RSA and Ceaser Cipher

Yash Verma(B23MT1046),Kartik Jain (B23ME1026),Aditya Jain(B23BB1003),Daksh Sohni

21 April 2024

1. Introduction

We have created an encryption algorithm using Ceaser Cipher and RSA in C. Also we have used Socket Programming demonstrate the working of the algorithm.

We have incorporated one way trapdoor functions in the algorithm. These functions are called one-way because they are easy to compute in one direction but (apparently) very difficult to compute in the other direction. They are called trapdoor functions since the inverse functions are in fact easy to compute once certain private trap-door information is known.

Ceaser Cipher

RSA

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.It is based on Trapdoor one way functions.

The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long.

GMP Library

The GNU MP (GMP) library is a powerful tool for performing arbitrary-precision arithmetic, which is essential in implementing RSA encryption and decryption algorithms. RSA, a widely used public-key cryptosystem, relies on large integer arithmetic for its key generation, encryption, and decryption processes. Here's how the GMP library contributes to RSA encryption:

1. Arbitrary Precision Arithmetic: RSA encryption involves operations on extremely large integers, often several hundred digits long. Standard data types provided by programming languages like int or long may not be sufficient to handle such large numbers accurately. The GMP library provides data types like mpz$_t$$, whichcanrepresentintegersofarbitrarylengthwithhighprecision, ensuringaccuratecalculationsinRSA.$

2. Efficiency: GMP is highly optimized for performance, offering efficient implementations of arithmetic operations on large integers. This efficiency is

crucial in RSA, where key generation, encryption, and decryption operations involve numerous arithmetic computations on large numbers. The GMP library's optimized algorithms contribute to the overall performance of RSA encryption, making it feasible to work with large keys and messages efficiently.

3. Security: RSA encryption relies on the computational complexity of certain mathematical operations, such as prime factorization, for its security. The GMP library provides reliable implementations of these operations, ensuring the cryptographic strength of RSA encryption. By using GMP, developers can trust that the arithmetic operations crucial to RSA security are performed correctly and securely.

Contributions And Problems Faced:

1.Yash Verma(B23MT1046):

- Generated random prime numbers of 1024 bits by using gmp module and miller rabin primality test.
- Debugged the prime number generation program and contributed in debugging the RSA algorithm
- Orchastrated the workflow and managed the group for efficient working

Problems faced :

- Finding appropriate library that supports big number operations studying about generating prime numbers which is deteministically a NP hard problem so opted for probablistic test and getting probability of generating prime numbers to 1- (1/(2**n))
- Studied various methods to implement a robust and close to modern day AES-256 encryption.
-
-

2. Kartik Jain(B23ME1026): Made the RSA Algorithm:

- Studied implementation of RSA and Caeser Cipher through multipe research papers.
- Using prime numbers,generated public and private keys.

Problems Faced:

- Counldn't implement random public key generation.
- Also faced problems during the generation of private key.

3. Aditya Jain(B23BB1003):

- Studied ceaser cipher and RSA

- Helped in debugging
- Implemented ceaser cipher
- performed padding for RSA
- traversed through strings for plain text
- Helped in encryption decryptyion of RSA

Problems faced :

- couldnt use a big key for ceaser cipher
- Errors in creating concentric ceaser cipher encryptions.

4. Daksh Sohni(B23MT1014): Contribution

- information about RSA and caeser cipher.
- Implementation of sockets (server and client) in RSA encryption , encryption of data in the buffer space of server

Problems faced

- Problem faced in connecting the client and the server to the RSA encryption function and due to time constraints was not able to combine all that