

**A NOVEL METHOD OF ENCRYPTION USING MODIFIED RSA
ALGORITHM AND CHINESE REMAINDER THEOREM**

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF**

Bachelor of Technology

In

Electronics and Communication Engineering

By

SANGEETA PATEL(10509017)

And

PARTHA PRITAM NAYAK(10509021)

Under the guidance of

Prof G.S Rath



Department of Electronics and Communication Engineering

National Institute of Technology

Rourkela-769008



National Institute of Technology

Rourkela

CERTIFICATE

This is to certify that the thesis entitled, “A NOVEL METHOD OF ENCRYPTION USING MODIFIED RSA ALGORITHM AND CHINESE REMAINDER THEOREM” submitted by SANGEETA PATEL and PARTHA PRITAM NAYAK in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Electronics & Communication Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof.G. S. Rath

Dept. of Electronics & Communication Engineering

National Institute of Technology

Rourkela-769008

ACKNOWLEDGEMENT

We would like to articulate our deep gratitude to our project guide Prof. G.S.Rath who has always been our motivation for carrying out the project. An assemblage of this nature could never have been attempted without reference to and inspiration from the works of others whose details are mentioned in reference section. We acknowledge our indebtedness to all of them. Last but not the least, our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.

SANGEETA PATEL

(Roll No: 10509017)

PARTHA PRITAM NAYAK

(Roll No: 10509021)

CONTENTS

		Page No
	ABSTRACT	1
Chapter 1	INTRODUCTION	2-3
Chapter 2	NUMBER THEORY	4
	2.1 Introduction	4-5
	2.2 Modular Arithmetic	5-6
	2.3 Euclidean algorithm	6
	2.4 Prime Numbers	6-7
	2.5 Fermat's and Euler's Theorem	7-8
	2.6 Chinese Remainder Theorem	8-11
Chapter 3	REVIEWS	12
	3.1 Applications of Chinese Remainder Theorem	13-14
	3.2 Hill Cipher and its applications	14
Chapter 4	RSA AND KEY ISSUES	15
	4.1 Introduction	16
	4.2 The Basic Principle	17-18
	4.3 Advantages and Disadvantages of Public Key	18-19
	4.4 The RSA Algorithm	19-23
	4.5 RSA using Chinese Remainder Theorem	23-24
	4.6 RSA Decryption using CRT	25-27
	4.7 MAPLE with MATLAB	27-29
	4.8 The RSA Cryptosystem with MAPLE	29-32

Chapter 5	IMPLEMENTATION & RESULTS	33
5.1	RSA without CRT	34
5.2	RSA with CRT	35-36
5.3	RSA in Variable Radix Number System	36-37
	CONCLUSION	38
	REFERENCES	39

ABSTRACT

Security can only be as strong as the weakest link. In this world of cryptography, it is now well established, that the weakest link lies in the implementation of cryptographic algorithms. This project deals with RSA algorithm implementation with and without Chinese Remainder Theorem and also using Variable Radix number System. In practice, RSA public exponents are chosen to be small which makes encryption and signature verification reasonably fast. Private exponents however should never be small for obvious security reasons. This makes decryption slow. One way to speed things up is to split things up, calculate modulo p and modulo q using Chinese Remainder Theorem. For smart cards which usually have limited computing power, this is a very important and useful technique. This project aims at implementing RSA algorithm using Chinese Remainder Theorem as well as to devise a modification using which it would be still harder to decrypt a given encrypted message by employing a Variable radix system in order to encrypt the given message at the first place.

Chapter 1

INTRODUCTION

Cryptography, defined as *the science and study of secret writing* concerns the ways in which communications and data can be encoded to prevent disclosure of their contents through eavesdropping or message interception, using codes, ciphers and other methods, so that only certain people can see the real message.

Security often requires that data be kept safe from unauthorized access. And the best line of defence is physical security (placing the machine to be protected behind physical walls). However, physical security is not always an option, due to cost and/or efficiency considerations. Instead, most computers are interconnected with each other openly, thereby exposing them and the communication channels that they use. With regards to confidentiality, cryptography is used to encrypt data residing on storage devices or travelling through communication channels to ensure that any illegal access is not successful. Also, cryptography is used to secure the process of authenticating different parties attempting any function on the system. Since a party wishing be granted a certain functionality on the system must present something that proves that they indeed who they say they are. That something is sometimes known as credentials and additional measures must be taken to ensure that these credentials are only used by their rightful owner. The most classic and obvious credential are passwords. Passwords are encrypted to protect against illegal usage.

Authorization is a layer built on top of authentication in the sense that the party is authenticated by presenting the credentials required (passwords, smart cards ... etc.). After the credentials are accepted the authorization process is started to ensure that the requesting party has the permissions to perform the functions needed.

Data integrity and Non-Repudiation are achieved by means of digital signature, a method that includes performing cryptography among other things.

Cryptography can essentially be classified into two types, the symmetric and asymmetric type. With a secret or symmetric key algorithm, the key is a shared secret between two communicating parties. Encryption and decryption both use the same key. The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are examples of symmetric key algorithms.

With a public key (PKA) or asymmetric key algorithm, a pair of keys is used. One of the keys, the private key, is kept secret and not shared with anyone. The other key, the public key, is not secret and can be shared with anyone. When data is encrypted by one of the keys, it can only be decrypted and recovered by using the other key. The two keys are mathematically related, but it is virtually impossible to derive the private key from the public key. The RSA algorithm is an example of a public key algorithm.

Chapter 2

NUMBER THEORY

2.1 Introduction

A number of mathematical concepts from number theory are essential in the design of cryptographic algorithms. This chapter provides an overview of the concepts along with the proofs of the theorems used in these algorithms. The various theorems have been elucidated which are further applied in RSA algorithm in our plan of work.

2.2 Modular Arithmetic

Modular arithmetic is a system of arithmetic for integers, where numbers *wrap around* after they reach a certain value - the *modulus*. Given any positive integer n and any nonnegative integer a , if we divide a by n , we get an integer quotient q and an integer remainder r that obey the following relationship:

$a = qn + r \quad 0 \leq r < n; q = \lfloor a/n \rfloor$ where x is the largest integer less than or equal to x . The remainder r is often referred to as a residue. If a is an integer and n is a positive integer, we define $a \bmod n$ to be the remainder when a is divided by n . The integer n is called the modulus. Thus, for any integer a , we can always write:

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

hence $12 \bmod 7 = 5$; $-12 \bmod 7 = 2$

Two integers a and b are said to be congruent modulo n , if $(a \bmod n) = (b \bmod n)$. This is written as $a \equiv b \pmod{n}$, for example $73 \equiv 4 \pmod{23}$.

Properties of Modular Arithmetic for Integers

Property	Expression
Commutative laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Associative laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive laws	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$ $[w + (x \times y)] \bmod n = [(w + x) \times (w + y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$

Additive inverse For each $w \in \mathbb{Z}_n$, there exists a z such that $w + z \equiv 0 \pmod n$

2.3 Euclidean Algorithm

Euclidean algorithm is a simple procedure for determining the greatest common divisor of two positive integers. Nonzero b is defined to be a divisor of a if $a = mb$ for some m , where a , b , and m are integers. We will use the notation $\gcd(a, b)$ to mean the greatest common divisor of a and b . The positive integer c is said to be the greatest common divisor of a and b if c is a divisor of a and of b and any divisor of a and b is a divisor of c .

An equivalent definition is the following: $\gcd(a, b) = \max\{k, \text{ such that } k|a \text{ and } k|b\}$.

Because we require that the greatest common divisor be positive, $\gcd(a, b) = \gcd(|a|, |b|)$.

Also, because all nonzero integers divide 0, we have $\gcd(a, 0) = |a|$.

We stated that two integers a and b are relatively prime if their only common positive integer factor is 1. This is equivalent to saying that a and b are relatively prime if $\gcd(a, b) = 1$.

The Euclidean algorithm is based on the following theorem: For any nonnegative integer a and any positive integer b , $\gcd(a, b) = \gcd(b, a \bmod b)$

2.4 Prime Numbers

A central concern of number theory is the study of prime numbers. Indeed, whole books have been written on the subject. An integer $p > 1$ is a prime number if and only if its only divisors are ± 1 and $\pm p$. Prime numbers play a critical role in number theory and in the cryptographic techniques discussed later. If P is the set of all prime numbers, then any positive integer a can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \quad \text{where each } a_p \geq 0$$

The right-hand side is the product over all possible prime numbers p ; for any particular value of a , most of the exponents a_p will be 0.

It is easy to determine the greatest common divisor of two positive integers if we express each integer as the product of primes.

Example: $300 = 2^2 \times 3^1 \times 5^2$

$$18 = 2^1 \times 3^2$$

$$\text{Gcd}(18,300) = 2^1 \times 3^1 \times 5^0 = 6$$

For many cryptographic algorithms, it is necessary to select one or more very large prime numbers at random. Thus we are faced with the task of determining whether a given large number is prime. There is no simple yet efficient means of accomplishing this task.

Miller-Rabin Algorithm yields a number that is not necessarily a prime. However, the algorithm can yield a number that is almost certainly a prime. It is based on the conclusion that if n is prime, then either the first element in the list of residues, or remainders, $(a^q, a^{2q}, \dots, a^{2^{k-1}q}, a^{2^kq})$ modulo n equals 1, or some element in the list equals $(n-1)$; otherwise n is composite (i.e., not a prime). On the other hand, if the condition is met, that does not necessarily mean that n is prime. For example, if $n = 2047 = 23 \times 89$, then $n-1 = 2 \times 1023$. Computing, $2^{1023} \bmod 2047 = 1$, so that 2047 meets the condition but is not prime.

2.5 Fermat's And Euler's Theorems

Two theorems that play important roles in public-key cryptography are Fermat's theorem and Euler's theorem. Fermat's theorem states the following: *If p is prime and a is a positive integer not divisible by p , then $a^{p-1} \equiv 1 \pmod{p}$*

Before presenting Euler's theorem, we need to introduce an important quantity in number theory, referred to as *Euler's totient function* and written $\phi(n)$, defined as the number of positive integers less than n and relatively prime to n . By convention, $\phi(1) = 1$.

Example: $\phi(37) = 36$ and $\phi(35) = 24$ for a prime number p , $\phi(p) = p-1$

Now suppose that we have two prime numbers p and q , with p not equal q . Then it can be show that for $n = pq$, $\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$

Euler's theorem states that for every a and n that are relatively prime:

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Example:

$$a = 2; n = 11; \varphi(11) = 10 \quad a^{\varphi(n)} = 2^{10} = 1024 \equiv 1 \pmod{11} = 1 \pmod{n}$$

2.6 The Chinese Remainder Theorem

In essence the Chinese remainder theorem (CRT) says it is possible to reconstruct integers in a certain range from their residues modulo a set of pair wise relatively prime moduli.

The integers 0 through 9 in \mathbb{Z}_{10} , can be reconstructed from their two residues modulo 2 and 5 (the relatively prime factors of 10). Say the known residues of a decimal digit $x \pmod{2} = 0$ and $x \pmod{5} = 3$. Therefore, x is an even integer in \mathbb{Z}_{10} whose remainder, on division by 5, is 3. The unique solution is $x = 8$.

One of the useful features of the Chinese remainder theorem is that it provides a way to manipulate (potentially very large) numbers mod M in terms of tuples of smaller numbers. This can be useful when M is 150 digits or more. However, note that it is necessary to know beforehand the factorization of M .

Theorem 1. Let m and n be integers with $\gcd(m, n) = 1$, $M = mn$ and let b and c be any integers. Then the simultaneous congruences

$$x \equiv b \pmod{m} \text{ and } x \equiv c \pmod{n}$$

have exactly one solution with $0 \leq x < M$.

Proof: We begin by solving the congruences $x \equiv b \pmod{m}$. The solution consists of all numbers of the form $x = my + b$. We substitute this into second congruence, which yields

$$my \equiv c - b \pmod{n}.$$

We are given that $\gcd(m, n) = 1$, so the linear congruence Theorem tells us that there is exactly one solution y_1 with $0 \leq y_1 < n$. Then the solution to the original is given by

$x_1 = my_1 + b$; and this will be the only solution x_1 with $0 \leq x_1 < M$, since there is only one y_i between 0 and n , and we multiplied y_1 by m to get x_1 . This completes the proof.

Examples:

1. Suppose we want to solve $x \equiv 8 \pmod{11}$ and $x \equiv 3 \pmod{19}$.

As stated in the proof, we write the solutions of the first congruence in the form of $x=11y + 8$ and substitute it into the second congruence, which yields $11y \equiv -5 \pmod{19}$ which is equal to $11y \equiv 14 \pmod{19}$ and equal to $11y \equiv 33 \pmod{19}$. Then we divide both sides of the congruences by 11 and we get $y \equiv 3 \pmod{19}$, now we can find the solution to the first congruence, $x=11y+8 = 11(3) + 8 = 41$.

Finally we want to check whether our answer is accurate, so substitute 41 for x and see that $41 \equiv 8 \pmod{11} = 33 = 0 \pmod{11}$ and $41 \equiv 3 \pmod{19} = 38 \equiv 0 \pmod{19}$.

2. Now, suppose that we want to solve three simultaneous congruences:

$$x \equiv 2 \pmod{3}, x \equiv 1 \pmod{5} \text{ and } x \equiv 3 \pmod{7}$$

We write the solutions to the first congruence as $x=3y+2$ and substitute it into the second and get $3y \equiv -1 \pmod{5}$ which is equivalent to $3y \equiv 4 \pmod{5}$ and equal to $3y \equiv 9 \pmod{5}$. If we divide by 3 we get that $y \equiv 3 \pmod{5}$ which can be rewritten as $y = 5z + 3$.

3. Now if we substitute the solutions to the first congruence into the third, we get $3y \equiv 1 \pmod{7}$, which is equivalent to $3(5z + 3) \equiv 1 \pmod{7}$ when substituting y . Evaluating, we get that $15z \equiv -8 \pmod{7}$ which is equal to $15z \equiv 6 \pmod{7}$. When we divide by 15 we get $z \equiv 6 \pmod{7}$.

When substituting we get that $x=3y+2 = 3[5(6) + 3] + 2 = 101$.

Finally to check our solution of x , we noted that $101 \equiv 2 \pmod{3} = 99 \equiv 0 \pmod{3}$; $101 \equiv 1 = 100 \equiv 0 \pmod{5}$; and $101 \equiv 3 = 98 \equiv 0 \pmod{7}$.

An alternative way to solve these congruences is the following:

Using the theorem, we get that $M=(3)(5)(7)=105$. Let $m_1 = 3, m_2 = 5$, and $m_3 = 7$; now let $b_1 = 2, b_2 = 1$ and $b_3 = 3$. The integers y_1, y_2 and y_3 are found by the congruence

$$\frac{M}{m_i} y_i \equiv 1 \pmod{m_i},$$

Thus we have $35y_1 \equiv 1 \pmod{3}$, $21y_2 \equiv 1 \pmod{5}$, and $15y_3 \equiv 1 \pmod{7}$. so $y_1 = -1, y_2 = 1$, and $y_3 = 1$ are possible values and

$$X = (35)(-1)(2) + (21)(1)(1) + (15)(1)(3) = -4 \pmod{105} = 101.$$

Thus, this leads us to define x as

$$x = \left(\sum_{i=1}^{\alpha} \frac{M}{m_i} y_i b_i \right) \pmod{M}.$$

Remarks: Notice that in the congruences $\frac{M}{m_i} y_i \equiv 1 \pmod{m_i}$, y_i is the multiplicative inverse of $\frac{M}{m_i}$ modulus m_i .

Extension of the Theorem to polynomials

When trying to extend the definition of the Chinese Remainder Theorem to polynomials we presented a problem of the following kind:

Example 1.

Assume you have a polynomial that when it is divided by $(x-1)$ you get remainder 3, when it is divided by $(x-2)$ you get remainder 2 and when it is divided by $(x-3)$, you get remainder -1. Find such a polynomial?

Using the theorem, we get $g(x) = (x-1)(x-2)(x-3)$. Notice that $P(x)$ can be a polynomial of degree at most 3.

Let $m_1 = x - 1, m_2 = x - 2$, and $m_3 = x - 3$. Now let $b_1 = 3, b_2 = 2$ and $b_3 = -1$. The polynomials y_1, y_2 , and y_3 (degree 0 in this particular case) are found by the congruence

$$\frac{M}{m_i} y_i \equiv 1 \pmod{m_i},$$

We now have $(x^2 - 5x + 6)y_1 \equiv 1 \pmod{(x - 1)}$,
 $(x^2 - 4x + 3)y_2 \equiv 1 \pmod{(x - 2)}$, and $(x^2 - 3x + 2)y_3 \equiv 1 \pmod{(x - 3)}$. So
 $y_1 = \frac{1}{2}$, $y_2 = -1$, and $y_3 = \frac{1}{2}$ are possible values and $P = (x-2)(x-3)(1/2)(3) + (x-1)(x-3)(-1)(2)$
 $+ (x-1)(x-2)(1/2)(-1) = -x^2 + 2x + 2$.

Thus notice the theorem could be extended to polynomials as long as the moduli m_i are relatively prime to each other.

Theorem 2. Let $m_1(x), m_2(x), \dots, m_r(x)$ denote r prime polynomials of degree p ($p \geq 1$) that are relatively prime in pairs, and let b_1, b_2, \dots, b_r denote any r prime polynomials of degrees at most $p-1$. Then the system of congruences $x \equiv b_i \pmod{m_i(x)}, i = 1, 2, \dots, r$ has a unique solution modulo $g(x)$, where

$$g(x) = \prod_{i=1}^r m_i(x).$$

Theorem 3⁴. Let $m_1(x), m_2(x), \dots, m_r(x)$ denote r prime polynomials of degree p ($p \geq 1$) that are relatively prime in pairs, and let b_1, b_2, \dots, b_r denote any r prime polynomials of degrees at most $p-1$. Then the system of congruences $x \equiv b_i \pmod{m_i(x)}, i = 1, 2, \dots, r$ has a unique solution modulo $g(x)$, where

$$g(x) = \prod_{i=1}^r m_i(x).$$

Proof: Let $g(x)$ denote the polynomial obtained by multiplying together all the $m_i(x)$, since $m_i(x)$ is a monic polynomial of degree one we can write it as $(x - a_i)$, where a_i is one of the x -coordinates of our points. For each point (a_i, b_i) , we can write the expression $\frac{b_i}{g'(a_i)} \cdot \frac{g(x)}{(x - a_i)}$

for which $\frac{g(x)}{(x - a_i)}$ is just $\frac{M}{m_i}$ and $\frac{1}{g'(a_i)}$ is algorithm for finding y_i . We now have our familiar

$x = \left(\sum_{i=1}^{\alpha} \frac{M}{m_i} y_i b_i \right) \pmod{M}$ in the form of $P(x) \left(\sum_{i=1}^{\alpha} \frac{g(x)}{(x - a_i)} \frac{b_i}{g'(a_i)} \right) \pmod{g(x)}$ for

polynomials. A note of remarkable importance is the fact that the algorithm for $P(x)$ is the familiar Lagrange Interpolation Formula found in Numerical Analysis.

Chapter 3

REVIEWS

3.1 Applications of Chinese Remainder Theorem

The conventional Chinese remainder theorem (CRT) is to determine a single integer from its remainders from a set of modulus. It has tremendous applications in various areas, such as cryptography and digital signal processing.

The Chinese Remainder Theorem (CRT) allows for an efficient implementation of the RSA algorithm. The theorem is as follows. Given input, m , raise it to the e -th (or d -th) power modulo p and modulo q . The intermediate results are then combined through multiplication and addition with some predefined constants to compute the final result (the modular exponentiation to N). This approach is often used for implementing RSA in embedded systems. It requires four times less execution time and smaller amount of memory for intermediate results, since modular exponentiation is performed on half the bit size of N .

The two most important considerations when designing Residue Number Systems are the choices of the moduli sets and the conversion from the residue to the weighted binary system. A new general conversion algorithm has been applied to the recently proposed conjugate moduli sets, which results in a more efficient design for the residue to binary conversion of the given moduli sets. This more efficient design for the converter will make the conjugate moduli sets more attractive compared to other moduli sets. The result also demonstrates the effectiveness of the New Chinese Remainder Theorems.

CRT has various generalizations. A different generalization of CRT has been recently proposed in, where (instead of a single integer in CRT) multiple integers need to be determined from (not a sequence of remainders but) a sequence of sets, residue sets, of remainders. A residue set consists of the remainders of multiple integers modulo a modulus integer, and the residue set is not ordered, i.e., the correspondence between the elements in the residue set and the multiple integers is not specified. The generalized CRT was motivated from the determination of multiple frequencies in a super positioned signal of multiple sinusoids from its multiple under sampled waveforms. This has applications in a sensor network, where multiple sensors have low power and low transmission rates, and their sampling rates may be low and much lower than the Nyquist rate of a signal of interest in the field. The generalized CRT has been used in synthetic aperture radar (SAR) imaging of moving targets and polynomial phase signal detection. It has been found that the error rates of multiple frequencies are significantly reduced with the proposed algorithm considering residue errors compared to the one without

Considering residue set errors. This algorithm finds application in a sensor network with low sampling rates.

Chinese Remainder Theorem has been used for hundreds of years and has been applied to many domains such as integers and polynomials as explained in last chapter. It can also be modified to design proxy signatures.

3.2 Hill Cipher and its Applications:

In classical cryptography, the **Hill cipher** is a polygraphic substitution cipher based on linear algebra. Invented by Lester S. Hill in 1929, it was the first polygraphic cipher in which it was practical (though barely) to operate on more than three symbols at once. The key size is the binary logarithm of the number of possible keys. There are 26^{n^2} matrices of dimension $n \times n$. Thus $\log_2(26^{n^2})$ or about $4.7n^2$ is an upper bound on the key size of the Hill cipher using $n \times n$ matrices. This is only an upper bound because not every matrix is invertible and thus usable as a key. The number of invertible matrices can be computed via the Chinese Remainder Theorem. i.e., a matrix is invertible modulo 26 if and only if it is invertible both modulo 2 and modulo 13. The number of invertible $n \times n$ matrices modulo 2 is equal to the order of the general linear group $GL(n, Z_2)$. In the improved version of the Hill cipher, a randomly generated non-singular matrix is used as an encryption key, and the inverse of the matrix is used as the decryption key. The weakness of Hill cipher is that the matrix may be revealed under the known-plaintext attack. But in this new version, a plaintext message first is partitioned into some suitable length of blocks and each block b concatenates with a random string r and a special controlled symbol c as $r||c||b$. The new string is converted to a vector and the components of the vector are the positive integers. To overcome the drawbacks of the Hill cipher, a more secure number system with different bases and an enforced transformation of the enciphering matrix are provided.

Matrix cryptosystems, like Hill cipher, are resistant to frequency analysis. The key is a non-singular matrix, for example 3×3 matrix K . A modular non-singular key-matrix for matrix ciphers can be generated and these results are applied to cryptography and computer security. For Example, a mutual authentication protocol based on Hamiltonian cycle in directed weight graphs and modular matrix algebra has been proposed.

Chapter 4

RSA AND KEY ISSUES

4.1 PUBLIC KEY ENCRYPTION

4.1.1 Introduction:

Public-Key Algorithms are asymmetric, that is to say the key that is used to encrypt the message is different from the key used to decrypt the message. The encryption key, known as the Public key is used to encrypt a message, but the message can only be decoded by the person that has the decryption key, known as the private key.

This type of encryption has a number of advantages over traditional symmetric Ciphers. It means that the recipient can make their public key widely available- anyone wanting to send them a message uses the algorithm and the recipient's public key to do so. An eavesdropper may have both the algorithm and the public key, but will still not be able to decrypt the message. Only the recipient, with the private key can decrypt the message.

Advantage of public-key algorithm is that they are more computationally intensive than symmetric algorithms, and therefore encryption and decryption take longer. This may not be significant for a short text message, but certainly is for bulk data encryption.

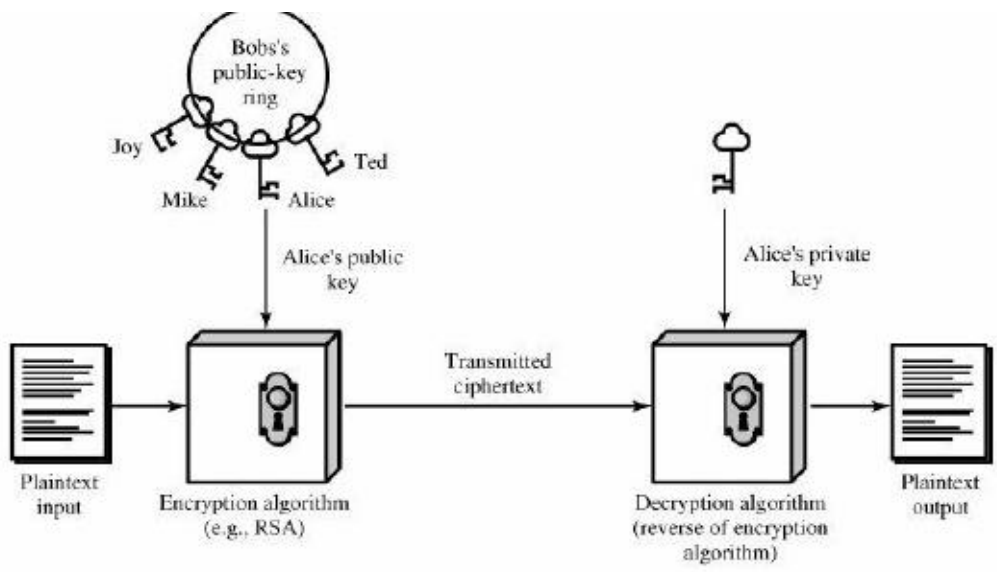
4.2 The Basic Principle:

In order to decrypt a message, Bob (the recipient) has to know the key. However, it may be difficult for Alice (the sender) to tell Bob what the key is. If they simply agree on a key by e-mail for example, Eve could be listening in on their e-mail conversation and thus also learn what the key is. Public key cryptography was invented to solve this problem.

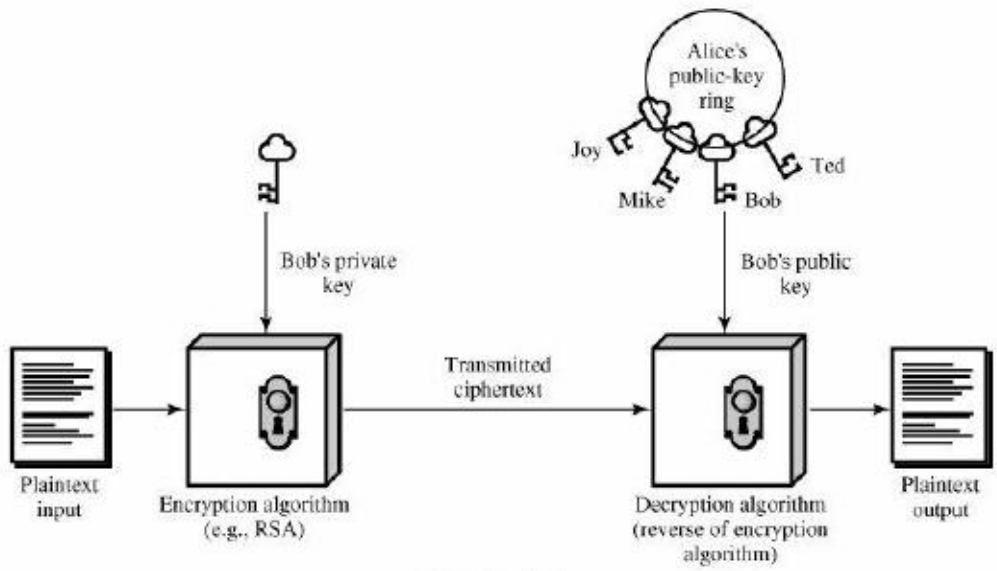
When using public-key cryptography, Alice and Bob both have their own key pairs. A key pair consists of a public key and a private-key. If the public-key is used to encrypt something, then it can be decrypted only using the private-key. And similarly, if the private-key is used to encrypt something, then it can be decrypted only using the public-key. It is not possible to figure out what the private-key is given only the public-key, or vice versa.

This makes it possible for Alice and Bob to simply send their public keys to one another, even if the channel they are using to do so is insecure. It is no problem that Eve now gets a copy of the public keys. If Alice wants to send a secret message to Bob, she encrypts the message using

Bob's public key. Bob then takes his private key to decrypt the message. Since Eve does not have a copy of Bob's private key, she cannot decrypt the message. Of course this means that Bob has to carefully guard his private key. With public key cryptography it is thus possible for two people who have never met to securely exchange messages. Figure below illustrates the public-key encryption process.



(a) Encryption



(b) Authentication

4.3 Public Key Issues:

Advantage and Disadvantage of Public-Key Cryptography Compared with Secret-Key Cryptography:

1. The primary advantage of public-key cryptography is increased security and convenience. Private keys never need to be transmitted or revealed to anyone. In a secret-key system, by contrast, the secret keys must be transmitted (either manually or through a communication channel), and there may be a chance that an enemy can discover the secret keys during their transmission.

2. Another major advantage of public-key systems is that they can provide a method for digital signatures. Authentication via secret-key systems requires the sharing of some secret and sometimes requires trust of a third party as well. As a result, a sender can repudiate a previously authenticated message by claiming that the shared secret was somehow compromised by one of the parties sharing the secret. For example, the Kerberos secret-key authentication system involves a central database that keeps copies of the secret keys of all users; an attack on the database would allow widespread forgery. Public-key authentication, on the other hand, prevents this type of repudiation; each user has sole responsibility for protecting his or her private key. This property of public-key authentication is often called non-repudiation.

3. A disadvantage of using public-key cryptography for encryption is speed; there are popular secret-key encryption methods that are significantly faster than any currently available public-key encryption method. Nevertheless, public-key cryptography can be used with secret-key cryptography to get the best of both worlds. For encryption, the best solution is to combine public- and secret-key systems in order to get both the security advantages of public-key systems and the speed advantages of secret-key systems. The public-key system can be used to encrypt a secret key, which is used to encrypt the bulk of a file or message. Such a protocol is called a digital envelope.

4. Public-key cryptography may be vulnerable to impersonation, however, even if users' private keys are not available. A successful attack on a certification authority will allow an adversary to impersonate whomever the adversary chooses to by using a public-key certificate from the compromised authority to bind a key of the adversary's choice to the name of another user.

5. In some situations, public-key cryptography is not necessary and secret-key cryptography alone is sufficient. This includes environments where secure secret key agreement can take place, for example by users meeting in private. It also includes environments where a single authority knows and manages all the keys (e.g., a closed banking system) Since the authority knows everyone's keys already, there is not much advantage for some to be "public" and others "private" Also, public-key cryptography is usually not necessary in a single-user environment. For example, if you want to keep your personal files encrypted, you can do so with any secret-key encryption algorithm using, say, your personal password as the secret key. In general, public-key cryptography is best suited for an open multi-user environment.

6. Public-key cryptography is not meant to replace secret-key cryptography, but rather to supplement it, to make it more secure. The first use of public-key techniques was for secure key exchange in an otherwise secret-key system; this is still one of its primary functions. Secret-key cryptography remains extremely important and is the subject of ongoing study and research. Some secret-key cryptosystems are discussed in the sections on Block Cipher and Stream Cipher.

4.4 The RSA Algorithm:

The RSA cryptosystem, named after its inventors R. Rivest, A. Shamir, and L. Adleman, is the most widely used public key Cryptosystem. It may be used to provide both secrecy and digital signatures and its security is based on the intractability of the integer factorization. The RSA scheme is a block cipher in which the plaintext and cipher text are integers between 0 and $n-1$ for some n . A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024} .

4.4a Description of the Algorithm

The scheme makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n . That is, the block size must be less than or equal to $\log_2(n)$; in practice, the block size is i bits, where $2^i < n < 2^{i+1}$. Encryption and decryption are of the following form, for some plaintext block M and cipher text block C :

$$C = M^e \bmod n$$

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the value of n . The sender knows the value of e , and only the receiver knows the value of d . Thus, this is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PU = \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. It is possible to find values of e, d, n such that $Med \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate $M^e \bmod n$ and C^d for all values of $M < n$.
3. It is infeasible to determine d given e and n .

We need to find a relationship of the form $M^{ed} \bmod n = M$.

The preceding relationship holds if e and d are multiplicative inverses modulo $\phi(n)$, where $\phi(n)$ is the Euler totient function. For p, q prime, $\phi(pq) = (p-1)(q-1)$. The relationship between e and d can be expressed as

$$ed \bmod \phi(n) = 1$$

This is equivalent to saying

$$ed \equiv 1 \pmod{\phi(n)} \quad \text{and} \quad d \equiv e^{-1} \pmod{\phi(n)}$$

That is, e and d are multiplicative inverses mod $\phi(n)$. According to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\phi(n)$. Equivalently, $\gcd(\phi(n), d) = 1$

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\text{gcd}(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod n$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod n$

4.4b Exponentiation in Modular Arithmetic:

Both encryption and decryption in RSA involve raising an integer to an integer power, mod n . If the exponentiation is done over the integers and then reduced modulo n , the intermediate values would be gargantuan, we can make use of a property of modular arithmetic: $[(a \pmod n) x$

$(b \bmod n) \bmod n = (a \times b) \bmod n$ Thus, we can reduce intermediate results modulo n . This makes the calculation practical.

Another consideration is the efficiency of exponentiation, because with RSA we are dealing with potentially large exponents. To see how efficiency might be increased, consider that we wish to compute x^{16} . A straightforward approach requires 15 multiplications: However, we can achieve the same final result with only four multiplications if we repeatedly take the square of each partial result, successively forming x^2, x^4, x^8, x^{16} . As another example, suppose we wish to calculate $x^{11} \bmod n$ for some integers x and n . Observe that $x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$. In this case we compute $x \bmod n, x^2 \bmod n, x^4 \bmod n,$ and $x^8 \bmod n$ and then calculate $[(x \bmod n) \times (x^2 \bmod n) \times (x^8 \bmod n)] \bmod n$.

4.4c Efficient Operation Using the Public Key

To speed up the operation of the RSA algorithm using the public key, a specific choice of e is usually made. The most common choice is 65537 ($2^{16} - 1$); two other popular choices are 3 and 17. Each of these choices has only two 1 bits and so the number of multiplications required to perform exponentiation is minimized. However, with a very small public key, such as $e = 3$, RSA becomes vulnerable to a simple attack. It is required that during key generation the user selects a value of e that is relatively prime to $\phi(n)$. Thus, for example, if a user has preselected $e = 65537$ and then generated primes p and q , it may turn out that $\gcd(\phi(n), e)$ is not equal to 1. Thus, the user must reject any value of p or q that is not congruent to 1 (mod 65537).

4.4d Key Generation

Before the application of the public-key cryptosystem, each participant must generate a pair of keys. This involves the following tasks:

- 1) Determining two prime numbers, p and q
- 2) Selecting either e or d and calculating the other

First, consider the selection of p and q . Because the value of $n = pq$ will be known to any potential adversary, to prevent the discovery of p and q by exhaustive methods, these primes must be chosen from a sufficiently large set (i.e., p and q must be large numbers). On the other hand, the method used for finding large primes must be reasonably efficient.

In summary, the procedure for picking a prime number is as follows.

- Pick an odd integer n at random (e.g., using a pseudorandom number generator).
- Pick an integer $a < n$ at random.

- Perform the probabilistic primality test, such as Miller-Rabin, with a as a parameter. If n fails the test, reject the value n and go to step 1.
- If n has passed a sufficient number of tests, accept n ; otherwise, go to step 2.

This is a somewhat tedious procedure. However, remember that this process is performed relatively infrequently: only when a new pair (PU, PR) is needed.

Having determined prime numbers p and q , the process of key generation is completed by selecting a value of e and calculating d or, alternatively, selecting a value of d and calculating e . Assuming the former, then we need to select an e such that $\gcd(\phi(n), e) = 1$ and then calculate $d \equiv e^{-1} \pmod{\phi(n)}$. Fortunately, there is a single algorithm that will, at the same time, calculate the greatest common divisor of two integers and, if the gcd is 1, determine the inverse of one of the integers modulo the other. The algorithm is referred to as the extended Euclid's algorithm. Thus, the procedure is to generate a series of random numbers, testing each against $\phi(n)$ until a number relatively prime to $\phi(n)$ is found.

4.5 RSA using Chinese Remainder Theorem

The complexity of the RSA decryption $M = C^D \pmod{N}$ depends directly on the size of D and N . The decryption exponent D specifies the numbers of modular multiplications necessary to perform the exponentiation, and the modulus N determines the size of the intermediate results. A way of reducing the size of both D and N is to take advantage of properties stated by the Chinese Remainder Theorem (CRT) and Fermat's Little Theorem.

Mathematical background

In the following, some basic facts and conclusions of the CRT are summarized. This mathematical background knowledge is of elementary importance for the efficient realization of RSA decryption.

Theorem 1 (Chinese Remainder Theorem) *Let the numbers n_1, n_2, \dots, n_k be positive integers which are relatively prime in pair, i.e. $\gcd(n_i, n_j) = 1$ when $i \neq j$. Furthermore, let*

$n = n_1 n_2 \dots n_k$ and let x_1, x_2, \dots, x_k be integers.

Then the system of congruences

$$x \equiv x_1 \pmod{n_1}$$

$$x \equiv x_2 \pmod{n_2}$$

⋮

⋮

⋮

$$x \equiv x_k \pmod{n_k}$$

has a simultaneous solution x to all of the congruences, and any two solutions are congruent to one another modulo n . Furthermore there exists exactly one solution x between 0 and $n-1$. The unique solution x of the simultaneous congruences satisfying $0 \leq x < n$ can be calculated as

$$x = \left(\sum_{i=1}^k x_i r_i s_i \right) \pmod{n}$$

$$= (x_1 r_1 s_1 + x_2 r_2 s_2 + \dots + x_k r_k s_k) \pmod{n}$$

where $r_i = \frac{n}{n_i}$ and $s_i = r_i^{-1} \pmod{n_i}$ for $i = 1, 2, \dots, k$.

This method is termed as *Gauss's algorithm*.

Corollary 1.1 *If the integers n_1, n_2, \dots, n_k are pairwise relatively prime and $n = n_1 n_2 \dots n_k$, then for all integers a, b it is always valid that $a \equiv b \pmod{n}$ if and only if $a \equiv b \pmod{n_i}$ for each $i = 1, 2, \dots, k$. As a consequence of the CRT, any positive integer $a < n$ can be uniquely represented as a k -tuple $[a_1, a_2, \dots, a_k]$ and vice versa, whereby a_i denotes the residue $a \pmod{n_i}$ for each $i = 1, 2, \dots, k$. The conversion of a into the residue system defined by n_1, n_2, \dots, n_k is simply done by modular reductions $a \pmod{n_i}$. Conversion back from residue representation to "standard notation" is somewhat more difficult.*

Theorem 2 (Fermat's Little Theorem) *Let p be a prime any integer a not divisible by p satisfies $a^{p-1} \equiv 1 \pmod{p}$.*

Fermat's little theorem is very useful for calculating the multiplicative inverse of an integer a because $a^{p-2} \equiv a^{-1} \pmod{p}$.

Corollary 2.1 *If an integer a is not divisible by p and if $n \equiv m \pmod{p-1}$, then $a^n \equiv a^m \pmod{p}$.*

Collorally (2.1) states that when working modulo a prime p , the exponents can be reduced $\pmod{p-1}$. This allows to perform the RSA decryption with significantly shorter exponents.

4.6 RSA decryption using the CRT

Since P and Q are primes, any message $M < N = PQ$ is uniquely represented by the tuple $[M_P; M_Q]$, where $M_P = M \bmod P$ and $M_Q = M \bmod Q$. Therefore, it is also possible to obtain M by computation of $M_P; M_Q$ and recombining them according to equation (6), rather than the usual computation of $M = CD \bmod N$. By using the corollary (2.1), the size of the exponent can be scaled down:

$$\begin{aligned} M_P &= M \bmod P = (C^D \bmod N) \bmod P = C^D \bmod P \text{ (since } N = PQ) \\ &= C^D \bmod (P-1) \bmod P = C^{D_P} \bmod P \text{ with } D_P = D \bmod (P - 1) \end{aligned}$$

Furthermore, it is easily observed that the ciphertext C can be reduced modulo P before computing M_P , so the lengths of all operands are scaled down by half. With the quantities $C_P = C \bmod P$ and $C_Q = C \bmod Q$, as well

as $D_P = D \bmod (P - 1)$ and $D_Q = D \bmod (Q - 1)$, we get the following equations for M_P and M_Q :

$$M_P = C_P^{D_P} \bmod P \text{ and } M_Q = C_Q^{D_Q}$$

The recombination of M_P and M_Q to get M can be done according to equation (6). For the special case of $k = 2$, $n_1 = P$, $n_2 = Q$ and $n = N = PQ$, we get $r_1 = N/P = Q$ and $r_2 = N/Q = P$. Moreover, equation (6) can be simplified by using Fermat's little theorem:

$$\begin{aligned} M &= (M_P Q(Q^{P-1} \bmod P) + M_Q P(P^{Q-1} \bmod Q)) \bmod N \\ &= (M_P Q(Q^{P-2} \bmod P) + M_Q P(P^{Q-2} \bmod Q)) \bmod N \\ &= (M_P (Q^{P-1} \bmod N) + M_Q (P^{Q-1} \bmod N)) \bmod N \end{aligned}$$

The last equality comes from the fact that $a(b \bmod c) = (ab) \bmod (ac)$ for any nonnegative integers $a; b; c$. Note that the coefficients $Q^{P-1} \bmod N$ and $P^{Q-1} \bmod N$ are constant and can be precomputed, thereby the effort for the recombination of M_P and M_Q is reduced to two multiplications, one addition and one reduction modulo N . When assuming that the exponents $D_P = D \bmod (P - 1)$ and $D_Q = D \bmod (Q - 1)$, as well as the constants which are needed for the recombination $R_P = Q^{P-1} \bmod N$ and $R_Q = P^{Q-1} \bmod N$ have been precomputed, the CRT based RSA decryption can take place according to the following steps :

1. Calculate $C_P = C \bmod P$ and $C_Q = C \bmod Q$.
2. Calculate the exponentiations $M_P = C_P^{D_P} \bmod P$ and $M_Q = C_Q^{D_Q} \bmod Q$
3. Calculate the coefficients $S_P = M_P R_P \bmod N$ and $S_Q = M_Q R_Q \bmod N$.

4. Calculate $M = S_p + S_q$. If $M \geq N$ then calculate $M = M - N$.

4.6a Algorithm for RSA using CRT:

1. Input the plain text (or message **M**).
2. Generate two random prime numbers (co-prime to each other) **p** and **q**.
3. Generate another set of random co- primes **r** and **s**.
4. Compute $n_1 = p * q$.
5. Compute $n_2 = r * s$.
6. Compute $\phi_1 = (p-1)(q-1)$.
7. Compute $\phi_2 = (r-1)(s-1)$.
8. Choose e_1 such that $1 < e_1 < \phi_1$
9. Choose e_2 such that $1 < e_2 < \phi_2$
10. Compute $d_1 = e_1^{-1} \text{ mod } n_1$
11. Compute $d_2 = e_2^{-1} \text{ mod } n_2$
12. Compute $t_1 = M \text{ mod } n_1$
13. Compute $t_2 = M \text{ mod } n_2$
14. Compute $y_1 = t_1^{e_1} \text{ mod } n_1$
15. Compute $y_2 = t_2^{e_2} \text{ mod } n_2$
16. Compute $a_1 = e_1^{-1} \text{ mod } \phi_1$
17. Compute $a_2 = e_2^{-1} \text{ mod } \phi_2$
18. Compute $b_1 = y_1^{a_1} \text{ mod } n_1$
19. Compute $b_2 = y_2^{a_2} \text{ mod } n_2$
20. Compute $f_1 = n_2^{-1} \text{ mod } n_1$
21. Compute $f_2 = n_1^{-1} \text{ mod } n_2$

22. Compute decoded text: $=(b_1 \cdot n_2 \cdot f_1 + b_2 \cdot n_1 \cdot f_2) \bmod (n_1 \cdot n_2)$

23. The decoded text is printed

4.6b Algorithm for RSA using Variable Radix Number System:

1. Generate two random prime numbers (co-prime to each other) p and q .
2. Generate another set of random co- primes r and s .
3. Input the plain text M such that $M < p \cdot q$.
4. Compute $n_2 = r \cdot s$.
5. Compute $\phi_2 = (r-1)(s-1)$.
6. Choose e_2 such that $1 < e_2 < \phi_2$
7. Compute $d_2 = e_2^{-1} \bmod n_2$
8. Compute $x_0 = m \bmod p$
9. Compute $x_1 = \text{floor}(m/p)$
10. Compute encrypted $y_0 = (x_0^{e_2}) \bmod n_2$
11. Compute encrypted $y_1 = (x_1^{e_2}) \bmod n_2$
12. Compute $x_0 = (y_0^{d_2}) \bmod n_2$
13. Compute $x_1 = (y_1^{d_2}) \bmod n_2$
14. Compute decrypted message $M = x_1 \cdot p + x_0$
15. Print decrypted output.

4.7 SOFTWARE USED: MAPLE WITH MATLAB

What is MAPLE?

MAPLE is essential technical computing software for today's engineers, mathematicians and scientists. Whether one needs to design sheets or produce sophisticated high fidelity simulation

models, MAPLE's world leading computation engine offers the breadth and depth to handle every type of mathematics.

MAPLE TOOLBOX for MATLAB:-

The MAPLE toolbox for MATLAB offers a technical computing solution that is lightly integrated with MATLAB, providing direct access to all commands, variables and functions of each product which working in either environment. This toolbox is available separately.

MAPLE and the symbolic Math Toolbox:-

The MAPLE engine is no longer included with the symbolic Math Toolbox and the extended Symbolic Math Toolbox from the Mathworks. However, it can be configured to call out an existing MAPLE installation, and MAPLE will perform the calculation, and MAPLE will perform the calculations and returns the results to MATLAB.

MATLAB to MAPLE code translation:-

The MATLAB to MAPLE code translator helps to convert the existing MATLAB code into MAPLE for use in new or expanded projects. It also offers a quick-on-the-fly translation if one is more familiar with MATLAB syntax:-

- Converts and automatically executes MATLAB commands by using the MAPLE equivalents.
- Works with logical commands or MATLAB.m files.
- Supports basic operations, matrices indexing and matrices constructions.
- Over 100 MATLAB commands are automatically mapped to their MAPLE equivalents.
- Collection of translatable commands is user expandable.

MATLAB Code Generation:-

MATLAB's code generation feature can generate MATLAB code from MAPLE expressions and procedures.

MATLAB LINK:-

The MATLAB link lets one to call on MATLAB to perform calculations from the MAPLE environment, and returns the results to MAPLE for further analysis.

4.8 THE RSA CRYPTOSYSTEM with MAPLE

In this section we show how Maple can be used to encipher and decipher messages using the RSA cryptosystem.

We begin by mentioning several Maple commands that are useful for finding large primes. The first command we will mention is the **nextprime** command, which returns the smallest prime larger than an integer input. For example, the following command returns the smallest prime larger than 400043344212007458000.

```
>> nextprime (400043344212007458000);
      400043344212007458013
```

A similar command is the **prevprime** command, which returns the largest prime smaller than an integer input. For example, the following command returns the largest prime smaller than 400043344212007458000.

```
>> prevprime (400043344212007458000);
      400043344212007457977
```

A final primality command we will mention is the **isprime** command, which returns *true* if an integer input is prime and *false* if not. For example, the following commands imply that 400043344212007457977 is prime while 400043344212007458000 is not.

```
>> isprime (400043344212007457977);
      True
```

```
>> isprime (400043344212007458000);
      False
```

We should mention that the **nextprime**, **prevprime**, and **isprime** commands are probabilistic routines that employ primality tests. This means that the output returned by Maple is in general guaranteed to be correct with extremely high probability, but not absolutely. We now

show how Maple can be used to perform the RSA encipherment and decipherment procedures. We begin by finding large primes p and q .

```
>> p := nextprime(400043344212007458000);
      p := 400043344212007458013
```

```
>> q:= nextprime(500030066366269001200);
      q := 500030066366269001203
```

Next, we define $n = pq$ and $m = (p - 1)(q - 1)$.

```
>> n := p*q;
      n := 200033699955714283345172521584008468989639
>> m := (p-1)*(q-1);
      m := 200033699955714283344272448173430192530424
```

And we will use the following encryption exponent a .

```
>> a := 10098768900987679000910003;
      a := 10098768900987679000910003
```

To verify that this value of a is a valid RSA encryption exponent, we enter the following Maple **igcd** command, which returns the greatest common divisor of the integers a and m . Note that, as required, $(a, m) = 1$.

```
>> igcd(a, m);
      1
```

We now use the RSA encipherment procedure to encipher the message, "RETURN TO HEADQUARTERS".

```
>> message := 'returntoheadquarters';
      message := returntoheadquarters
```

Next, we convert this message into a list of 2-digit integers and combine these integers into a single block. To do this, we have to provide the user written procedure **to number**, for which code is given. If this procedure is saved as the text file *to number* in the directory from which we are running Maple, then we can include the **to number** procedure in this Maple session by entering the following command.

```
>> read to_number;
```

We can then convert message into its numerical equivalent as a single block by entering the following command.

```
>> plaintext := to_number(message);
      plaintext := 1704192017131914070400031620001719041718
```

Because this plaintext integer is smaller than n , we can encipher this message as a single block. That is, we can encipher this message by raising plaintext to the power a and reducing modulo n . To do this, we enter the following command. (Because this modular exponentiation involves

such a large exponent, we use the Maple **&^** command instead of just **^** for the exponentiation. By using **&^**, we cause Maple to do the exponentiation in a very efficient way.

```
>> ciphertext := plaintext &^ a mod n;
      ciphertext := 39705667751051336812284136334817473485289
```

To decipher this message, we must find a decryption exponent b that satisfies $ab = 1 \pmod{m}$. We can do this by entering the following Maple **igcdex** command. Like the preceding **igcd** command, the following **igcdex** command returns the greatest common divisor of the integers a and m . However, the following **igcdex** command also takes two additional user defined variable inputs, which it leaves as integers b and y that satisfy $ab + my = (a,m)$. Since $(a,m) = 1$, these will be values of b and y that satisfy $ab + my = 1$ or, equivalently, $ab = 1 \pmod{m}$. Thus, we can find a decryption exponent b by entering the following command.

```
>> igcdex (a, m, 'b', 'y');
```

```
1
```

To see the decryption exponent b defined by the previous command, and to express this value as a positive number less than m , we enter the following command.

```
>> b := b mod m;
      b := 54299300950841826990071853678997985400035
```

Next, by entering the following command we verify that this value of b satisfies $ab = 1 \pmod{m}$.

```
>> a*b mod m;
      1
```

To recover the plaintext integer, we must only raise ciphertext to the power b and reduce modulo n .

```
>> plaintext := ciphertext &^ b mod n;
      plaintext := 1704192017131914070400031620001719041718
```

To see the original plaintext letters, we must split this single block into a list of 2-digit integers and convert these integers back into letters. To do this, we have provided the user-written procedure **to letter**, for which code is given.

```
>> read to_letter;
```

We can then convert plaintext back into a list of letters by entering the following command.

```
>> to_letter(plaintext);
      Returntoheadquarters
```

A final command we will mention is the **ifactor** command, which returns the prime factorization of an integer input. For example, the following command very quickly returns the prime factorization of the 43-digit integer 1118516508138307725195354324934560155358253.

```
>> ifactor (1118516508138307725195354324934560155358253);  
(17)5 (389) (45001200019828331)2
```

Recall that the security of the RSA cryptosystem is based on the apparent difficulty of factoring the value of n . Hence, in order for the RSA system used in this section to be secure, it should be very difficult for an intruder to factor the 42-digit value of n used in this section. Although this value of n is one digit shorter than the integer used in the preceding command, because the prime factors of n are both very large, it will take **ifactor** much more time to return these prime factors. For example, the reader may wish to enter the preceding and following commands to see the difference.

```
>> ifactor (200033699955714283345172521584008468989639);
```

And recall, if p and q are both around 100 digits long, then the fastest known factoring algorithms, including the one employed by the **ifactor** command, would in general take millions of years to factor $n = pq$, even when programmed on a computer that can perform millions of operations per second.

Chapter 5

IMPLEMENTATION RESULTS

5.1 RSA without Chinese Remainder Theorem

Outputs:

PLAIN TEXT MESSAGE :

36489656439701145972457523098132091447225668989528382573694967128594581341696354706942814918
02846120607876041957626485782602709228491414478078017857900999654234151658896273828484907531
96984483917921633499173120483800789218174753482435328041525968282689242448449908738367668775
6808250983298803378215978241

Parameters

$p=185078731400523062481441254377509455858583223247383377301344032467716615394479165221010471$
 $49794805095611172985373938066279653462203253151628946347482717349$

$q=$

$26755500621476182476235043624053751860767956969924425426492412835499485491845324611561266421$
 $975797211492947350804792680468942874570562668055997954531542783$

$n=495187411300871824656376487782363847751829121569677212389942954570700536290488450866636581$
 $91635002703350352059016504245111847283095045661395721838825182381783961806338360633127086800$
 $71999941015692047948711008483131977249198637073517807044053933919948846733144219783696975992$
 $28690575040503478802952570089842267$

$e =166955546873566405943$

$d=411305813285699030320744082906675650673709593595480594601554177052927559073441034057405225$
 $59183470966766074552454388118899621630028234832642438569719904468489402895376177381421241309$
 $46929696772713787256998849857096217774430733153998323488420734724604440331554109930382784394$
 $91275545398070345654314377543097679$

$CIPHER_TEXT=43098068030045515712435290551031006913230930156162662286651969290649839951713817$
 $43267074751119067424973489745483641968711152988210654612746688048470072863780344978020290306$
 $54067734912563568110910406019165813544213939413336518829979873903310013049936429396863299883$
 $766901028992702271697643499912469270040942711$

$DECODED_TEXT=$

36489656439701145972457523098132091447225668989528382573694967128594581341696354706942814918
02846120607876041957626485782602709228491414478078017857900999654234151658896273828484907531
96984483917921633499173120483800789218174753482435328041525968282689242448449908738367668775
6808250983298803378215978241

5.2 RSA with Chinese Remainder Theorem

Outputs:

ENTER MESSAGE:

36489656439701145972457523098132091447225668989528382573694967128594581341696354706942814918
02846120607876041957626485782602709228491414478078017857900999654234151658896273828484907531
96984483917921633499173120483800789218174753482435328041525968282689242448449908738367668775
6808250983298803378215978241

Parameters

p1=20334809283673287114475443902995393136895869842563408442900517639422906549503817059239485
459840968702208580902920918814783385527387093734816832862666480269

q1=15735928157609261156586763760288430468897665819183625164940708181149013215784181187471504
247853323491608819750564251897925339269785978664223462535647161271

n1=31998709798656868851656146799194061992560005500514888120035752309554571769302577500433622
37989030204552162015402276191935720512262556353914281078058111663481518101882535744072757982
7123790533430748642299850595999596325415853508217514561999142702718587569987866273160664373
880903122993937664036029734982461899

e1=96793620484551651428704035909150537691415559822529992641932845825398014868657604537299035
91298413638363501200646166049412748270322240286348900341916673768229627887099814712082539915
00164790545910426893016557542117392898740043090349066292068212883484861928709987780045004308
41670956519003480439675193017447287

d1=13000807100173770262325774950532580819678288838900669855034070742097228798543979594670979
47778124161814923658638436157271311607535261384870527778352394680558062286588653612990818084
73430752472145480503228568196836770271160617515190797097213878747039533983218646772288260212
933733729067720818293162580159229143

p2=19181670784033298213789584327055747423274384647840278754796144376884868313478386925534025
289327478423366661359103402159244341643926335962290064015285892911

q2=21661276037257051743918043298143237220402945465556565871729893005497663953924460387283591
41705504745117386461786917763909498061895922347742636985262125097

n2=41549946570873416673473276613020853574920730254296238329931698625869808808456050446342631
13945163456257402685941481373166831432672265669867930732406849005247195732557836644037228784
67265623224799054866099341007051488534528539237158511834378480424350625129980772167216578972
911211455877852343600937497027487367

e2=13514653650239443389548737232300683824610475744702779252218278159054235386079040939860387
04263925010707321734358621097603318285448582321231050881734941971028604089270394051198422274
78674589916513536626986363634645102630791440185893491407782514034406976279860390939443841421
966307411733702668477785214218380327

d2=27438863311629338289438145005760040948953032770821014768010082582962189355415206403458317
46637351070099318687315455895994586893676303002339561543350452459850435135717567635635496129
86908616106186851955720657549554614834355134418104359997880434070546553611997435800846618114
394318736034371953125614236176594503

ENCRYPTED=152487905740931869620782429502939633170180729262271207480762213532883310229954654
66248428345995397325375987414819585067114529059808473980165678824398597510601506812953452238
06018427208154561225656186169585127154964021733187708502028183457569123046272728866119705209
66338620891217313772978963018779277616510918,
39239899928892944329101400472414348364938317979573928306420001860509837278178621936253443224
81914537566627550444400879287970193130826572243219332957816359223488028572904543249263051449
25044964043052315535780424251166329454022428852273436289668176048638227655771311288340199854
686193249643500893610441493837269

DECRYPTED=364896564397011459724575230981320914472256689895283825736949671285945813416963547
06942814918028461206078760419576264857826027092284914144780780178579009996542341516588962738
28484907531969844839179216334991731204838007892181747534824353280415259682826892424484499087
383676687756808250983298803378215978241

5.3 RSA in Variable Radix Number System:

Outputs:

ENTER MESSAGE

:3648965643970114597245752309813209144722566898952838257369496712859458134169635470694281491
80284612060787604195762648578260270922849141447807801785790099965423415165889627382848490753
19698448391792163349917312048380078921817475348243532804152596828268924244844990873836766877
56808250983298803378215978241

Parametersp1=17841791540423651247489093804867051385957956484294846306706888129279202029850457
413946942319263736948306792375769913037774324136628387493318030044975813559

q1=24516859756580670800638761780809277237685001137254423395484340517475136458663359430242471
870896714205615274245379727549343940721063078183173707949913269311

n1=43742470100271406983875224839944332834385486933598285799335047538003577141993784220079698
34659855714727054184648951981553745389893658595380838567017062703655814986940272435607588707

26504886069742885255092385022638351481143221362814634540198666033016325365711840712480093554
811322150145184027992640751692387849

e1=26407319545155054320101324025006771983013753549987330522975714068153377296738006904203639
60292586399216447930826740271717402644147726388259459019028670200344286682874452107776226857
53729679084615265698141308014435430273331688477375458505516171645723261761348607898489399047
753140983798859028020576877634882983

d1=24600169071784984165204750150326773998985675237419542703352599237792435469720905261946704
00778058234961356061028232097009957345177919059501694008301816574703834139717346643816444466
61434954011299241817865450036063490839964603974574710702241933235580921337000187202483549224
544574125421301520257071204209115647

p2=26185453278454694621182255835437115990151760849418124065502557004503894887230275649413142
399223476187156259249154334339753949125783229203333751968837066139

q2=26674377954991172500372241347599278007962788485926096729848977405815808346929873830468492
553207069546515674114611665652844221642153067489440986225649032087

n2=69848067767226323059045279069305533607567234155705720435844813916680376306858222955073058
01981095990820487151184372041324392033680025489880570838398654128259910099433491404108687495
17948331382895253012348647266937229482496534353186645453611523912879618600299041656744155359
434699657286482805587615274152202093

e2=48802084065629773950750933671773618203956227834785482907592460729746740238527885217076675
46527984211316468170613044609045871800122628640584410163155248858239087641561447852413759766
02633033717553096392519786992652990693607810972028613359253880337633769045317182345921449973
46987624622967747214766398588312911

d2=33275112826972048168187262492758619702054151070895606679525247318334767771623726906103942
69121151262089684229748487596709327090894832239165618930124884976086637125349297317141028506
42870907343055866140527920464005417088725697093109651718471017907711386725063780034504923177
175224313807464107132768000547353599

x0=32671783901545143089073824877125657685883895594357169747787454043915971963511242709114479
61228803297639954261723658879678728039958035978454953232146088307

x1=20451789472501989732751083340490860853539868586524272698904186189202274686859588550530476
5335443329165957332098536114951713602736689716268356580108251

DECRYPTED=364896564397011459724575230981320914472256689895283825736949671285945813416963547
06942814918028461206078760419576264857826027092284914144780780178579009996542341516588962738
28484907531969844839179216334991731204838007892181747534824353280415259682826892424484499087
383676687756808250983298803378215978241

CONCLUSIONS

The above project was under taken in order to develop better and faster algorithms for implementation of RSA system in Cryptography. Various types of Symmetrical and Asymmetrical Cryptography methods were studied. The direct form of RSA algorithm was implemented using MATLAB. In order to improve the speed of calculations Chinese Remainder Theorem was used in order to encrypt and decrypt text message in RSA algorithm. Finally a inventive stage of Variable Radix Number System was implemented along with the usual RSA cryptosystem in order to achieve an extra layer of encryption.

REFERENCES:

- [1] W.Stallings; "Cryptography and Network Security" 2nd Edition, Prentice Hall, 1999
- [2] Bruce Schneir: Applied Cryptography, 2nd edition, John Wiley & Sons, 1996
- [3] Cryptography and Network Security – Behrouz Forouzan
- [4] Michael Welschenbach "Cryptography in C and C++"
- [5] Eskiciogiu,A. Litwin,L " Cryptography and Network Security" LOS Alamitos,CA: IEEE computer society press,1987
- [6] Applications of Abstract Algebra with MAPLE – Richard E.Kilma,Neil Sigmon,Ernest Stizinger
- [7] A Generalized CRT for residue sets with errors and its application in frequency determination from multiple sensors with low sampling rates – Xiang-Gen Xia and Kejing Liu
- [8]Applications of New Chinese Remainder Theorem to RNS with two pairs of conjugate moduli – Alex Skavantzoz,Yuke Wang
- [9]A New cryptosystem using matrix transformation – Yi Shiung Yeh,Tzong-Chen Wu,Chin-Chen Chang
- [10]Modular Matrix Cipher and its application in authentication protocol – Bao Ngoc TRAN,Thuc Dinh NGUYEN
- [11]www.ieee.org
- [12]www.acm.org