FINDING CASES OF

CIPHERTEXT EQUAL TO PLAINTEXT

IN THE RSA ALGORITHM

By

BEHNAZ SADR

Bachelor of Science in Electrical and Electronics

Engineering

Bilkent University

Ankara, Turkey

1992

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

FINDING CASES OF CIPHERTEXT EQUAL TO PLAINTEXT IN THE RSA ALGORITHM

Thesis Approved:

Dr. John M. Acken

Thesis Adviser

Dr. Carl Latino

Dr. Sohum Sohoni

Dr. Mark E. Payton

Dean of the Graduate College

ACKNOWLEDGMENTS

Here I would like to thank a few people who have been sources of encouragement; people who without them it would have been very difficult if at all possible to do this job.

Special thanks to:

My advisor: Dr. John M. Acken, who have been an amazing mentor and encourager for me. I truly appreciate your help in the past four years.

Members of my thesis committee: Dr. Sohum Sohoni and Dr. Carl Latino for accepting to be members of my thesis committee.

My mother: Mehri (Mary) Zandieh who encouraged me to continue with my studies and come back to school one more time after many years.

My dear husband: David (Reza) Nadri who made it possible by being there for me and picking up the slack whenever I was overwhelmed. David, I wouldn't be able to do this without you. Thank you my love.

Finally, I dedicate this thesis to my daughter, Shadi Nadri, a beautiful, intelligent, young lady who will always be my joy just like the meaning of her name, Shadi. You can help me next time on my PhD dissertation. We might get our PhD's together! Just joking!

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Information security is a current and heavily discussed topic. After all, this is the information age. Information is everywhere and at our fingertips. It needs to be protected from falling into the wrong hands. The Webster definition of the word “secure”, in its verb form is, “To relieve from exposure to danger; act to make safe against adverse contingencies” [1]. In other words, to secure information is to relieve it from exposure to danger. One method of doing just that is by changing the information beyond recognition. The act of changing or garbling a piece of information in a reversible fashion, (that is the message can be transferred back to its original form) is called encryption.

Encryption is a form of disguising or garbling information in a way that it won’t be recognized by anybody without a conscious effort to decrypt it. In order to encrypt and/or decrypt a message two things are necessary: a method and a key. When using the same key to encrypt and decrypt, the type of encryption is called symmetric key encryption. When using different keys for encryption and decryption, the type of encryption is called asymmetric key encryption. The concept of asymmetric encryption, or having different keys for encryption and decryption, was introduced by Whitfield Diffie and Martin Hellman. This innovation set the stage for the RSA algorithm that became very popular and has been used widely ever since. RSA is an acronym taken from the last names of the people who invented it, Ronald Rivest, Adi Shamir and Leonard Adleman. The RSA algorithm, with its asymmetric key method, was invented in 1977. It was the first attempt to realize Public Key Cryptosystems (PKC), the novel concept invented by Diffie and Hellman. PKC is another name for asymmetric encryption.

The task of studying and analyzing the RSA algorithm was the topic of my directed studies. While working on the material and finding good candidates for a small example, an interesting observation was made. There were cases in which after encrypting a message, the ciphertext would stay the same as the original message; as if, the encryption never occurred. This observation was the motivator for my thesis. In this thesis, these cases of plaintext equal ciphertext were researched and some other significant additional observations were made. These anomalies, the frequency of their occurrence, and their symmetrical behavior were characterized. Also, research was expanded to find out whether these cases are anomalies or regular occurrences. The result of the research shows that these cases are not anomalous and they occur no matter how large the numbers get.

The topics covered in the remaining chapters of this thesis are: Chapter II, a general background to communication security, encryption and the RSA algorithm; Chapter III, explanation of the research method used in the quest for a simple example of asymmetric key encryption; Chapter IV, observations demonstrating anomalies in RSA encryption; Chapter V, a summary of anomalous RSA behavior and the resulting conclusions.

CHAPTER II

BACKGROUND

2.0 Introduction

Information security is a very broad topic. One subtopic is communication security. At any given moment information is either in use, in motion or in storage. The security of this information may be compromised at any of these fronts. Communication security is a look at data in motion and the methods that may be employed to keep this moving data secure. This chapter is an overview of communication security with an emphasis on encryption.

2.1 Communication Security

When two or more parties want to communicate, sometimes they need to keep their communication secure. As mentioned in chapter I, to secure information is to relieve it from exposure to danger. There are different elements used to describe the parts of a secure communication system such as, sender, receiver, transmission media, message, encrypted message, encryption method, key, storage, etc. Also, there are different roles for participants in a secure system such as: sender, receiver, key creator, unauthorized listener, imposter, etc. These different roles can be referred to by using nicknames such as Alice, Bob, Eve and so on. Related to communication security is the issue of long term versus short term security which corresponds to the life span of information. There are goals in securing communication such as, user authentication, content authentication, confidentiality, integrity and non-repudiation. Encryption is an underlying technology that can be used for communication security and as such, an essential topic in any security system. Encryption is part of the solution to a number of problems related to achieving the goals of communication security.

2.1.1 Elements of a Secure Communication System

To talk about the elements of a secure communication system one needs to first define what a secure system is. The word “system” has one meaning that fits this content best, “a group of units so combined as to form a whole and to operate in unison.”[1] The word “secure” in its adjective form means, “free from danger or risk of loss; safe”. [1] To secure a system is to protect it from danger; to make it safe. Danger or risk of loss can come from inside or outside a system. One step of securing a system is to control access to the system. When it comes to people or entities that have access to a system, they can be divided into at least two categories: authorized users and non-authorized users. The concepts of access and authorization are different concepts related to a system. Let’s first look at an example about access. When a customer goes to a supermarket and walks around, he has access to the commodities that are on the shelves.

If the door to the office of the manager is open and his computer is on, and he has forgotten to logout of the accounting system, the customer now has access to whatever the manager can access! On the other hand, in the same supermarket, people don’t have access to the things that are out of reach, very high on the shelves, things that are in the storage area or locked somewhere. So having access to something, includes being “able to do” something rather than with being “allowed to” do it. Now let’s look at the concept of authorization using the above example. Having access to the accounting system of a supermarket doesn’t make a person an authorized user. If the manager in the last example forgets the only key to his office at home, he temporarily doesn’t have access to his office, but that doesn’t make him un-authorized to enter that office.

To give authorization to a user for doing a certain task is the job of an authority in a given system. There are different methods of authorization: There might be a list of authorized personnel or there might be some criteria that must be met in order to be authorized to do a task. Even the list system usually works based on some regulation that puts people on or off a list. Here are some scenarios regarding authorization and access. In the first scenario, Alice is authorized to buy a certain book 50% off only if she is one of the first one hundred applicants. Here, being one of the first one hundred applicants is the rule. The second scenario considers three situations. In an Online Classroom (OC) system, a teaching assistant (TA) Bob, is authorized to enter the grades for labs, tests, etc. by being assigned by the professor or the department as an authorized user. Bob doesn’t have access to the grades if he doesn’t have a computer or doesn’t remember his password. In comparing the concept of authorization and access in the same example consider a second situation. A teaching assistant isn’t authorized to see all the grades of a student for his other courses however, if a student leaves his OC account open and doesn’t logout, Bob (the TA!) will have access to all the grades but cannot change any of them! Now consider a third situation. If Charles who is a TA for another course, leaves his OC account logged on, Bob not only will have access to some grades, but would also be able to change them. For that matter anybody else, another student, a staff, a stranger can also go in and change the grades. The second and third situations demonstrate that Bob has access regardless of authorization.

Many times there is more focus on outsider danger rather than on insider danger. A non-trustworthy insider can bring a lot of serious damage to a system, worse than an outsider. The concept of walls or borders, being inside or outside a system, brings the idea of a key or a password to mind. Authentication means determining whether someone should be allowed inside. In a secure system, when the identity of a user is authenticated and he is found to be an authorized user, he is in. This is also called being “accepted”. So authenticating the identity of an authorized user means to let an authorized user in, “Accepting”, and keep the unauthorized user out, “Rejecting”. There are four situations that can happen with a combination of “Authorized”, “Non-authorized”, “Accepted” and “Rejected”. The first combination is when an authorized person is accepted. This is a desirable outcome. The second combination is when a non-authorized person is accepted. This is a non-desirable and dangerous outcome. The third combination is when an authorized person is rejected. This is also a non-desirable outcome but rather than being dangerous it’s annoying. The final combination is when a non-authorized person is rejected. This is something expected of a secure system; to keep the non-authorized people out. These four combinations are shown in Figure 2.1.

Authorized user Accepted Desirable outcome	Non-authorized user Accepted Non-desirable (dangerous) outcome
Authorized user Rejected annoying outcome	Non-authorized user Rejected Expected outcome

Figure 2.1 Authorization and Access

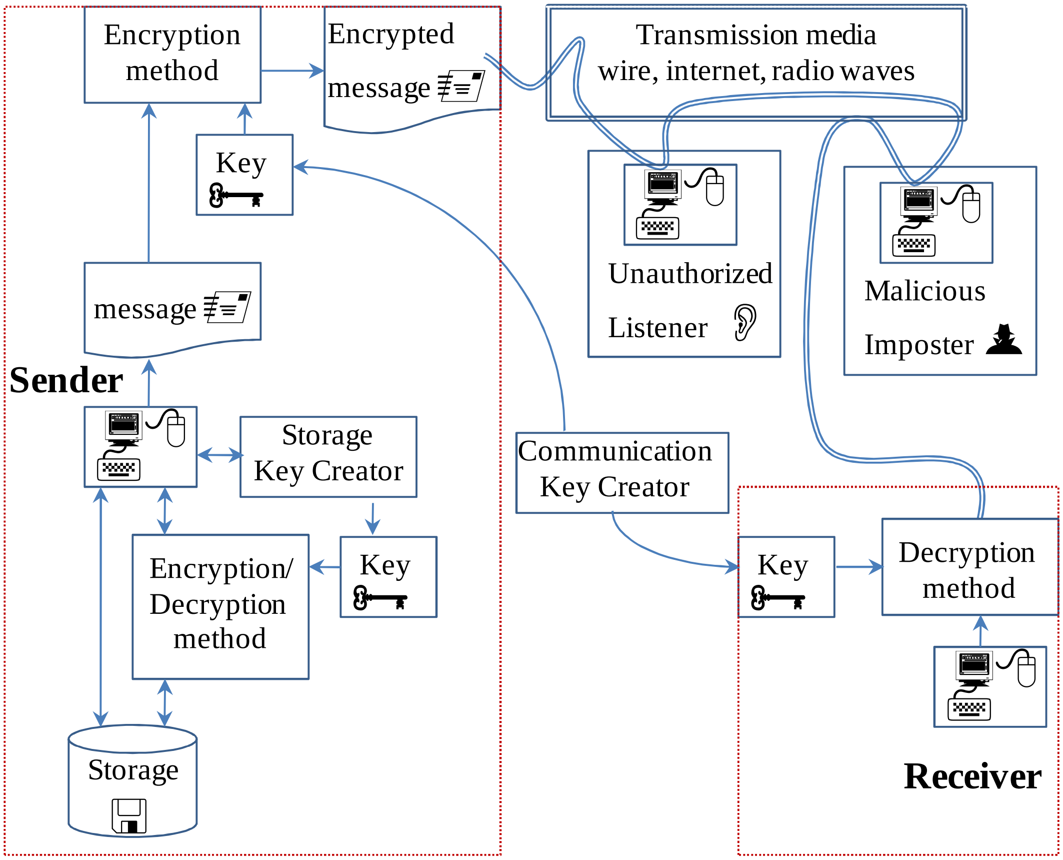


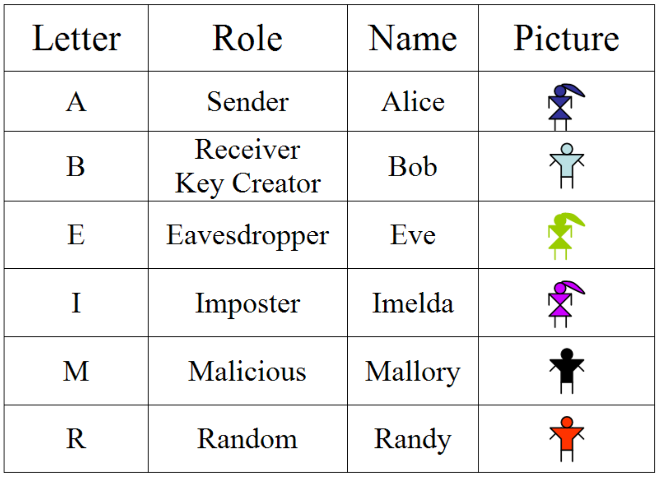
Figure 2.2 Elements of a Secure Communication System

After the brief look at the concepts of authorization and access in the context of security systems, let us consider the elements of a secure communication system. These elements include: Encrypted message, Decrypted message, Encryption method, Key(s), Key management, Message, sender, receiver, transmission media, non- authorized listener, Malicious imposter, Decryption method, encryption/decryption/key used for storage,

storage. Figure 2.2 shows the interrelation between these elements and shows where encryption may play a part in securing a communication system. There are three main areas in Figure 2.2 where encryption and decryption are shown: 1- On Sender side where a message is encrypted before it gets transmitted, 2- On Sender side where data is encrypted and decrypted in order to be safely stored and retrieved, 3- On Receiver side where the received message is decrypted and used. For each of these encryption or decryption methods there is a key involved. This key might be a shared secret key, which would be the same for sender and receiver or two different keys in case of asymmetric encryption. Key management is another element in a secure communication system. Keys need to be created and distributed according to the method used. In RSA, the public and the private key are both created by the receiver and the public key is publicly distributed to be used by any sender. The private key will stay at the receiving side, possibly stored in a storage unit using some sort of encryption. The private key will be used once a message is received and the decryption method will decrypt the message using the private key. At any part of the communication line there might be intruders, imposters or eavesdroppers. These are unauthorized entities who try to gain access to either the plaintext message, the ciphertext, to the key or any combination of these. The unauthorized entities goal is to passively or actively attack a secure system for different kinds of gain.

2.1.2 Security Players: Alice, Bob, etc.

In cryptography there has been a preference to use names instead of letters when describing different characters. For this reason instead of saying, “A sends the message m to B”, it is said: “Alice sends the message m to Bob”. So Alice and Bob were used instead of person A and person B. This effort throughout the years has resulted in a rather elaborate library of names representing different characters in cryptography and other fields that would benefit from this convention.



Letter	Role	Name	Picture
A	Sender	Alice	
B	Receiver Key Creator	Bob	
E	Eavesdropper	Eve	
I	Imposter	Imelda	
M	Malicious	Mallory	
R	Random	Randy	

Figure 2.3 Security Players Names and Roles

Figure 2.3 shows a list of some of these names [2][3][4][5]. Alice and Bob are probably the most famous characters in this list. Alice sends Bob private encrypted messages. Eve is an observer who is not authorized to read Alice’s message. Eve is an eavesdropper who might do her job anywhere in the communication line. The encryption must be strong enough that although Eve has intercepted the message, she cannot read it. Imelda wishes to send Bob a message as if the message is sent by Alice. Imelda is an imposter, someone who would pretend to be someone else. Mallory is a malicious character. The encryption system must be secure enough that Mallory's active efforts to circumvent it will fail. Randy is a random stranger who has just happened to intercept the communication and is not necessarily a negative character. A weak encryption is sufficient to keep information protected from Randy. Figure 2.4 shows some of these characters in action. As Alice is requesting access to communicate with Bob, Eve is listening to the conversation. Malory and Randy also have tapped into the communication line. At the other end, the communication line has reached Bob.

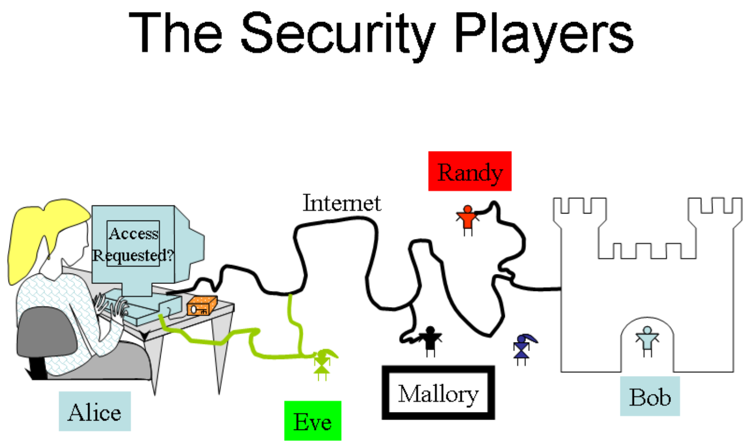


Figure 2.4 Security Players in Action

2.1.3 Long Term versus Short Term Secure

Securing information is a time dependant issue. The required strength and amount of effort applied to keep data secure depends on the sensitivity of the data as well as the duration in which that data needs to stay secure. An example from daily life is the requirement for encryption used for securing the data of a movie. The manufacturer needs to use a type of encryption that would be hard to break for the duration of the copyright. A copyright lasts at least 70 years [6]. The example of movie encryption is an example of long term security. For short term encryption let's consider the case of bidding on stock, which is a time sensitive matter. If an attacker gains access to the data related to a biding process, during the process, this is dangerous. If the attacker gains access to the same data after the sale, this will pose no problem. The example about stock bidding was an example of short term security. Another example for short term security needed for very sensitive data is a president's itinerary. There is a considerable amount of effort put into securing data regarding the whereabouts of a president. If an attacker can manage to break the code and gain access to the itinerary, before a president goes to a specific function, this would be considered a major security breech. On the other hand breaking the code after the president has attended that same function is not going to matter. This is the reason for considering the type of security in hand when determining the level of protection required.

2.1.4 Security Goals and Their Definitions

There are a number of goals when considering security in communication. These goals are authentication, confidentiality, integrity and non-repudiation. Authentication applies to both the users and the message. Authentication of a user means checking whether he is who he claims to be. Authentication of a message means checking the message and its originator to be authentic. Confidentiality is about protecting information from unauthorized access. Information integrity means protecting information from change. Non-repudiation means making sure the users can not deny their involvement in the communication. Many tools are available to achieve these goals; encryption is one of them.

2.1.5 Encryption or Cryptography in Context of Security

Encryption is a solution to a number of problems that may arise in reaching security goals and as such, an essential topic in any security system. The security goals, mentioned in the previous section, can be met from time to time using encryption. Here, some of these problems are addressed and how their solution might include or be related to encryption. Let's look at identity authentication as one of these goals. To authenticate whether a user is who he claims to be, his identity is checked. A user's identity may be checked by "what he knows", "what he has", and "what he is". Examples of "what he knows" are PIN (Personal Identification Number), DOB (Date of Birth) and SSN (Social Security Number). Examples of "what he has" are an identification card, a bank card or a physical key. Examples of biometric measures that show "what he is" include retina scan, voice print or fingerprint. The step of user identification depends upon encryption when the digital information of a user is stored or being transmitted and needs to be secure. For this encryption can be used. If the goal is authenticating a message, a hash from the user might be appended to the message. The hash value can be encrypted to prevent tampering. This is the basis for digital signatures. If the goal is confidentiality, hiding, or concealment then one of the methods used to achieve this goal is encryption. Information integrity, which is about protecting information from change, can be met using a hash function. One such hash process uses encryption to create an electronic signature. To meet the goal of non-repudiation and making sure the parties can not deny their involvement in the communication, digital signatures are used. Digital signature uses encryption.

2.1.6 Summary of Communication Security

Communication security revolves around securing data while in motion. In this section elements involved in a secure communication such as Sender, Receiver, Encrypted message, Decrypted message, Encryption method, Decryption method, Key(s), Key management, transmission media, storage, etc. were mentioned and the role of encryption in relation to these elements was considered. The security participants, their roles and a relatively new "nicknaming" system that has been used in communication and security literature recently, were glanced at. Securing information depends on the lifetime of that information and the sensitivity of it. The goals of securing a communication system and how encryption may be used in achieving some of these goals were explained.

2.2 Cryptography

Cryptography is “The science and art of transforming messages to make them secure and immune to attacks,” [7]. A more formal definition by Man Young Rhee in his book, Cryptography and Secure Communication is, “Cryptography is the study of cryptosystems by which privacy (confidentiality) and authentication of data can be ensured.” [8] Cryptography includes the study of encryption or ciphers. Later in this section some classical ciphers are mentioned, followed by some modern ciphers which are in use today.

2.2.1 Encryption an Old Ally or Foe?

Encryption is not a modern concept. In the olden days a message sent by one ruler to another was usually vital information. If the message fell into the wrong hands the lives of many were at stake. But what if the message was intercepted? Encryption is an ancient method used to conceal information and delaying the unauthorized person from seeing the plaintext. The issue of data security is not a matter of preference but a matter of life and death. As time passed, more and more sophisticated ways of hiding the data were found and little by little more modern and newer technology was applied. The nations and rulers, who had the technology to make better ciphers, had a huge advantage over people who were not able to hide whatever big or small amount of vital information they had. That is why encryption could be your ally or your foe depending on which side of the fence you were sitting. These sentences are not an attempt to consider encryption in a moral or philosophical context but to emphasize the importance of it. As it is said in my culture if a thief comes in with a light he can be choosy in what he steals. A strong tool in the hands of a foe makes him even stronger and more dangerous. This is true of any tool, for a weapon or for an encryption method.

2.2.2 Some Encryption Methods and Their Timeline

There are a few ways of categorizing encryption methods. Encryption methods or ciphers may be classical or modern, symmetric or asymmetric and a block cipher or a stream cipher. Modern ciphers are conceptually based on the classical ciphers. There are three main types of classical ciphers: transposition, substitution and product ciphers. All of these ciphers are symmetric methods. A symmetric encryption method is one that uses the same key for encryption and decryption. The act of transposing, substituting or creating a mixture of transposition and substitution, which gives you the product cipher, may be done manually or using modern technology. Symmetric algorithms, whether classic or modern, are divided into two types: block cipher and stream cipher. In his book, “Applied Cryptography”, Bruce Schneier gives a simple definition for the block and stream ciphers: “Block ciphers operate on blocks of plaintext and ciphertext – usually of 64 bits but sometimes longer. Stream ciphers operate on streams of plaintext and ciphertext one bit or byte (sometimes even one 32-bit word) at a time. With a block cipher, the same plaintext block will always encrypt to the same ciphertext block, using the same key. With a stream cipher the same plaintext bit or byte will encrypt to a different bit or byte every time it is encrypted.” [3] According to Forouzan, in his book “Data Communications and Networking”, a transposition cipher is, “a character-level encryption method in which the position of the character changes.” [7] and Schneier says, “In a transposition cipher the plaintext remains the same but the order of characters is shuffled around." [3] The next classical category of ciphers is a substitution cipher which Schneier defines as, "A substitution cipher is one in which each character in the plaintext is substituted for another character in the ciphertext." [3] There are different types of substitution cipher: monoalphabetic, homophonic, polyalphabetic and polygram which as the scope of this thesis is concerned, it will suffice to just mention their names. The next category of block ciphers is a product cipher. Rhee says, "A product cipher is a composition of two or more ciphers such that the ciphertext space of one cipher becomes the message plaintext space of the next. The combination of ciphers, called cascaded superencipherment, is done in such a manner that the final product is superior to either of its components." [8] In other words, when a combination of transposition and substitution is applied to a plaintext, in a cascaded manner, the resulting cipher is called a product cipher.

Here is a list of a few ciphers and their approximate dates and users. According to Rhee, "The oldest known transposition cipher is the Scytale cipher used by the ancient Greeks as early as 400 BC." [8] So Scytale would be a classic, block, transposition cipher. For all the classic ciphers being symmetrical is a given. The Caesar cipher, the most well known cipher, is an example of a classic substitution cipher which was used by Julius Caesar. In this cipher every letter of the alphabet in the plaintext is encrypted as the third letter following it in the alphabet (D would substitute for A). So the word AND is encrypted as DQG. A few other substitution ciphers are Beale cipher from 1880's, the Vigenère cipher from 16th century, the Rotor cipher from WWII and the one-time pad which is unbreakable. In 1917, the one-time pad was used for the first time in the Vernam cipher, designed by Gilbert Vernam. Two examples of product ciphers are the German ADFGVS cipher used in WWI and a modern product cipher is the famous Data Encryption Standard widely known as DES. Public Key Cryptosystems (PKC's) are asymmetric ciphers, use two different keys and are in the modern category of ciphers. Some examples of public key cryptosystems are algorithms such as RSA, ElGamal, Rabin and Elliptic Curves.

2.2.3 Symmetric versus Asymmetric Encryption

Encryption methods in general are divided into two main categories, symmetric-key and asymmetric-key. In symmetric-key cryptography, the same key is used by the sender for encryption, and by the receiver for decryption.

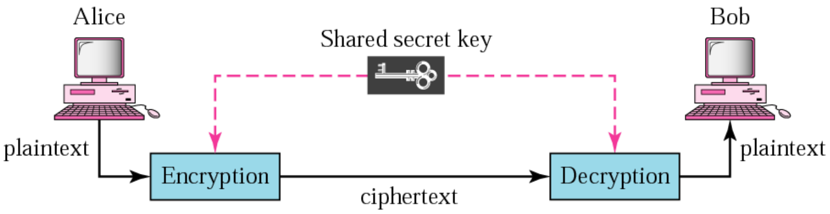


Figure 2.5 Symmetric-Key Cryptography

The key is called a shared secret key since it is a secret only known by the transmitting parties. This key needs to be kept secret because whoever has the key can both encrypt and decrypt the message. Due to the nature of this key, special care is used when delivering the key to the other party. Figure 2.5 shows a visual representation of symmetric key cryptography. Alice has a plaintext message to send to Bob. Alice uses the shared secret key to translate the plaintext to ciphertext. The ciphertext is now sent to Bob. Bob then uses the same secret key to decrypt the message and read the plaintext. The public key cryptosystems invented by Diffie and Hellman is a completely new system of cryptography. Until this point in history, all the encryption methods used one shared secret key. The public key cryptosystems introduced the idea of having two keys instead of one, hence the name asymmetric key encryption. Asymmetric-key cryptography, Figure 2.6, starts when a party wants to communicate with others and creates two keys. The first key will be publicly distributed; hence it is not a secret anymore and is called the public key. The second key is kept only by the initiator and is called a private key.

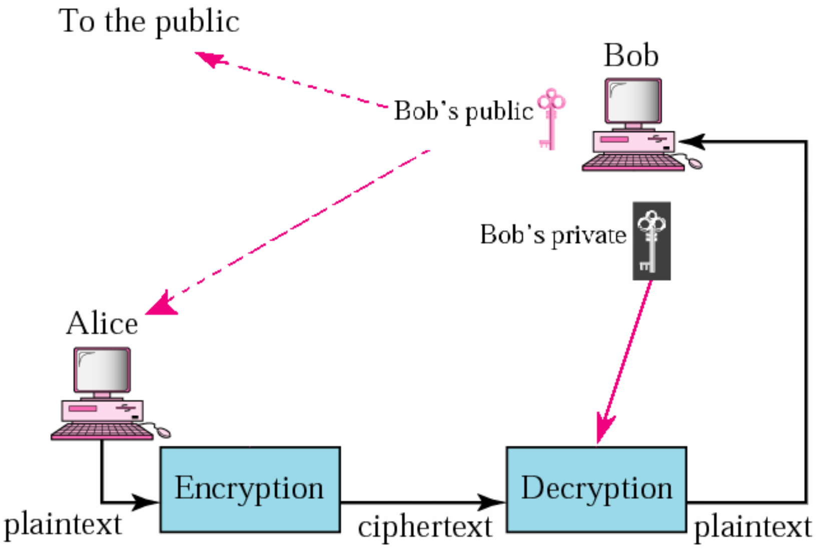


Figure 2.6 Asymmetric-Key Cryptography

Although anybody who has access to the public key can send encrypted messages, only those possessing the private key can decrypt these messages. For example, in Figure 2.7, Alice has a plaintext message to send to Bob. She needs a key. This key is a public key, created by Bob and broadcast to public. Alice uses Bob's public key to translate the plaintext to ciphertext. The ciphertext is now sent to Bob. Bob then uses his private key to decrypt the message and read the plaintext. There are different methods for implementing asymmetric encryption. One of these methods is the RSA algorithm introduced by Rivest, Shamir and Adleman in 1978[2].

2.3 RSA, an Asymmetric Encryption

When it comes to asymmetric encryption methods, RSA is the most widely known. In this section the historical context in which RSA was born is shown. Next the terminology used in this thesis to represent the formulas and limitations in RSA are explained. Later, the process of message preparation, encryption, decryption and message retrieval and the order they are applied in RSA are presented.

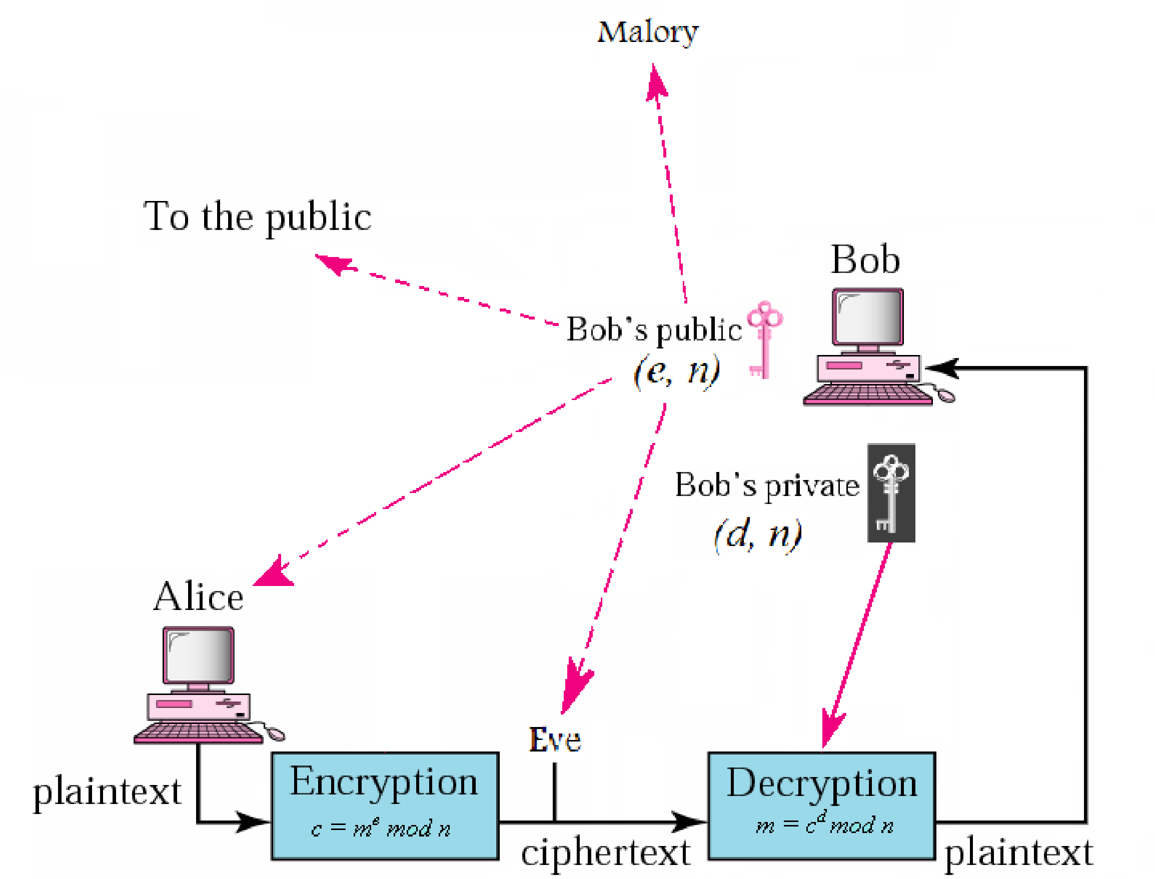


Figure 2.7 RSA as an Asymmetric-Key Cryptosystem

2.3.1 RSA: History

It was the year 1977. In January, Jimmy Carter became the 39th president of the United States. In May, the first Star Wars movie was released. In July, the New York Blackout happened. In August, Elvis Presley died at the age of 42. In April of this eventful year, three MIT students published a paper which changed the face of cryptography significantly. Their names were Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman and the title of their paper was “A method for Obtaining Digital Signatures and Public-Key Cryptosystems”. Today more than 30 years later, RSA is still widely used for cryptography and although it was the first implementation of a public-key cryptosystem (PKC), it has not retired or become obsolete.

2.3.2 RSA: Terms, Variables and, Limitations

There are two sets of equations in RSA which are explained in detail in section 2.3.3:

$\varphi(n) = (p-1)(q-1)$ *and* $d = (1 + k * \Phi(n)) / e$ for key generation and,
$c = m^e \bmod n$, *and* $m = c^d \bmod n$ for encryption and decryption.

The first set, top line, is used to create the keys for encryption and decryption and the second set, bottom line, is used to encrypt and decrypt the message. In Table 2.1 is the list of variables used in this paper and their definitions. There are a few limitations on these terms that will be mentioned throughout the chapter.

Table 2.1 Table of Terms, Definitions and Limitations

m is the message or plaintext in numerical form, $0 < m < n, m^e > n$
c is the encrypted message or ciphertext in numerical form
p & q are two prime numbers used to generate keys
n is the product of p & q
$\varphi(n)$ is the totient function of n
e is the public key in conjunction with n
d is the private key in conjunction with n
k is a positive integer used to generate d
$0 < m < n; m^e > n; 0 < c < n; c^d > n$ (Limitations)

2.3.3 RSA: Message Preparation, Formulas and Message Retrieval

RSA is a block cipher, which means the message is encrypted, one block at a time, using the same key. The message or plaintext is first prepared and then encrypted. This preparation includes encoding, blocking and padding. RSA keys are produced using a set of algebraic formulas and constraints. With the message prepared and the keys in hand, the RSA encryption and decryption formulas are ready to be applied. Decoding is done after decryption in order to retrieve the original message.

Message preparation is a necessary step in RSA encryption. A plaintext goes through a number of steps before it is ready to be encrypted. RSA is an encryption method able to encrypt integers and integers only. The RSA formulas are based on modular arithmetic. Therefore, to use RSA one needs to apply it when the plaintext is in integer form or can be interpreted as such. Figure 2.8 is a diagram which includes the encoding, encrypting, decrypting and, decoding in it as four boxes and shows how the message changes as it passes through. The message preparation starts with turning or transforming the text into a number, a sequence of digits. This sequence of digits corresponds to a sequence of bits which will be divided into blocks and padded accordingly. There are different standards used for blocking and padding depending on where RSA is being used.[3] For example, Privacy Enhanced Mail (PEM) uses RSA. According to Schneier, “PEM is the Internet Privacy-Enhanced Mail Standard, adopted by the Internet Architecture Board (IAB) to provide secure electronic mail over the Internet,” and, “PEM also supports public-key certificates for key management, using the RSA algorithm (key length up to 1024 bits) and the X.509 standard for certificate structure.” [3] Pretty Good Privacy (PGP) is another security program which uses RSA (key length up to 2047 bits) for key management and digital signatures. The Public-Key Cryptography Standards (PKCS) are RSA Data Security, Inc.’s “attempt to provide an industry standard interface for public-key cryptography.”[3] The combination of encoding, blocking and padding will make the message suitable for RSA encryption.

Inside the computer, the original message which is in text form corresponds to an ASCII equivalent. Each letter of the text has an ASCII representation and an ASCII code.

Turning the whole text into its ASCII form is a way of encoding the text. The string of characters is now a string of bits which can be considered the representation of a very large integer. After the message is encoded, it needs to be turned to blocks. Blocking a message means cutting it into smaller blocks of a given size. Each block can be turned into an integer *m* less than *n.* After the blocking is done, there will usually be a left over number of digits which are not enough to form a complete block. This is where padding is done. Padding a message means adding enough bits to complete a block and reach a certain block size. In other words, if the leftover bits are not enough to make a complete block, an adequate number of bits will be added to them. Sometimes padding must be done in order to increase the value of m by a known amount to be able to satisfy the inequality $m^e > n$. These padding schemes must be done according to the considerations of blocking and padding standards. Blocking and padding were mentioned here just to give the reader a general understanding of the process. The details and standards will not de discussed in this paper. Once the steps of message preparation are completed, the message is ready to be encrypted and then transmitted.

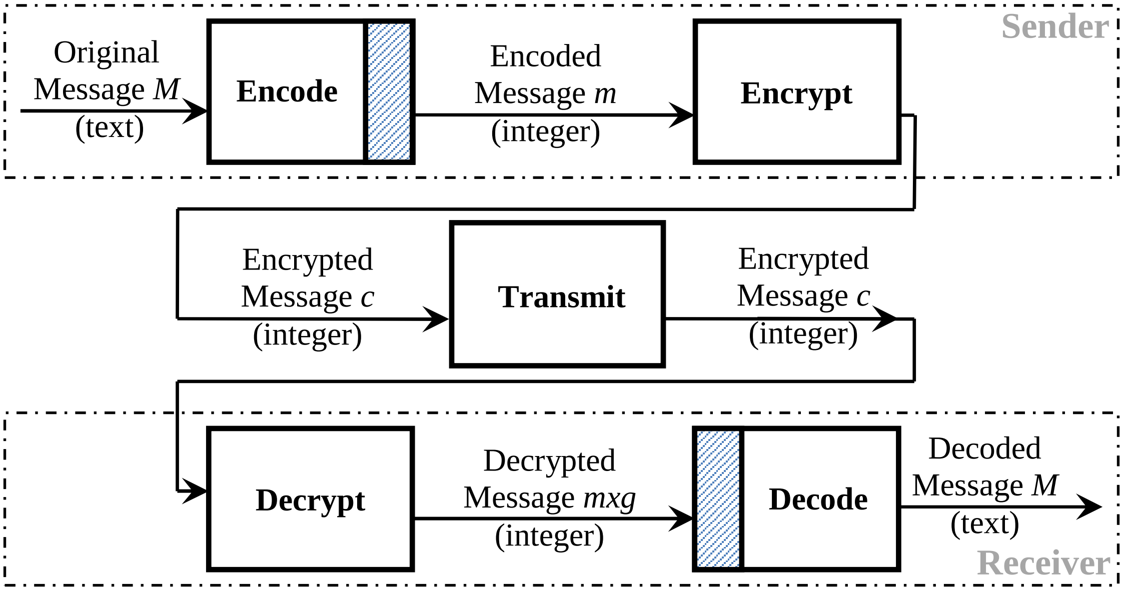

Sender
Original Message M (text)
Encode
Encoded Message m (integer)
Encrypt
Encrypted Message c (integer)
Transmit
Encrypted Message c (integer)
Decrypt
Decrypted Message mxg (integer)
Decode
Decoded Message M (text)
Receiver

Figure 2.8 Communication Block Diagram

There are two sets of equations in RSA. The first set is used to create *n, e* and *d,* the keys to be used for encryption and decryption (equations 2.1 through 2.3). The second set is used to encrypt and decrypt the message (equations 2.4 and 2.5). RSA uses modular arithmetic in these calculations and the following steps to generate the public and the private keys:

1. Choose two prime numbers, preferably large. These are called *p* and *q.* The reason for choosing large primes is explained in step 2.
2. Calculate their product. This is *n.* The product of two large primes is a very large number. The reason for choosing *p* and *q* to be large is due to the difficulty of factoring large n. This is known as the factorization problem and as of yet does not have an easy

solution. Formally this problem has a complexity called np complete. This means as the problem size grows the time to solve the problem grows exponentially.

3. Calculate the totient of *p* and *q*. Totient of a semiprime number by definition is:

$$\varphi(n) = (p-1)(q-1) \qquad 2.1$$

4. Choose a number *e* less than *n* that is coprime with the totient and satisfies the relation $m^e > n$. To be coprime with another number means the two numbers have no common factors other than 1. For example 38 is coprime with 15 because they have no common factors other than 1.

5. Find a number *d* such that *e* and *d* can be multiplicative inverses in modulus *n*. This means that the product of *d* and *e* in modulus *n* must be 1. Another way of saying this is, there exists an integer *k* where *d* and *e* satisfy the equation:

$$d = (1 + k * \Phi(n)) / e \qquad 2.2$$

or

$$d * e \equiv 1 \pmod{\Phi(n)} \qquad 2.3$$

Encryption and decryption formulas are the second set of formulas used in RSA. These formulas are equation 2.4 and 2.5. When Alice wants to transmit a message M to Bob, her message will pass through five conceptual blocks before it reaches Bob (Figure 2.8). She first encodes M, which is a text, and turns it into *m* which is an integer. Then, she uses the public key *(e, n)* to encrypt the message m, transmits the encrypted message c. She uses 2.4 for encryption.

$$c = m^e \bmod n, \qquad 2.4$$

When Bob receives the ciphertext, c, he uses the private key *(d, n)* to decrypt the ciphertext and get the original message *m* back.

$$m = c^d \bmod n \qquad 2.5$$

The encoded message *m* can in turn be decoded back to M. This process is illustrated in the Communication Block Diagram, Figure 2.8.

Decoding happens at the other end of the communication block. After the steps of message preparation are done, the result is a set of integers, *m1, m2, m3,…* all less than *n.* These integers will be encrypted one by one into their counterparts *c1, c2, c3, …* . This set of encrypted integers will be transmitted and reach the receiving side. At the receiving side, they will be decrypted and decoded. Decoding, the reverse of encoding, is the act of turning the decrypted message which is in an encoded form into text form. In this way, the receiver will use decryption and decoding to reverse what was done on the sender side and gets back the original message.

2.3.4 RSA: Comments

RSA claims that due to the difficulty of factoring large *n* breaking the RSA code is infeasible. The size of *n* required to support this claim depend on the computational power available at any given time. The higher the computational power, the larger is the value of *n* necessary to withstand attacks. There are two places in the original RSA paper where the authors ask the readers to break their method. The First place is in "Computing D in some other way" where "The reader is challenged to find a way to "break'" their method. The second place is in conclusion where "The reader is urged to find a way to 'break' the system".

There have been a few attacks on the RSA algorithm. One is called "The timing attack" and is based on the fact that the algorithm takes different amounts of time to decrypt different inputs. This can be used by Eve the eavesdropper and give her an idea about the key. This depends on the attacker, Eve, having access to *c*, the encrypted message. This attack is based on finding the private key bit by bit and the fact that finding the entire key may start with bit 0 of it and repeating the same procedure until the key is completely found.

2.4 Chapter II Summary

The broad topic of Information Security includes protecting data in use, in motion and in storage. Communication security is the security of data in motion. The use for encryption emphasized in this thesis is as a tool used in communication security. The background for this thesis is different types and uses for encryption. The two primary types of encryption used for communication are the symmetric key and the asymmetric key encryption. Symmetric Key encryption uses the same key for encryption and decryption whereas in Asymmetric encryption there are two keys involved: a public key and a private key. One of the most popular asymmetric encryption methods is RSA. In this chapter RSA, as an example of asymmetric encryption, was explained and message preparation, key generation, encryption and decryption formulas were overviewed.

RSA is an encryption method used in the digital age. Before Diffie and Hellman introduced Public Key Cryptosystems and the idea of having two keys for an encryption method, this was either unheard of or, not been put in practice. It was a group of young MIT students who took the challenge upon themselves to find the correct set of algebraic functions that would do the job. This is how RSA was born. Table 2.2 is a chart which is an attempt to put all of the formulas used for key generation, encryption and decryption in one place. Some definitions, usages, conditions and limitations are also included. The Summary of RSA Terms and Formulas table, Table 2.2, contains the terms and variables used in this paper as well as some definitions, formulas and limitations. In different literature, different letters are used for the keys, the message, the ciphertext, etc. So although the letters used might be different in different literature, the names and definitions are consistent. The condition / limitation column is either the range of validity for a given term or the formulas that needs to be satisfied and impose a restriction on that term. The terms and variables for RSA formulas are: A semiprime number also referred to as 2-almost-prime is the product of two primes. [9] Two numbers are coprime when they have no common factors other than one. The variables *p* and *q* are two prime numbers used in key generation. The variable n is the product of p and q and hence a semiprime number. The variable n is used in RSA as the modulus for encryption and decryption. The function phi of n or totient of n, $\varphi(n)$, is the Euler's totient of n . The Euler's totient of n by definition is the number of integers less than n and coprime with it. [9] For a semiprime number, n, this is, $\varphi(n) =(p-1)(q-1)$. The ordered pair (e, n) is the public key which is used for encryption. The variable k is a positive integer that evenly divides $(d * e) - 1$. The ordered pair (d, n), is the private key which is used for decryption. The relationship between the public and the private key is that they are both coprime with n and their product in modulus $\varphi(n)$ is 1.

Table 2.2 Summary of RSA Terms and Formulas

Letter	Name	Definition / Use / Formula	Condition / Limitation
	semiprime	product of two primes also called 2-almost-prime	
	coprime	two numbers with no common factors other than one	
p, q		key generation	prime numbers
n		a semiprime number, modulus for encryption and decryption $n = p * q$	should be large enough to make factorization difficult
$\varphi(n)$	totient of n	Euler's Totient function: Number of integers less than n and coprime with it. For a semiprime number n: $\varphi(n) =(p-1)(q-1)$ modulus for key generation	
(e,n)	public key	encryption key	both e and d are coprime with n; e, d, k and $\varphi(n)$ satisfy: $d = (1 + k * \Phi(n))/e$ or $d * e \equiv 1$ $(mod\ \varphi(n))$
(d,n)	private key	decryption key	
k		divisor of $(d * e) - 1$	
m	message	cleartext or plaintext	$0 < m < n$, $m^e > n$
c	ciphertext	encrypted message or cryptotext $c = m^e \bmod n$.	$0 < c < n$, $c^d > n$
mxg	decrypted message	$mxg = c^d \bmod n$	ideally $mxg = m$

Another way of saying this is: there exists a positive integer k, which divides one less than the product of the two keys, evenly. The variable m is the message, or plaintext. It has to be an integer smaller than n . The variable c is the ciphertext, also known as encrypted message or cryptotext. The formula for encryption is $c = m^e \bmod n$. mxg is the decrypted message and is calculated as: $mxg = c^d \bmod n$. The decrypted message is

ideally, exactly equal to the original message. Both m and c have to be smaller than n for the modular arithmetic to work properly in the context of encryption. Specifically, if $m > n$ then multiple message values encrypt to the same ciphertext value c. There are two inequalities that need to be satisfied as well:
$m^e > n$ *and* $c^d > n$. If $m^e < n$ then no encryption is done. The description of the RSA asymmetric algorithm may not present an intuitive understanding of the process without an example. For students a good example of RSA encryption is very instructive. The next chapter describes a search for a good example.

CHAPTER III

RESEARCH: Quest for a Simple Example

3.0 Introduction

Teaching about the RSA algorithm is enhanced by using a simple example. As described in chapter II, the RSA algorithm is a mathematically difficult concept. RSA works with modular arithmetic and exponentiation. Both of these concepts are easier to grasp using smaller examples. Searching the literature disclosed that even the smallest published examples were too large for simple hand calculations. This research started as a search for a small example. During the search some unexpected results were observed. Additional tools were developed to spot the unexpected results. In the process of research, a few tables were developed to be used as tools. These tools can be used as lookup tables, both for key generation and for encryption and decryption. At the end of the chapter, two simple small examples will be presented using these look up tables. These examples can help a student of the RSA algorithm to learn and analyze it.

3.1 Mechanics of the Simple Example: Why a Simple Example?

Studying and making observations on a mathematical concept calls for using simple examples; examples that are simple enough that they can be done either manually or using standard computational tools. However, the examples must be large or complex enough to avoid being trivial. The RSA algorithm uses exponentiation and modular arithmetic. Integer exponentiation creates numbers that are very large. This means it is even more important to use very small, double-digit values for p and q. The original RSA paper [2], uses the primes 47 and 59 for p and q. The public and private keys used are (e, n) = (17, 2773) and (d, n) = (157, 2773) respectively. The totient is 2668. This n is relatively large for hand calculation. This research however, started searching for RSA encryption examples with primes less than 20 for p and q and then, extended to include primes between 20 and 30. The main reason for extending the domain to include numbers between 20 and 30 was interesting observations which will be explained later in chapter IV.

3.1.1 The First Set of Key Generation and Encryption Tools

The first step in the search for a small example was to develop a chart to be used as a tool to calculate and display a simple RSA encryption/decryption set of messages. Next, a chart was created as a tool to generate keys, small public and private keys, to be used in RSA encryption. These first two charts, developed with my advisor, were the first set of tools and the starting point of this research. The Tiny Key Encryption Calculator was the first tool developed. The Tiny Key Encryption Table, Table 3.1, is a picture of the Tiny Key Encryption Calculator.

This encryption calculator encrypts and decrypts values of m between two and $n - 1$, inclusive, for any given *e*, *n* and their corresponding *d*. The imposed limitation on *m*, ($1 < m < n$) is due to the properties of modular arithmetic as described in chapter II.

Table 3.1 Tiny Key Encryption Table

e = 11 *n* = 15 *d* = 3

cleartext		ciphertext		decrypted text
m	*m*^2 mod(*n*)	*c*	*c*^2mod(*n*)	*mxg*
2	4	8	4	2
3	9	12	9	3
4	1	4	1	4
5	10	5	10	5
6	6	6	6	6
7	4	13	4	7
8	4	2	4	8
9	6	9	6	9
10	10	10	10	10
11	1	11	1	11
12	9	3	9	12
13	4	7	4	13
14	1	14	1	14

In the top row portion of Table 3.1 there is a row of information regarding the public and private keys used for a particular instance of the Tiny Key Encryption Calculator. In this case the public key is $(e, n) = (11, 15)$ and the private key is $(d, n) = (3, 15)$. Below the row including *e*, *n* and *d* values is a table of five columns. The five columns correspond to the original cleartext, an intermediate calculation for encryption, the ciphertext, an intermediate calculation for decryption and the decrypted message. The two intermediate calculation columns were either hidden or omitted in later encryption calculators. The horizontally shaded areas in Table 3.1 are rows of calculation and each line identifies a case where the plaintext equals the ciphertext. For example, one of these shaded rows corresponds to a cleartext *m* equal to 4. When cleartext is 4, the ciphertext is also calculated to be 4. The decryption of 4 is also equal to 4. In Table 3.1, such cases of cleartext equal to ciphertext is true for values of *m* equal to: 4, 5, 6, 9, 10, 11 and 14. Each of these values of *m* will produce values of *c*, equal to *m* respectively. When the encryption of a message *m* produces an equal value of *c*, a hole has occurred. This doesn’t mean that, the value of *m* is a hole but a hole happens at that *m*, as if that value falls through without getting encrypted. The last row of this chart, $m = n$-1, always corresponds to a hole. Since this happens regardless of the value of *n*, this row is omitted in later encryption calculators. For explanation on the limits of *m*, see chapter II. The suspected cause for these holes was that this was just a bad example. Therefore the search for a small example needed to be extended.

The second step in the search for a small example was to develop a chart to be used as a tool to generate and display the keys for different values of *n*, *e* and *k*. This resulted in the creation of the Tiny Samples Key Generator. The Tiny Samples Key Table, Table 3.2, is a picture of the Tiny Samples Key Generator. Table 3.2, is a collection of tiny *p* and *q* values with their corresponding *n* and totient calculated. For every *p* and *q* combination, different values of *e* were considered and corresponding *d* values, were calculated and displayed.

Table 3.2. Tiny Samples Key Table

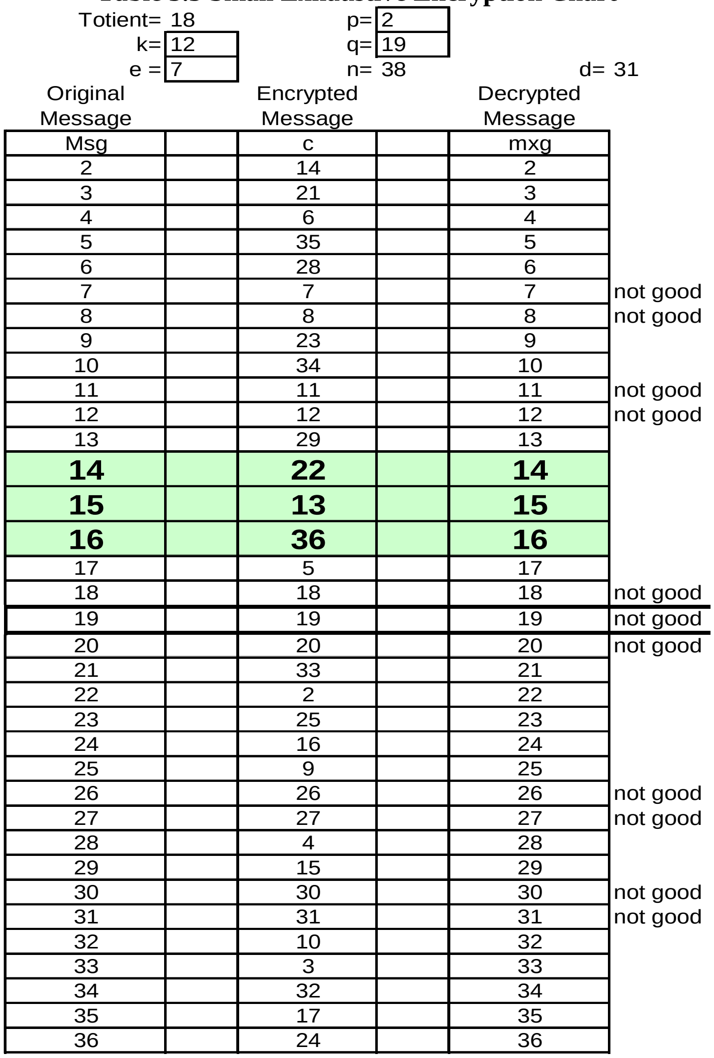
p	2	2	2	2	**3**	3	3	5	5
q	3	5	7	11	**5**	7	11	7	11
n	6	10	14	22	**15**	21	33	35	55
totient	2	4	6	10	**8**	12	20	24	40
e	3	3	5	3	3	5	3	5	3
k	1	2	4	2	4	2	1	1	2
d	1	3	5	7	11	5	7	5	27
e	5	5	7	7	5	7	7	7	7
k	2	1	8	2	3	4	8	2	4
d	1	1	7	3	5	7	23	7	23
e	7	7	11	11	7	11	11	11	11
k	3	5	9	1	6	10	6	5	3
d	1	3	5	1	7	11	11	11	11
e	11	11	13	13	**11**	13	13	13	13
k	5	8	2	9	**4**	1	11	7	12
d	1	3	1	7	**3**	1	17	13	37

In this table, the horizontal shading of *e* values is done to separate each group of *e*, *k*, *d*, from the next group. On rows one and two, tiny primes were chosen and on rows three and four, the product of *p* and *q*, *n*, as well as their totient was calculated. There are groups of three rows under the top four rows. These sets of rows contain values for *e* and their corresponding *d* using a specific *k*. The leading *e*, is normally 3 as the first prime to consider. Then the list of primes greater than 3, whose values were coprime with the totient were considered and displayed in the column below. The reason for not starting with 2 as the first prime candidate for *e* is that the totient of *n*, which is a semiprime number, is always even and not coprime with 2. Let's look at an example from Table 3.2 and consider the case for *p* and *q* being 3 and 5, respectively. This example is the vertically shaded area and some cells in that column are in bold for emphasis. For *p* and *q* values of 3 and 5, *n* is 15 and totient is 8, all in bold. So *e* can start with 3 and continue with 5, 7 and 11. Here, you may see if *e* is chosen to be 11, for a value of *k* equals to 4, *d* is calculated to be 3. The *e*, *k*, *d* combination of 11, 4, 3 is in bold in the bottom of that column. The public key is (*e*, *n*) = (11, 15) and the private key is (*d*, *n*) = (3, 15). This set of keys is the set that was used in Table 3.1.

3.1.2. Developing the Small Exhaustive Encryption Chart: Holes!

The Small Key Encryption Calculator is a tool for encrypting message *m* under a given set of values for *p*, *q*, *k* and *e*. Table 3.3, the Small Exhaustive Encryption Chart, is an instance of the Small Key Encryption Calculator with *p*, *q*, *k* and e being 2, 19, 12 and 7. This new Encryption Calculator is an improved version of the Tiny Key Encryption Calculator. On the top, instead of showing only *n*, *e*, & *d*, all of the following values are displayed: *p, q, n, φ*(*n*), *e, k* and *d*. Out of the above seven values, four are inputs to this excel worksheet and three are calculated based on them. In other words, given *p, q, e* and *k*, the values of *n, φ*(*n*) and *d* are calculated. Here, the idea of introducing a vector which would contain all of these values, an ordered septuple, came to mind. The seven parameters for RSA which are: *p, q, n, φ, e, k* and *d*, can be kept in order and be treated as an ordered septuple or a vector of dimension seven. In this way, they can be referred to as a single vector (*p, q, n, φ, e, k, d*). Some of the elements of this vector are dependant on the other elements. For example a vector for the primes 11 and 13, with *e* being 23 and *k* being 9 can be calculated and shown as (11, 13, 143, 120, 23, 9, 47).

Table 3.3 Small Exhaustive Encryption Chart



Totient= 18 p= 2
k= 12 q= 19
e = 7 n= 38 d= 31

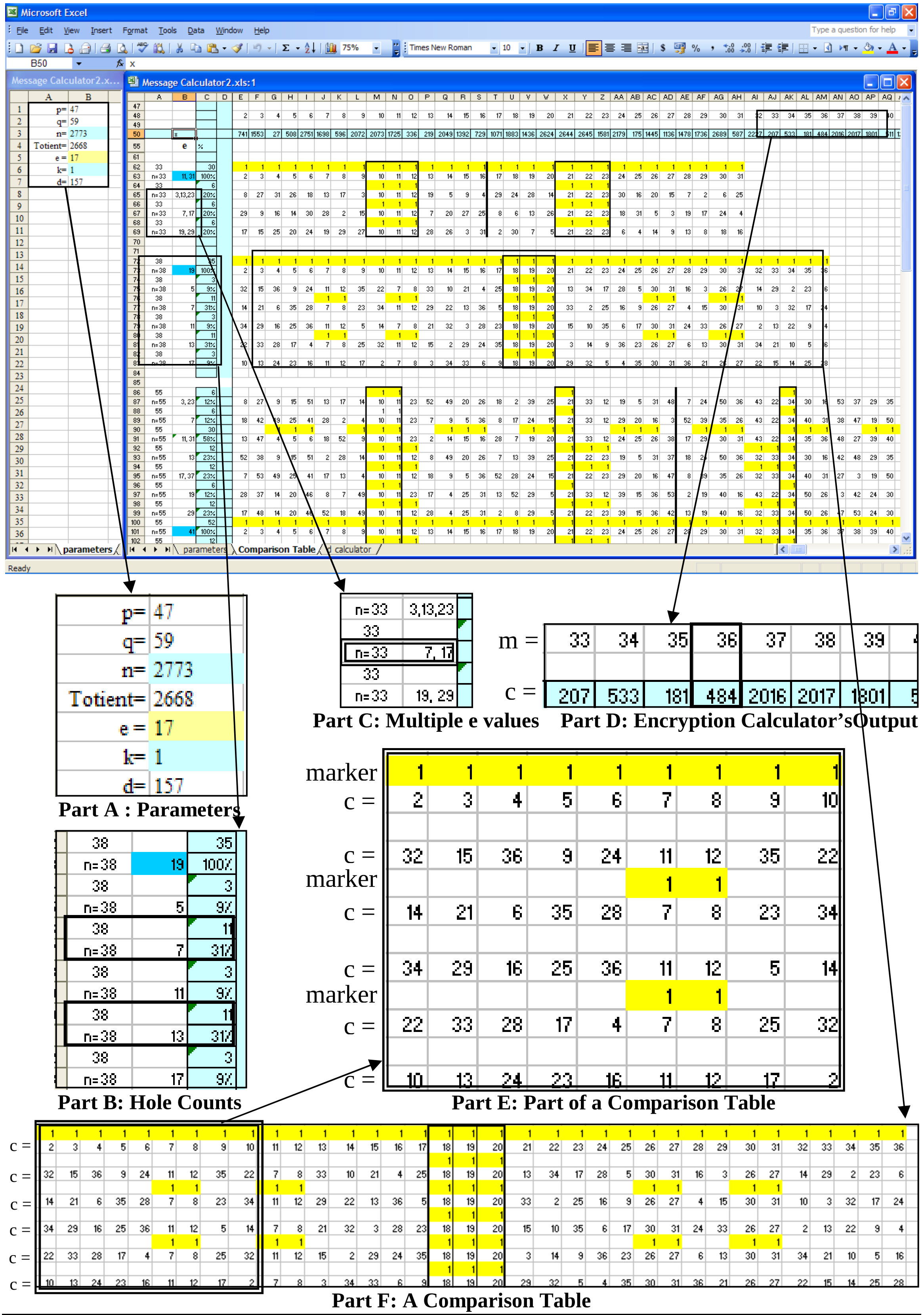
Original Message		Encrypted Message		Decrypted Message	
Msg		c		mxg	
2		14		2	
3		21		3	
4		6		4	
5		35		5	
6		28		6	
7		7		7	not good
8		8		8	not good
9		23		9	
10		34		10	
11		11		11	not good
12		12		12	not good
13		29		13	
14		**22**		**14**	
15		**13**		**15**	
16		**36**		**16**	
17		5		17	
18		18		18	not good
19		19		19	not good
20		20		20	not good
21		33		21	
22		2		22	
23		25		23	
24		16		24	
25		9		25	
26		26		26	not good
27		27		27	not good
28		4		28	
29		15		29	
30		30		30	not good
31		31		31	not good
32		10		32	
33		3		33	
34		32		34	
35		17		35	
36		24		36	

The Small Key Encryption Calculator has two main sections. The top portion of it has the information of a given septuple (*p, q, n, φ, e, k, d*). Below that is a table of encrypted and decrypted values of a range of messages, *m*, between 2 and *n*-2. The Table 3.3 is an instance of the Small Key Encryption Calculator, for a specific septuple (2, 19, 38, 18, 7, 12, 31). The top portion contains information of a septuple and this information is used for the calculation of the ciphertext and the decrypted messages.

The description for the three columns of data follows: The leftmost column under the heading, "Original Message" or "Msg" is an exhaustive list of all values of *m* between 2 and *n*-2, inclusive. The column under the heading "Encrypted Message" or "*c*" is the encrypted values for each message. For example in Table 3.3 the septuple (2, 19, 38, 18, 7, 12, 31) results in the values 14, 15 and 16 to be encrypted as 22, 13 and 36 respectively, as shown in the shaded area. The column under the heading "Decrypted Message" or *mxg* is the calculated decryption of column "*c*" using the decryption key, *d*. These numbers are calculated to show that after encrypting and decrypting a message, the end result is equal to the original message. In Table 3.3, the values 14, 15 and 16, which were encrypted as 22, 13 and 36 respectively, are decrypted to 14, 15 and 16; the decrypted message is equal to the original message, as shown in the shaded area. The rightmost column is a column of labels or markers. Looking at the rows which are labeled, "not good" shows that in these rows, the encrypted value of a message is equal to the original value of that message. Or to put it in another way, ciphertext equals cleartext. These are examples of where the RSA encryption algorithm has failed to encrypt. For example, when the value of message is 11, the encrypted value is also 11 and the row is marked: not good. The Small Exhaustive Encryption Chart, Table 3.3, is an instance of the Small Exhaustive Encryption Calculator for a given septuple, (2, 19, 38, 18, 7, 12, 31).

3.1.3 Developing the Message Calculator

In order to compare encrypted messages from various different septuples, the Message Calculator was developed. Figure 3.1, is a screenshot of the Message Calculator. The Message Calculator is an Excel workbook that has three portions. These portions are called: 1) Parameters, 2) Encryption Calculator and, 3) Comparison Tables. The Parameters portion is located in a worksheet called, "Parameters", and it takes values of *p*, *q*, *e* and *k* to calculate the corresponding values of *n*, totient and *d* in return. This is shown in Part A, of Figure 3.1. The Encryption Calculator, which is in a hidden area, inputs a septuple from Parameters and calculates the encrypted values of *m*. These encrypted values will be displayed in a row right below their corresponding plaintext values. In this row the encrypted values are ready to be harvested and pasted where needed. A portion of this row is blown up in Figure 3.1, part D. Here you can see the encryption of $m = 36$ is $c = 484$, under a septuple (47, 59, 2773, 17, 1, 157). The values of the septuple are shown in Figure 3.1 part A. The Comparison Tables portion is a set of tables where these encrypted values are placed. Each individual Comparison Table contains information related to the same value of *n*; what changes within that table is *e* and *m*. There are two rows for each *e* value, the bottom row is a collection of encrypted values and the top row is a marker for the holes. Anywhere there is a hole, a label "1," marks it. These 1's are added to be used for calculation of the total number of holes per case. If there are two or more septuples or vectors that create the same encrypted values, their corresponding *e* values are written together in the column for *e*. For example: in Figure 3.1 Part C, for $n = 33$, $e = 7$ and $e = 17$ create the same encrypted values. Part B of Figure 3.1 shows a blown up version of the row headings of the comparison table for $n = 38$. For a value of $n = 38$ and $e = 7$, there are 11 holes. This number of holes is 31 percent of the total number of encrypted messages and for $e = 13$, there are 11 holes as well.



p=	47
q=	59
n=	2773
Totient=	2668
e =	17
k=	1
d=	157

Part A : Parameters

n=33	3,13,23	
33		
n=33	7, 17	
33		
n=33	19, 29	

Part C: Multiple e values

m =	33	34	35	36	37	38	39
c =	207	533	181	484	2016	2017	1801

Part D: Encryption Calculator's Output

38		35
n=38	19	100%
38		3
n=38	5	9%
38		11
n=38	7	31%
38		3
n=38	11	9%
38		11
n=38	13	31%
38		3
n=38	17	9%

Part B: Hole Counts

marker	1	1	1	1	1	1	1	1	1
c =	2	3	4	5	6	7	8	9	10
c =	32	15	36	9	24	11	12	35	22
marker						1	1		
c =	14	21	6	35	28	7	8	23	34
c =	34	29	16	25	36	11	12	5	14
marker						1	1		
c =	22	33	28	17	4	7	8	25	32
c =	10	13	24	23	16	11	12	17	2

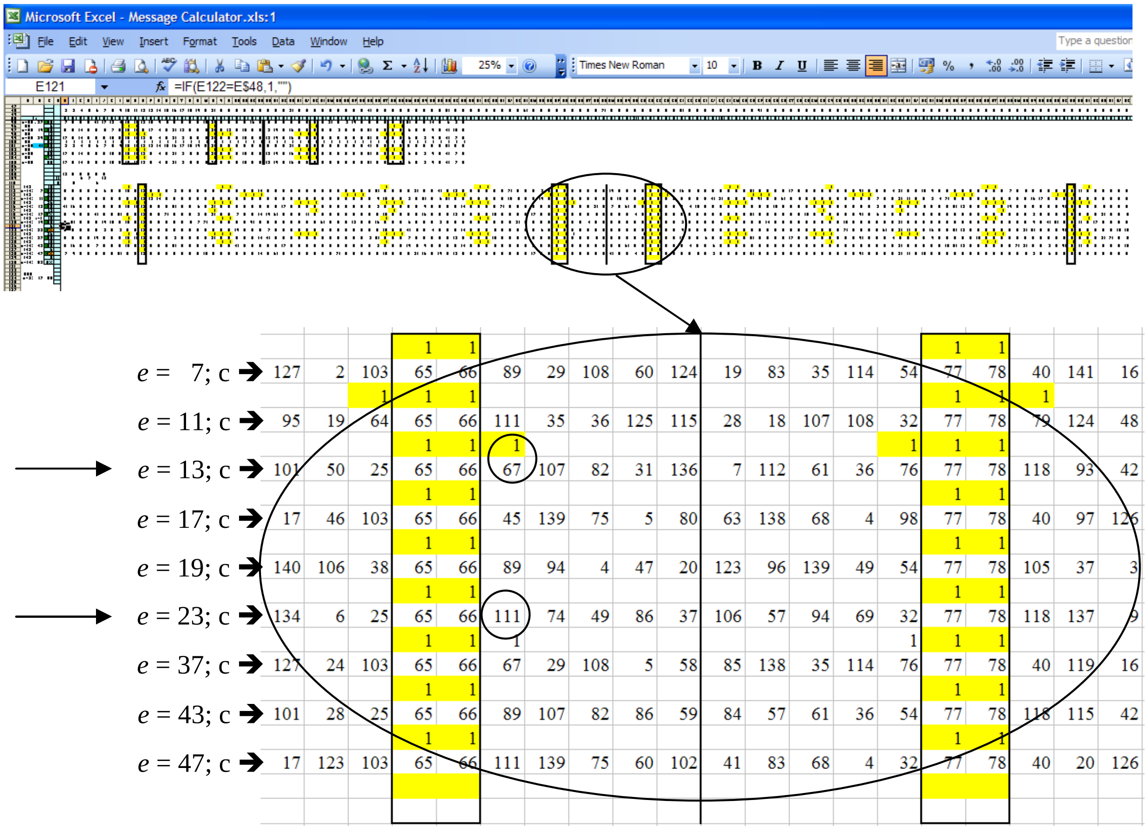
Part E: Part of a Comparison Table

	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c =	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
																	1	1	1																
c =	32	15	36	9	24	11	12	35	22	7	8	33	10	21	4	25	18	19	20	13	34	17	28	5	30	31	16	3	26	27	14	29	2	23	6
						1	1			1	1						1	1	1						1	1			1	1					
c =	14	21	6	35	28	7	8	23	34	11	12	29	22	13	36	5	18	19	20	33	2	25	16	9	26	27	4	15	30	31	10	3	32	17	24
																	1	1	1																
c =	34	29	16	25	36	11	12	5	14	7	8	21	32	3	28	23	18	19	20	15	10	35	6	17	30	31	24	33	26	27	2	13	22	9	4
						1	1			1	1						1	1	1						1	1			1	1					
c =	22	33	28	17	4	7	8	25	32	11	12	15	2	29	24	35	18	19	20	3	14	9	36	23	26	27	6	13	30	31	34	21	10	5	16
																	1	1	1																
c =	10	13	24	23	16	11	12	17	2	7	8	3	34	33	6	9	18	19	20	29	32	5	4	35	30	31	36	21	26	27	22	15	14	25	28

Part F: A Comparison Table

Figure 3.1 Screenshot of Message Calculator

The encrypted values for different *e* are kept together to form a table, one table for each *n*. Figure 3.1, part F, shows the table for *n* = 38. In part E, the view is zoomed even further. This way of displaying the results makes a visual comparison possible. The number of holes and the percentage of them per septuple is also calculated and written in the table, Figure 3.1 part B.



E121 =IF(E122=E$48,1,"")

				1	1											1	1			
e = 7; c ➔	127	2	103	65	66	89	29	108	60	124	19	83	35	114	54	77	78	40	141	16
			1	1	1											1	1	1		
e = 11; c ➔	95	19	64	65	66	111	35	36	125	115	28	18	107	108	32	77	78	79	124	48
				1	1	1									1	1	1			
e = 13; c ➔	101	50	25	65	66	67	107	82	31	136	7	112	61	36	76	77	78	118	93	42
				1	1											1	1			
e = 17; c ➔	17	46	103	65	66	45	139	75	5	80	63	138	68	4	98	77	78	40	97	126
				1	1											1	1			
e = 19; c ➔	140	106	38	65	66	89	94	4	47	20	123	96	139	49	54	77	78	105	37	3
				1	1											1	1			
e = 23; c ➔	134	6	25	65	66	111	74	49	86	37	106	57	94	69	32	77	78	118	137	9
				1	1	1									1	1	1			
e = 37; c ➔	127	24	103	65	66	67	29	108	5	58	85	138	35	114	76	77	78	40	119	16
				1	1											1	1			
e = 43; c ➔	101	28	25	65	66	89	107	82	86	59	84	57	61	36	54	77	78	118	115	42
				1	1											1	1			
e = 47; c ➔	17	123	103	65	66	111	139	75	60	102	41	83	68	4	32	77	78	40	20	126

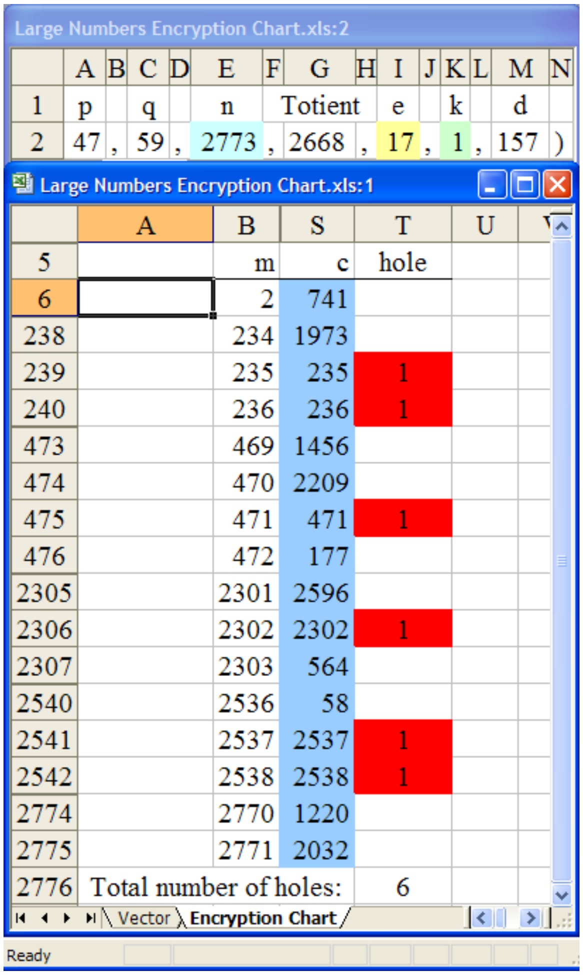
Figure 3.2 Symmetry of Holes

Symmetry is an interesting aspect of visual observation. In Figure 3.2, Symmetry of Holes, a view of the Message Calculator is shown for *n* = 143. The markers for holes are shaded. Also, the holes which are common to all cases within a comparison table are outlined. A larger view of the center of the sub-table shows how the holes are equidistant from an axis of symmetry. The manual search for holes continued using these tools. For all the values of *n* that were studied, no matter how large, there are still some holes and the symmetry continued. The Large Numbers Encryption Calculator was developed to help study larger values.

3.1.4 The Large Numbers Encryption Calculator

The Large Numbers Encryption Calculator which is shown partially in Figure 3.3 was the next Excel worksheet developed in this series of tools. The Message Calculator, although a great tool in showing visual symmetry of holes, was limited by the number of columns in Excel. The Message Calculator can only be used for values of *n* less than 255. For larger values of *n*, a different tool had to be developed.

The Large Numbers Encryption Calculator is the tool developed for larger *n*. This tool is a giant calculator which is used to find encrypted values of *m*, when *n* is larger than 255. In Figure 3.3, a portion of the calculator is shown as a screenshot where it has calculated encrypted messages for the septuple (47, 59, 2773, 2668, 17, 1, 157). This septuple is based on the values used in an example in the original RSA paper [2].



Large Numbers Encryption Chart.xls:2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	p		q		n		Totient		e		k		d	
2	47	,	59	,	2773	,	2668	,	17	,	1	,	157	)

Large Numbers Encryption Chart.xls:1

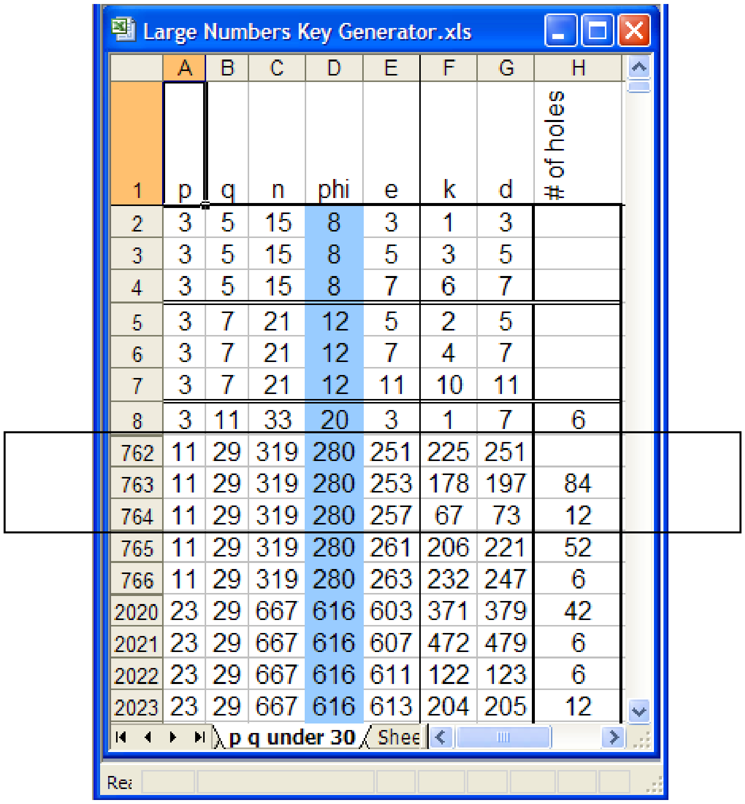
	A	B	S	T	U
5		m	c	hole	
6		2	741		
238		234	1973		
239		235	235	1	
240		236	236	1	
473		469	1456		
474		470	2209		
475		471	471	1	
476		472	177		
2305		2301	2596		
2306		2302	2302	1	
2307		2303	564		
2540		2536	58		
2541		2537	2537	1	
2542		2538	2538	1	
2774		2770	1220		
2775		2771	2032		
2776	Total number of holes:			6	

Figure 3.3 Calculator Screenshot

Most of the rows of data have been hidden in order to make the chart more readable. If a student would like to use this calculator as a look up table to find encrypted values for 2, 472 and 2771, they may read these values as, 741, 471 and 2032 respectively. The values where the holes occur are shaded and marked by a marker "1" on the side. The total number of holes is calculated and written on the last row of the table. An attempt to read this chart to find the total number of holes and the message values corresponding to them, for the RSA paper's septuple, shows that there are holes at *m* = 235, 236, 471, 2302, 2537 and 2538, and the total number of holes is six. Even for this relatively large value of *n*, there are still 6 holes present. At this stage, in order to investigate the occurrence of holes, it was necessary to gather more information and extend the domain of the research. The domain of the values of *p* and *q* was increased to include values between 20 and 30 and all acceptable values of *e* less than the totient were considered. To do such a big range of calculations using a programming language is more practical than using Excel. The results from a program can be used by Excel to be manipulated and formatted if necessary.

3.1.5 The Large Numbers Key Generator

The search for a simple RSA example continued, yet it had spawned an additional investigation about the cases where the plaintext equals ciphertext. To organize the search, the idea of using a seven element vector, or a septuple was presented, in section 3.1.2. An ordered septuple would represent a unique combination of *p*, *q*, *e*, and *k* as well as values that are calculated using them i.e. *n*, phi or *φ* and *d*. Each vector has seven elements in this order: *p, q, n, φ, e, k* and *d*. In order to organize the investigation of holes further, the hole count was associated with each septuple.



	A	B	C	D	E	F	G	H
1	p	q	n	phi	e	k	d	# of holes
2	3	5	15	8	3	1	3	
3	3	5	15	8	5	3	5	
4	3	5	15	8	7	6	7	
5	3	7	21	12	5	2	5	
6	3	7	21	12	7	4	7	
7	3	7	21	12	11	10	11	
8	3	11	33	20	3	1	7	6
762	11	29	319	280	251	225	251	
763	11	29	319	280	253	178	197	84
764	11	29	319	280	257	67	73	12
765	11	29	319	280	261	206	221	52
766	11	29	319	280	263	232	247	6
2020	23	29	667	616	603	371	379	42
2021	23	29	667	616	607	472	479	6
2022	23	29	667	616	611	122	123	6
2023	23	29	667	616	613	204	205	12

Figure 3.4 Chart of Hole Counts

The Large Numbers Key Generator is a matrix of more than 2000 rows. Each row corresponds to a septuple and its associated hole count. The "more than 2000 values" is due to ranging *p* and *q* over all the primes under 30 while varying *e* from 3 to phi. This calculator doesn't generate the primes. It uses the list of primes entered manually, to calculate the product, and the totient. Then, as the values of *e*, (3 ≤ *e* < phi), are checked for coprimality and entered to the calculator, it will find the *d* that corresponds to each value of *e*. The value of *k* has also been manually entered because it needs to be a *k* that divides *e*. The value of *d* is calculated based on phi, *e* and *k*. Finally, the hole count or the "# of holes" was calculated using a Perl program, the Hole Finder, and was manually entered as the last column of data in the Large Numbers Key Generator. The Chart of

Hole Counts, Figure 3.4, is a screenshot of the Large Numbers Key Generator with a lot of its rows hidden. An attempt to read the chart in order to see the effect of changing *e* on the number of holes can show that in rows 763, 764 and 765 of this chart, choosing *e* as 253, 257 and 261 respectively has produced 84, 12 and 52 number of holes. This may be an indicator that the septuple corresponding to *e* = 257 is the best choice since it creates the least number of holes among these three cases. Also this chart may be used to find the private key corresponding to a given public key. For example, if the original primes used are *p* = 11 and *q* = 29, for *e* = 257, *d* = 73. A listing of the "Hole Finder" Perl program, the partial list of entries in the Large Numbers Key Generator and a list of septuples, hole counts and their corresponding list of holes may be found in the Appendices.

3.2 Two Simple Examples

Simple examples enhance teaching and make observation of interesting results easier. The original RSA paper used the septuple (47, 59, 2773, 2668, 17, 1, 157). These calculations are too cumbersome to serve as a demonstration example for a simple lesson or lecture. Following are two simple examples of encryption using smaller primes for *p* and *q*. The first example is encrypting a number and the second is encrypting a short sentence. Both examples use the same public and private keys.

3.2.1 Simple Example #1: Encrypting/Decrypting a Number

Using relatively smaller primes, a simple example is created which can be used later to make some observations regarding the RSA algorithm. The first step is generating the keys. For this, the five steps to generate the private and public keys introduced in section 2.3.3 will be used as follows,

1. Choose two prime numbers, *p* and *q*.

   $p = 11, q = 13.$

2. Calculate their product, *n*.

   $n = p \,.\, q = 11 \text{ x } 13 = 143.$

3. Calculate the totient of *143*.

   $\varphi(n) = (p\text{-}1)(q\text{-}1),$
   $\varphi(143) = (11 - 1)\ (13 - 1),$
   $\varphi(143) = 120.$

4. Choose a number *e* less than 143 that is coprime with 120. *e* = 23 is a valid candidate since 120 and 23 have no common factors.

5. Find a number *d* such that *e* and *d* can be multiplicative inverses in modulus *n* arithmetic. A valid value for *d* is 47 (for *k* = 9) since it satisfies equation 2.2:

$$d = (1 + k * \Phi(n)) / e$$
$$d = (1 + (9 * 120)) / 23$$
$$d = 1081 / 23 = 47$$
$$d = 47$$

or,

$$d * e \equiv 1 \qquad (\bmod \Phi(n))$$
$$23 * 47 \equiv 1 \qquad (\bmod 120)$$

Using the above five steps Bob created his RSA keys. Bob kept (*d*, *n*) = (47, 143) for himself and posted his public keys (*e*, *n*) = (23, 143) on his website. Since the keys *e* and *d* are calculated, the process of encrypting and decrypting can be performed. Let's choose the message to be the number 7.

Alice wants to send the number 7 to Bob, therefore *m* = 7:

$$c = m^e \; mod \; n = 7^{23} \bmod 143 = 27368747340080916343 \bmod 143 = 2$$

After Bob receives the encrypted message, "2", he uses *d* = 47 to decrypt it:

$$m = c^d \; mod \; n = 2^{47} \bmod 143 = 140737488355328 \bmod 143 = 7$$

In summary, Bob did the five steps to generate the keys and announced the public key on his website. Alice used the public key to encrypt the message which was the number 7. The resulting encryption was the number 2. Alice sent the number 2 to Bob. Bob used the private key and decrypted the ciphertext, number 2, coming up with the number 7. Of course, many times Alice might want to send a text instead of a number to Bob. The next section describes applying RSA to a message that is a text.

3.2.2 Simple Example #2: Encrypting /Decrypting a Short Sentence

Encryption is used to conceal information from unauthorized users and reveal information to the intended recipient. In the Simple Example #2, a sample text: "I loVe cRyptoGraphy! ;-)" is encrypted using the public key (*e, n*) = (23, 143) and decrypted using the private key (*d, n*) = (47, 143). These are the same public and private keys calculated in Simple Example #1. Let the text "I loVe cRyptoGraphy! ;-)" be a message Alice is trying to send to Bob. The encryption and decryption is done using the Message Calculator as a lookup table. The first step for Alice is to encode the message from text to numbers. The block size used here is one letter or 7 bits. It is important to notice that current cryptography uses larger key sizes of 768 bits and higher. So the block size needed will be slightly smaller than the key size (*m* is less than *n*). When these large block sizes are used, the text, “I loVe cRyptoGraphy! ;-)” will correspond to one large integer: 21412941336931961136287122536275264957456991747037 7. Coming back to our example, this conversion to ASCII is the first box in the Communication Block Diagram, Figure 2.8 in chapter II. In Table 3.4, each letter is turned into its decimal ASCII code:

Table 3.4. ASCII Equivalents of a Text

Letter	ASCII Code	Letter	ASCII Code	Letter	ASCII Code	Letter	ASCII Code
I	73		32	o	111	y	121
	32	c	99	G	71	!	33
l	108	R	82	r	114		32
o	111	y	121	a	97	;	59
V	86	p	112	p	112	-	45
e	101	t	116	h	104	)	41

I		l	o	V	e		c	R	y	p	t	o	G	r	a	p	h	y	!		;	-	)
73	32	108	111	86	101	32	99	82	121	112	116	111	71	114	97	112	104	121	33	32	59	45	41

The next step for Alice is to encrypt the encoded blocks. This is the second box in Figure 2.8. The line of data in the bottom of Table 3.4 is a set of 24 integers, the encoded version of the message (seven bits per character) Alice is sending Bob. For the public key (*e, n*) = (23, 143), the corresponding *d* is 47. The septuple used for this encryption is (11, 13, 143, 120, 23, 9, 47). Using the above message, the Message Calculator (Section 3.1.3) and this septuple, the cipher text can be calculated as shown in Table 3.5:

Table 3.5 Table of Encrypted Values

Message (ASCII)	Encrypted Message (ASCII)	Letter	Message (ASCII)	Encrypted Message (ASCII)	Letter	Message (ASCII)	Encrypted Message (ASCII)	Letter	Message (ASCII)	Encrypted Message (ASCII)	Letter
73	57	9	32	76	L	111	67	C	121	88	X
32	76	L	99	44	,	71	37	%	33	132	oor
108	36	$	82	114	r	114	108	1	32	76	L
111	67	C	121	88	X	97	102	f	59	119	w
86	135	oor	112	96	`	112	96	`	45	89	Y
101	30	RS	116	51	3	104	26	Subs	41	72	H

57	76	36	67	135	30	76	44	114	88	96	51	67	37	108	102	96	26	88	132	76	119	89	72
9	L	$	C	oor	RS	L	,	r	X	`	3	C	%	l	f	`	Subs	X	oor	L	w	Y	H

The line at the bottom of the Table 3.5 is the list of letters corresponding to the encrypted message if decoded without decryption: “9L$CoorRSL,rX`3C%lf`SubsXoorLwYH”. The underlined characters are special characters which are not even letters. The third step is to transmit the ciphertext. This is the middle box of Figure 2.8. So encoding and encryption is happening on Alice’s side. What Alice is sending to Bob is a stream of 24 integers. The only person who is able to decrypt the stream of integers is Bob, the owner of the one and only private key, in this case, *d* = 47, (*d* was generated in Simple Example #1). The fourth step is decrypting the ciphertext received from Alice. This is the fourth box of Figure 2.8. Once Bob the receiver, gets this stream of 24 integers, he will decrypt each integer. The fifth step is decoding the decrypted message. This is the fifth and last box of Figure 2.8. Bob will find the corresponding letters in the ASCII code table and decode the numbers into text. Here in Table 3.6, the Message Calculator was used as a lookup table for decryption:

Table 3.6 Table of Decrypted Code

Encrypted Message (ASCII)	Decrypted Message (ASCII)	Letter	Encrypted Message (ASCII)	Decrypted Message (ASCII)	Letter	Encrypted Message (ASCII)	Decrypted Message (ASCII)	Letter	Encrypted Message (ASCII)	Decrypted Message (ASCII)	Letter
57	73	I	76	32		67	111	o	88	121	y
76	32		44	99	c	37	71	G	132	33	!
36	108	l	114	82	R	108	114	r	76	32	
67	111	o	88	121	y	102	97	a	119	59	;
135	86	V	96	112	p	96	112	p	89	45	-
30	101	e	51	116	t	26	104	h	72	41	)

After placing these letters side by side the original text will be revealed:

I loVe cRyptoGraphy! ;-)

Table 3.7 displays all of the steps for Simple Example #2.

Table 3.7 Sample Message Transformation Table

Original Plaintext	I		l	o	V	e		c	R	y	p	t	o	G	r	a	p	h	y	!		;	-	)
Encoded (ASCII)	73	32	108	111	86	101	32	99	82	121	112	116	111	71	114	97	112	104	121	33	32	59	45	41
Encrypted (ASCII)	57	76	36	67	135	30	76	44	114	88	96	51	67	37	108	102	96	26	88	132	76	119	89	72
Ciphertext	9	L	$	C	oor	RS	L	,	r	X	`	3	C	%	l	f	`	Subs	X	oor	L	w	Y	H
Received (ASCII)	57	76	36	67	135	30	76	44	114	88	96	51	67	37	108	102	96	26	88	132	76	119	89	72
Decrypted (ASCII)	73	32	108	111	86	101	32	99	82	121	112	116	111	71	114	97	112	104	121	33	32	59	45	41
Decoded Message	I		l	o	V	e		c	R	y	p	t	o	G	r	a	p	h	y	!		;	-	)

3.3 Chapter III Summary

Many difficult concepts can be better explained using small simple examples. The RSA algorithm is an asymmetric key encryption method which uses modular arithmetic and exponentiation for encrypting a number. During the course of exponentiation, small numbers grow to become very large numbers and modular arithmetic cuts them short again. It can be hard to work with these large intermediate numbers. Having lookup tables to find encrypted values of small messages is beneficial. That is the underlying reason for developing a few lookup tables. Some of these lookup tables would be used for encryption and others for analysis. A glance at the first lookup table was enough to realize there is an interesting observation to be made. Encryption with small numbers seemed to fail. Specifically, the failure was that the encrypted value was equal to the unencrypted message value. One observation lead to another and gradually tables were developed to look up slightly bigger values to see if the observations are still true. These observations will be fully explained in the next chapter. In this chapter, a few tools were described which would become lookup tables for the simple examples or help in observations. The Tiny Key Encryption Calculator was created, to encrypt and decrypt an exhaustive list of values of m using tiny keys. The Tiny Samples Key Generator was made to help see all different e and d combinations. The Small Key Encryption Calculator was created to serve as a calculator as well as being a lookup table for encryption under a given set of then called parameters. This lead to the introduction of ordered septuple, ($p, q, n, \varphi , e, k, d$). This septuple is very useful for organizing the results and communicating example observations. The Message Calculator was the next tool developed to help see the holes and also as a lookup table for encryption and decryption. The Large Numbers Encryption Calculator was developed to be a lookup table for values of n larger than 255. This lead to the creation of the Large Numbers Key Generator. Each row of this chart would include the total number of holes for a given septuple.

Finally, in this chapter two simple small examples of RSA encryption and decryption were presented. The first small example was the encryption and decryption of number 7 under the septuple (11, 13, 143, 23, 9, 47). The second example was the encryption and decryption of the short sentence, “I loVe cRyptoGraphy! ;-)” step by step. Here the intention was to show the elements of encoding and decoding as well as encryption and decryption. The same septuple was used for the second small example. The main tool used for these two examples was the Message calculator. While developing these tools, a number of interesting observations were made which are presented in Chapter IV.

CHAPTER IV

OBSERVATIONS

4.0 Introduction

Observations are the result of watching an object intently. In science, observations are the stepping stones for further research. In order to observe well, sometimes you need to look closer and sometimes you need to step back and look at the object in a panoramic view. This is a chapter on observations made while studying the RSA algorithm. The first observation shows that there are cases where the ciphertext becomes equal to the plaintext. Mathematically, this means that there are values of *m*, where $c = m^e \bmod n = m$. In this paper, these specific values of *m* are said to have caused or created a hole, or a hole occurred at that *m*. The second observation is that, for all septuples used in this study, there were holes. In other words, in the search space of this thesis, holes are always there. The third observation is that there is a pattern of symmetry for the holes. The fourth observation is that for a given *n* and different values of *e*, there are some values of *m* where a hole will occur repeatedly and changing *e* does not save that *m* from producing a hole. The fifth observation is that there is a minimum number of holes detected in this study and that minimum is greater than zero. For odd *n*, the minimum number of holes is six. The sixth observation is that there are six holes which keep occurring for all values of *e*, for a given *n*. These holes are the same as the six minimum holes. These holes are called the principal holes. In the rest of this chapter, these observations will be visited and explained using examples.

Note:

1. The public key is *e* in conjunction with *n* and the private key is *d* in conjunction with *n.* To simplify the following discussion, hereafter *e* will be called the public key and *d*, the private key. Specific mention of *n* as a part of the keys will not be made.
2. Before proceeding with these observations, it is important to notice that although *n* can theoretically be even, for all practical purposes, it is always an odd number. An even *n* shows that one of the two prime factors of *n* is 2 and the other prime is half of *n*. Whenever the values of *p*, *q* and *e* are known, anyone can easily calculate the private key *d* and decrypt any message.

4.1 Observation 1: There are Holes

Encrypting a message is supposed to make that message hard to read. If a message is encrypted and the result is the same message, the method has failed to encrypt.
The first time this anomaly was detected was in Table 3.1. The shaded area showed that for a few values of *m* such as 4, 5 and 6, the ciphertext was the same as the cleartext.

This was first considered a problem due to using small n. Holes were the name chosen for cases where $c = m^e \; mod \; n = m$. This means when m is encrypted, the encrypted message c, is equal to the original message m. In other words, encrypting m does not change it. A message $m = 7$ is encrypted using a public key $(e, n) = (7, 38)$ and the ciphertext is calculated as $c = m^e \; mod \; n = 7^7 \bmod 38 = 823543 \bmod 38 = 7$. This value of m producing a hole is illustrated in Figure 4.1. Some examples will show the concept of holes.

Observation One	
Used Septuple: (2,19,38,18,7,12,31)	
m	*c*
2	14
3	21
…	…
6	28
7	7
…	…
35	17
36	24

For this value of *m*, the message and the ciphertext are equal, $m = c$. There is a hole here.

Figure 4.1 There Are Holes

Example 4.1.1 Consider the case for the septuple: $(p, q, n, \varphi, e, k, d) = (11, 13, 143, 120, 23, 9, 47)$. This is the same septuple that was used in the simple examples in chapter III. Encrypting all possible values of m ranging from 2 to 141, reveals six cases where $c = m$. These holes occur at: $m = 12, 65, 66, 77, 78, 131$. These six holes are out of 140 total possible values of m.

Let's look at the encryption of $m = 12$ under $n = 143$ and $e = 23$:

$$c = m^e \; mod \; n = 12^{23} \bmod 143 = 6624737266949237011120128 \bmod 143 = 12 = m$$

There is a hole at $m = 12$ because for this value of m, $c = m = 12$.

Example 4.1.2 Let $p = 47$ and $q = 59$. These are the two primes used in the original RSA paper. Using the same procedure as in Simple Example #1, n and $\varphi(n)$ can be calculated and choosing a valid e , d can be calculated. For $(p, q, e) = (47, 59, 17)$

$n = 2773,$
$\varphi(n) = \varphi(2773)= 2668$
for $e = 17$ and $k = 1$,
$d = 157$

So the corresponding septuple for the RSA small example is:
$(p, q, n, \varphi, e, k, d) = (47, 59, 2773, 2668, 17, 1, 157)$.
Here is an excerpt from section VIII of the original RSA paper, "A small example":

> “With n = 2773 we can encode two letters per block, substituting a two-digit number for each letter: blank = 00, A = 01, B = 02, . . . , Z = 26. Thus the message
>
> ITS ALL GREEK TO ME
>
> (Julius Caesar, I, ii, 288, paraphrased) is encoded:
>
> 0920 1900 0112 1200 0718 0505 1100 2015 0013 0500
>
> Since e = 10001 in binary, the first block (M = 920) is enciphered:
>
> $M^{17} = (((((1)^2 . M)^2)^2)^2)^2 . M = 948$ (mod 2773).
>
> The whole message is enciphered as:
>
> 0948 2342 1084 1444 2663 2390 0778 0774 0219 1655.
>
> The reader can check that deciphering works: $948^{157} \equiv 920$ (mod 2773), etc.”

Let’s look at the case for m = 235 for the septuple formed by the example in the RSA paper. The septuple is (47, 59, 2773, 2668, 17, 1, 157) and for m =235:

$c = m^e \; mod \; n = 235^{17}$ mod 2773 = 235 = m

The rest of the holes for the RSA septuple are at m = 236, 471, 2302, 2537 and, 2538. These six holes are out of 2770 total possible values of m.

4.2 Observation 2: Holes are Always there

In the study space of this research, for all septuples used, every case had holes. This observation was made also in 1979 by Blakely and Borosh in their paper, “Rivest-Shamir-Adleman Public Key Cryptosystems Do Not Always Conceal Messages,” [10]. In this research, the observation of the holes being always there is made on two parts of the study space. One part is produced using Excel and is exhaustive. These septuples are more than 2000 combinations of (p, q, e) where: $p \neq q$; $3 \leq p < 30$; $3 \leq q < 30$; $3 \leq e < \varphi(n)$ and e is coprime with $\varphi(n)$. The other part of the study space is the results of a Perl hole counter which so far has covered p, q, n, φ, and e values where the highest value for p is 5009, for q is 9029, for n is 45172087, for phi is 45158056 and for e is 89. (d is calculated based on e.) The highest number of holes observed, so far, is 6886.

4.3 Observation 3: Holes Occur Symmetrically

The values of holes are symmetrically distributed around $\frac{n}{2}$. For all odd n, the point $\frac{n}{2}$ falls between two integers on the numbers axis. Let $\frac{n}{2}$ be called the “point of symmetry”. For every hole on one side of the point of symmetry, there is a corresponding hole on the opposite side exactly at the same distance from the point $\frac{n}{2}$. These two holes are named “complementary holes” in this thesis. The addition of m (or c) values at two complimentary holes is always equal to n.

Proof:

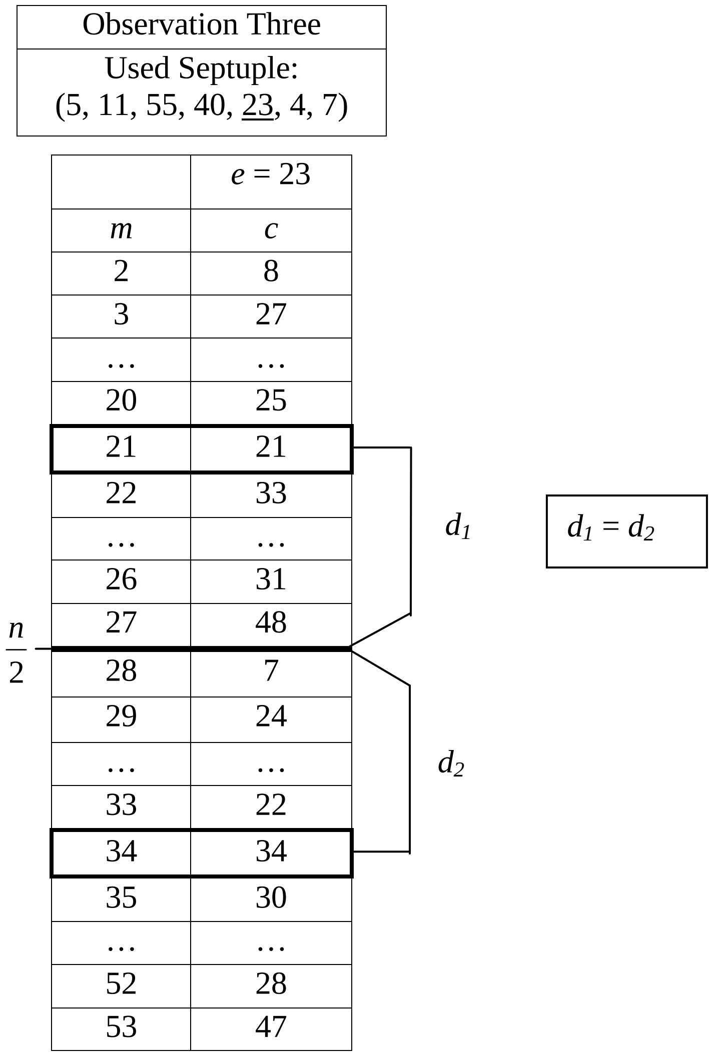
Consider any two messages m_1 and m_1'. These messages are chosen to be equidistant from the point $\frac{n}{2}$. If d_1 is the distance between m_1 and $\frac{n}{2}$ when d_2 is the distance between $\frac{n}{2}$ and m_1'. If $d_1 = d_2$ then

$$\frac{n}{2} - m_1 = m_1' - \frac{n}{2},$$

$$\frac{n}{2} + \frac{n}{2} = m_1' + m_1 ,$$

$$\therefore m_1 + m_1' = n .$$

If m_1 and m_1' are defined as complementary holes, and this is defined if and only if $d_1 = d_2$, then the addition of two complimentary holes is always equal to n.



Observation Three
Used Septuple: (5, 11, 55, 40, 23, 4, 7)

	e = 23
m	c
2	8
3	27
…	…
20	25
21	21
22	33
…	…
26	31
27	48
28	7
29	24
…	…
33	22
34	34
35	30
…	…
52	28
53	47

Figure 4.2 Symmetrical Locations of Holes

Holes come in pairs which are equidistant from a point $\frac{n}{2}$. Another way of saying this is that there is an even number of holes for each public key (e,n) and every hole is paired with a complimentary hole. The two complimentary holes are equidistant from the point of symmetry. This symmetry can be best observed in an example. For this example, n is chosen to be 55.

Let $(p,q) = (5,11) \Rightarrow n = 55, \varphi(n) = \varphi(55) = 40.$

For $e = 23$ and $k = 4$, the corresponding septuple is $(p,q,n,\varphi,e,k,d) = (5,11,55,40,23,4,7)$. There are two complimentary holes at *m* = 21 and *m* = 34. These holes are equidistant from $\frac{n}{2}$ = 27.5:

$\frac{n}{2} - m_1 = d_1 = m_1{}' - \frac{n}{2}$; 27.5 – 21 = 6.5 = 34 – 27.5. This example is illustrated in Figure 4.2.

4.4 Observation 4: Some Holes Are Repeated For Different Values of *e*

When there is a hole that occurs for a given combination of *n* and *e*, changing *e* might not save that *m* from producing a hole. In other words, with *n* constant and *e* varied, there are some *m* values where a hole will occur repeatedly. There are even a few values of *m* that for a constant *n*, no matter what the value of *e*, such *m* will keep producing holes. This was first observed when working with the message calculator. In Figure 3.2, Symmetry of Holes, the comparison table that is magnified corresponds to *n* = 143. On the side of the magnified portion as *e* is varied, some values of *m*, for example *m* = 65 and 66, keep producing a hole. On the other hand at *m* = 67, a hole occurs at *e* = 13, and changing *e* to *e* = 23 will produce *c* = 111 which is not equal to 67 and as such, not a hole. In Figure 4.3 the following two different encryption keys are used. With *n* = 55, the two keys are *e* =13 and *e* =23. At *m* = 10, both keys create a hole, however at *m* = 12, *e* = 13 does create a hole and *e* = 23 does not.

4.5 Observation 5: The Minimum Number of Holes is 6

Observation two mentioned that in the study space so far, there have always been some holes. Observation four showed how some holes are repeated and some are not as *e* is varied. Since there were different number of holes associated with different septuples, it was interesting to find out what is the minimum number of holes that occur. The research so far has shown that the minimum is six. For odd *n*, there is no septuple in the study space that has less than 6 holes. This analysis which was initially based on: $p \neq q$; $3 \leq p < 30$; $3 \leq q < 30$; $3 \leq e < \varphi(n)$ and *e* is coprime with *φ(n)*. The analysis has been extended to include extreme values such as: $3 \leq p < 5009$, $3 \leq q < 9029$, $n < 45172087$, phi < 45158056 and $e < 89$. The minimum number of holes being six was also verified using larger *p*, *q* and *e* using Perl programs. For the Simple Example septuple the minimum of 6 is out of 140, for the RSA example septuple the minimum of 6 is out of 2770, and for a very large *n*, such as is used in practice today, the minimum of 6 is out of 2^{768} possible values. There are six holes per septuple that not only are the least number of holes but have another interesting property.

4.6 Observation 6: The Principal Holes

For a given *n*, there are six cases of ciphertext equal plaintext occurring at any *e*. As *e* varies and the number of holes changes accordingly, the only six holes that are always there for a constant *n* are the same holes that create the set of six minimum holes. These are called the “principal holes”. For example for *n* = 55, four keys have been used for encryption in Figure 4.4. All four keys have encrypted the list of messages 10, 11, 21, 34, 44 and 45 as holes. Since these holes occur at all values of *e*, they are the principal holes. For *m* = 26, there is a hole at *e* = 11 but there is no hole at other values of *e* that were used for encryption. So the hole at *m* = 26 is not a principal hole.

Observation Four
Used Septuples: (5, 11, 55, 40, 13, 12, 37) (5, 11, 55, 40, 23, 4, 7)

	e = 13	*e* = 23
	12 holes	6 holes
m	*c*	*c*
2	52	8
3	38	27
…	…	…
8	28	17
9	14	14
10	10	10
11	11	11
12	12	23
13	8	52
…	…	…
52	17	28
53	3	47

This hole is repeated for both e values

This hole occurs for *e* = 13 and not for *e* = 23

Figure 4.3 Repeated and Unrepeated Holes

Observation Six
Used Septuples: (5, 11, 55, 40, 3, 2, 27) (5, 11, 55, 40, 23,4,7) (5, 11, 55, 40, 11, 3, 11) (5, 11, 55, 40, 13, 12, 37)

	e = 3	*e* = 23	*e* = 11	*e* =13
	6 holes	6 holes	30 holes	12 holes
m	*c*	*c*	*c*	*c*
2	8	8	13	52
3	27	27	47	38
…	…	…	…	…
9	14	14	9	14
10	**10**	**10**	**10**	**10**
11	**11**	**11**	**11**	**11**
12	23	23	23	12
…	…	…	…	…
20	25	25	20	25
21	**21**	**21**	**21**	**21**
22	33	33	33	22
…	…	…	…	…
26	31	31	26	31
27	48	48	38	37
28	7	7	17	18
29	24	24	29	24
…	…	…	…	…
33	22	22	22	33
34	**34**	**34**	**34**	**34**
35	30	30	35	30
…	…	…	…	…
43	32	32	32	43
44	**44**	**44**	**44**	**44**
45	**45**	**45**	**45**	**45**
46	41	41	46	41
…	…	…	…	…
52	28	28	8	17
53	47	47	42	3

Some holes that are not principal holes

The Six Principal Holes

Figure 4.4 Principal Holes

4.7 Chapter IV Summary

Unexpected observations such as a ciphertext equal to its plaintext are interesting and intriguing. One observation leads to another and a compilation of all of these interesting observations may lead to future discovery by the same person or another. As the quest for a simple example for the RSA algorithm started, developing new tools and arranging the data in different ways and platforms lead to many observations. At times, observation was a lot easier than explanation; however, science requires an observer to explain the observations accurately as well as the method of research.

The existence of a ciphertext equal to the original message is an anomalous behavior for an encryption method. According to the study space of this thesis, holes ($c = m$) are always present. They occur in pairs and the two complementary holes of a pair are equidistant from a point of symmetry located at $\frac{n}{2}$. Some holes occur at the same m for a given n even when the value of e is changed. There are a minimum of 6 holes in our study space. Finally, these six holes were given the name, the principal holes, and one of their properties is that the principal holes show up in every septuple with the same n. The likelihood of one of the minimum 6 occurring for a very large n such as is used today is infinitesimal.

There are still more observations being made and the research goes on. I had to wrap up this chapter and land this plane. One interesting thing about observations is that the observer needs to constrain herself not to jump to early conclusions. There is a funny story about a man who decided to study the hearing of flies. He gathered his pen and paper and sat in a room. He captured the first fly passing by and kept it in his fist without squeezing it. When he was ready to make the first observation, he released the fly on the table and said, “Fly, fly!” Obviously the fly took off. He wrote, “Observation one: when the fly is asked to fly in a mild tone of voice, it flies.” Then he got up, captured the fly again and came back to his seat. He removed one of the wings of the fly and repeated the same command, “Fly, fly!” This time the fly didn’t fly. He repeated the command with a louder voice and the fly, jumped and landed a short distance away. He recorded the second and third observations: “Observation two: after removing one of the wings, when the fly is asked to fly in a mild tone of voice, it paused. After repeating the command with a louder voice, it flew.” “Observation three: This time the distance flown was shorter.” He stretched his hand and captured the poor one-winged fly and removed the other wing. He repeated the command, “Fly, fly.” No action. He raised his tone of voice and he repeated a second time; still no action. This time he got even louder. He shouted, “FLY, FLY!” Still, no flying was done. He recorded his observations and his conclusion as follows: “When the fly was commanded to fly, after one wing was removed, it took a louder tone to make it fly. After the removal of both wings, no attempt was successful in making the fly respond to the command. We conclude that after the removal of one wing, the fly’s hearing got impaired and once both wings were removed, the fly became deaf. Conclusion: the ears of a fly are attached to his wings.” The moral of the story is that observations can be very intriguing and they might push you to make conclusions that are wrong! In this chapter I have refrained from making conclusions that are sweeping and general because the study is still going on. The next chapter is a summary of this thesis, from the study of RSA and the quest for a simple non-trivial example to the land of holes, where an encryption method fails to encrypt.

CHAPTER V

SUMMARY & CONCLUSION

The RSA algorithm is the most widely used encryption method yet there is lack when it comes to finding a good simple example. RSA is an asymmetric encryption method which means it uses two different keys for encryption and decryption. These keys are generated by the receiver of the message to enable his correspondents communicate with him securely. The receiver chooses two preferably large prime numbers p and q. He finds the product and calls it n. He calculates the totient using the formula $\varphi(n) = (p-1)(q-1)$. Then he chooses an integer e that is coprime with φ and less than n. The variable e in conjunction with n will be called the public key. The receiver then finds another integer d that would satisfy the equation $d = (1 + k * \varphi)/e$. The variable d in conjunction with n will be called the private key. The private key will be distributed to the correspondents of the creator of the keys. Now that the keys are ready, a message m from a sender Alice can be sent to a receiver Bob securely using RSA encryption. Alice will use the formula $c = m^e \mod n$ to calculate the ciphertext c. She then transmits the message to the creator of the key, Bob. Bob uses the formula $m = c^d \mod n$ to decrypt the message and read the plaintext that Alice sent him. This is the RSA algorithm in a nutshell. In answer to a need for a good example for RSA a search started. A good example would use the smallest possible values of p and q and the best candidate for e to encrypt and decrypt a short sentence. The search for a good e, lead to observing some anomalies in the RSA algorithm. Before considering these anomalies, a notion was introduced to help refer to different sets of values used in RSA. There are seven values involved in the RSA formulas. These values are p, q, n, φ, e, k and d; some are given and some are calculated. An ordered septuple containing all of these values was introduced $(p, q, n, \varphi, e, k, d)$. Since different values of e produced different number of anomalies for the same n, a good septuple would contain small p and q and create the least number of anomalies. This septuple was found to be
$(p, q, n, \varphi, e, k, d) = (11, 13, 143, 120, 23, 9, 47)$ The short sentence encrypted was "I loVe cRyptoGraphy! ;-)" and was encrypted as "9L$CoorRSL,rX`3C%lf`SubsXoor LwYH", where the underlined characters are special characters which are not letters. The mission to find a small nontrivial example for RSA was accomplished. What about the anomalies?

During the search for a simple example for RSA, anomalous behavior was observed. In RSA for some values of message *m*, when *m* is encrypted using the formula $c = m^e \bmod n$, the ciphertext *c* is equal to the original message *m*. A “hole” was defined to have occurred at a message value where $c = m^e \bmod n = m$. The existence of holes was the first observation. In our original study space there were more than 2000 septuples and all of them created holes. Even when the search space was extended to include larger *p* and *q* as high as 523 for *p*, 709 for *q* and 89 for *e*, there were always some holes present. The hole values displayed some symmetry. Specifically, holes come in pairs that are equidistant from a point $\frac{n}{2}$. When there is a hole that occurs for a given key (*e, n*), changing *e* might not save that *m* from producing a hole. The minimum number of holes found in the study space is six. For any given *n*, there are six holes that occur for any key (*e, n*); changing *e* doesn’t get rid of these holes. These are the same holes as the minimum six holes. These were named the principal holes.

Existence of holes is an important phenomenon. The percentage of holes among all possible messages gets smaller and smaller as *n* gets larger (for a 768 bit *n*: 6 out of 2^{768} possibilities), however, it is still true that they exist. Future work on them may include: Finding relationships between a set of holes especially the principal holes such that if and when a hole is found, the other holes may be predicted; Finding a relationship between holes and the values of their respective septuple and using these relationships in factorizing *n*; Studying the holes further more in order to improve the RSA and avoiding bad *e* values as reference mentions [10]. Even a simple example can be used for understanding RSA and seeing these holes.

REFERENCES

[1] "Secure." *The New Merriam-Webster Dictionary for LARGE PRINT Users,* Thorndike Pr. 1989. (ISBN 0-8161-4754-X)

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, v. 21, n. 2, pp 120–126, Feb 1978.

[3] B. E. Schneier, *Applied Cryptography*, 2nd ed. John Wiley & Sons, Inc. 1996.

[4] J. M. Acken and L. E. Nelson, "Statistical Basics for Testing and Security of Digital Systems for Identity Authentication," CCCT, International Institute of Informatics and Systematic, Orlando, FL, pp. 122-128, June 29th – July 2nd 2008.

[5] S. Sohoni, C.D. Shaver, J.M. Acken, D. Mertz, L. E. Nelson, J. Remington, B. Sadr, G. Sundararajan, "Evaluation Criteria for Biometric Based Identity Authentication Systems", *Proceedings of ISSSIS2009*, Coimbatore, India, 8-10 January 2009.

[6] On copyright duration: http://www.copyright.gov/title17/92chap3.pdf

[7] B. A. Forouzan, *Data Communications and Networking*, 4th ed. McGraw-Hill, 2007.

[8] M. Y. Rhee. *Cryptography and Secure Communications,* McGraw-Hill Book Co. – Singapore, 1994.

[9] D. G. Wells, *Prime Numbers: The Most Mysterious Figures in Math*, John Wiley & Sons, Inc. 2005.

[10] G. R. Blakely and I. Borosh, "Rivest-Shamir-Adleman Public Key Cryptosystems Do Not Always Conceal Messages," *Computers and Mathematics with Applications*, v. 5, n. 3, pp. 169-178, 1979.

APPENDICES

A-1: The Extensive Table of Holes

The extensive table of holes contains a collection of septuples and their corresponding holes. The original excel sheet containing this information has above 28,000 entries which is an equivalent of a 3,153 page document. Each row of data contains a septuple, number of holes for that septuple, left holes and right holes. The left and right holes are the message values to the left or to the right of the point of symmetry on the number axis where a hole has occurred. The point of symmetry for all odd n is at $\frac{n}{2}$. The septuples used in this thesis are highlighted in the table that follows. The upper and lower limits on p, q and e for the original excel sheet are:

$p \neq q$, $3 \leq p \leq 233$, $5 \leq q \leq 251$, $3 \leq e \leq$ phi or 89, and e coprime with phi.

Since the collection presented here, is taken from the exhaustive search above, the selected septuples will also satisfy the above conditions.

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes	
3,5,15,8,3,1,3	6	4 5 6	9 10 11	
3,5,15,8,5,3,5	12	2 3 4 5 6 7	8 9 10 11 12 13	
3,5,15,8,7,6,7	6	4 5 6	9 10 11	
3,5,15,8,11,4,3	**6**	**4 5 6**	**9 10 11**	☺
3,5,15,8,13,8,5	12	2 3 4 5 6 7	8 9 10 11 12 13	
5,11,55,40,3,2,27	6	10 11 21	34 44 45	
5,11,55,40,7,4,23	6	10 11 21	34 44 45	
5,11,55,40,11,3,11	30	4 5 6 9 10 11 14 15 16 19 20 21 24 25 26	29 30 31 34 35 36 39 40 41 44 45 46 49 50 51	
5,11,55,40,13,12,37	12	10 11 12 21 22 23	32 33 34 43 44 45	
5,11,55,40,17,14,33	12	10 11 12 21 22 23	32 33 34 43 44 45	
5,11,55,40,19,9,19	6	10 11 21	34 44 45	
5,11,55,40,23,4,7	**6**	**10 11 21**	**34 44 45**	☺
5,11,55,40,29,21,29	12	10 11 12 21 22 23	32 33 34 43 44 45	

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes
5,11,55,40,31,24,31	30	4 5 6 9 10 11 14 15 16 19 20 21 24 25 26	29 30 31 34 35 36 39 40 41 44 45 46 49 50 51
5,11,55,40,37,12,13	12	10 11 12 21 22 23	32 33 34 43 44 45
5,11,55,40,41,42,41	52	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27	28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
5,11,55,40,43,29,27	6	10 11 21	34 44 45
5,11,55,40,47,27,23	6	10 11 21	34 44 45
5,11,55,40,53,49,37	12	10 11 12 21 22 23	32 33 34 43 44 45
11,13,143,120,7,6,103	18	10 12 22 23 43 55 56 65 66	77 78 87 88 100 120 121 131 133
11,13,143,120,11,1,11	30	12 13 14 25 26 27 38 39 40 51 52 53 64 65 66	77 78 79 90 91 92 103 104 105 116 117 118 129 130 131
11,13,143,120,13,4,37	36	10 11 12 21 22 23 32 33 34 43 44 45 54 55 56 65 66 67	76 77 78 87 88 89 98 99 100 109 110 111 120 121 122 131 132 133
11,13,143,120,17,16,113	12	12 21 34 44 65 66	77 78 99 109 122 131
11,13,143,120,19,3,19	18	10 12 22 23 43 55 56 65 66	77 78 87 88 100 120 121 131 133
11,13,143,120,23,9,47	**6**	**12 65 66**	**77 78 131** ☺
11,13,143,120,29,7,29	12	12 21 34 44 65 66	77 78 99 109 122 131
11,13,143,120,31,8,31	74	3 4 9 10 12 13 14 16 17 22 23 25 26 27 29 30 35 36 38 39 40 42 43 48 49 51 52 53 55 56 61 62 64 65 66 68 69	74 75 77 78 79 81 82 87 88 90 91 92 94 95 100 101 103 104 105 107 108 113 114 116 117 118 120 121 126 127 129 130 131 133 134 139 140
11,13,143,120,37,4,13	36	10 11 12 21 22 23 32 33 34 43 44 45 54 55 56 65 66 67	76 77 78 87 88 89 98 99 100 109 110 111 120 121 122 131 132 133

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes
11,13,143,120,41,14,41	52	5 8 12 13 14 18 21 25 26 27 31 34 38 39 40 44 47 51 52 53 57 60 64 65 66 70	73 77 78 79 83 86 90 91 92 96 99 103 104 105 109 112 116 117 118 122 125 129 130 131 135 138
11,13,143,120,43,24,67	18	10 12 22 23 43 55 56 65 66	77 78 87 88 100 120 121 131 133
11,13,143,120,47,9,23	6	12 65 66	77 78 131
11,13,143,120,53,34,77	12	12 21 34 44 65 66	77 78 99 109 122 131
11,13,143,120,59,29,59	6	12 65 66	77 78 131
11,13,143,120,61,31,61	140	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71	72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141
11,13,143,120,67,24,43	18	10 12 22 23 43 55 56 65 66	77 78 87 88 100 120 121 131 133
11,13,143,120,71,42,71	30	12 13 14 25 26 27 38 39 40 51 52 53 64 65 66	77 78 79 90 91 92 103 104 105 116 117 118 129 130 131
11,13,143,120,73,59,97	36	10 11 12 21 22 23 32 33 34 43 44 45 54 55 56 65 66 67	76 77 78 87 88 89 98 99 100 109 110 111 120 121 122 131 132 133
11,13,143,120,79,52,79	18	10 12 22 23 43 55 56 65 66	77 78 87 88 100 120 121 131 133
11,13,143,120,83,74,107	6	12 65 66	77 78 131
11,13,143,120,89,66,89	12	12 21 34 44 65 66	77 78 99 109 122 131

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes
47,59,2773,2668,3,2,1779	6	235 236 471	2302 2537 2538
47,59,2773,2668,5,3,1601	6	235 236 471	2302 2537 2538
47,59,2773,2668,7,6,2287	6	235 236 471	2302 2537 2538
47,59,2773,2668,11,9,2183	6	235 236 471	2302 2537 2538
47,59,2773,2668,13,4,821	6	235 236 471	2302 2537 2538
47,59,2773,2668,17,1,157	**6**	**235 236 471**	**2302 2537 2538** ☺
47,59,2773,2668,19,7,983	6	235 236 471	2302 2537 2538
47,59,2773,2668,31,15,1291	6	235 236 471	2302 2537 2538
47,59,2773,2668,37,9,649	6	235 236 471	2302 2537 2538
47,59,2773,2668,41,27,1757	6	235 236 471	2302 2537 2538
47,59,2773,2668,43,21,1303	6	235 236 471	2302 2537 2538
47,59,2773,2668,53,50,2517	6	235 236 471	2302 2537 2538
47,59,2773,2668,61,42,1837	6	235 236 471	2302 2537 2538
47,59,2773,2668,67,28,1115	6	235 236 471	2302 2537 2538
47,59,2773,2668,71,45,1691	6	235 236 471	2302 2537 2538
47,59,2773,2668,73,31,1133	6	235 236 471	2302 2537 2538
47,59,2773,2668,79,22,743	6	235 236 471	2302 2537 2538
47,59,2773,2668,83,76,2443	6	235 236 471	2302 2537 2538
47,59,2773,2668,89,45,1349	6	235 236 471	2302 2537 2538
47,67,3149,3036,5,4,2429	6	469 470 939	2210 2679 2680
47,67,3149,3036,17,5,893	6	469 470 939	2210 2679 2680
47,67,3149,3036,29,13,1361	6	469 470 939	2210 2679 2680
47,67,3149,3036,41,20,1481	6	469 470 939	2210 2679 2680
47,67,3149,3036,53,7,401	6	469 470 939	2210 2679 2680
47,67,3149,3036,59,24,1235	6	469 470 939	2210 2679 2680
47,67,3149,3036,71,46,1967	6	469 470 939	2210 2679 2680
47,67,3149,3036,83,19,695	6	469 470 939	2210 2679 2680
53,83,4399,4264,3,2,2843	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,7,6,3655	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,11,3,1163	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,19,7,1571	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,23,5,927	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,31,20,2751	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,43,6,595	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,47,29,2631	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,59,11,795	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,67,14,891	6	582 1908 1909	2490 2491 3817
53,83,4399,4264,71,53,3183	6	582 1908 1909	2490 2491 3817

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes
97,109,10573,10368,7,6,8887	46	326 327 546 872 873 935 1199 1200 1262 1745 1807 1808 2072 2134 2135 2461 2680 3006 3007 3333 3334 3879 4206	6367 6694 7239 7240 7566 7567 7893 8112 8438 8439 8501 8765 8766 8828 9311 9373 9374 9638 9700 9701 10027 10246 10247
97,109,10573,10368,11,9,8483	6	872 873 1745	8828 9700 9701
101,103,10403,10200,7,6,8743	18	102 2322 2323 2425 2827 2828 2930 5150 5151	5252 5253 7473 7575 7576 7978 8080 8081 10301
101,103,10403,10200,11,7,6491	30	102 309 721 825 926 1854 3297 3398 4326 4430 4531 4842 4943 5150 5151	5252 5253 5460 5561 5872 5973 6077 7005 7106 8549 9477 9578 9682 10094 10301
101,103,10403,10200,13,8,6277	32	102 515 1808 2313 2322 2323 2425 2827 2828 2838 2930 3343 4636 4737 5150 5151	5252 5253 5666 5767 7060 7473 7565 7575 7576 7978 8080 8081 8090 8595 9888 10301
101,103,10403,10200,19,13,6979	18	102 2322 2323 2425 2827 2828 2930 5150 5151	5252 5253 7473 7575 7576 7978 8080 8081 10301
101,103,10403,10200,23,2,887	6	102 5150 5151	5252 5253 10301
101,103,10403,10200,29,11,3869	12	102 515 4636 4737 5150 5151	5252 5253 5666 5767 9888 10301

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes
101,103,10403,10200,31,30,9871	74	102 309 469 721 825 926 974 1498 1602 1854 2003 2014 2107 2322 2323 2425 2519 2632 2827 2828 2930 3044 3137 3249 3297 3398 3549 3754 4177 4326 4430 4531 4682 4842 4943 5150 5151	5252 5253 5460 5561 5721 5872 5973 6077 6226 6649 6854 7005 7106 7154 7266 7359 7473 7575 7576 7771 7884 7978 8080 8081 8296 8389 8400 8549 8801 8905 9429 9477 9578 9682 9934 10094 10301
101,103,10403,10200,37,34,9373	32	102 515 1808 2313 2322 2323 2425 2827 2828 2838 2930 3343 4636 4737 5150 5151	5252 5253 5666 5767 7060 7473 7565 7575 7576 7978 8080 8081 8090 8595 9888 10301
101,103,10403,10200,41,32,7961	60	102 309 515 721 825 926 1648 1854 1958 2059 2061 2162 2266 2885 2986 3090 3193 3297 3398 3503 3604 4326 4430 4531 4636 4737 4842 4943 5150 5151	5252 5253 5460 5561 5666 5767 5872 5973 6077 6799 6900 7005 7106 7210 7313 7417 7518 8137 8241 8342 8344 8445 8549 8755 9477 9578 9682 9888 10094 10301
101,103,10403,10200,43,19,4507	18	102 2322 2323 2425 2827 2828 2930 5150 5151	5252 5253 7473 7575 7576 7978 8080 8081 10301
101,103,10403,10200,47,46,9983	6	102 5150 5151	5252 5253 10301
101,103,10403,10200,53,11,2117	12	102 515 4636 4737 5150 5151	5252 5253 5666 5767 9888 10301
101,103,10403,10200,59,17,2939	6	102 5150 5151	5252 5253 10301

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes
101,103,10403,10200,61,14,2341	144	57 102 262 309 365 469 515 562 675 721 767 825 870 926 974 1180 1498 1602 1648 1808 1854 1958 2003 2014 2059 2061 2107 2162 2266 2313 2322 2323 2425 2519 2632 2827 2828 2838 2885 2930 2986 3044 3090 3137 3193 3249 3297 3343 3398 3503 3549 3604 3754 3971 4177 4326 4382 4430 4476 4485 4531 4589 4636 4682 4737 4842 4887 4943 4990 5094 5150 5151	5252 5253 5309 5413 5460 5516 5561 5666 5721 5767 5814 5872 5918 5927 5973 6021 6077 6226 6432 6649 6799 6854 6900 7005 7060 7106 7154 7210 7266 7313 7359 7417 7473 7518 7565 7575 7576 7771 7884 7978 8080 8081 8090 8137 8241 8296 8342 8344 8389 8400 8445 8549 8595 8755 8801 8905 9223 9429 9477 9533 9578 9636 9682 9728 9841 9888 9934 10038 10094 10141 10301 10346
101,103,10403,10200,67,46,7003	18	102 2322 2323 2425 2827 2828 2930 5150 5151	5252 5253 7473 7575 7576 7978 8080 8081 10301
101,103,10403,10200,71,3,431	30	102 309 721 825 926 1854 3297 3398 4326 4430 4531 4842 4943 5150 5151	5252 5253 5460 5561 5872 5973 6077 7005 7106 8549 9477 9578 9682 10094 10301
101,103,10403,10200,73,11,1537	32	102 515 1808 2313 2322 2323 2425 2827 2828 2838 2930 3343 4636 4737 5150 5151	5252 5253 5666 5767 7060 7473 7565 7575 7576 7978 8080 8081 8090 8595 9888 10301
101,103,10403,10200,79,35,4519	18	102 2322 2323 2425 2827 2828 2930 5150 5151	5252 5253 7473 7575 7576 7978 8080 8081 10301
101,103,10403,10200,83,37,4547	6	102 5150 5151	5252 5253 10301

(p,q,n,Phi,e,k,d)	#holes	Left holes	Right holes
101,103,10403,10200,89,28,3209	12	102 515 4636 4737 5150 5151	5252 5253 5666 5767 9888 10301
233,251,58483,58000,73,48,38137	24	1254 2008 3262 3263 5270 6525 16065 19327 20834 22589 24096 27358	31125 34387 35894 37649 39156 42418 51958 53213 55220 55221 56475 57229
233,251,58483,58000,79,62,45519	6	3262 3263 6525	51958 55220 55221
233,251,58483,58000,83,44,30747	6	3262 3263 6525	51958 55220 55221
233,251,58483,58000,89,35,22809	24	1254 2008 3262 3263 5270 6525 16065 19327 20834 22589 24096 27358	31125 34387 35894 37649 39156 42418 51958 53213 55220 55221 56475 57229
_____ End of Key Generation. _____			

A-2 The Perl Program Listing

Here is a listing of the Perl program used in this research. This program will find the holes for a list of triples (p, q, e) and encrypt the ASCII equivalent of the message “ABoV;-)” which is 32, 65, 66, 111, 86, 59, 45,41 using each of these 10 encryption keys. In each of these 10 iterations, the holes, the total number of holes and the percentage of holes are found and printed.

```
#
# RSA encryption check for cleartext
#          File name:      HoleFinder2329FromList.pl
# Original Author:      John M Acken
#   Current Author:     Behnaz Sadr
# Revision history:     initial version 20 May 2010
#                       Current version 21 May 2011
#
# Description: This program checks for holes in RSA.
# That means looping through integers comparing the cleartext
# to the ciphertext.
#
# ## A hole has been found when the cleartext equals ciphertext. ##
#
# This program requires a list of p's, q's and e's as input.
# The output is the list of plaintexts that are holes for a given ordered triple (p, q, e)
#
##########################################################################
# ...~Desktop\perl> bin\perl HoleFinder2329FromList2.pl > t4holes23_?.txt
##########################################################################
#
# First the list of p and q values are entered as two arrays.
# There are 10 values of e, each one used with the same two prime factors p and q.
# In other words this program will encrypt 8 values of m using 10 different keys with
# a common n.
#
# Each key is e in conjunction with n: (e, n)
#
# In this program p = 23 and q = 29.

@listOfp = (

23, 23, 23, 23, 23,
23, 23, 23, 23, 23

);
```

```
@listOfq = (

29, 29, 29, 29, 29,
29, 29, 29, 29, 29

);

# Values of e are chosen such that they are: odd, between 3 and phi, and they are coprime
# with phi. In this example phi is 22x28. The factors for phi are: 2, 7 & 11. These
# are invalid candidates for e.
#
# Another rule of thumb for choosing e is even though choosing the values of p or q for e
# is admissible since they might be odd and coprime with phi, they usually create a lot
# of holes. This program demonstrates this fact, as well as being an example.
# So as you may notice, two of the e values are 23 and 29.

@listOfe = (

3,      5,      13,     17,     19,
23,     29,     31,     37,     41

);

# After the three arrays @listofp, @listOfq and @listOfe are defined and filled
# with values, the ASCII values for a Test message " ABoV;-)" is kept in @message
# and will be printed on top of the output file.

@message = ( 32, 65, 66, 111, 86, 59, 45, 41);

print STDOUT "\n Test message  = @message      \n";
print STDOUT " ________________________________________\n";
print STDOUT "\n__________________________________________________\n";


# Next, an outer for loop is formed to be executed as many times as the number of e
# values which in this case is 10. The outer for loop's counter is $example.
# The counter is incremented one at a time.
# In each iteration of the outer for loop the values of p, q and e are taken from their
# respective place in their array and n, n/2 and phi are calculated.
# The number of holes is initialized to zero.

for ($example = 0; $example < 10; $example++)  {  # beginning of the outer for loop
        $p = $listOfp[$example];                # put the next element of listofp in $p
        $q = $listOfq[$example];
        $n = $p * $q;
        $midpoint = $n / 2;                     # midpoint is defined as n/2
        $phi = ($p - 1) * ($q - 1);            # phi is defined as (p-1)(q-1)
        $e = $listOfe[$example];
        $holes = 0;                            # initialize the number of holes to zero

# Next comes the information that will be printed out for each value of e:
# Values of p, q, n and e are printed.

        print STDOUT "____________________________________";
        print STDOUT "For example $example p = $p; q = $q; n = p*q = $n; e = $e \n";

# This for loop goes through the 8 elements of the @message array one by one
#
#
        for ($BN = 0; $BN < 8; $BN++ ) {               # beginning of the first inner for loop
                $m = $message[$BN];
                $c  = ($m * $m) % $n;                  # c = m*m mod n ,
                                                       # to find c, m will be
                                                       # multiplied by it self e times

                for ($BN2 = 2; $BN2 < $e; $BN2++ ) {
                                                       # beginning of innermost for loop

                        $c = ($m * $c) % $n;          # each time multiplying the last found
                                                       # product by m and take modulus n
                                                       # eventually c = m^e mod n
                }                                      # end if innermost for loop
                print STDOUT " $m => $c ||";           # print the message value m and
                                                       # its corresponding ciphertext c
                                                       # separate from the next m,c, by "||"

        }                                              # end of the first inner for loop
#
# At this stage we will have a list of all 8 m values from the list @message as well as
# their encryptions.
#
#
        print STDOUT "\n ____________________ \n List of holes: \n";
#
# The following nested for loops and their corresponding if statements will encrypt m
# and compare the encrypted value c with m. Whenever this encrypted value becomes
# equal to m i.e. c = m, two things are done: first, the number of holes is incremented,
# second the value of m creating the hole is printed followed or preceded by a pipe '|'
# character.
# The position of the pipe is chosen in such a way that there will be two pipe characters
# representing the midpoint. At the end of this second inner for loop the total number of
# holes for a given e value is calculated and printed. Having the total number of holes by
# now, the total number of holes as well as the percentage of holes are printed.

    for ($m = 2; $m < $n - 1; $m++) {     # beginning of the second inner for loop

        $c  = ($m * $m) % $n;
        for ($BN2 = 2; $BN2 < $e; $BN2++ ) {
                                          # beginning of the 2nd innermost for loop

            $c = ($m * $c) % $n;
        }                                 # end of second innermost for loop
                                          # the previous three lines are exactly
                                          # the same as the for loop used on top.

        if ($m == $c) {                   # if there is a hole
            $holes++;                     # There will be a "|" after holes before
                                          # reaching the midpoint n/2
                                          # when there is a hole after the midpoint
                                          # the "|" is printed before the hole.
                                          # In this way there will be a double pipe
                                          # in the middle, representing the midpoint.

            if ($m > $midpoint) {print STDOUT "|";}
                                          # if the hole happened where m > n/2,
                                          # print "|"
            print STDOUT " $m";
                                          # Now print m
            if ($m < $midpoint) {print STDOUT "|";}
                                          # if the hole happened where m < n/2,
                                          # print "|"
        }                                 # end of if statement
    }                                     # end of the second inner for loop
    $percent = int(10000 * $holes / $n) / 100;
                                          # percentage of holes are calculated
    print STDOUT "\n number of holes = $holes      = $percent %\n";
                                          # number of holes and percentage
                                          # of holes are printed.
    print STDOUT "_____________________________________________\n";
                                          # This line shows the end of one
                                          # iteration of e.
}                                         # end of the outer for loop
                                          # Printing the next line shows the end of
                                          # the program that ran for a given list of e
                                          # in our case, 10 e values.
print STDOUT "\n ____________________  All Done. ____________________\n";
exit;                                     # end of program.
```

A-3: Simple Example in RSA

There are two simple examples in chapter three. These examples use the septuple, (11, 13, 143, 120, 23, 9, 47). The text used in example 2 is: “I loVe cRyptoGraphy! ;-)” The block size is one letter. This text is turned into a large integer using its ASCII code:

I		l	o	V	e		c	R	y	p	t	o	G	r	a	p	h	y	!		;	-	)
73	32	108	111	86	101	32	99	82	121	112	116	111	71	114	97	112	104	121	33	32	59	45	41

In order to encrypt the plaintext message above, the Table A-3.1 is used.

Table A-3.1 Encryption Table for Septuple (11, 13, 143, 120, 23, 9, 47)

m	c	m	c	m	c	m	c	m	c
		31	47	61	29	91	104	121	88
2	85	32	76	62	134	92	53	122	34
3	126	33	132	63	6	93	59	123	63
4	75	34	122	64	25	94	139	124	93
5	125	35	107	65	65	95	101	125	31
6	128	36	82	66	66	96	138	126	81
7	2	37	97	67	111	97	102	127	95
8	83	38	103	68	74	98	54	128	123
9	3	39	117	69	49	99	44	129	116
10	43	40	79	70	86	100	133	130	91
11	110	41	72	71	37	101	30	131	131
12	12	42	113	72	106	102	71	132	33
13	52	43	10	73	57	103	64	133	100
14	27	44	99	74	94	104	26	134	140
15	20	45	89	75	69	105	40	135	60
16	48	46	41	76	32	106	46	136	141
17	62	47	5	77	77	107	61	137	15
18	112	48	42	78	78	108	36	138	18
19	50	49	4	79	118	109	21	139	68
20	80	50	84	80	137	110	11	140	17
21	109	51	90	81	9	111	67	141	58
22	55	52	39	82	114	112	96		
23	56	53	14	83	73	113	16		
24	19	54	98	84	24	114	108		
25	38	55	22	85	28	115	136		
26	130	56	23	86	135	116	51		
27	92	57	8	87	120	117	13		
28	7	58	115	88	121	118	105		
29	35	59	119	89	45	119	124		
30	127	60	70	90	129	120	87		

The large integer is encrypted as:

57 76 36 67 135 30 76 44 114 88 96 51 67 37 108 102 96 26 88 132 76 119 89 72

This integer if it was converted to text would read as:
“9L$CoorRSL,rX`3C%lf`SubsXoorLwYH” where the underlined characters are special characters which are not letters.

Table 3.7 is reprinted here to summarize the steps done in encrypting and decrypting the simple example, “I loVe cRyptoGraphy! ;-)”.

Table 3.7 Sample Message Transformation Table

Original Plaintext	I		l	o	V	e		c	R	y	p	t	o	G	r	a	p	h	y	!		;	-	)
Encoded (ASCII)	73	32	108	111	86	101	32	99	82	121	112	116	111	71	114	97	112	104	121	33	32	59	45	41
Encrypted (ASCII)	57	76	36	67	135	30	76	44	114	88	96	51	67	37	108	102	96	26	88	132	76	119	89	72
Ciphertext	9	L	$	C	oor	RS	L	,	r	X	`	3	C	%	l	f	`	Subs	X	oor	L	w	Y	H
Received (ASCII)	57	76	36	67	135	30	76	44	114	88	96	51	67	37	108	102	96	26	88	132	76	119	89	72
Decrypted (ASCII)	73	32	108	111	86	101	32	99	82	121	112	116	111	71	114	97	112	104	121	33	32	59	45	41
Decoded Message	I		l	o	V	e		c	R	y	p	t	o	G	r	a	p	h	y	!		;	-	)

VITA

Behnaz Sadr

Candidate for the Degree of

Master of Science

Thesis: FINDING CASES OF CIPHERTEXT EQUAL TO PLAINTEXT IN THE RSA ALGORITHM

Major Field: Electrical Engineering

Biographical:

Education:
Completed the requirements for the Master of Science in Electrical Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2011.

Experience:
2007 – 2011 Teaching Assistant, Oklahoma State University, Tulsa, Oklahoma
2005 – 2007 Math Tutor, Oklahoma State University, Tulsa, Oklahoma

Name: Behnaz Sadr Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: FINDING CASES OF CIPHERTEXT EQUAL TO PLAINTEXT IN THE RSA ALGORITHM

Pages in Study: 58 Candidate for the Degree of Master of Science

Major Field: Electrical Engineering

Scope and Method of Study: The RSA Algorithm is the most widely used public key encryption method that has survived the past 34 years of scrutiny and criticism. Studying and understanding RSA is important for a student of cryptography or information security. Finding a good example that is small enough for easy standard calculation and large enough to be non-trivial was the main and initial goal of this study.

Findings and Conclusions: In the course of studying the RSA algorithm, there were cases where the ciphertext would equal the plaintext. In other words, at these values, the encryption would not change the value of the original message as if the plaintext is falling through a hole and not get encrypted. These cases were called holes. characterization of the holes become the second objective of this study.

ADVISER’S APPROVAL: Dr. John M. Acken