REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

A MULTIFACETED ANALYSIS OF THE RSA ALGORITHM: OPTIMIZATION, SECURITY AND CRYPTANALYSIS

Zülfiye Melisa SAYGINER

MASTER OF SCIENCE THESIS
Department of Mathematics
Program of Mathematics

Supervisor
Assoc. Prof. Dr. Muttalip ÖZAVŞAR

November, 2025

REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

A MULTIFACETED ANALYSIS OF THE RSA ALGORITHM: OPTIMIZATION, SECURITY AND CRYPTANALYSIS

A thesis submitted by Zülfiye Melisa SAYGINER in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 21.11.2025 in Department of Mathematics, Program of Mathematics.

Assoc. Prof. Dr. Muttalip ÖZAVŞAR
Yildiz Technical University
Supervisor

Approved By the Examining Committee

Assoc. Prof. Dr. Muttalip ÖZAVŞAR, Supervisor
Yildiz Technical University _____

Assoc. Prof. Dr. İsmail AYDOĞDU, Member
Yildiz Technical University _____

Assist. Prof. Dr. Ferhat KÜRÜZ, Member
Istanbul Sabahattin Zaim University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of A Multifaceted Analysis of the RSA Algorithm: Optimization, Security and Cryptanalysis supervised by my supervisor, Assoc. Prof. Dr. Muttalip ÖZAVŞAR. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Zülfiye Melisa SAYGINER

Signature

Dedicated to my family
and loved ones

ACKNOWLEDGEMENTS

I owe my gratitude to my advisor, Muttalip Özavşar, who guided me with his experiences in determining the thesis topic and supported me in the subsequent stages of this process.
I would like to extend my infinite thanks to my valuable family, whose support I have always felt throughout this life journey where I have gained many experiences.

Zülfiye Melisa SAYGINER

TABLE OF CONTENTS

LIST OF SYMBOLS vii

LIST OF ABBREVIATIONS viii

LIST OF FIGURES ix

LIST OF TABLES x

ABSTRACT xi

ÖZET xiii

1 INTRODUCTION 1

1.1 Introduction 1

1.2 Organization of the Thesis 1

2 BASIC CONCEPTS 3

2.1 Modular Arithmetics 3

2.2 Groups 6

2.2.1 Binary Operations 6

2.2.2 Groups 7

3 ENCRYPTION ALGORITHMS 9

3.1 Symmetric Encryption Algorithm 10

3.2 Asymmetric Encryption Algorithm 11

3.3 Comparison of Symmetric Encryption Algorithm and Asymmetric Encryption Algorithm 12

4 RSA ALGORITHM 13

4.1 Processes of the Algorithm 15

4.2 Other Applications in the RSA 19

4.2.1 Exponentiation by Repeated Squaring and Multiplication Algorithm 19

4.2.2 Computing e from d and phi(n) 21

4.2.3 Computing phi(n) Without Factoring n 25

5 ANALYSIS OF THE RSA ALGORITHM: SPEED PERFORMANCE AND OPTIMIZATIONS 28
5.1 Chinese Remainder Theorem 29
5.2 Montgomery Algorithm 29
5.3 Barrett Algorithm 30
5.4 Comparison of the Algorithms 31

6 ATTACKS on the RSA ALGORITHM 34
6.1 Passive Attacks on RSA 34
6.1.1 Side Channel Attacks 34
6.2 Active Attacks on RSA 40
6.2.1 Chosen Ciphertext Attacks (CCA) 40
6.2.2 Man in the Middle Attack 42
6.3 Scenarios of Active and Passive Attacks in the RSA Algorithm . 44

7 CRYPTANALYSIS of the RSA ALGORITHM 45

8 CONCLUSION 51

REFERENCES 53

PUBLICATIONS FROM THE THESIS 58

LIST OF SYMBOLS

$D(C)$	Decryption function of the ciphertext C
$E(M)$	Encryption function of the message M
$\phi(n)$	Euler's totient function
$< G, * >$	Group notation
C	The ciphertext, i.e., the encrypted version of the message M
$E^{G,H}(x)$	The encryption function that is used in the OAEP schema
D_A	The function that is used by Alice for decryption
D_B	The function that is used by Bob for decryption
E_A	The function that is used to encrypt by Alice
E_B	The function that is used to encrypt by Bob
M	The message to be encrypted
(M, S)	The message-signature pair
$\lambda_1(L)$	The minimal Euclidean norm
$P(t)$	The power consumption for time t
$\oplus$	The XOR operation

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
ARP	Address Resolution Protocol
CCA	Chosen Ciphertext Attack
CRT	Chinese Remainder Theorem
DES	Data Encryption Standard
DH	Diffie-Hellmann
DNS	Domain Name Server
DPA	Differential Power Analysis
gcd	Greatest common divisor
LLL	Lenstra, Lenstra, and Lovasz (a lattice-based algorithm)
MitM	Man in the Middle Attack
OAEP	Optimal Asymmetric Encryption Padding
RSA	The algorithm that is named after Rivest, Shamir and Adleman
SCA	Side-Channel Attack
SPA	Simple power Analysis
SSL	Secure Sockets Layer
SVP	Shortest Vector Problem
TLS	Transport Layer Security

LIST OF FIGURES

Figure 3.1 Symmetric encryption algorithm schema 10
Figure 3.2 Asymmetric encryption algorithm schema 11
Figure 5.1 Encryption and decryption times for 1024, 2048, and 4096 bit lengths of algorithms 33

LIST OF TABLES

Table 3.1 Comparison of symmetric encryption algorithm and asymmetric encryption algorithm 12
Table 5.1 Encryption and decryption times of algorithms for 1024 bits. 31
Table 5.2 Encryption and decryption times of algorithms for 2048 bits. 32
Table 5.3 Encryption and decryption times of algorithms for 4096 bits. 32
Table 6.1 Attack scenarios of passive and active attacks on the RSA algorithm 44

ABSTRACT

A Multifaceted Analysis of the RSA Algorithm: Optimization, Security and Cryptanalysis

Zülfiye Melisa SAYGINER

Department of Mathematics
Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Muttalip ÖZAVŞAR

The contributions of the studies conducted towards digitalization since the 20th century have undeniably gained visible importance in today's world. These studies started in areas of significant importance, such as military, and with their effective presence in daily life, they are occupying a broader spectrum. In today's digital age, the rapid advancement of technology has increased the need for secure communication as well as the importance of ensuring the security of digital data. Especially the confidentiality, integrity and accuracy of data transmitted over the internet are of great importance for both individual users and institutional structures. In this context, information security plays an effective role in preventing the misuse of sensitive data, aiming to protect fundamental principles such as confidentiality, integrity, authentication and non-repudiation. In achieving these goals, cryptography, which forms the basis of modern information security approaches, emerges as an important tool. Cryptography enables the development of solutions to establish objectives such as privacy, integrity and authentication for security in digital communication. In this context, the RSA algorithm, developed about half a century ago, has become one of the most common and effective encryption methods based on theoretical foundations such as number theory, prime numbers and modular arithmetics. This thesis presents a multifaceted approach to the RSA algorithm. First, its mathematical foundations are examined in detail; then, the algorithm is analyzed along with several methods affecting its speed performance. Finally, potential attacks threatening the security of the algorithm, such as side-channel attacks, are addressed and attack methods like Coppersmith's are included

within the scope of attack analyses. The aim is to provide a holistic perspective on the RSA algorithm in both theoretical and practical aspects, contributing to the academic and technical knowledge in the field of information security.

Keywords: RSA, modular arithmetics, groups, attacks, cryptanalysis.

ÖZET

RSA Algoritmasının Çok Yönlü Analizi: Optimizasyon, Güvenlik ve Kriptanaliz

Zülfiye Melisa SAYGINER

Matematik Anabilim Dalı
Yüksek Lisans Tezi

Danışman: Doç. Dr. Muttalip ÖZAVŞAR

20. yüzyıldan itibaren dijitalleşme yolunda yapılan çalışmaların günümüze olan katkısı, gözle görülür derecede önem kazandığı bir gerçektir. Bu çalışmalar askeri gibi önem arz eden alanlarda başlayıp gündelik hayatta da etkili oluşuyla beraber daha geniş yelpazede yer edinmektedir. Dijital çağın hakim olduğu günümüzde teknolojinin hızla gelişmesi, hem güvenli iletişiminin ihtiyacını hem de dijital verilerin güvenliğinin sağlanmasının önemini artırmıştır. Özellikle internet üzerinden iletilen verilerin gizliliği, bütünlüğü ve doğruluğu, hem bireysel kullanıcılar hem de kurumsal yapılar için büyük önem arz etmektedir. Bu doğrultuda bilgi güvenliği, hassas verilerin kötü amaçla kullanılmasını önlemede etkin bir rol oynamakta olup gizlilik, bütünlük, kimlik doğrulama ve inkar edilemezlik gibi temel ilkeleri korumayı hedeflemektedir. Bu hedeflerin gerçekleştirilmesinde modern bilgi güvenliği yaklaşımlarının temelini oluşturan kriptografi bilimi önemli bir araç olarak öne çıkmaktadır. Kriptografi, dijital iletişimdeki güvenliği sağlayan gizlilik, bütünlük, kimlik doğrulama gibi hedefleri belirlemek için çözümler üretilmesini sağlar. Bu bağlamda yaklaşık yarım asır önce geliştirilen RSA algoritması, sayılar teoremi, asal sayılar, modüler aritmetik gibi teorik temellere dayanan en yaygın ve en etkili şifreleme yöntemlerinden biri olmuştur. Bu tezde RSA algoritması çok yönlü olarak ele alınmıştır. İlk olarak matematiksel temelleri detaylı şekilde incelenmiş, ardından hız performansını etkileyen birkaç yöntemle beraber algoritmanın analizi yapılmıştır. Son olarak, algoritmanın güvenliğini tehdit eden yan kanal saldırısı gibi olası saldırılar ele alınmış ve saldırı analizleri kapsamında da Coppersmith gibi saldırı yöntemlerine yer verilmiştir. Amaç, teorik ve uygulamalı yönleriyle RSA algoritmasına

bütüncül bir bakış açısı sunarak bilgi güvenliği alanındaki akademik ve teknik bilgi birikimine katkı sağlamaktır.

Anahtar Kelimeler: RSA, modüler aritmetik, gruplar, saldırılar, kriptanaliz.

1 INTRODUCTION

1.1 Introduction

In today's world, where technology has developed towards the digital age, the internet, computers and smartphones have become indispensable elements of our lives. These developments also bring security issues. At this point, there is a need for information security. Information security, which has become a necessity of this age, is important in applications such as online shopping and banking, which are common and require high levels of security. Cryptography, one of the most important elements of information security, ensures the secure production and transmission of data and includes algorithms that perform encryption/decryption processes and key generation [1]. One of these algorithms, the RSA algorithm, emerged in the 1970s and is now encountered in applications such as digital signatures and authentication.

1.2 Organization of the Thesis

This thesis work contains information about the RSA algorithm, one of the most discussed algorithms from past to present. The first section provides a brief introduction to the thesis. The second section will discuss some basic concepts necessary for a better understanding of the logic of the RSA algorithm. The third section will outline encryption algorithms in general. By the end of this section, it will be determined which encryption algorithms RSA belongs to. The fourth section contains fundamental information that can be considered basic about RSA. In the following sections, more comprehensive topics about RSA will be addressed. The fifth section will explain how the speed performance of RSA can be measured by certain methods. In the thesis, this section is considered as the analysis of RSA. The sixth section will describe the attacks applied to RSA. This part is also quite important because RSA has been an algorithm vulnerable to attacks since it was developed. The seventh section will address the cryptanalysis of RSA. With the processing of this section, it will

be understood how different areas the concepts of analysis and cryptanalysis cover. The eighth and final section contains the conclusion of the thesis.

2
BASIC CONCEPTS

This chapter has a significant place due to application of the RSA. The section includes some subjects in mathematics such as functions, modular arithmetics and groups. Functions are quite important subject and are widely used in almost all areas of mathematics, especially in areas such as applied mathematics and abstract algebra. Functions are generally notated as $f(a) = b$ for $a \in A$ and $b \in B$, where A is a set of the 'domain' and B is a set of the 'codomain' [2]. The notation 'function' can be described as 'mapping' in some books [2, 3]. The inverse function, this time, is represented as $f^{-1}(b) = a$ if $f(a) = b$, where $b \in B$ is an element of the domain and $a \in A$ is an element of the range [4]. Modular arithmetics and groups will be examined in detail in this study. In addition, the theorems and definitions included in this section will provide benefit for the main idea of the RSA algorithm.

2.1 Modular Arithmetics

Some subjects in modular arithmetics are quite important in abstract algebra, especially applying the RSA algorithm (which will be mentioned in Chapter 4). In this section, they will mention shortly.

Theorem 2.1 (Bézout's Theorem) If $a, b \in \mathbb{Z}^+$, then there are $s, t \in \mathbb{Z}$ that provides

$$gcd(a, b) = sa + tb \tag{2.1}$$

where $gcd(a, b)$ is the common divisor of a and b [6].

This theorem is named after the French mathematician Bézout and the equation 2.1 gives the **Bézout's identity** [7]. In this theorem, the integers s and t are defined as **Bézout's coefficients** of a and b [7].

Another subject that is very significant is Euclidean algorithm. This algorithm is based on calculating the gcd and is used in the application of the RSA

algorithm.

Theorem 2.2 (Euclidean Algorithm) The implementation of the algorithm is formed by the following system

$$\begin{aligned} a &= bq_0 + r_0 \\ b &= r_0q_1 + r_1 \\ r_0 &= r_1q_2 + r_2 \\ r_1 &= r_2q_3 + r_3 \\ &\vdots \\ r_{k-1} &= r_kq_k + 0 \end{aligned}$$

where $a, b \in \mathbb{Z}^+$ and $a > b$ [8, 9].

In this system 2.2, the integers q_i and r_i are obtained, where $i = [0, k]$. The last remaining element in this system of equations must satisfy $gcd(a, b) = r_k$, where $r_k \in \mathbb{Z} - \{0\}$.

As seen, Euclidean Algorithm uses two passes through the steps [7]. However, there is another algorithm that is more practical than this algorithm. This algorithm, which uses one pass through the steps [7], is called **the Extended Euclidean Algorithm**.

Theorem 2.3 (Extended Euclidean Algorithm) Let $a, b \in \mathbb{Z}^+$. Then,

$$gcd(a, b) = s_na + t_nb \tag{2.2}$$

where s_n and t_n are the nth term that are defined by

$$\begin{aligned} s_0 = 1,\ t_0 = 0 \\ s_1 = 0,\ t_1 = 1 \end{aligned}$$

and

$$\begin{aligned} s_j &= s_{j-2} - q_{j-1}s_{j-1} \\ t_j &= t_{j-2} - q_{j-1}s_{j-1} \end{aligned}$$

for $j = [2, n]$ where the q_i are the quotients in the divisions of the Euclidean algorithm when it is used to find $gcd(a, b)$ [5].

Looking at the equation 2.2 in this theorem, it is very similar to the equation in

Bézout's theorem, see 2.1. The terms s_n and t_n here give the Bézout coefficients.

For a better understanding of theorems 2.2 and 2.3, it would be a good idea to show an example application using these theorems.

Example 1 Find $gcd(99, 78)$.
The solution will be in the following steps using theorem 2.2:

- The goal is to find $gcd(99, 78) = sa + tb$.
- The first application will be in the Euclidean Algorithm, see 2.2. It states that when finding gcd, the operations continue until the last remainder is zero. Then,

$$\begin{aligned} 99 &= 78 \times 1 + 21 \\ 78 &= 21 \times 3 + 15 \\ 21 &= 15 \times 1 + 6 \\ 15 &= 6 \times 2 + 3 \\ 6 &= 3 \times 2 + 0 \end{aligned} \tag{2.3}$$

  Because the last remainder is 3, the result is $gcd(99, 78) = 3$.

The theorem 2.3 allows operations to be performed starting from the result of the previous theorem. In other words, in order to apply theorem 2.3, the operations applied in theorem 2.2 are needed. We can compare the result of operations in these two theorems in this way:

Euclidean	**Extended Euclidean**
$15 = 6 \times 2 + 3$	$3 = 15 - 6 \times 2$
$21 = 15 \times 1 + 6$	$6 = 21 - 15 \times 1$
$78 = 21 \times 3 + 15$	$15 = 78 - 21 \times 3$
$99 = 78 \times 1 + 21$	$21 = 99 - 78 \times 1$

The application of the Extended Euclidean Algorithm cannot start from equation 2.3 since it has a 'zero' term. In other words, it is required a remainder which is different than 0 by the definition of the algorithm 2.3. In the final equation, the numbers 99 and 78 are reached. Accordingly, it is necessary to obtain the equation $3 = 15 - 6 \times 2$ in terms of 99 and 78. Based on all the

operations performed, this result is obtained.

$$\begin{aligned} 3 &= 78 \times 3 - (99 - 78 \times 1) \times 11 \\ &= 78 \times 3 - 99 \times 11 + 78 \times 11 \\ &= 78 \times (3 + 11) - 99 \times 11 \\ &= 78 \times 14 - 99 \times 11 \end{aligned}$$

The result becomes $3 = 99 \times (-11) + 78 \times 14$. So, by the equation 2.1, $s = -11$ and $t = 14$.

Theorem 2.4 (Fermat's Little Theorem) If p is prime and a is an integer such that p is not divisible by a, then

$$a^{(p-1)} \equiv 1 \pmod{p}$$

[5, 7].

2.2 Groups

Groups are one of the most well-known topics in mathematics, especially in abstract algebra. In addition, it is extremely important to mention this issue because groups are the main subject of this study. In this section, first of all, the binary operations part, which is the first topic of the groups, will be discussed; then, content such as the definition, characteristics and types of groups will be processed.

2.2.1 Binary Operations

Definition 2.1 A **binary operation** $*$ on a set S is a function for each ordered pair (a, b) such that $a, b \in S$. It is denoted by $a * b$ [3].

Example 2 Addition and multiplication are binary operations on $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$. It is also valid for subtraction on $\mathbb{Z}$, but not on $\mathbb{Z}^+$ [2].

Binary operations have different notations. For instance, $+$ is a binary operations for addition, $*$ is also a binary operation for multiplication [3]. However, the symbol $*$ is generally used for them [3].

Definition 2.2 Let $*$ be a binary on a set S and A be a subset of S. Then, A is **closed** under the binary operation $*$ if $a, b \in A$, then $a * b \in A$ [10].

Definition 2.3 Let $*$ be a binary operation on a set S and $a, b, c \in S$.

- If $a * b = b * a$, then it is **commutative** and
- if $(a * b) * c = a * (b * c)$, then it is **associative**.

Example 3 Addition and multiplication are commutative and associative on $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$, but subtraction is not [2].

Example 4 Let $M_n(\mathbb{R})$ be the set of all $n \times n$ matrices over $\mathbb{R}$. Then addition and multiplication are binary operations on the defined set. Addition and multiplication are commutative and associative (Note that multiplication is commutative when only $n = 1$) [2].

Definition 2.4 An element $e \in S$ is an **identity element** if $e * a = a * e = a$ $\forall a \in S$.

Example 5 In $M_n(\mathbb{R})$, the zero matrix 0_n is an identity for matrix addition and the eye matrix I_n is an identity for matrix multiplication [2].

2.2.2 Groups

In mathematics, groups have an important place. Groups are generally defined by a set G and a binary operation $*$ on G. Groups are expressed as $< G, * >$ [2].

Definition 2.5 $< G, * >$ can be defined as a group if the following axioms are satisfied:

- G is closed under a binary operation,
- The binary operation is associative,
- There is an identity element in G, i.e, there is an element $e \in G$ such that $e * g = g * e = g$ for all $g \in G$,
- Every element of G has an inverse in G, i.e, there is an inverse element $g' \in G$ for the element $g \in G$ such that $g * g' = g' * g = e$

[3, 10].

Definition 2.6 If the group $< G, * >$ has a associative binary operation $*$, then the group is **semigroup** [11].

Note that by the definition of the group, every group is also a semigroup [11].

Definition 2.7 Let S be a set. A **monoid** is a semigroup of functions on the set S that has an identity [12].

Definition 2.8 If the group $< G, * >$ has a commutative binary operation $*$, then the group is **abelian** [2].

Example 6 $< \mathbb{Z}, + >$, $< \mathbb{Q}, + >$, $< \mathbb{C}, + >$ and $< \mathbb{R}, + >$ are abelian groups [2].

Definition 2.9 The number of elements in the group G is called **order of** G and denoted by $|G|$ [3].

Definition 2.10 Let R represent $\mathbb{Z}_m$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{C}$, $\mathbb{R}$. Then the set $M_n(R)$ of all $n \times n$ matrices over R is a monoid using matrix multiplication. It is called **general linear group of degree** n **over** R and denoted by $GL_n(R)$ [3].

Definition 2.11 A **permutation** of a nonempty set S is a bijective function such that $S : S \to S$ [13].

Definition 2.12 A **permutation group** $(G, *)$ on a nonempty set S if the elements of the group are the permutations of the set and the binary operation is the composite function [13].

Definition 2.13 The set $S_n = \{1, 2, 3, \ldots, n\}$ is a group under the composition is called **symmetric group of degree** n.

Note that the order of the symmetric group S_n is $n!$ [2].

3
ENCRYPTION ALGORITHMS

In recent years, digital applications have been preferred when performing daily transactions such as shopping and banking transactions thanks to smartphones. This also requires security because tools containing personal information such as credit and debit cards are used while doing this. Such tools can sometimes be vulnerable to fraudulent jobs such as changing information, copying identity, etc [14]. Preventing this situation is related to the formation of information security [14]. The secure use of the information in these tools requires confidentiality and this confidentiality is ensured by encryption. This encryption is called 'cryptology' in mathematics [15]. This concept is formed by the combination of the words 'cruptos' (hidden) and 'logos' (science) [16].

Cryptology has been used many times by many people in history; one of the most well-known of these is Caesar [1, 17]. The encryption method invented by Caesar was based on the encryption of each letter in any open text by replacing it with the next k letters [18]. Here, k is considered a positive integer since it represents the number of keys. For instance, take $k = 3$ [1, 14]. If the word 'cryptology' is encrypted according to the Caesar's method, the word 'cryptology' becomes 'fubswrorjb' in English alphabet, 'etbşvrorıb' in Turkish alphabet. As can be seen, although the encryption method applied is the same, the encrypted texts may differ since different alphabets are used. In the World War 2, the Germans used the Enigma machine which was invented by Arthur Scherbius and later solved by Alan Turing and his team [1]. To date, many algorithms have been invented and improved, some of them have been broken. These studies continue today.

Cryptology is divided into two headings: cryptography and cryptanalysis. Cryptography is the science that examines the steps in the formation of any cipher with mathematical methods. Cryptanalysis, on the other hand, examines the analysis of the resulting cipher [19]. Cryptography is examined under two headings: encryption algorithms and decryption algorithms. In this section,

the concept 'encryption algorithms' will be examined which is one of the sub-disciplines of cryptography. This topic is widely seen in fields such as mathematics, electronics, banking systems etc. It has also an important place today with the development of technology.

Encryption algorithms have two different sub-disciplines: symmetric encryption algorithms and asymmetric encryption algorithms. To understand how these algorithms work, it is need to define some concepts in encryption. For example, the text that is requested to be encrypted is called 'plaintext' and the encrypted version of this text is called 'ciphertext' [19]. The process by which a sender converts a plaintext into ciphertext is called 'encryption', and the process by which the receiver reconstructs the plaintext through the ciphertext is called 'decryption' [18]. These concepts will be illustrated in detail through the figures in the next sections.

3.1 Symmetric Encryption Algorithm

Encryption algorithms contain keys and text. In the symmetric encryption algorithm, only a secret key is required to encrypt and decrypt it [16]. The purpose of this algorithm is to ensure that the key used in an encrypted text is also used when decrypting the same text [16]. The security of a symmetric encryption algorithm depends on the key used [1]. It is shown in the figure 3.1 how the symmetric encryption algorithm works.

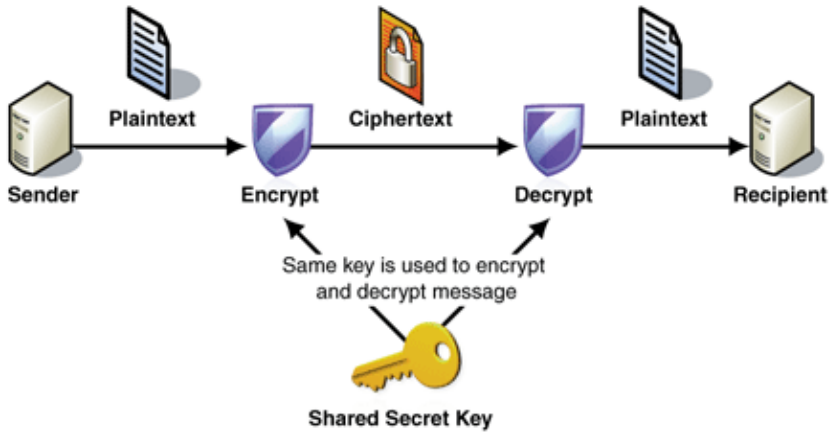


Figure 3.1 Symmetric encryption algorithm schema
[20]

Symmetric encryption algorithms are examined under two topics: block ciphers and stream ciphers. **Block cipher algorithms** are algorithms that process data or information made up of groups of bits into blocks [14, 17]. In block

cipher algorithms, when an input block is encrypted with the same key, it always produces the same output block [14]. The most common examples of these algorithms are DES (Data Encryption Standard) and AES (Advanced Encryption Standard) [17]. **Stream ciphers** are algorithms that take sequences of bits as data and produce them as output [14]. In stream encryption algorithms, the same blocks of inputs in different locations of a message generate different outputs [14]. The RC4 can be give as an example for stream ciphers algorithms [21].

3.2 Asymmetric Encryption Algorithm

Asymmetric encryption algorithms are algorithms used to solve mathematically difficult problems [16]. These algorithms are common in both encryption and transactions where identity verification is required, defined as a digital signature [1]. They are a secure way to perform operations such as authentication, privacy policies and so on. It is shown in the figure 3.2 how the asymmetric encryption algorithm works.

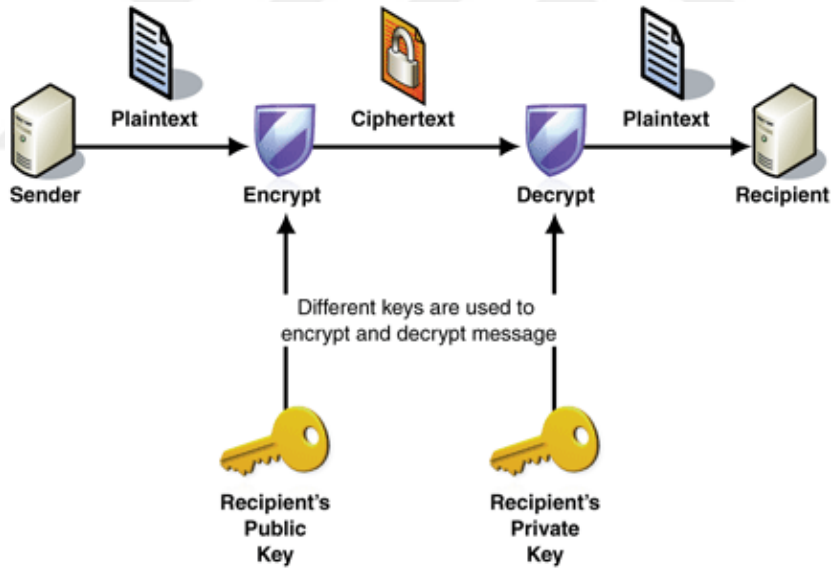


Figure 3.2 Asymmetric encryption algorithm schema [20]

Asymmetric encryption algorithm differs from symmetric encryption algorithm due to the key used. In asymmetric encryption algorithms, there are two different keys used: 'public key' and 'private key' [19]. In these algorithms, the public key is used for the encryption process, while the private key is used for the decryption process [19]. The working principle of encryption and digital signature processes is almost the same and is based on an asymmetric encryption algorithm. According to this algorithm, a user can encrypt a message using the

public key and decrypt it using only his private key [18]. Likewise, a user has to use the corresponding public key to verify the digital signature generated through the private key [18].

RSA is one of the most well-known asymmetric encryption algorithms. This algorithm, like other asymmetric encryption algorithms, has a pair of public and private keys. The details about the RSA will mentioned in the next chapter.

3.3 Comparison of Symmetric Encryption Algorithm and Asymmetric Encryption Algorithm

This section includes the differences between the symmetric encryption algorithms and the asymmetric encryption algorithms. In this section, it will done comparison of encryption algorithms according to their advantages and disadvantages. The comparison table 3.1 has been prepared using the bibliographies [14, 17, 22]:

Table 3.1 Comparison of symmetric encryption algorithm and asymmetric encryption algorithm

Symmetric Encryption Algorithm	**Asymmetric Encryption Algorithm**
It works faster	It works slower
The key size and number of bits used are smaller	The key size and number of bits used are bigger
Because the key needs to be changed frequently, it is difficult to distribute securely.	The key is not needed to have to changed for long times, so keys are easy to distribute securely.
It is mandatory to know and keep the key confidential for the communication between two different parties	It is sufficient for the parties to keep their private key secret.
DES, AES, Blowfish are the examples of the related algorithm.	DH (Diffie-Hellmann), RSA are the examples of the related algorithm.

4
RSA ALGORITHM

In this section, the definition, operation and applications of the RSA algorithm will be discussed in detail. For more information, see [23].

The RSA algorithm is an asymmetric encryption algorithm discussed by Ron Rivest, Adi Shamir and Leonard Adleman as "The fact that an encryption key is publicly available does not require the decryption key to be open.". Like other asymmetric encryption algorithms, the RSA algorithm has a pair of public and private keys. The public key is used to encrypt, while the secret key is used for decryption [19]. This algorithm, developed in the 70s, is generally based on calculating the product of two large prime numbers and the difficulty of factoring this number. The RSA algorithm is an algorithm that has been frequently encountered recently, especially in situations that require high-level security, such as the creation of passwords for bank cards.

In a public key cryptosystem, every user has the right to encrypt a public file, provided that it is subject to a certain procedure; and this procedure is considered as 'E'. The decryption procedure is kept only by the user who did the decryption and is hidden to other users; and this procedure should also be considered as 'D'. Assuming a defined message as M, the following properties are satisfied:

- When an encrypted M message is deciphered, the M message is obtained.

$$D(E(M)) = M \tag{4.1}$$

- Even if a user knows E, it is not easy for him/her to be able to calculate D; however, thanks to E, the user can learn the passwords of the encrypted messages, and thanks to these ciphers, an effective way to decrypt D.
- When a decrypted M message is encrypted, the M message is also

obtained.

$$E(D(M)) = M \tag{4.2}$$

Note that the equations 4.1 and 4.2 have the same result, M. This shows that the functions E and D are inverse in $\mathbb{Z}_n^*$.

The RSA algorithm is found in tools that provide security, such as digital signatures. There are mathematical operations behind the formation and transmission of this signature. These processes are given in the following steps, respectively:

- Bob who wants to convey an M message must be able to deliver that message to Alice, who is the receiver, using E_B. This message is denoted by the equation

  $$S = E_B(M) \tag{4.3}$$

  where S is the signature of Bob, M is the signed message and E_B is the function that is used by Bob to encrypt.

- Then, in order to keep the message secret, Bob must encrypt S without having to send M, and send the result of $E_A(S)$ to Alice, where E_A is the function that is used by Alice to encrypt.

- Alice then uses the D_A function, where D_A is the function that is used by Alice for decryption, to decrypt the ciphertext to compute S; this way, it knows that Bob is the person who sent the signature, and if necessary, it can be reflected in plain text attached to the S. Then the encryption procedure is extracted according to the sender of the message

  $$M = D_B(S) \tag{4.4}$$

  with the D_B function, where D_B is the function that is used by Bob for decryption.

As a result, Alice has the (M, S), called 'message-signature pair', which has similar characteristics to those of a signed document.

Each message transmitted and received must have its own unique signature. Mathematically transferred, the situation is explained as follows: When an $S = E_B(M)$ signature of the message $M = D_B(S)$ is generated, the recipient cannot intercept this message, edit the message or forge the corresponding

signature. If a message is requested to be changed, a message different from M must be generated as $M' = D_B(S')$ and the signature $S' = E_B(M')$ must be created. The creation of a signature for a message is based on this procedure.

4.1 Processes of the Algorithm

So far, the RSA algorithm has been mentioned in general and how the signature of a message produced in this algorithm is created. In this section, it will be mentioned that the steps of RSA in the formation of encryption and decryption keys are created with the help of **key generation**, **encryption** and **decryption** cases.

In the previous section, it was mentioned that the encryption procedure represents 'E' and the decryption represents 'D'. In the RSA algorithm, the encryption and decryption functions are defined in general in the following way:

Let $e, d, n \in \mathbb{Z}^+$, $M \in (0, n-1)$ and $C \in (0, n-1)$. Then, the following equations are provided:

$$E(M) = M^e \pmod{n} = C \tag{4.5}$$

for a message M, and

$$D(C) = C^d \pmod{n} = M \tag{4.6}$$

for a ciphertext C.

Equation 4.5 (e, n) constitutes the encryption key which is public and equation 4.6 (d, n) constitutes the decryption key that is private [24]. The goal here is not to encrypt the message, but simply to put it into the digital form required for encryption. The message is first converted into a number and then the encryption process is performed. This encryption process is done by taking the eth exponent in the encryption key of the message in numerical form and then taking the mod n value of the result found. The process of deciphering a ciphertext consists of similar steps. This time, it is necessary to take the dth exponent in the decryption key and calculate the value of mod n. Thus, a ciphertext is returned back to the M message.

After this step, it is necessary to determine the encryption and decryption keys. Before determining, some concepts have to known.

Definition 4.1 Let $n > 0$. **Euler's totient function**, denoted by $\phi(n)$, is the number of integers that are less than and relatively prime to n. Note that $\phi(n)$ has an interval $[1, n-1]$ for the integer n [6].

Corollary 4.1 Define $k \in \mathbb{Z}^+$. There is a group $\mathbb{Z}_n^* = \{k|\ k \in \mathbb{Z}_n,\ \gcd(k,n) = 1\}$. Then, $|\mathbb{Z}_n^*| = \phi(n)$ [8].

Example 7 $\phi(1) = 1$ because 1 has no positive integers less than it.

Example 8 $\phi(2) = 1$ because 1 is coprime to 2, i.e. there is only one number coprime to 2.

Example 9 $\phi(3) = 2$ because 1 and 2 are coprime to 3, i.e. there are two numbers coprime to 2.

Properties of $\phi(n)$

- If n is prime, then $\phi(n) = n - 1$ [24]. The example 9 can be given as an example.
- If m and n are integers that are relatively prime, then $\phi(mn) = \phi(m)\phi(n)$ [24].

  Example 10 $\phi(6) = \phi(3 * 2) = \phi(3)\phi(2) = 2 * 1 = 2$
- Let p and k be positive integers. If n is the k th power of p, i.e. $n = p^k$, then

  $$\phi(n) = \phi(p^k) = p^k - p^{k-1} = p^k(1 - \frac{1}{p})$$

  [9].

  Example 11 $\phi(4) = \phi(2^2) = 2^2 - 2^{2-1} = 4 - 2 = 2$ is correct since there are two numbers that is coprime to 4: 1 and 3.
- If n contains more than one prime factor, i.e. for $n = p_1^{e_1} p_2^{e_2} \dots p_s^{e_s}$ where p_i are prime factors of n, then the equation

  $$\phi(n) = n \prod_{i=s} \left(1 - \frac{1}{p^i}\right)$$

  [25] is generally used.

  Example 12 Take $n = 35 = 5 * 7$. Then,

  $$\phi(35) = 35(1 - \frac{1}{5})(1 - \frac{1}{7}) = 24$$

Theorem 4.1 (Euler's Theorem) Let $n \in \mathbb{Z}^+$ and $a \in \mathbb{Z}$. If $gcd(a, n) = 1$, i.e., if a and n are relatively prime, then

$$a^{\phi(n)} \equiv 1 \bmod n$$

[6, 8].

Key generation is a fundamental step of the RSA algorithm. In this step, two different keys, the public key and the private key, are generated. The following is the process of generating these keys in order:

1. A number n is taken so that

   $$n = pq$$

   where p and q are the two major prime numbers. Although everyone knows the number n here, the numbers p and q will remain hidden because this number is hardly factored. This means that the derivation of d from e is hidden.

2. Using the number n, the Euler function, i.e. $\phi(n)$, is calculated. Since p and q are prime, the value of this function is obtained as

   $$\phi(n) = (p-1)(q-1)$$

   using the property 4.1.

3. In the interval $(1, \phi(n))$, a random number e is chosen that satisfies the condition $gcd(e, \phi(n)) = 1$. With the number of e found, the public key (e, n) that performs the encryption process is created.

4. In the interval $(1, \phi(n))$, the number d is obtained that satisfies the condition

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

   With the number d obtained, the secret key (d, n) is generated. Decryption is done with this secret key.

The Euler's totient function and Euler's Theorem are also important topics in RSA because they play a critical role in determining the correctness of encryption and decryption stages. The accuracy of a decryption algorithm

depends on the encryption and decryption stages, which are shown in the following steps.

Choose a message M so that it is a coprime to n, where M is a positive integer:

$$M^{\phi(n)} \equiv 1 \pmod{n} \tag{4.7}$$

The equation 4.7 represents the Euler's theorem. In this case, by the property 4.1 of the totient function, it becomes

$$\begin{aligned} \phi(n) &= \phi(p)\phi(q) \\ &= (p-1)(q-1) \\ &= pq - p - q + 1 \\ &= n - (p+q) + 1 \end{aligned} \tag{4.8}$$

Since d is relatively prime to $(p-1)(q-1)$, is equal to $\phi(n)$ and has a multiplicative inverse e, the result is

$$ed = 1 \bmod \phi(n) \tag{4.9}$$

As a result of these steps, it is seen that deciphering works properly if e and d are selected correctly.

What needs to be done next is to show that the previously mentioned functions E and D satisfy this state. Now, recall the equations 4.5 and 4.6. If the processes above are applied, then the equations become

$$E(D(M)) = (D(M))^e = (M^d)^e = M^{ed} \pmod{n} \tag{4.10}$$

$$D(E(M)) = (E(M))^d = (M^e)^d = M^{ed} \pmod{n} \tag{4.11}$$

In the equation 4.10, first encryption, then decryption is done; whereas in the equation 4.11 first decryption, then encryption is done.

The equation 4.9 provides the equality

$$M^{ed} \pmod{n} = M^{k\phi(n)+1} \pmod{n}$$

for some $k \in \mathbb{Z}$.

From 4.7 and using the property 4.1, it can be written

$$M^{(p-1)} \equiv 1 \pmod{p} \tag{4.12}$$

for all M. Since $(p-1)$ divides $\phi(n)$, it can be written step by step as

$$\begin{aligned} M^{(p-1)} &\equiv 1 \pmod{p} \\ M.M^{(p-1)} &\equiv M.1 \pmod{p} \\ M^{p-1+1} &\equiv M \pmod{p} \\ M^{k\phi(n)+1} &\equiv M \pmod{p} \end{aligned} \tag{4.13}$$

Similarly, since $(q-1)$ divides $\phi(n)$, the formula below is also valid:

$$M^{k\phi(n)+1} \equiv M \pmod{q} \tag{4.14}$$

Together 4.13 and 4.14 imply

$$M^{ed} \equiv M^{k\phi(n)+1} \equiv M \pmod{n} \tag{4.15}$$

for all M, in the interval $[0, n)$. Therefore, D and E are inverse permutations.

4.2 Other Applications in the RSA

4.2.1 Exponentiation by Repeated Squaring and Multiplication Algorithm

This algorithm occupies an important place in RSA. In the algorithm, it is explained that the expression $M^e \pmod{n}$ contains at most $2\log_2(e)$ multiplication and division. Here,

- 2 determines that the coefficient in $2\log_2(e)$ contains two operations, multiplication and division,
- $\log_2$ determines that the algorithm works in a **binary system**,
- e determines the encryption key.

The algorithm is generally applied for encryption. The way the algorithm works is explained in these steps:

1. Suppose $e_0, e_1, e_2, \ldots, e_m$ be binary representations of e.

2. Take $C = 1$.

3. For $i = [0, m]$,

   - if $e_i \pmod{n} \equiv 0$, i.e. e_i is even, then set C to C^2.
   - else set C to $C \times M$ until $e_i \pmod{n} \equiv 0$.

4. As a result of all the steps, we can state that C is the encrypted form of M.

Note that these steps can be applied for decryption process. This time, d is used instead of e.

Here's an example of Python code for a better understanding of the algorithm:

```
def exponentiation(base, exponent, modulus):
result = 1;
base = base % modulus;
while exponent > 0:
    if exponent % 2 == 1:
        result = (result * base) % modulus;
    base = (base * base) % modulus;
    exponent //= 2;
return result;

base = 13
exponent = 4
modulus = 17
print(exponentiation(base, exponent, modulus))
```

In this code, the values of base, exponent and modulus are given as an example. According to this code, $result = 1$ represents $C = 1$ in the step 2 of the algorithm. Since 1 is actually an identity (unit element) in multiplication groups, it is taken for both. $base = base \ \% \ modulus$ determines the base. In the while loop, exponent, i.e. the e, is expected to be positive and the steps in the loop are the code of the procedures in the step 3.

In this code, base=13 is chosen. When the $base = base \ \% \ modulus$ command is applied, the first base is determined as 13 because $13 = 13 \pmod{17}$. Then, through the while loop, it is checked that exponent, i.e. e is a positive

number. In the example, since exponent is selected as 4, the loop is entered and the operations in the loop are performed. Since e is an even number, the $base = (base * base)\ \%\ modulus$ command will be used directly. This command will give the result $(13 * 13) \pmod{17} = 169 \pmod{17} = 16$, and thus 16 will be the other base specified. Since the modulus value is 17, operations have been performed according to the $\pmod{17}$ value throughout the code. The result of the code is found to be $(16^2) \pmod{17} = 256 \pmod{17} = 1$.

4.2.2 Computing e from d and $\phi(n)$

One of the other applications of RSA is computing e using d and $\phi(n)$. The aim is to determine that e and d are multiplicative inverses. In this section, the Euclidean and the Extended Euclidean algorithms in the 'Basic Concepts', see chapter 2, will be used.

Here, the main aim is to calculate $gcd(\phi(n), d)$ by computing series $x_0, x_1, x_2, \ldots$ where the following properties satisfy:

$$\begin{aligned} x_0 &= \phi(n) \\ x_1 &= d \\ x_{i+1} &= x_{i-1} \pmod{x_i} \end{aligned} \tag{4.16}$$

The equation 4.16 is based on Euclidean Algorithm. It is used continuously until an $x_k = 0$ is found, i.e., until the equality $gcd(x_0, x_1) = gcd(\phi(n), d) = x_{k-1}$ is provided.

Compute for each x_i numbers a_i and b_i such that

$$x_i = a_i x_0 + b_i x_1 \tag{4.17}$$

Recalling the Bézout's Theorem, it is seen that constants x_0 and x_1 replace the constants a and b in the equation 2.1. Using equation 4.17, the goal is to compute b_{k-1} is the multiplicative inverse of $x_1 \pmod{x_0}$ if $x_{k-1} = 1$ is found.

For a better understanding of this topic, it can be reinforced with a simple example.

Example 13 Let $\phi(n) = 35$ and $d = 12$.
Using the properties above, it can be said $x_0 = 35$ and $x_1 = 12$.

Next, the equation 4.16 will be used. Then,

$$\begin{aligned} n = 1 \to x_2 &= x_0 \pmod{x_1} \\ &= 35 \pmod{12} = 11 \\ n = 2 \to x_3 &= x_1 \pmod{x_2} \\ &= 12 \pmod{11} = 1 \\ n = 3 \to x_4 &= x_2 \pmod{x_3} \\ &= 11 \pmod{1} = 0 \end{aligned}$$

The process is over when $x_4 = 0$ is found. Since the $gcd(x_0, x_1)$ has to be different from 0, the result is $gcd(35, 12) = 1$.
Now, let's determine the Bézout's coefficients using $x_0 = 35$ and $x_1 = 12$. This step is required to find d. Using equation 2.1, it is represented as

$$\begin{aligned} gcd(x_0, x_1) &= s \times x_0 + t \times x_1 \\ &= s \times 35 + t \times 12 \end{aligned}$$

It is known that Extended Euclidean Algorithm is based on the Bézout's Theorem, so the Extended Euclidean Algorithm will be used in this step.

Euclidean	**Extended Euclidean**
$35 = 12 \times 2 + 11$	$11 = 35 - 12 \times 2$
$12 = 11 \times 1 + 1$	$1 = 12 - 11 \times 1$
$11 = 1 \times 11 + 0$	$-$

Now, substitute the equation $11 = 35 - 12 \times 2$ into the equation $1 = 12 - 11 \times 1$. Then, this implies

$$\begin{aligned} 1 &= 12 - 11 \times 1 \\ &= 12 - 1 \times (35 - 12 \times 2) \\ &= -1 \times 35 + 12 \times (2 + 1) \\ &= (-1) \times 35 + 12 \times 3 \end{aligned} \tag{4.18}$$

Then, by the equation 4.18, the coefficients are $s = -1$ and $t = 3$.

Now, let's create the encryption and decryption key pairs in RSA through this example.
Recall example 13. It is given $\phi(n) = 35$ and $d = 12$ and as a result, it has found the value $t = 3$ as one of the Bézout's coefficients. In this point, by the

equation below

$$d \times e \equiv 1 \pmod{\phi(n)} \tag{4.19}$$
$$e = d^{-1} \pmod{\phi(n)} \tag{4.20}$$

which exists in the RSA, it is written as $12 \times e \equiv 1 \pmod{35}$. Thus, since $t = 3$ is the multiplicative inverse of $d = 12$, by the equation 4.18, it becomes the encryption key, i.e., it can be written as $e = 3$.

In the first operations, $d = 12$ is any constant, but when substituted in the formulas in the RSA algorithm, it can be said that d in the $d = 12$ is the decryption key. Thus, our existing key pairs are $(e, n) = (3, n)$ for encryption and $(d, n) = (12, n)$ for decryption.

The next example will include the basic steps of the RSA that are key generation, encryption and decryption. In this example, the encryption and decryption processes will be done via given two prime numbers.

Example 14 Let $p = 47$ and $q = 59$.
Recalling the key generation step of RSA, the values n and $\phi(n)$ will become

$$n = p * q = 47 * 59 = 2773$$

and

$$\begin{aligned} \phi(n) &= (p-1)(q-1) \\ &= 46 * 58 \\ &= 2668 \end{aligned}$$

Now, let's determine the encryption key e and the decryption key d. Then, by the step 3 in the key generation, say $d = 157$. Then, the processes in example 13 will be used. From this example, it is known that $x_0 = \phi(n) = 2668$ and $x_1 = d = 157$. Then, this implies

$$
\begin{aligned}
n = 1 \rightarrow x_2 &= x_0 \pmod{x_1} \\
&= 2668 \pmod{157} = 156 \\
n = 2 \rightarrow x_3 &= x_1 \pmod{x_2} \\
&= 157 \pmod{156} = 1 \\
n = 3 \rightarrow x_4 &= x_2 \pmod{x_3} \\
&= 156 \pmod{1} = 0
\end{aligned}
$$

At this point, the Euclidean and the Extended Euclidean Algorithms will be used. Thus,

Euclidean	**Extended Euclidean**
$2668 = 16 \times 157 + 156$	$156 = 2668 - 16 \times 157$
$157 = 156 \times 1 + 1$	$1 = 157 - 156 \times 1$
$156 = 156 \times 1 + 0$	$-$

Now, substituting the equation $156 = 2668 - 16 \times 157$ into the equation $1 = 157 - 156 \times 1$, it becomes

$$
\begin{aligned}
1 &= 157 - 1 \times (2668 - 16 \times 157) \\
&= -1 \times 2668 + 157 \times (16 + 1) \\
&= -1 \times 2668 + 157 \times 17 \qquad (4.21)
\end{aligned}
$$

and by applying the equations 4.19 and 4.20, e becomes

$$
\begin{aligned}
d \times e &\equiv 1 \pmod{\phi(n)} \\
157 \times e &\equiv 1 \pmod{2668} \qquad (4.22)
\end{aligned}
$$

Since the equation 4.21 is a representation of the equation 4.22, by the result 4.22, $e = 17$ will be determined. The public and private keys are

$$
\begin{aligned}
(e, n) &= (17, 2773) \\
(d, n) &= (157, 2773)
\end{aligned}
$$

Now, choose a message M that provides $M < n$. Suppose $M = 920$. Then, by

the equation 4.5, this implies

$$C = M^e \pmod{n} = 920^{17} \pmod{2773} \tag{4.23}$$

Since $17 = (10001)_2$, 4.23 can be written as

$$C = M^e \pmod{n} = \left(\left((920^2)^2\right)^2\right)^2 \times 920 \pmod{2773} \tag{4.24}$$

Then, 4.24 becomes

$$C = M^e \pmod{n} = \left(\left((920^2)^2\right)^2\right)^2 \times 920 = 948 \pmod{2773} \tag{4.25}$$

The result in 4.25 gives the ciphertext. The equation 4.6 gives the message M back, which is shown below

$$M = 948^{157} \pmod{2773} = 920 \tag{4.26}$$

4.2.3 Computing $\phi(n)$ Without Factoring n

It was known that RSA is an algorithm based on the difficulty of factoring the number obtained from the product of two large prime numbers. Also, RSA is an algorithm where $\phi(n)$ is also difficult to find; which explains that RSA is a secure algorithm. If $\phi(n)$ could be easily calculated, the number n would also be known, and thus the factors of n would be easy to find. This means that the system can be attacked by an attacker. This is how this topic is treated mathematically:

In the previous sections,

$$n = pq \tag{4.27}$$

and

$$\phi(n) = (p-1)(q-1) \tag{4.28}$$

are given. By using the equation 4.28, it becomes

$$\begin{aligned} \phi(n) &= (p-1)(q-1) \\ &= pq - p - q + 1 \\ &= pq - (p+q) + 1 \\ &= n - (p+q) + 1 \\ \implies & \boxed{p+q = n+1-\phi(n)} \end{aligned} \tag{4.29}$$

Now, we know the equations 4.27 and 4.29. Then, by using the equations, $p-q$ becomes

$$\begin{aligned}(p+q)^2-(p-q)^2 &= 4pq \\ (p-q)^2 &= (p+q)^2-4n \\ \Longrightarrow & \boxed{p-q=\sqrt{(p+q)^2-4n}} \end{aligned} \tag{4.30}$$

Adding and subtracting the terms $(p+q)$ and $(p-q)$ give p and q. Thus,

$$\begin{aligned}(p+q)+(p-q) &= 2p \\ \Longrightarrow & \boxed{p=\frac{(p+q)+(p-q)}{2}} \end{aligned} \tag{4.31}$$

and

$$\begin{aligned}(p+q)-(p-q) &= 2q \\ \Longrightarrow & \boxed{q=\frac{(p+q)-(p-q)}{2}} \end{aligned} \tag{4.32}$$

An example can be given for a better understanding of the subject.

Example 15 Let $n=55$. The number 55 can be written as a product of two prime numbers, for instance $55=11\times 5$. Then, the values $p=11$ and $q=5$ can be taken. Thus,

$$\begin{aligned}\phi(n) &= (p-1)(q-1) \\ &= (11-1)(5-1) \\ &= 10\times 4 \\ &= 40 \end{aligned}$$

Now, the equations 4.29, 4.30, 4.31 and 4.32 can be applied respectively:

$$\begin{aligned}p+q &= n+1-\phi(n) \\ 11+5 &= 55+1-40 \\ 16 &= 16 \end{aligned}$$

$$
\begin{aligned}
p - q &= \sqrt{(p+q)^2 - 4n} \\
11 - 5 &= \sqrt{(11+5)^2 - 4 \times 55} \\
6 &= \sqrt{16^2 - 220} \\
6 &= \sqrt{256 - 220} \\
6 &= \sqrt{36} \\
6 &= 6
\end{aligned}
$$

$$
\begin{aligned}
p &= \frac{(p+q) + (p-q)}{2} \\
11 &= \frac{(11+5) + (11-5)}{2} \\
11 &= \frac{16+6}{2} \\
11 &= \frac{22}{2} \\
11 &= 11
\end{aligned}
$$

$$
\begin{aligned}
q &= \frac{(p+q) - (p-q)}{2} \\
5 &= \frac{(11+5) - (11-5)}{2} \\
5 &= \frac{16-6}{2} \\
5 &= \frac{10}{2} \\
5 &= 5
\end{aligned}
$$

When the two-sided operation is applied in all equations, it is clear that the values of p and q are found correctly because equality is achieved on both sides.

5
ANALYSIS OF THE RSA ALGORITHM: SPEED PERFORMANCE AND OPTIMIZATIONS

RSA is a common and useful asymmetric encryption algorithm; however, it has major issues such as high cost and slow performance. At this point, an analytical study has been conducted to increase the speed of RSA using various optimization techniques. In this way, it is aimed to make the algorithm work more efficiently.

It is known that RSA has a performance issue due to its operations with large prime numbers, which causes the algorithm to run slowly, but it is possible to make RSA run faster, that is, to improve its performance. The purpose of this section is to solve the performance problem. In this context, the Chinese Remainder Theorem, Montgomery and Barrett algorithms will be discussed as optimization techniques, respectively. In previous studies, the Chinese Remainder Theorem [26, 27] in papers, the Montgomery algorithm [28] in a paper, and the Barrett algorithm [29] in a paper have been examined both in the context of RSA and individually on their own. Additionally, the application of the Chinese Remainder Theorem in the RSA algorithm is briefly mentioned in [30]. In a recent publication [31], Montgomery and Barrett algorithms are compared, while in another paper [32], a comparison is made between the Chinese Remainder Theorem and RSA. In the studies conducted so far, there has been no comparative study on the application of the Montgomery and Barrett algorithms in encryption and decryption over RSA; this situation has formed the foundation of this study. This study is significant as it compares encryption and decryption by considering RSA, the Chinese Remainder Theorem, Montgomery and Barrett optimizations together. In this section, it is expected to analyze the RSA algorithm, Chinese Remainder Theorem, Montgomery and Barrett methods through Python, compare the

results obtained from these analyses with the help of tables and graphs; in this way, determine the most suitable method in terms of speed and efficiency.

5.1 Chinese Remainder Theorem

The Chinese Remainder Theorem, also known as CRT in mathematics, has applications in the field of cryptography [33]. The CRT method is a method based on finding solutions to more than one modular equation. In particular, it is often used to calculate large numbers quickly. It is known that the history of CRT dates back to about 1795 [33].

The Chinese Remainder Theorem is briefly explained as follows:

Theorem 5.1 Let the numbers $m_1, m_2, \ldots, m_n$ be defined which satisfies $gcd(m_i, m_j) = 1$ provided that $i, j \leq n$ and $i \neq j$. Then, there is a system of modular equation

$$\begin{aligned} X &\equiv a_1 \pmod{m_1} \\ X &\equiv a_2 \pmod{m_2} \\ &\vdots \\ X &\equiv a_n \pmod{m_n} \end{aligned}$$

and there is a unique modular solution $X = \sum_{i=1}^{k} a_i M_i y_i \pmod{M}$ of the integers $m_1, m_2, \ldots, m_n$ in this system satisfying the conditions $M_i = \frac{M}{m_i}$ and $M_i y_i = 1 \pmod{m_i}$ for some y_i [26].

5.2 Montgomery Algorithm

The Montgomery algorithm is an algorithm developed in the 80s and takes its name from Peter Montgomery [28]. This algorithm was developed for the purpose of comprehensively calculating some algorithms, such as RSA [28]. In this article, the Montgomery algorithm is shown as one of the optimizations that affect the speed of the RSA algorithm.
The notation of Montgomery's algorithm is as follows [28]:

Algorithm 1 Montgomery Algorithm

function $REDC(T)$
$m \leftarrow (T \bmod R)N' \bmod R$
$t \leftarrow (T + mN)/R$
if $t \geq N$
return $t - N$

else
return t

In this algorithm, the terms represent

$REDC \rightarrow$ Montgomery Reduction algorithm,
$T \rightarrow$ the number for which modular reduction is desired,
$R \rightarrow$ the Montgomery base, which is generally denoted by $R = 2^k$,
$N \rightarrow$ the modulus,
$N' \rightarrow$ the inverse of N with respect to mod$R$ $\left(NN' \equiv -1 \bmod R\right)$,
$m \rightarrow$ intermediate correction coefficient,
t in the step $t \leftarrow (T + mN)/R \rightarrow$ intermediate modular value,
t in the step $return\ t \rightarrow$ the final modular result.

The goal of the algorithm is to reduce T under mod N and convert operations performed in Montgomery form to normal form. As can be seen, since there is no direct division in the algorithm, even if large numbers are used, a faster result is obtained than classical modular arithmetic.

5.3 Barrett Algorithm

The Barrett algorithm is a technique developed in 1986 and named after Paul Barrett [29]. The purpose of the algorithm is to find the value c that satisfies the value $c = d \bmod n$, as stated in the 'output' section in algorithm shown below. The Barrett algorithm is represented as follows [34]:

Algorithm 2 Barrett Algorithm

Input: $n \leq 2^N, \quad d \leq 2^D, \quad \mu = \lfloor \frac{2^L}{n} \rfloor$ where $N \leq D \leq L$
Output: $c = d \bmod n$
$c_1 \leftarrow d \gg (N - 1)$
$c_2 \leftarrow c_1 \mu$
$c_3 \leftarrow c_2 \gg (L - N + 1)$
$c_4 \leftarrow d - nc_3$
while $c_4 \geq n$ **do**
$c_4 \leftarrow c_4 - n$
return c_4

In this algorithm, the terms represent

$n \rightarrow$ the modulus,

$d \rightarrow$ the number that the mode will be taken,

$\mu \rightarrow$ the value to be used for the calculation of the n module, also known as the precalculation value in the Barrett algorithm [35],

$N \rightarrow$ the bit length of n,
$D \rightarrow$ the bit length of d,
$L \rightarrow$ the length of bits for calculating the number μ,
$c_1 \rightarrow$ the value in the first step of calculating the approximate value of d,
$c_2 \rightarrow$ the value obtained by multiplying c_1 and μ,

$c_3 \rightarrow$ with the help of the obtained c_2 value, the value that reduces d to its approximate value,

$c_4 \rightarrow$ the value c in the step $c = d \bmod n$.

The point to note here is that the values c_1, c_2 and c_3 are intermediate values in the algorithm. Thanks to these three values, the algorithm reaches a solution faster.

5.4 Comparison of the Algorithms

The performance of each algorithm is different from each other. This section uses an environment with an Intel(R) Core(TM) i5-2410M CPU 2.30 GHz processor and Windows 10 operating system to compare the algorithms in terms of speed. This comparison was made by first creating the values using Python codes, and then showing the relationships between each other with the help of tools such as tables and graphs. The following tables 5.1, 5.2 and 5.3 show the encryption and decryption times in seconds for the algorithms with bit lengths of 1024, 2048 and 4096, respectively.

Table 5.1 Encryption and decryption times of algorithms for 1024 bits.

Algorithm	**Encryption Duration (s)**	**Decryption Duration (s)**
RSA	0.00095984	0.00234954
CRT	0.00011984	0.00320810
Montgomery	0.00002016	0.00002652
Barrett	0.00010220	0.00662896

Table 5.2 Encryption and decryption times of algorithms for 2048 bits.

Algorithm	**Encryption Duration (s)**	**Decryption Duration (s)**
RSA	0.00159344	0.00731684
CRT	0.00026782	0.01310590
Montgomery	0.00004086	0.00006210
Barrett	0.00034214	0.04480350

Table 5.3 Encryption and decryption times of algorithms for 4096 bits.

Algorithm	**Encryption Duration (s)**	**Decryption Duration (s)**
RSA	0.00455008	0.03967240
CRT	0.00099742	0.09114390
Montgomery	0.00012394	0.00019608
Barrett	0.00104042	0.29904300

The chart is given in the figure 5.1 that is based on the values in the tables. In this graph, encryption and decryption times are discussed separately. On the graph, the bit lengths of the algorithms are shown on the horizontal axis and the running times of the algorithms in seconds are shown on the vertical axis. In order to make the graph more readable, each algorithm is shown in a different color. In the graph,

- the Standart RSA, also known as RSA, is shown in blue,
- the CRT is shown in yellow,
- the Montgomery Algorithm is shown in green,
- the Barrett Algorithm is shown in red.

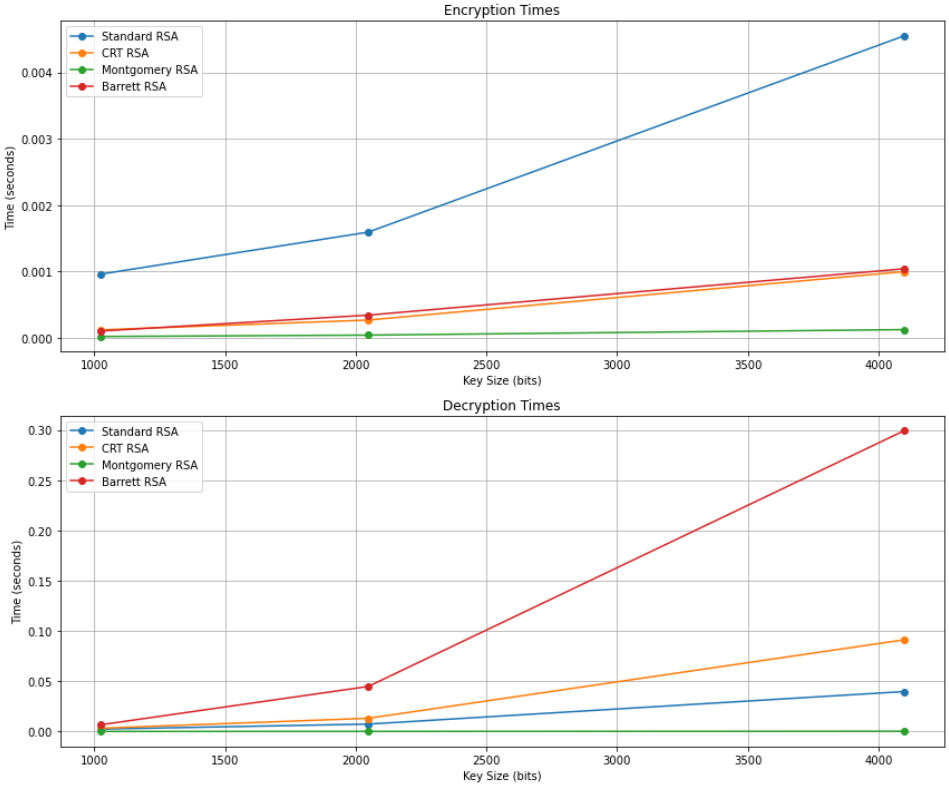


Figure 5.1 Encryption and decryption times for 1024, 2048, and 4096 bit lengths of algorithms

As can be seen from this graph, the encryption process of the RSA algorithm is completed more slowly than other algorithms in all bit lengths, while the decryption process is performed faster. The point of note here is that the Montgomery algorithm is the fastest to perform both the encryption and decryption steps for all bit lengths. Moreover, the encryption and decryption times of this algorithm are observed to be closer to each other numerically. On the other hand, just like the Montgomery algorithm, the Chinese Remainder Theorem is stable in terms of encryption and decryption times, while the Barrett algorithm is found to be the slowest algorithm in decryption for a length of 4096 bits.

6
ATTACKS on the RSA ALGORITHM

RSA has been widely used, especially in recent years. Although this algorithm is secure, it can be vulnerable to attack and this can have some bad consequences. In this section, the types of attacks of RSA and the techniques to prevent these attacks will be discussed.

An attack can be defined as a deliberate technical action to collapse the security policy of a system [36]. Attacks are generally examined in two main categories: active attacks and passive attacks. Passive attacks are attacks made by the attacker to observe and collect data. The system is not damaged, but the confidentiality of the data is threatened [37]. An attacker who conducts a passive attack does not interfere with the system; therefore, these attacks are very difficult to detect. Active attacks, on the other hand, are carried out to change the system and directly interfere with the information in the system. When this attack is carried out, important information such as the confidentiality of data and authentication is threatened [37]. Since the target of the attack is directly the operation of the system, this attack can be easily detected.

RSA has been subjected to many attacks since it was invented. These attacks are categorized under passive and active attacks.

6.1 Passive Attacks on RSA

6.1.1 Side Channel Attacks

Side-Channel Attack (SCA) is one of the most frequently mentioned attack types in RSA and has been the subject of significant study in the 90s [38]. A side-channel attack is a passive attack that analyzes behaviors such as timing, power consumption, electromagnetic radiation and targets the secret key [39, 40]. The purpose of this attack is to obtain the secret key in the RSA. Since the attack is applied directly to the system, it can be said to be hardware-based

[41].
Side channel attacks can generally be examined under two subheadings: timing attacks and power analysis attacks.

6.1.1.1 Timing Attacks

Timing attack is one of the attacks on RSA. The goal of this attack is conveyed by Kocher as follows: To reveal the secret key of an encrypted message in a system with the help of decryption time [42, 43]. In short, the attack is done to reveal the secret key d.

In RSA, the timing attack is done with the 'Exponentiation by Repeated Squaring and Multiplication Algorithm', which is mentioned in the previous sections. Detailed information about this algorithm can be found 4.2.1. In this section, the way the algorithm is used will be based on the number of bits; that is, it is calculated by processing into the exponent bits used. If the number of bits in the algorithm is 1, then both square and multiplication are used; if it is 0, only the square operation is used [44, 45]. Since multiplication is additionally performed in 1-bit operations, the processing time will also be extended. By examining the measured times, the attacker can guess the bits of the secret key from the difference between them.

Preventing a timing attack means preventing the relationship between the system's uptime and the hidden base. One solution to create this situation is to add a delay for each bit to process in a fixed time using modular exponentiation [46]. This process consists of treating each step as if multiplication is being performed. Another solution is the more preferred 'blinding' technique recommended by Rivest [44]. This technique was developed to prevent the attack by generating random data [45]. The technique is demonstrated mathematically with the help of the following steps using the RSA private-key operation $M = C^d \bmod N$ [36]:

1. A large number r is chosen randomly so that $gcd(r, N) = 1$. New random r must be selected for each message.

2. Instead of solving C directly, it is arranged so that

$$C' = (C \times r^e) \bmod N$$

   In this embodiment, e gives the public exponent value and $r^e \bmod N$ gives the random masking factor. At this point, the attacker sees C'

instead of C.

3. Now because C' is seen, $M' = (C')^d \bmod N$ is solved. Since $C' = C \times r^e$, the system is solved in this way:

$$\begin{aligned} M' = (C')^d \bmod N &= (C \times r^e)^d \bmod N \\ &= (C^d \times (r^e)^d) \bmod N \\ &= (C^d \times r^{ed}) \bmod N \end{aligned} \tag{6.1}$$

4. From the definition of RSA, since $ed = 1 \bmod \phi(n)$, $r^{ed} = r^1 = r \bmod N$ will be satisfied. As a result, the equation 6.1 becomes

$$M' = (C^d \times r) \bmod N$$

via the following steps:

$$\begin{aligned} M' &= (C^d \times r) \bmod N \\ &= C^d \bmod N \times r \bmod N \\ &= M \times r \bmod N \\ \implies M &= M' \bmod N \times r^{-1} \bmod N \\ M &= (M' \times r^{-1}) \bmod N \end{aligned}$$

6.1.1.2 Power Analysis Attack

Power analysis attack is one of the types of side-channel attacks. This attack aims to reveal the key of a device by measuring and analyzing the power consumption in electronic devices, which can also be characterized cryptographically [47, 48]. This power consumption varies based on the device's data and can be defined as an operation that the attacker must perform during both encryption and decryption processes to find the key [48].

In power analysis attack, the 'gate' and the 'noise' are quite significant concepts. A **gate** (or **logic gate**) is a fundamental element that performs logic operations in digital circuits by operating through digital signals. Digital signals generally consist of bits, 0 and 1, and logic operations use AND, OR, and XOR. For detailed information on this topic, see [49]. A **noise** is an element that remains outside the targeted operations in power consumption. Thanks to it, the attacker cannot directly access the information they are targeting.

Power analysis attack is examined under two different headings: **simple power**

analysis (SPA) and **differential power analysis (DPA)**. It has been emphasized that 'the first approach mentioned in the literature' for SPA, which is a type of power analysis attack [50]. SPA is an attack that helps to obtain information about the system and keys through the patterns created by power consumption [51]. In this attack, it is expected that the attacker first observes the behavior of the device through the patterns created, to capture all the necessary information about the device, and then, thanks to this information, to easily carry out targeted operations such as key recovery [51].

The change in power consumption over time in SPA can be expressed mathematically. This expression gives the total power consumed at gate g at time t. Noise also accompanies this total because noise is an important factor in power consumption. The change in power consumption can be represented via the mathematical model below [52]:

$$P(t) = \sum_{g} f(g,t) + N(t) \tag{6.2}$$

In the equation 6.2,

- g represents the gate,
- t represents the time,
- $f(g,t)$ represents the function that indicates the gate g consumes power over time t,
- $N(t)$ represents the noise function.

The attacker examines the behavior of the attack with the help equation 6.2, especially the term $f(g,t)$. This term helps to extract 0 and 1 bit patterns with the assistance of the Square and Multiply algorithm in RSA. Due to the operations performed between bits using this algorithm, different values will emerge; thus, different $P(t)$ values will be created for different bits. In other words, the attacker can analyze the power consumption values $P(t)$ corresponding to each time t with the help of the equation 6.2 and can deduce the pattern of the attack through the bits. With this information, they can gain insights into the encryption and decryption processes.

DPA, like SPA, is a type of attack aimed at revealing the sensitive information such as key, but it measures power consumption using statistical analysis

methods [50]. In the attack, the analysis method is carried out with the following steps [50]:

- A target to be analyzed is selected (for instance, a key)
- A 'leak' that occurs during power consumption is measured
- Predictions about the leak are made using a model formed from this leak
- These predictions are compared to reach the target using statistical calculations such as correlation coefficient and average.

RSA and the applications of this algorithm (such as decryption, digital signing) may be vulnerable to DPA; this makes it easier for an attacker to achieve their goal. The Square and Multiply algorithm in RSA is used to execute the attack [52]. For details of this algorithm, you can check the section 4.2.1. In a DPA attack, bits are at the forefront, and the target is usually the decryption key. The implementation of the Square and Multiply algorithm in DPA will be significant during the multiplication operation, where power consumption varies, that is, when the bit is 1. Thus, when it is noticed how frequently the multiplication operation is performed at each step, the bits of the private key will emerge.

As in every attack, there are various techniques to prevent SPA and DPA attacks. One of the best-known techniques is the blinding technique. The important point to note here is that the processes of this technique differs in SPA and DPA.

The 'exponent blinding' technique is used in the prevention of SPA. This technique, proposed by Kocher, is also known to be cost-effective; therefore, it is widely preferred [53].

The target of this technique is to determine the message M using the secret key d. Also, the equation 4.15 will be helpful. It is aimed to perform each deciphering operation using the formula $d' = d + r\phi(n)$ with a different exponent. In this way, it is ensured that the power trace is different. Exponent blinding operations consist of the following steps:

- Choose a random integer r.

- Take $d' = d + r\phi(n)$. Then,

$$\begin{aligned} M = C^d \bmod n \rightarrow C^{d'} &= C^{d+r\phi(n)} \\ &= C^d (C^{\phi(n)})^r \bmod n \end{aligned}$$

  also implies. According to the Euler's theorem, it is known that $C^{\phi(n)} = 1 \bmod n$. This implies

$$C^{\phi(n)} = 1 \bmod n \rightarrow (C^{\phi(n)})^r = 1^r = 1 \bmod n$$

  Thus,

$$\begin{aligned} C^{d'} &= C^d (C^{\phi(n)})^r \bmod n \\ &= C^d.1 \bmod n = C^d \bmod n \end{aligned}$$

- Calculate

$$M = C^{d'} \bmod n = C^{d+r\phi(n)} \bmod n = C^d \bmod n \tag{6.3}$$

  The equation 6.3 shows that the message M can be calculated using d' instead of d.

This technique provides strong defense against SPA; however, it may not provide as much advantage against DPA as it does against SPA. For this, one of the important defense techniques in DPA, namely the masking technique, has been developed. The aim of this technique, which is a good idea to use for asymmetric encryption algorithms, is to prevent the attacker from establishing a correlation between the intermediate values and the key bits [47, 54].

The DPA attack conducted in RSA is prevented by masking technique. The masking technique is based on the ability to re-express the processes through masked values and still produce the desired results [54]. In this technique, just like in exponent blinding, $M = C^d \bmod n$ is targeted [55]. The first step of the masking technique is to apply the masking operation

$$C' = C \cdot r \bmod n \tag{6.4}$$

by taking a random number $r \in \mathbb{Z}_n^*$. Then, by using $M = C^d \bmod n$, this implies

$$M' = (C')^d \bmod n \tag{6.5}$$

Thanks to the eq 6.5, this equality

$$\begin{aligned} M' = (C')^d \bmod n \implies M' &= (C \cdot r)^d \bmod n \\ M' &= C^d \cdot r^d \bmod n \\ M' &= M \cdot r^d \bmod n \end{aligned} \tag{6.6}$$

is satisfied. By the equation 6.6, and since $r \in \mathbb{Z}_n^*$, the equation

$$M = M' \cdot (r^d)^{-1} \bmod n$$

is obtained.

6.2 Active Attacks on RSA

6.2.1 Chosen Ciphertext Attacks (CCA)

The chosen ciphertext attack (CCA) is defined as an attack carried out by an attacker who converts encrypted text into plain text with the help of a secret key [36]. In other words, the attacker's aim is to decrypt and reach the plaintext containing the secret information. When the attacker gains access to the plaintext, they can both encrypt this text and decrypt the encrypted messages they have created [56]. This situation renders the algorithm under attack vulnerable.

The RSA algorithm is vulnerable against to the CCA [36]. The process of the attack on RSA is presented in the following steps [36, 37, 57]:

- From RSA, the equations 4.5 and 4.6 are known.
- Take a random number s where $s \in \mathbb{Z}_n^*$. Then, there is a C' such that

  $$C' = C \cdot s^e \bmod n \tag{6.7}$$

  where e is the public key.
- Since $M = C^d \bmod n$, $M' = (C')^d \bmod n$ is also valid. Then,

  $$\begin{aligned} M' = (C')^d \bmod n \implies M' &= (C \cdot s^e)^d \bmod n \\ &= C^d \cdot s^{ed} \bmod n \\ &= M^{ed} \cdot s^{ed} \bmod n \\ &= M \cdot s \bmod n \end{aligned} \tag{6.8}$$

- The original message is obtained with the help of the formula $M = M' \cdot s^{-1} \bmod n$. Thus,

$$\begin{aligned} M = M' \cdot s^{-1} \bmod n \quad &\Longrightarrow \quad M = M \cdot s \cdot s^{-1} \bmod n \\ &\Longrightarrow \quad M = M \bmod n \end{aligned} \tag{6.9}$$

  is provided.

CCA differs in terms of access timing and attack power. For this reason, this attack has been examined under two different headings: CCA1 and CCA2. CCA1 is also known as 'lunchtime chosen ciphertext attack' and it can be accepted as the basic CCA [58]. In the CCA1, the attacker's access time to decrypt the password on the target system will be limited. The attacker can query until they receive the encrypted text (challenge ciphertext) in the system; after receiving it, they cannot make further queries. Therefore, CCA1 is also defined as 'non-adaptive chosen ciphertext attack' [59]. In CCA2, unlike CCA1, the attacker can query even after receiving the encrypted text (challenge ciphertext) in the system. However, this querying can only be done not on the text itself but in order to decrypt other ciphertexts. This situation allows CCA2 to be defined as 'adaptive chosen ciphertext attack' [59]. According to these explanations, CCA2 is stronger than CCA1 [60].

Although it is quite strong and effective against RSA, CCA is an attack that can be prevented by certain techniques. One of the best-known of these techniques is OAEP (Optimal Asymmetric Encryption Padding). This system, presented by Bellare and Rogaway in 1994 [61], is considered a reliable technique against CCA attacks [62]. OAEP is generally expressed mathematically as follows [61]:

Definition 6.1 Assume k is the security parameter, f a function from k-bits to k-bits and n is the message length as a bit that provides $n = k - k_0$. The **OAEP** schema is represented with choosing a random 'generator', which is $G : \{0,1\}^{k_0} \to \{0,1\}^n$ and a random 'hash function', that is $H : \{0,1\}^n \to \{0,1\}^{k_0}$, such that

$$E^{G,H}(x) = f(x \oplus G(r) \;\; \| \;\; r \oplus H(x \oplus G(r)))$$

where

- x is a message such that $x \in \{0,1\}^n$,
- r is a random k_0-bit number,

- $\|$ represents concatenation,
- $\oplus$ is the XOR operation.

Here, the masking of the x message is achieved with the help of $G(r)$. By using $G(r)$, the operation $H(x \oplus G(r))$ is performed, and through this operation, r is masked. These operations display the encrypted text randomly; thus, leaking the text becomes almost impossible. In summary, the G and H functions are components that prevent attacks by keeping x and r secret.

6.2.2 Man in the Middle Attack

Man in the middle attack is one of the active attacks on RSA. A man-in-the-middle attack, also known as MitM, is a type of attack where there is a third person (the attacker) between two users [63]. In this case, both users are unaware of the attacker's presence; therefore, the attacker can carry out the attack by communicating with both parties [64]. This transfer consists of processes such as altering or eavesdropping on the message [63]. Additionally, it disrupts communication among users and leaves the system vulnerable, as the attacker gains access to confidential information. This attack targets systems containing important information, such as credit cards and bank accounts [64]. The aim of the attack is to gain access to confidential information within the system.

MitM attacks can be carried out using various techniques. These techniques are fundamentally divided into two types: spoofing based and SSL/TLS based.

6.2.2.1 Spoofing based MitM Attack

Spoofing based MitM attacks can be defined as a type of attack where the attacker intervenes in communication to mislead users in the system using fake data [63]. It can be understood from this that the attacker's goal is to secretly enter the system and steal data.
Spoofing-based MitM attacks include these topics:

1. IP Spoofing
   In IP spoofing, the attacker presents modified packet headers in the IP address to users as a trusted source and intercepts data through this source [65]. To prevent this, techniques such as implementing packet filters, authenticating all IP addresses and strengthening these authentications, and using intrusion prevention systems are utilized [65].

2. ARP Spoofing
   ARP is defined as a protocol that maps the IP addresses of devices to their MAC addresses and stores these addresses in its cache [65]. In ARP spoofing, the attacker's goal is to seize users' IP addresses by using the MAC address they send, and the attacker accomplishes this by transmitting a fake ARP message [64, 65]. This spoofing is prevented by techniques such as S-ARP and dynamic ARP testing [65].

3. DNS Spoofing
   In DNS spoofing, the attacker's general goal is to gain access to the DNS server [64]. The attacker misleads users by redirecting them to a different IP address with the help of a fake DNS server; thus achieving their objective [65]. DNSsec is considered a protocol used to prevent DNS spoofing [65].

6.2.2.2 SSL/TLS-based MitM Attack

SSL/TLS-based MitM attacks are defined as a type of attack in which the attacker reads the messages sent by users through two different SSL connections [63]. The attacker aims to steal information by accessing both the users and the server while carrying out this attack.
SSL/TLS-based MitM attacks consist of the following topics:

1. SSL Hijacking
   In SSL hijacking, the attacker gains access to SSL/TLS connections with the help of fake certificates [66]. Using a VPN, keeping the system updated to the latest version, and authentication measures can prevent this attack.

2. SSL Stripping
   The main goal of the SSL stripping is to convert an HTTPS connection into HTTP [64]. In this attack, the attacker breaks the TLS authentication in a system, making all session information of the victim visible to themselves; thus, they manage to capture all the data in the system [64, 66]. The address bar must have HTTPS, and attention should also be paid to the lock symbol for privacy.

6.3 Scenarios of Active and Passive Attacks in the RSA Algorithm

An attacker resorts to various methods to gain access to a system that wants to attack. These methods are briefly referred to as 'scenario'. In RSA, there are various attack scenarios included in both active and passive attacks. The most common of these attack scenarios are listed in the table 6.1 [36].

Table 6.1 Attack scenarios of passive and active attacks on the RSA algorithm

Attack Scenario	**Category**	**Description**
Ciphertext Only	Passive	Encryption algorithm and the ciphertext is known.
Known Plaintext	Passive	In addition to the criteria in the ciphertext only, it also includes *one or more plaintext-ciphertext pairs generated with a related key.*
Chosen Plaintext	Active	In addition to the criteria in the ciphertext only, it also includes *plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key.*
Chosen Text	Active	Chosen Text is also known as *a scenario where chosen ciphertext attacks and chosen plaintext scenarios are blended.* In this scenario, the secret key can be determined through both ciphertext and plaintext.

The main goal of an attacker is to reveal the secret key; therefore, the more information the attacker has knowledge about the algorithm they want to attack, the stronger that attack scenario becomes [67]. The strength of the scenarios varies depending on the level of control the attacker has and their interaction with the system. In this context, the strongest attack scenario is Chosen-Text, while the weakest is Ciphertext-Only. This does not provide information about whether active attacks or passive attacks are stronger; in other words, the applicability of the scenarios against an attack does not indicate whether the attack is directly active or passive.

7
CRYPTANALYSIS of the RSA ALGORITHM

Cryptanalysis is the branch of science that develops mathematical techniques aimed at breaking cryptosystems and is concerned with the analysis of information security services [37]. In some sources such as [38, 68], cryptanalysis is also defined as an 'art', as well. Cryptanalysis aims to investigate the confidentiality of information systems and targets elements such as cryptographic security systems and encrypted messages [68, 69]. Cryptanalysis holds a significant place especially today because it explicitly and concretely determines whether a cryptosystem is secure or not; this makes cryptanalysis an important part of cryptology [38, 69]. A cryptanalyst is someone who deals with this science [37] and continues this work.

In this section, the cryptanalysis of the RSA algorithm, which is one of the most well-known asymmetric encryption algorithm, will be discussed.

Cryptanalysis on RSA relies on attacks and aims to exploit the weaknesses of encryption algorithms. It is examined under two main categories: **mathematical attacks** and **implementation attacks**. In mathematical attacks, operations such as factorization and special modulus conditions are considered. One of the most well-known of these attacks is factorization-based attacks. The goal of this attack is to ensure the factorization of a number n which is obtained from the product of two large prime numbers. Rivest, Shamir and Adleman [23], Knuth [70] and Pollard [71] have explained this attack in detail.

Another mathematical attack is the Coppersmith attack, which is a valuable research topic. The applications of this attack in RSA are quite broad; scenarios such as key certification, finding factors using the private key and finding factors with the help of known bits can be given as examples of these applications [72]. The general idea of this attack, introduced by Coppersmith in 1996 and aimed at finding small integer roots of a modular polynomial, is as follows [73]:

Definition 7.1 Let N be an integer that defines a modulus value.

Let $P(x)$ be a monic polynom that has a degree σ

$$P(x) = x^{\sigma} + p_{\sigma-1}x^{\sigma-1} + \ldots + p_2x^2 + p_1x + p_0$$

Assume that there is an integer solution x_0 that satisfies the equation

$$P(x_0) = 0 \bmod N \quad \text{where} \quad |x_0| < N^{\frac{1}{\sigma}}$$

for the polynomial $P(x)$.

One of the works done by Coppersmith is the development of an algorithm that finds small roots of polynomial equations; this algorithm is the LLL algorithm developed by Lenstra, Lenstra, and Lovasz in 1982 [74, 75]. The LLL algorithm is a lattice-based algorithm. The lattice technique is explained as follows, along with some related concepts:

Definition 7.2 Let $n \in \mathbb{Z}^+$. A subset L that is defined in the n-dimensional real vector space $\mathbb{R}^n$ is called **lattice** and formulated as [76]

$$L = \sum_{i=1}^{n} z_i b_i$$

where

- $z_i = z_1, z_2, \ldots, z_n \in \mathbb{Z}$,
- b_i is a basis for L such that $b_i = [b_1 b_2 \ldots b_n]$ and $b_1, b_2, \ldots, b_n \in \mathbb{R}^n$,
- n is the rank of the lattice L.

Definition 7.3 The **determinant of the lattice** L is a positive real number that is independent from the choice of the basis and is denoted by [76]

$$\det(L) = |\det(b_i)| = |\det(b_1, b_2, \ldots, b_n)|$$

Definition 7.4 Let $b_1, b_2, \ldots, b_n \in \mathbb{R}^n$ be linearly independent. The **Gram-Schmidt Orthogonalization** states that if there are vectors b_i^*, then

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \tag{7.1}$$

where

$$\mu_{i,j} = \frac{b_i, b_j^*}{b_j^*, b_j^*} = \frac{b_i \times b_j^*}{||b_j^*||^2}$$

for $j \in [1, i-1]$ [75, 76]. The symbol (,) denotes the inner product on $\mathbb{R}^n$ [76].

Note that the Gram-Schmidt process is an important concept for the LLL algorithm. The set of vectors b^* mentioned in this definition are orthogonal basis for the vector space expanded by b; however, b^* cannot be considered as the basis of the original lattice L [75].

Definition 7.5 Two conditions are important in the application of the LLL algorithm. These conditions mentioned below are also known as the **Lovasz conditions** [75].

$$||\mu_{i,j}|| \leq \frac{1}{2} \quad for \quad all \quad 1 \leq j < i \leq n \tag{7.2}$$

$$||b_i^* + \mu_{i,i-1} b_{i-1}^*||^2 \geq \delta ||b_{i-1}^*||^2 \quad for \quad all \quad 1 < i \leq n \tag{7.3}$$

In the equation 7.3, any real number that satisfies $\delta = (\frac{1}{4}, 1)$ can be taken; however, generally $\frac{3}{4}$ is used for the constant δ [76].

Definition 7.6 The lattice L is a norm of the shortest vector, also known as **Shortest Vector Problem (SVP)** if

$$\lambda_1(L) = \min\{||b|| : b \in L - \{0\}\}$$

This provides to get an approximately solution for the LLL [77].

Lemma 7.1 For any LLL-reduced basis, the equations below are valid

$$||b_j||^2 \leq 2^{i-1} ||b_i^*||^2 \tag{7.4}$$

where $1 \leq j \leq i \leq n$ [76] and

$$\prod_{i=1}^{n} ||b_i|| \leq 2^{\frac{n(n-1)}{4}} \det L \tag{7.5}$$

[75].

Proof Recall the inequality 7.2. Then, since $j < i$, this implies

$$\begin{aligned} ||\mu_{i,j}|| \leq \frac{1}{2} \quad &\Longrightarrow \quad ||\mu_{i,i-1}|| \leq \frac{1}{2} \\ &\Longrightarrow \quad \mu_{i,i-1}^2 \leq \frac{1}{4} \\ &\Longrightarrow \quad -\mu_{i,i-1}^2 \geq -\frac{1}{4} \end{aligned} \tag{7.6}$$

Also, recall the inequality 7.3, i.e. the Lovasz condition, applying 7.6, it becomes

$$\begin{aligned}
||b_i^* + \mu_{i,i-1}b_{i-1}^*||^2 &\geq \frac{3}{4}||b_{i-1}^*||^2 \\
(b_i^*)^2 + (\mu_{i,i-1})^2(b_{i-1}^*)^2 &\geq \frac{3}{4}(b_{i-1}^*)^2 \\
(b_i^*)^2 &\geq (b_{i-1}^*)^2\left(\frac{3}{4} - (\mu_{i,i-1})^2\right) \\
(b_i^*)^2 &\geq \frac{1}{2}(b_{i-1}^*)^2 \implies (b_{i-1}^*)^2 \leq 2(b_i^*)^2 \qquad (7.7)
\end{aligned}$$

The inequality 7.7 guarantees the inequality

$$||b_j^*||^2 \leq 2^{i-j}||b_i^*||^2 \qquad (7.8)$$

which is essential for this process.
By the definition of the Gram-Schmidt orthogonalization, i.e. the equation 7.1, it can be computed as

$$\begin{aligned}
||b_i||^2 &= \left|\left|b_i^* + \sum_{j=1}^{i-1}\mu_{i,j}b_j^*\right|\right|^2 \\
&= ||b_i^*||^2 + \sum_{j=1}^{i-1}\mu_{i,j}^2||b_j^*||^2 \\
&\leq ||b_i^*||^2 + \sum_{j=1}^{i-1}\frac{1}{4}||b_j^*||^2 \\
&\leq ||b_i^*||^2 + \sum_{j=1}^{i-1}\frac{1}{4}2^{i-j}||b_i^*||^2 \quad \text{(since 7.8)} \\
&\leq ||b_i^*||^2 + \frac{1}{4}||b_i^*||^2\sum_{j=1}^{i-1}2^{i-j} \\
&\leq ||b_i^*||^2 + \frac{1}{4}||b_i^*||^2\sum_{j=1}^{i-1}2^{k} \quad (k = i-j) \\
&\leq ||b_i^*||^2 + \frac{1}{4}||b_i^*||^2(2^i - 2) \quad \text{(by the geometrical series)} \\
&\leq ||b_i^*||^2\left(1 + \frac{1}{4}(2^i - 2)\right) \\
&\leq 2^{i-1}||b_i^*||^2
\end{aligned}$$

By $||b_i||^2 \leq 2^{i-1}||b_i^*||^2$, it also can written $||b_j||^2 \leq 2^{j-1}||b_j^*||^2$. Since $j < i$, the equation can be written as

$$||b_j||^2 \leq 2^{i-1}||b_i^*||^2$$

This proves the equation 7.4.
Now, recall the equation 7.5 for the proof. Then, by the equation 7.4, it becomes

$$\begin{aligned}\prod_{i=1}^{n}||b_i||^2 \leq \prod_{i=1}^{n}2^{i-1}||b_i^*||^2 &= 2^{\sum_{i=1}^{n}(i-1)}\prod_{i=1}^{n}||b_i^*||^2 \\ &\leq 2^{\frac{n(n-1)}{2}}\prod_{i=1}^{n}||b_i^*||^2\end{aligned}$$

Since $\det(L) = \prod_{i=1}^{n}||b_i^*||$, by taking square roots, this implies

$$\prod_{i=1}^{n}||b_i|| \leq 2^{\frac{n(n-1)}{4}}\det L$$

which gives the equation 7.5.

Theorem 7.1 The LLL algorithm must satisfy the conditions

$$\begin{aligned}||b_1|| &\leq 2^{\frac{n-1}{4}}\det(L)^{\frac{1}{n}} \\ ||b_1|| &\leq 2^{\frac{n-1}{2}}\lambda_1(L)\end{aligned}$$

where $\lambda_1(L)$ is the norm of the shortest vector [75, 76].

Proof By the definition of the Gram-Schmidt process, $||b_i|| = ||b_i^*||$. Taking the square for each term of the equation 7.5, it becomes

$$\begin{aligned}\prod_{i=1}^{n}||b_i||^2 \quad &\leq \quad 2^{\frac{n(n-1)}{2}}\prod_{i=1}^{n}||b_i^*||^2 = 2^{\frac{n(n-1)}{2}}\det(L)^2 \\ \Longrightarrow \quad &\prod_{i=1}^{n}||b_i|| \leq 2^{\frac{n(n-1)}{4}}\det(L) \qquad (7.9)\end{aligned}$$

In LLL, this inequality is used to find approximate values.

$$||b_1|| \leq \left(\prod_{i=1}^{n}||b_i||\right)^{\frac{1}{n}} \qquad (7.10)$$

Using the inequalites 7.9 and 7.10, this implies

$$\begin{aligned}||b_1|| &\leq \left(\prod_{i=1}^{n}||b_i||\right)^{\frac{1}{n}} \\ &\leq \left(2^{\frac{n(n-1)}{4}}\det(L)\right)^{\frac{1}{n}} \\ &\leq 2^{\frac{(n-1)}{4}}\det(L)^{\frac{1}{n}}\end{aligned}$$

This proves the first equation of the theorem 7.1. Now, by the definition of the Gram-Schmidt process, this implies

$$b = \sum_{j=1}^{n} c_j b_j = \sum_{j=1}^{n} d_j b_j^*$$

where $c_j, d_j \in \mathbb{R}$. By the orthogonality, this implies

$$||b||^2 = \left|\left| \sum_{j=1}^{n} d_j b_j^* \right|\right|^2 = \sum_{j=1}^{n} d_j^2 ||b_j^*||^2$$

Let $d_k \neq 0$ for the smallest k. Then,

$$\begin{aligned} ||b||^2 &= \left|\left| \sum_{j=1}^{n} d_j b_j^* \right|\right|^2 \\ &= d_1^2||b_1^*||^2 + d_2^2||b_2^*||^2 + \ldots + d_k^2||b_k^*||^2 + \ldots + d_n^2||b_n^*||^2 \\ &\geq ||b_k^*||^2 \end{aligned}$$

By the definition of SVP, it becomes

$$\lambda_1(L)^2 = ||b||^2 \geq ||b_k^*||^2 \geq \min_j ||b_j^*||^2 \tag{7.11}$$

Recall the inequality 7.4. Then,

$$\begin{aligned} ||b_1||^2 &\leq 2^{k-1}||b_k^*||^2 \\ &\leq 2^{k-1}\lambda_1(L)^2 \quad \text{(by 7.11)} \end{aligned}$$

Since $||b_k|| \geq 1$ and $k \leq n$, by taking the square roots, this implies

$$||b_1|| \leq 2^{\frac{k-1}{2}}\lambda_1(L) \leq 2^{\frac{n-1}{2}}\lambda_1(L)$$

This gives the exactly second equation of the theorem.

Cryptanalysis in RSA is also performed through implementation attacks. The aim of these types of attacks is to damage factors such as hardware and timing in the system. The most common implementation attack is the 'Side Channel Attack.' In some studies such as [68, 78], it is also referred to as 'Side Channel Analysis Attack'; however, both terms refer to the same situation. This type of attack is explained in detail in the previous section, see 6.1.1.

8 CONCLUSION

The study addresses the RSA algorithm. This algorithm, developed nearly 50 years ago, is still used effectively today. First, a brief introduction to the topic has been made in the 'Introduction' section. In the following section, concepts such as functions, modular arithmetics and groups have been discussed to gain a better understanding of RSA. Since the fundamental logic of RSA is based on groups, this section has been included. The third section covers encryption algorithms in general terms. Here, encryption algorithms are discussed under two headings: symmetric encryption algorithms and asymmetric encryption algorithms, emphasizing that RSA is an asymmetric encryption algorithm. In the next section, an introduction to the RSA algorithm has been provided. This section contains detailed information about RSA, including its definition and proof. With the proof of RSA, the logic of the algorithm has been made clearer. Since its development, RSA has been the subject of various studies. One of the research topics on RSA, performance speed, is presented in the fifth chapter. In this chapter, some optimizations are suggested to improve RSA's performance and the comparison of RSA with these optimizations is discussed. These comparisons have been made with the help of tables and graphs. It has been observed in this study that the best method is the Montgomery algorithm. Attacks on RSA, which is also a significant research topic of RSA, is shown in the sixth chapter. The attacks are addressed under two main headings: active attacks and passive attacks. The most well-known attacks such as side channel and chosen ciphertext are discussed in this study. Additionally, in this chapter, potential attack scenarios that can be used against RSA are also addressed through tables. In previous sections, the analysis of RSA was highlighted through speed performance. In the last section, a brief explanation was given on how RSA's cryptanalysis can be performed. Since the cryptanalysis in RSA is based on the attacks, this topic is categorized into two main headings: mathematical attacks and implementation attacks. Because implementation attacks are also known as side channel attacks and the side channel attacks

were previously discussed in detail, this section focuses on a more detailed examination of mathematical attacks. As a result, this study aims to provide a more comprehensive explanation of RSA, including some fundamental concepts like groups, as well as topics such as analysis, cryptanalysis, and attacks in RSA.

REFERENCES

[1] Ü. Günden, "Şifreleme algoritmalarının performans analizi," M.S. thesis, Sakarya Universitesi (Turkey), 2010.

[2] G. Ehrlich, *Fundamental concepts of abstract algebra*. Courier Corporation, 2011.

[3] W. K. Nicholson, *Introduction to abstract algebra*. John Wiley & Sons, 2012.

[4] G. B. T. Jr., M. D. Weir, J. Hass, *Thomas' Calculus: Early Transcendentals*, 13th. Pearson, 2014.

[5] K. H. Rosen, *Elementary number theory*. Pearson Education London, 2011.

[6] Z. J. Luo, R. Liu, A. Mehta, M. L. Ali, "Demystifying the rsa algorithm: An intuitive introduction for novices in cybersecurity," *arXiv preprint arXiv:2308.02785*, 2023.

[7] S. Edition, K. H. Rosen, *Discrete mathematics and its applications*, 2019.

[8] F. Çetin, "Kriptografide kullanılan asal sayı test yöntemleri üzerine bir çalışma," M.S. thesis, Necmettin Erbakan University (Turkey), 2021.

[9] İ. Okumuş, "Rsa kriptosisteminin hızını etkileyen faktörler," 2012.

[10] J. B. Fraleigh, *A first course in abstract algebra*. Pearson Education India, 2003.

[11] J. A. Gallian, *Contemporary Abstract Algebra*, 9th. Cengage Learning, 2017.

[12] J. D. H. Smith, *Introduction to Abstract Algebra*. Chapman and Hall/CRC, 2009.

[13] D. S. Malik, J. N. Mordeson, M. Sen, *Fundamentals of abstract algebra*. McGraw-Hill, 1997.

[14] İ. Ciğer, "Data şifreleme algoritmaları ve performans analizi," M.S. thesis, Fen Bilimleri Enstitüsü, 2012.

[15] D. Z. Demir, "An introduction to cryptography," Bachelor's thesis, Yıldız Technical University, İstanbul, Türkiye, 2024.

[16] T. Yerlikaya, "Yeni şifreleme algoritmalarının analizi," 2006.

[17] F. Şahin, "Modern blok şifreleme algoritmaları," *İstanbul Aydın Üniversitesi Dergisi*, vol. 7, no. 26, pp. 23–40, 2015.

[18] O. Fındık, "Şifrelemede kaotik sistemin kullanılması," 2004.

[19] A. Beşkirli, D. Özdemir, M. Beşkirli, "Şifreleme yöntemleri ve rsa algoritması üzerine bir inceleme," *Avrupa Bilim ve Teknoloji Dergisi*, pp. 284–291, 2019.

[20] T. S. Obaid, "Study a public key in rsa algorithm," *European Journal of Engineering and Technology Research*, vol. 5, no. 4, pp. 395–398, 2020.

[21] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, N. S. A. Shakir, M. M. Deris, "A survey on the cryptographic encryption algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, 2017.

[22] C. Aslanyürek, "Şifreleme algoritmalarının hızını etkileyen faktörler," M.S. thesis, Fen Bilimleri Enstitüsü, 2018.

[23] R. L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[24] Z. J. Luo, R. Liu, A. Mehta, "Understanding the rsa algorithm," *arXiv preprint arXiv:2308.02785*, 2023.

[25] D. Burton, *Ebook: Elementary number theory*. McGraw Hill, 2010.

[26] R. S. Abdeldaym, H. M. Abd Elkader, R. Hussein, "Modified rsa algorithm using two public key and chinese remainder theorem," *IJ of Electronics and Information Engineering*, vol. 10, no. 1, pp. 51–64, 2019.

[27] V. Ďuriš, V. Bojdová, T. Šumnỳ, "Solving selected problems on the chinese remainder theorem," *Annales Mathematicae et Informaticae*, 2022, pp. 196–207.

[28] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.

[29] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor," *Conference on the Theory and Application of Cryptographic Techniques*, Springer, 1986, pp. 311–323.

[30] K. Somsuk, "The special algorithm based on rsa cryptography for signing and verifying digital signature," *Heliyon*, vol. 11, no. 4, 2025.

[31] M. Mohammadi, A. Zolghadrasli, M. Pourmina, "A fast and secure rsa public key cryptosystem," *Journal of Mathematical Extension*, vol. 14, no. 4, pp. 1–20, 2020.

[32] Y. Suryawanshi, S. Khade, D. B. Bhoyar, A. R. B. Patil, P. Zode, "Chinese remainder theorem based performance analysis of rsa cryptosystem," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 8, pp. 523–526, 2023.

[33] S. Singh, G. Agarwal, "Use of chinese remainder theorem to generate random numbers for cryptography," *International Journal of Applied Engineering Research*, vol. 1, no. 1, pp. 168–174, 2010.

[34] R. Géraud, D. Maimuţ, D. Naccache, "Double-speed barrett moduli," in *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, Springer, 2016, pp. 148–158.

[35] S. J. Aboud, E. Prakash, "Modular reduction methods," *International Journal of Scientific & Engineering Research*, vol. 5, no. 5, 2014.

[36] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Pearson Education Limited, 2017.

[37] P. C. Van Oorschot, A. J. Menezes, S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[38] C. Paar, J. Pelzl, *Understanding cryptography*. Springer, 2010, vol. 1.

[39] A. Lo'ai, H. H. Tawalbeh, F. Turki, et al., "Review of side channel attacks and countermeasures on ecc, rsa, and aes cryptosystems," *J Internet Technol Secured Trans*, 2016.

[40] Y. Lyu, P. Mishra, "A survey of side-channel attacks on caches and countermeasures," *Journal of Hardware and Systems Security*, vol. 2, pp. 33–50, 2018.

[41] A. S. Nath, A. Jeyasekar, "Survey on multimedia security and visual cryptography," *International Journal of Advance Computational Engineering and Networking (IJACEN)*, vol. 9, pp. 1–4, 2021.

[42] N. Somani, D. Mangal, "An improved rsa cryptographic system," *International Journal of Computer Applications*, vol. 105, 2014.

[43] P. C. Kocher, "Timing attacks on implementations of die-hellman, rsa, dss, and other systems," *Advances in Cryptology| Crypto*, vol. 96, 1996, p. 104 113.

[44] D. Boneh et al., "Twenty years of attacks on the rsa cryptosystem," *Notices of the AMS*, vol. 46, no. 2, pp. 203–213, 1999.

[45] D. Savić, P. Milić, B. MAŻINJANIN, P. Spalević, "Cryptanalytic attacks on rsa algorithm and its variants.," *Przeglad Elektrotechniczny*, vol. 98, no. 2, 2022.

[46] V. Shukla, A. Choubey, "A comparative analysis of the attacks on public key rsa cryptosystem," *INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY & CREATIVE ENGINEERING (ISSN: 2045-8711)*, vol. 2, no. 2, pp. 6–11, 2012.

[47] S. Mangard, E. Oswald, T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. New York, NY, USA: Springer Science+Business Media, LLC, 2007.

[48] H. Gamaarachchi, H. Ganegoda, "Power analysis based side channel attack," *arXiv preprint arXiv:1801.00932*, 2018.

[49] J. Feher, "Introduction to digital logic with laboratory exercises," *Creative Commons Attribution*, 2009.

[50] L. Lerman, G. Bontempi, O. Markowitch, "Power analysis attack: An approach based on machine learning," *International Journal of Applied Cryptography*, vol. 3, no. 2, pp. 97–115, 2014.

[51] M. A. N. Rony Chowdhury, "Improved study of side-channel attacks using recurrent neural networks," 2019.

[52] M. Aigner, E. Oswald, *Power analysis tutorial*, 2000.

[53] K. Gandolfi-Villegas, N. Hamzi, "Dynamic runtime methods to enhance private key blinding," *Cryptology ePrint Archive*, 2013.

[54] H. J. Mahanta, A. K. Khan, "Partitioned-exponent blinding: A countermeasure against power analysis attacks," *Indian Journal of Science and Technology*, vol. 10, p. 14, 2017.

[55] A. K. Khan, H. J. Mahanta, A. Chakraborty, "Investigating the blinding approach to resist power analysis attacks on modular exponentiation," *International Journal of Computational Intelligence & IoT*, vol. 1, no. 1, 2018.

[56] J. Katz, Y. Lindell, *Introduction to Modern Cryptography*, 2nd. CRC Press, Taylor & Francis Group, 2015.

[57] D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1," *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*, Springer, 1998, pp. 1–12.

[58] R. Cramer, V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*, Springer, 1998, pp. 13–25.

[59] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, "Relations among notions of security for public-key encryption schemes," *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*, Springer, 1998, pp. 26–45.

[60] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, "Rsa-oaep is secure under the rsa assumption," *Journal of cryptology*, vol. 17, pp. 81–104, 2004.

[61] M. Bellare, P. Rogaway, "Optimal asymmetric encryption – how to encrypt with rsa," *Advances in Cryptology — EUROCRYPT '94*, ser. Lecture Notes in Computer Science, vol. 950, Springer-Verlag, 1994, pp. 92–111.

[62] H. C. A. van Tilborg, S. Jajodia, Eds., *Encyclopedia of Cryptography and Security*, Second Edition. Springer, 2011.

[63] D. Javeed, U. MohammedBadamasi, C. O. Ndubuisi, F. Soomro, M. Asif, "Man in the middle attacks: Analysis, motivation and prevention," *International Journal of Computer Networks and Communications Security*, vol. 8, no. 7, pp. 52–58, 2020.

[64] A. Mallik, "Man-in-the-middle-attack: Understanding in simple words," *Cyberspace: Jurnal Pendidikan Teknologi Informasi*, vol. 2, no. 2, pp. 109–134, Oct. 2018.

[65] M. Tareq, "Man in the middle attack in wireless network," *Proceedings of the Man In The Middle Attack Conference*, 2021.

[66] Z. Cekerevac, P. Cekerevac, L. Prigoda, F. Al-Naima, "Security risks from the modern man-in-the-middle attacks.," *MEST Journal*, vol. 13, no. 1, 2025.

[67] D. R. Stinson, M. B. Paterson, *Cryptography: Theory and Practice*, 4th ed. CRC Press, Taylor & Francis Group, 2019.

[68] S. Raviteja, V. Vani, A. Swathi, P. Sudheer Babu, "Cryptanalysis on RSA," *Proceedings of the National Conference on Internet of Things (IoT)–2K18*, Secunderabad, India: St. Martin's Engineering College, Secunderabad, 2018.

[69] F. A. Harahap, Y. Yusfrizal, M. Sovina, I. Lazuly, "Cryptanalysis of rsa using algebraic and lattice methods," *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, vol. 3, no. 3, pp. 746–750, 2024.

[70] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd. Addison-Wesley, 1998.

[71] J. M. Pollard, "Theorems on factorization and primality testing," *Mathematical proceedings of the Cambridge philosophical society*, Cambridge University Press, vol. 76, 1974, pp. 521–528.

[72] A. May, "Lattice-based integer factorisation: An introduction to coppersmith's method," *Computational Cryptography: Algorithmic Aspects of Cryptology*, pp. 78–105, 2021.

[73] D. Coppersmith, "Small solutions to polynomial equations, and low exponent rsa vulnerabilities," *Journal of cryptology*, vol. 10, no. 4, pp. 233–260, 1997.

[74] A. May, "Using lll-reduction for solving rsa and factorization problems," in *The LLL algorithm: survey and applications*, Springer, 2009, pp. 315–348.

[75] J. Hoffstein, J. Pipher, J. H. Silverman, *An Introduction to Mathematical Cryptography* (Undergraduate Texts in Mathematics), 1st. Springer, 2008.

[76] A. Lenstra, H. Lenstra, L. Lovász, "Factoring polynomials with rational coefficients," *Math. ann*, vol. 261, no. 4, pp. 515–534, 1982.

[77] P. Q. Nguyen, B. Vallée, *The LLL algorithm*. Springer, 2010.

[78] K. Berlin, S. Dhenakaran, "An overview of cryptanalysis of rsa public key system," *International Journal of Engineering and Technology*, vol. 9, no. 5, pp. 3575–3579, 2017.

PUBLICATIONS FROM THE THESIS

Conference Papers

1. Z. M. Saygıner, M. Özavşar, "Comparison of Techniques for Increasing RSA Speed", 12th International Congress on Fundamental and Applied Sciences 2025 (ICFAS2025), Istanbul, Turkiye, ICFAS2025-ID: 1218, 2-4 September 2025.