**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**

**A diagram of a display menu

AI-generated content may be incorrect.**

**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 (char - 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 φ(n) = (p-1)(q-1).
3. Use public key e, calculate private key d such that e\*d ≡ 1 (mod φ(n)).
4. Encrypt message character as:  
   cipher = (msg^e) % n
5. Decrypt cipher as:  
   msg = (cipher^d) % n

**RSA Flow**

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

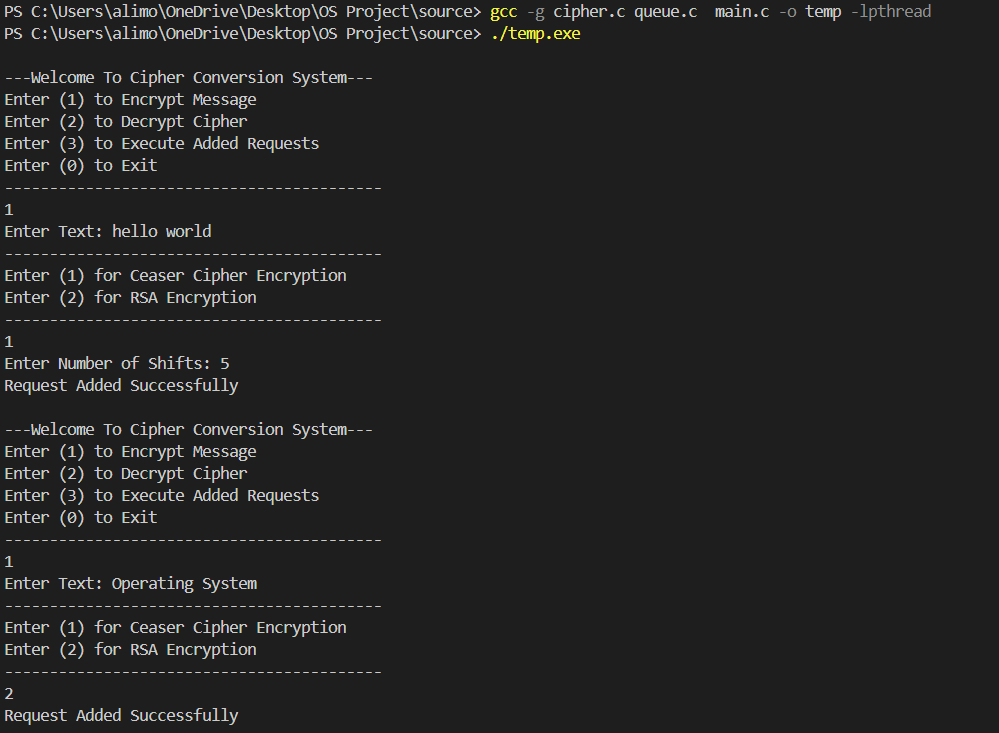
Ciphertext -> Decrypt (d,n) -> Plaintext

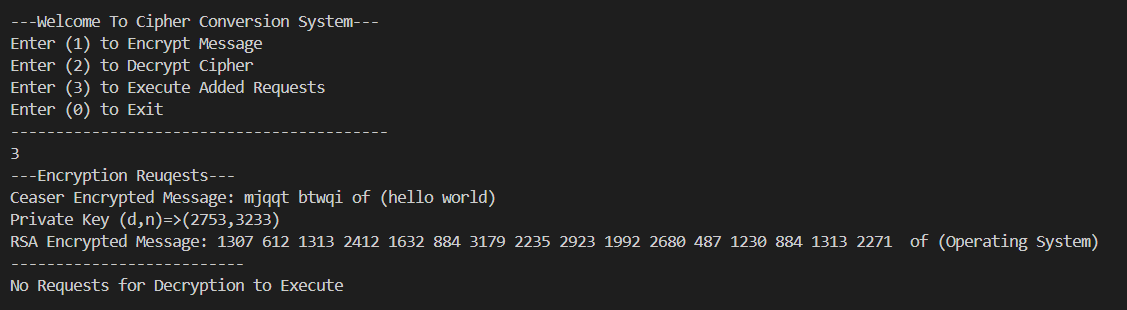
**Dry Run Example**

**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:**





A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

**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