

Cipher Conversion System - Project Report

Introduction

The **Cipher Conversion System** is a multithreaded C-based project that allows users to encrypt and decrypt messages using two methods: **Caesar Cipher** and **RSA Encryption**. The system efficiently manages requests using queues and executes them in parallel using pthreads.

Objectives

- Encrypt and decrypt user messages using Caesar and RSA algorithms.
 - Queue and execute multiple requests concurrently.
 - Display appropriate encryption/decryption results.
 - Emphasize thread safety using pthread_mutex.
-

Technologies Used

- Programming Language: C
 - Libraries: pthread, string, stdlib, math
 - Data Structures: Queue
 - Algorithms: Caesar Cipher, RSA
-

System Design Overview

1. Main Modules:

- main.c: Handles user input and manages encryption/decryption threads.
- queue.c: Implements queue operations for request management.
- cipher.c: Implements Caesar and RSA cipher functions.

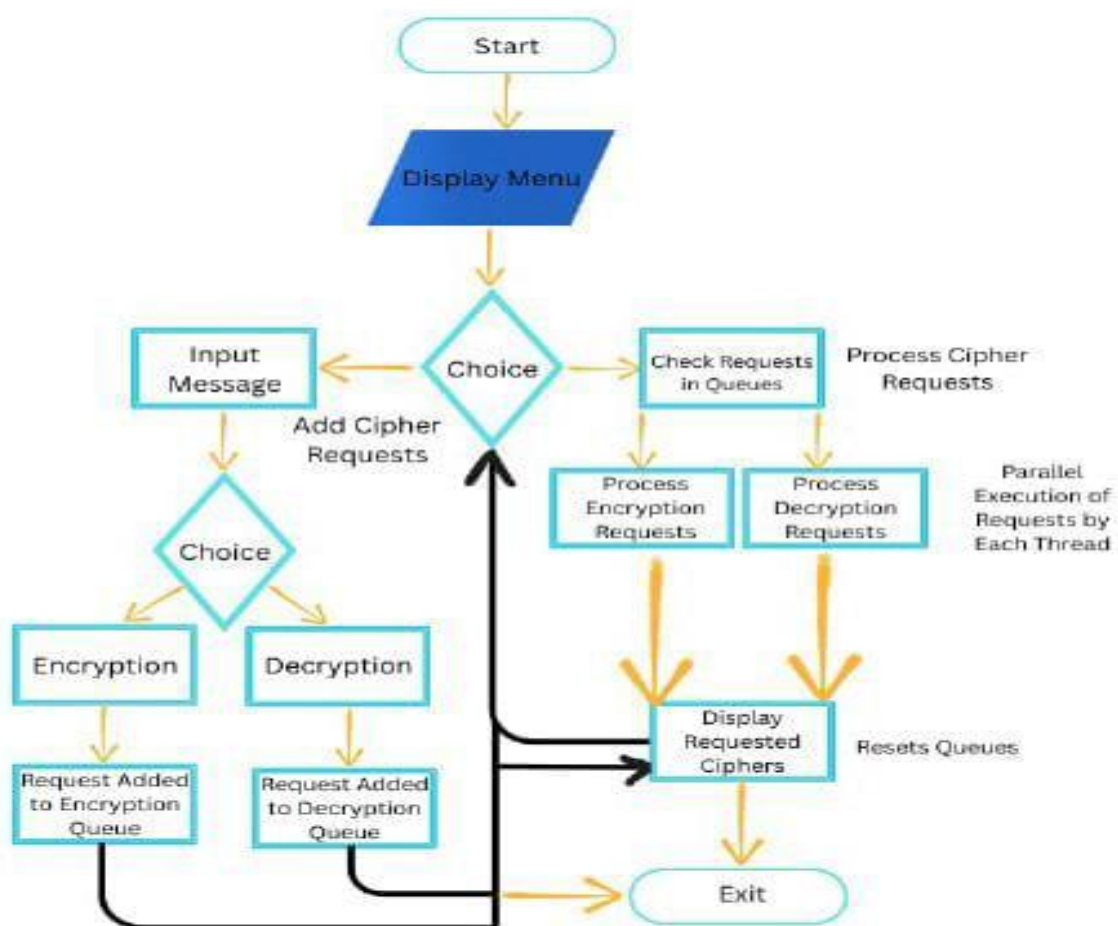
2. Headers:

- queue.h: declares functions to handle requests, implemented in queue.c.
- cipher.h: declares functionality for encryption and decryptions, implemented in cipher.c.

3. Data Structure

- A QUEUE Data structure is used to carry message content, size, and encryption/decryption details.

System Flowchart



Algorithm 1: Caesar Cipher

Encryption

1. Iterate over each character.
2. If alphabetic, shift it by shifts % 26.
3. Wrap around using modular arithmetic.
4. Return the modified string.

Decryption

1. Reverse the shift using $(\text{char} - \text{shift} + 26) \% 26$.
2. Maintain non-alphabet characters.

Caesar Cipher

Plaintext: HELLO

Shift: 3

Encrypted: KHOOR

Algorithm 2: RSA Encryption

1. Generate p and q, calculate $n = p * q$.
2. Compute $\phi(n) = (p-1)(q-1)$.
3. Use public key e, calculate private key d such that $e * d \equiv 1 \pmod{\phi(n)}$.
4. Encrypt message character as:
 $\text{cipher} = (\text{msg}^e) \% n$
5. Decrypt cipher as:
 $\text{msg} = (\text{cipher}^d) \% n$

RSA Flow

Plaintext -> Encrypt (e,n) -> Ciphertext

Ciphertext -> Decrypt (d,n) -> Plaintext

Dry Run Example

Input: Encrypt "HELLO" using Caesar with shift = 3

1. Queue receives: {message: "HELLO", cipherType: Caesar, shifts: 3}
2. Message goes to encryptionQueue.
3. User selects Execute → thread is created.
4. Thread calls messageToCeaser() → returns "KHOOR"
5. Result is printed: "Ceaser Encrypted Message: KHOOR of (HELLO)"

OUTPUT DRY RUN TEST CASES:

```
PS C:\Users\alimo\OneDrive\Desktop\OS Project\source> gcc -g cipher.c queue.c main.c -o temp -lpthread
PS C:\Users\alimo\OneDrive\Desktop\OS Project\source> ./temp.exe

---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
1
Enter Text: hello world
-----
Enter (1) for Ceaser Cipher Encryption
Enter (2) for RSA Encryption
-----
1
Enter Number of Shifts: 5
Request Added Successfully

---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
1
Enter Text: Operating System
-----
Enter (1) for Ceaser Cipher Encryption
Enter (2) for RSA Encryption
-----
2
Request Added Successfully
```

```

---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
3
---Encryption Requests---
Caesar Encrypted Message: mjqqt btwqi of (hello world)
Private Key (d,n)=>(2753,3233)
RSA Encrypted Message: 1307 612 1313 2412 1632 884 3179 2235 2923 1992 2680 487 1230 884 1313 2271 of (Operating System)
-----
No Requests for Decryption to Execute

```

```

---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
2
Enter Text: mjqqt btwqi
-----
Enter (1) for Caesar Cipher Decryption
Enter (2) for RSA Decryption
-----
1
Enter Number of Shifts: 5
Request Added Successfully

---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
2
Enter Text: 1307 612 1313 2412 1632 884 3179 2235 2923 1992 2680 487 1230 884 1313 2271
-----
Enter (1) for Caesar Cipher Decryption
Enter (2) for RSA Decryption
-----
2
Enter Private Key (d n): 2753 3233
Request Added Successfully

```

```






---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
2
Enter Text: 5435 6433 2425 7854 4563
-----
Enter (1) for Caesar Cipher Decryption
Enter (2) for RSA Decryption
-----
2
Enter Private Key (d n): 2732 3233
Request Added Successfully

```

```
---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
3
No Requests for Encryption to Execute
---Decryption Requests---
Caesar Decrypted Message: hello world of (mjqqt btwqi)
RSA Decrypted Message: Operating System of (1307 612 1313 2412 1632 884 3179 2235 2923 1992 2680 487 1230 884 1313 2271)
-----

---Welcome To Cipher Conversion System---
Enter (1) to Encrypt Message
Enter (2) to Decrypt Cipher
Enter (3) to Execute Added Requests
Enter (0) to Exit
-----
0
```

Strongest Points of Project

-  **Threaded Execution:** Uses pthreads for real parallelism.
-  **RSA Implementation:** Incorporates fundamental public-key encryption logic.
-  **Clear Design:** Proper modularization between queue and cipher logic.
-  **User Interface:** Simple, menu-driven UI for interactive use.
-  **Scalable Queues:** Manages multiple encryption and decryption jobs.

Conclusion

This project demonstrates strong concepts in concurrency, cryptography, and system programming. The mix of classical (Caesar) and modern (RSA) encryption makes it ideal for learning and expanding in future versions.

Appendix

- **Group Leader:** Muhammad Ali Hadi(23K-0663)
- **Group Members:** Ali Mobin(23K-0622), Muhammad Mustafa(23K-5545), Abdul Qadir(23K-0674)
- **Submitted To:** Prof. Muhammad Kashif
- **Semester:** Spring 2025
- **Institution:** FAST NUCES Karachi