Programming in C

Password encryption and decryption

Submitted By

Name: Piyush Singh

SAP ID: 590026614

Submitted to Faculty

Mohsin F. Dar

Date of Submission: 05/12/2025

Btech-CSE First Semester

Project Report

Project Title: Secure Banking System with Password Encryption using Shift Cipher

Language: C Programming

Focus: Data Security and File Handling

1. Abstract

In modern software systems, user security is paramount. Storing passwords in plain text creates significant vulnerabilities. This project implements a Bank Management System that demonstrates fundamental cryptography concepts. It utilizes a Shift Cipher (Caesar Cipher) algorithm to encrypt user passwords before storing them in a persistent database (text file) and decrypts them dynamically during user authentication.

2. Problem Statement

When a standard C program writes strings to a file, they are legible to anyone who opens that

file.

- **Vulnerability: If an attacker opens bank_database.txt, they can read every user's password.**
- **Objective: To implement a mechanism where the stored data is unintelligible to the human eye but retrievable by the system.**

3. Methodology & Algorithm

The Encryption Logic (Shift Cipher)

The project uses a symmetric substitution cipher. The key used in this implementation is an integer shift of +3.

- **Encryption Formula: $C = P + 3$**
  - **Where P is the plaintext character (user input).**
  - **Where C is the ciphertext character (stored data).**
- **Decryption Formula: $P = C - 3$**

Example:

If the user's password is "BAT":

1. **B (+3) $\->$ E**
2. **A (+3) $\->$ D**
3. **T (+3) $\->$ W**

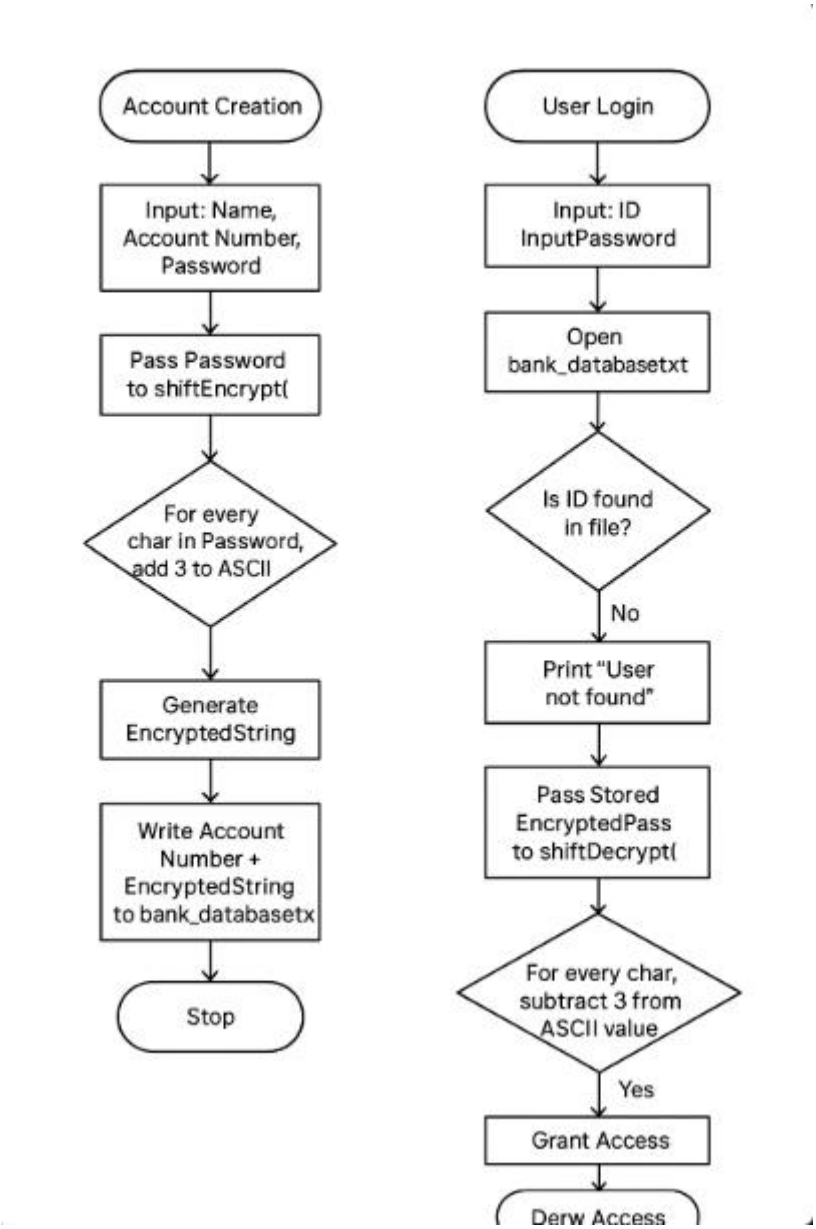
Stored in file: "EDW"

File Handling Logic

The system uses standard C File I/O:

1. **Registration: Opens file in Append mode ("a"), encrypts the password, and writes the record.**
2. **Login: Opens file in Read mode ("r"), reads the encrypted string, applies the reverse shift (Decryption), and compares it with the user's input using strcmp.**

4. System Flowchart (Text Representation)



here is the text logic for your flowchart diagrams.

A. Account Creation (Encryption Flow)

1. **Start**
2. **Input: User enters Name, Account Number, Password.**
3. **Process: Pass Password to shiftEncrypt().**

4. **Loop: For every character in Password, add 3 to ASCII value.**
5. **Output: Generate Encrypted++String.**
6. **I/O Operation: Write Account Number + EncryptedString to bank_database.txt.**
7. **Stop**

B. User Login (Decryption Flow)

1. **Start**
2. **Input: User enters ID and InputPassword.**
3. **I/O Operation: Open bank_database.txt.**
4. **Decision: Is ID found in file?**
    - ***No:*** **Print "User not found" $\->$ Stop.**
    - ***Yes:*** **Fetch StoredEncryptedPass.**
5. **Process: Pass StoredEncryptedPass to shiftDecrypt().**
6. **Loop: For every character, subtract 3 from ASCII value.**
7. **Process: Result = strcmp(InputPassword, DecodedPass).**
8. **Decision: Is Result == 0?**

- ***Yes:*** **Grant Access.**
- ***No:*** **Deny Access.**

9. **Stop**

5. Implementation Details (Key Functions)

- **shiftEncrypt(char *real, char *coded): Iterates through the input string array. Modifies the ASCII value of the character and stores it in the target array.**
- **shiftDecrypt(char *coded, char *real): Performs the inverse operation to restore the original string for verification.**
- **createAccount(): Handles the UI for registration and calls the encryption function before writing to the database.**
- **changePassword(): Demonstrates a complex file operation (Copy $\rightarrow$ Modify $\rightarrow$ Rename) to update credentials securely.**

6. Testing and Results

Test Case 1: Account Creation

- **Input Name: Rohan**
- **Input Password: Code123**
- **System Action: Shifts characters by +3.**
- **File Content Verified: The file bank_database.txt contains Frgh456 instead of Code123.**

Test Case 2: Successful Login

- **Input ID: 101**
- **Input Password: Code123**
- **System Action: Reads Frgh456**
- **$\->$ Decrypts to Code123**
- **$\->$ Matches Input.**
- **Result: Access Granted.**

7. Conclusion & Future Scope

This project successfully secures user credentials using a basic cryptographic technique. While a Shift Cipher provides a basic layer of security against casual observation, it is vulnerable to frequency analysis attacks.

- **Future Enhancement: Implement a hashing**

algorithm (like SHA-256) instead of a reversible cipher for industry-standard security.