## SAVEETHA SCHOOL OF ENGINEERING

##### SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

**CHENNAI-602105**

**Securing Data Transmission: Implementation and Analysis of RSA Algorithm for Cryptographic Applications**

## A CAPSTONE PROJECT REPORT

## Code: CSA5193

## Subject: Cryptography and Network Security for Secure System Design

### Submitted in the partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING**

## IN

**COMPUTER SCIENCE AND ENGINEERING**

## Submitted by

## P. Harsha Vardhan (192210047)

## M. Manoj (192211728)

## Under the Supervision of

Mrs. J. Alphonsa

**DECLARATION**

## I P. Harsha Vardhan, M. Manoj student of Bachelor of Engineering in the Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha School of Engineering, Chennai, hereby declare that the work presented in this Capstone Project Work entitled Preparation And Conductivity Analysis of High Performance Polymers In Electronics is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

## P. Harsha Vardhan(192210047)

## M. Manoj(192211728)

## 

## Date:

**Place:**

**CERTIFICATE**

This is to certify that the project entitled “Securing Data Transmission: Implementation and Analysis of RSA Algorithm for Cryptographic Applications” submitted by M. Manoj(192211728) P. Harsha Vardhan (192210047) has been carried out under my supervision. The project has been submitted per the requirements in the current B. Tech Information Technology semester.

Teacher-In-Charge

Mrs. J. Alphonsa

## 

## Table of Contents

|  |  |
| --- | --- |
| **S.NO** | **TOPICS** |
| 1 | Abstract |
| 2 | Introduction |
| 3 | Key Generation |
| 4 | Analysis of RSA Algorithm |
| 5 | Practical Analysis |
| 6 | Conclusion |
| 7 | References |

## 

## 

## Abstract:

## In the digital era, securing data transmission is paramount. The RSA algorithm, one of the most widely used public-key cryptographic systems, plays a pivotal role in ensuring confidentiality, data integrity, and authentication. This report explores the implementation of the RSA algorithm, its operational mechanics, and its analysis concerning security guarantees, performance, and practical applications in cryptographic contexts.

## 1. Introduction

## As the internet facilitates global communication, the need for secure data transmission increases. Cryptography serves as a critical tool to protect sensitive information. Among various cryptographic methods, the RSA algorithm stands out due to its asymmetric key structure which allows secure data exchange and digital signatures.

## Background of Cryptography

## Cryptography has evolved over centuries, transitioning from classical techniques to modern computational algorithms. Public-key cryptography, introduced in the 1970s, revolutionized the field by enabling secure exchanges without the need for a pre-shared secret.

## Overview of RSA Algorithm

## RSA (Rivest-Shamir-Adleman) was developed in 1977 and is based on the mathematical properties of large prime numbers. It enables secure data transmission and digital signatures through a pair of keys: a public key for encryption and a private key for decryption.

## 2. Implementation of RSA Algorithm

## This section outlines the steps involved in the RSA algorithm's implementation, including key generation, encryption, and decryption processes.

## Key Generation

## 1. Select Two Distinct Prime Numbers: Choose large prime numbers ( p ) and ( q ).

## 2. Compute ( n ): ( n = p \* q ). The modulus ( n ) is part of the public key and is used in both encryption and decryption.

## 3. Calculate Euler's Totient: ( phi(n) = (p-1) \* (q-1) ).

## 4. Choose Public Exponent: Select an integer ( e ) such that ( 1 < e < phi(n) ) and ( text{gcd}(e, phi(n)) = 1 ).

## 5. Determine Private Exponent: Compute ( d ) as ( d \* e^{-1} mod phi(n) ).

## 2.2 Encryption:

## The encryption process converts plaintext ( M ) into ciphertext ( C ) using the public key: C \*M^e mod n .

## 2.3 Decryption:

## The decryption process retrieves plaintext from ciphertext using the private key: M \* C^d mod n .

## 2.4 Python Code:

## import random

## from sympy import isprime, mod\_inverse

## def generate\_large\_prime(bits):

## while True:

## num = random.getrandbits(bits)

## if isprime(num):

## return num

## def generate\_keys(bits):

## p = generate\_large\_prime(bits)

## q = generate\_large\_prime(bits)

## n = p \* q

## phi\_n = (p - 1) \* (q - 1)

## e = 65537 # Common choice for e

## d = mod\_inverse(e, phi\_n)

## return (e, n), (d, n) # public\_key, private\_key

## def encrypt(public\_key, plaintext):

## e, n = public\_key

## plaintext\_int = int.from\_bytes(plaintext.encode('utf-8'), 'big')

## ciphertext = pow(plaintext\_int, e, n)

## return ciphertext

## def decrypt(private\_key, ciphertext):

## d, n = private\_key

## plaintext\_int = pow(ciphertext, d, n)

## plaintext = plaintext\_int.to\_bytes((plaintext\_int.bit\_length() + 7) // 8, 'big').decode('utf-8')

## return plaintext

## # Example usage

## public\_key, private\_key = generate\_keys(512)

## ciphertext = encrypt(public\_key, "Secure Message")

## plaintext = decrypt(private\_key, ciphertext)

## print(f"Ciphertext: {ciphertext}")

## print(f"Plaintext: {plaintext}")

## 3. Analysis of RSA Algorithm:

## 3.1 Security Considerations:

## RSA's security relies on the difficulty of factoring the product of two large prime numbers. Key length significantly affects the algorithm’s security; currently, a minimum of 2048 bits is recommended to ensure adequate protection against brute-force attacks.

## 3.2 Performance Metrics:

## Speed: RSA encryption is slower than symmetric encryption algorithms (e.g., AES) due to computational complexity.

## Key Size: The larger the key size, the greater the security and the slower the performance.

## 

## 3.3 Strengths and Weaknesses:

## Strengths:

## - Public key distribution eliminates the need for a secure channel for key sharing.

## - Supports digital signatures, enhancing authenticity.

## Weaknesses:

## - Vulnerable to attacks if improper key sizes or poor random number generation methods are used.

## - Slower than symmetric key algorithms for bulk data encryption.

## 4. Practical Applications

## RSA is widely used in various applications including:

## - Secure web communications (HTTPS)

## - Digital certificates (SSL/TLS certificates)

## - Secure email protocols (PGP)

## - Code signing and document verification

## 5. Conclusion:

## The RSA algorithm is a cornerstone of modern cryptographic systems, enabling secure data transmission and authentication. While it has some performance limitations and is sensitive to key management, its strengths in safety and broad applicability make it one of the most trusted methods for cryptography.

## 6. References:

## Rivest, R. L., Shamir, A., & Adelman, L. (1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". \*Communications of the ACM\*.

## Stallings, W. (2019). "Cryptography and Network Security: Principles and Practices". Pearson.

## Kahn, D. (1996). "The Codebreakers: The Story of Secret Writing". Scribner.

## 4. Chen, L., & Wang, J. (2021). "A Survey of RSA Cryptosystem: Current Status and Future directions". \*Journal of Cryptographic Engineering\*.