB, indexA, indexB, RndValA, RndValB, primeA, primeB, opA, opB As
Long
Dim chkA, chkB, tstA, tstB As Boolean
Dim cont As Boolean
Dim PRIME1, PRIME2, PROD, PHIE, PUBLICKEY, SECRETKEY, POS As Long
Dim CIPHER As String
Dim y, x, N As Long
Dim store(9999), mtp, temp, flmt, disp(15), t, rvf(15), res, pcnt, suma(15) As Long
Dim pow(15), ch(14), a, c, il, cnt, j, jp, kt, lt, l, j1, ic, jc As Long
Dim op, t2 As String
Dim gt(15), rv(15), lent, chlen, dval As Integer
Dim gate, fin As Boolean
Dim fname As Variant
Private Sub Command2_Click()
   Form2.Show
   Unload Me
   Dim d As String
   MsgBox ("")
   d = (UCase(Right(Label6.Caption, 3)))
   MsgBox (d)
End Sub
Private Sub Command3_Click()
   End
End Sub
Private Sub cmdDecrypt_Click()
On Error Resume Next
   txtTemp.Text = txtFileName.Text
   If (UCase(Right(txtFileName.Text, 3)) <> "TXT") Then
      MsgBox "Cannot Encrypt This Type Of File!", vbExclamation, "Type Mismatch!!!"
```

```
    Exit Sub
End If
SECRETKEY = InputBox("Enter The Secret Key", "RSA")
PROD = InputBox("Enter The Value Of Phi", "RSA")
Open fn For Input As #1
    While Not EOF(1)
        Line Input #1, lin
        dsize = Len(lin)
        If dsize > 0 Then
           ReDim pdata(1 To dsize)
        Else
           ReDim pdata(1)
        End If
        rlinefeed = ""
        j = ""
        i = 1
        Do Until i = dsize + 1
           j = Mid(lin, i, 1)
           rlinefeed = rlinefeed & j
           i = i + 1
        Loop
        x = rlinefeed
        y = SECRETKEY
        N = PROD

      'FINDING THE BINARY OF Y
         t2 = ""
         a = y
         Do While a >= 2
            c = a Mod 2
            a = Fix(a / 2)
            t2 = t2 & c
         Loop
         t2 = t2 & a
         op = StrReverse(t2)

      'FINDING THE LENGTH
        a = 1
        Do While (gate = False)
           c = Mid$(op, a, 1)
           gt(a) = c
           If (c <> 1 And c <> 0) Then
              gate = True
           End If
           cnt = a
           a = a + 1
```

```
Loop
lcnt = cnt - 1

'FINDING THE REVERSE
a = 1
For c = lcnt To 1 Step -1
  rv(a) = gt(c)
  a = a + 1
Next c

'FINDING THE POWERS
l = 2
j = 1
kt = 2
pow(1) = rv(1)
ch(1) = pow(1)
lt = 2
chlen = 1
Do While (lt <= lcnt)
    pow(j + 1) = rv(kt) * 2 ^ j
    ch(l) = pow(j + 1)
    chlen = chlen + 1
    j = j + 1
    kt = kt + 1
    l = l + 1
    lt = lt + 1
Loop

'FINDING THE SQUARES OF X
store(1) = x Mod N
temp = store(1)
disp(1) = temp
mtp = 2
flmt = 1
t = 2
Do While ((flmt / 2) <= Val(y))
   store(mtp) = (temp ^ 2) Mod N
   temp = store(mtp)
   disp(t) = (store(mtp))
   mtp = mtp * 2
   flmt = mtp * 2
   t = t + 1
Loop

'ELIMINATIN ZEROS
i1 = 1
```

```
        j1 = 2
        rvf(1) = pow(1)
        Do While (i1 < 15)
            If ch(i1) <> 0 Then
                rvf(j1) = ch(i1)
                j1 = j1 + 1
            End If
            i1 = i1 + 1
        Loop

        'CALLING VALUES
        pcnt = 1
        jc = 1
        For ic = 1 To chlen
            If ch(ic) <> 0 Then
                suma(jc) = (disp(ic))
                pcnt = pcnt + 1
                jc = jc + 1
            End If
        Next ic

        'RESULT
        res = 1
        Dim q1 As Integer
        res = (suma(1) * suma(2)) Mod N
        For q1 = 2 To (pcnt - 2)
            res = (res * suma(q1 + 1)) Mod N
        Next q1
        pout = Chr(res)
        Dim loc As Integer
        filepout = filepout & pout
        x = y = N = res = 0
    Wend
    Open fnt For Append As #2
        Print #2, filepout
    Close #2
  Close #1
  FileCopy fnt, fn
  Kill (fnt)
  MsgBox ("Decryption Is Completed Successfully")
  End
nd Sub

rivate Sub cmdEncrypt_Click()
)n Error Resume Next
  If (UCase(Right(txtFileName.Text, 3)) <> "TXT") Then
```

```
   MsgBox "Cannot Encrypt This Type Of File!", vbExclamation, "Type Mismatch!!!"
   Exit Sub
End If
'Generating Two Random Prime Numbers
   If cont = False Then
      While (chkA = False)
         tstA = False
         Randomize
         GetRndA = Rnd() * 100
         RndValA = Round(GetRndA, 1)
         For indexA = 2 To RndValA - 1
            primeA = RndValA Mod indexA
            If (primeA = 0) Then
               tstA = True
            End If
         Next indexA
         If (tstA = False) Then
            If RndValA <= 2 Then
               'do nothing
            Else
               PRIME1 = Round (RndValA, 0)
               chkA = True
            End If
         End If
      Wend
      While (chkB = False)
         tstB = False
         Randomize
         GetRndB = Rnd() * 100
         RndValB = Round(GetRndB, 1)
         For indexB = 2 To RndValB - 1
            primeB = RndValB Mod indexB
            If (primeB = 0) Then
               tstB = True
            End If
         Next indexB
         If (tstB = False) Then
            If RndValB <= 2 Then
            Else
               PRIME2 = Round(RndValB, 0)
               chkB = True
            End If
         End If
     Wend
     If (PRIME1 = PRIME2) Then
        'do nothing
```

```
    ElseIf (PRIME1 <= 2) Then
        'do nothing
    ElseIf (PRIME2 <= 2) Then
        'do nothing
    Else
        cont = True
    End If
End If

'FINDING THE VALUE OF N
PROD = PRIME1 * PRIME2

'FINDING THE VALUE OF PHIE
PHIE = (PRIME1 - 1) * (PRIME2 - 1)

'FINDING THE PUBLIC KEY
cont = False
For i1 = 2 To (PHIE - 1)
   If cont = False Then
       a1 = PHIE Mod i1
       If a1 = 0 Then
           'do nothing
       Else
           PUBLICKEY = i1
           cont = True
       End If
   End If
Next i1

'FINDING THE SECRET KEY
cont = False
For q = 1 To 100000
   If cont = False Then
       If ((PUBLICKEY * q) Mod PHIE) = 1 Then
           SECRETKEY = q
           cont = True
       End If
   End If
Next q

'GETTING THE CHARACTER ONE BY ONE FROM THE FILE
Open fn For Input As #1
While Not (EOF(1))
   Line Input #1, lin
   dsize = Len(lin)
   If dsize > 0 Then
```

```
          ReDim pdata(1 To dsize)
       Else
          ReDim pdata(i)
       End If
       linefeed = ""
       i = 1
       Do Until i = dsize + 1
          pdata(i) = Mid(lin, i, 1)
          i = i + 1
       Loop
    Wend


    'FINDING THE ASCII OF CHARACTER
    For i = 1 To dsize
      adata(i) = Asc(pdata(i))
    Next i

    'FINDING THE CIPHER TEXT
    Open fnt For Append As #2
    For jval = 1 To dsize
       x = adata(jval)
       y = PUBLICKEY
       N = PROD
       Powers
       CIPHER = res
       Print #2, CIPHER
  Next jval
  Close #2
  Close #1
  FileCopy fnt, fn
  Kill (fnt)
  g = MsgBox("Encryption Is Complete", vbInformation, "RSA")
  Form3.lblSecret.Caption = SECRETKEY
  Form3.lblN.Caption = PROD
  'Unload Me
  Form3.Show
nd Sub

ivate Sub cmdExit_Click()
End
nd Sub

ivate Sub cmdSend_Click()

nd Sub
```

```
Private Sub Dir1_Change()
   File1.Path = Dir1
End Sub
Private Sub Drive1_Change()
   Dir1.Path = Drive1
End Sub

Private Sub File1_Click()
   txtFileName.Text = Dir1 & "\" & File1
   txtFileName.FontBold = True
   txtFileName.FontSize = "8"
   fn = txtFileName.Text
   fnt = Dir1 & "\" & "temp.tmp"
End Sub
Private Sub Form_Load()
   Dim g As Integer
   txtTemp.Visible = False
End Sub

Private Sub Powers()
'FINDING THE BINARY OF Y
   t2 = ""
   a = y
   Do While a >= 2
      c = a Mod 2
      a = Fix(a / 2)
      t2 = t2 & c
   Loop
   t2 = t2 & a
   op = StrReverse(t2)

   'FINDING THE LENGTH
    a = 1
   Do While (gate = False)
      c = Mid$(op, a, 1)
      gt(a) = c
      If (c <> 1 And c <> 0) Then
         gate = True
      End If
      cnt = a
      a = a + 1
   Loop
   lent = cnt - 1
```

```
 'FINDING THE REVERSE
 a = 1
 For c = lent To 1 Step -1
   rv(a) = gt(c)
   a = a + 1
 Next c

'FINDING THE POWERS
l = 2
j = 1
kt = 2
pow(1) = rv(1)
ch(1) = pow(1)
lt = 2
chlen = 1
 Do While (lt <= lent)
    pow(j + 1) = rv(kt) * 2 ^ j
    ch(l) = pow(j + 1)
    chlen = chlen + 1
    j = j + 1
    kt = kt + 1
    l = l + 1
    lt = lt + 1
 Loop

 'FINDING THE SQUARES OF X
 store(1) = x Mod N
 temp = store(1)
 disp(1) = temp
 mtp = 2
 flmt = 1
 t = 2
 Do While ((flmt / 2) <= Val(y))
    store(mtp) = (temp ^ 2) Mod N
    temp = store(mtp)
    disp(t) = (store(mtp))
    mtp = mtp * 2
    flmt = mtp * 2
    t = t + 1
 Loop
 'ELIMINATIN ZEROS
 i1 = 1
 j1 = 2
 rvf(1) = pow(1)
 Do While (i1 < 15)
    If ch(i1) <> 0 Then
```

```
        rvf(j1) = ch(i1)
        j1 = j1 + 1
    End If
    i1 = i1 + 1
  Loop
  'CALLING VALUES
  pcnt = 1
  jc = 1
  For ic = 1 To chlen
    If ch(ic) <> 0 Then
        suma(jc) = (disp(ic))
        pcnt = pcnt + 1
        jc = jc + 1
    End If
  Next ic
  'RESULT
  res = 1
  Dim q1 As Integer
  res = (suma(1) * suma(2)) Mod N
  For q1 = 2 To (pcnt - 2)
    res = (res * suma(q1 + 1)) Mod N
  Next q1
End Sub
```

ENTER PLAIN TEXT

Dear Mary,

I am writing to you, to disclose to you the fact that my knowledge of Computer Science has taken a new dimension. I have now discovered through the course (post-graduate diploma in Computer Science) that I have just completed, that Cryptography is needed in many aspects of communication.

I shall explain more of that to you in my next mail.

Your sister,

Ada.

CIPHER TEXT

156

1074

782

946

861

485

485

1074

246

677

640

485

861

794

1071

1074

451

1132

459

246

1074

1132

678

1074

576

946

861

576

1074

576

1132

1074

354

1132

1066

CIPHERTEXT DECRYPTED INTO PLAINTEXT

ar Mary,

I am writing to you, to disclose to you the fact that my knowledge Computer Science has taken a new dimension. I have now scovered through the course (post-graduate diploma in Computer ience) that I have just completed, that Cryptography is needed in ny aspects of communication.

I shall explain more of that to you in my next mail.

Your sister,

Ada.