# **Project Report: Console-Based Banking System**

# **Introduction**

This project implements a **console-based banking system** in C that enables users to create, manage, and delete bank accounts. The system includes features such as account creation, password protection, deposit, and withdrawal, all while securely storing and retrieving account balances using **XOR encryption**. The system also ensures user privacy and data integrity by encrypting sensitive information like passwords and balances.

.

## **Key Features of the System**

### **1. Account Management**

* **Account Creation:** Users can create a new account by entering a unique account number, the account holder’s name, and an initial deposit amount. Additionally, users are required to set a password for their account, which is encrypted before storage.
* **Account Information Viewing:** Users can view their account details (account number, account holder's name, and balance). To ensure security, users must provide their account password to access this information.
* **Deposit and Withdrawal:** The system allows users to deposit money into or withdraw money from their accounts. The balance is encrypted before saving, ensuring that sensitive information is not exposed in plaintext.
* **Account Deletion:** Users can delete their accounts, along with all associated information (balance and password). Upon deletion, all files related to the account (such as account details, balance, and password) are removed from the system.

### 

### 

### **2. Security Features**

* **Password Protection:** Every account is secured with a password, which is set by the user during account creation. To access or modify account details, the user must enter the correct password. The password is encrypted using **XOR encryption** to prevent unauthorized access.
* **XOR Encryption for Balance Storage:** The system uses **XOR encryption** to encrypt the account balance before storing it in a file. This ensures that the balance cannot be directly accessed, even if someone gains unauthorized access to the system files.
* **Password Encryption:** Similarly, the password for each account is encrypted using XOR and stored in a separate password file. This approach prevents direct access to the password and adds an additional layer of security.

### **3. File-Based Data Storage**

* **Account Information:** Account details (account number and account holder's name) are stored in individual files named account\_<account\_number>.txt. Each account has a separate file to store its information, ensuring organized and accessible storage.
* **Balance Storage:** Each account’s balance is stored in a separate file (balance\_<account\_number>.txt), and the balance is encrypted before being written to the file. This ensures that even if someone gains access to the balance files, the actual balance value remains obscured.
* **Password Storage:** Each account’s password is stored in a separate file (password\_<account\_number>.txt). The password is encrypted using XOR to prevent unauthorized access.
* **Central Accounts List:** A central list of all account numbers is stored in the accounts.txt file. This file is updated whenever a new account is created or an account is deleted.

### **4. Encryption for Data Privacy**

* **XOR Cipher:** The XOR cipher is applied to encrypt both the account password and balance before saving them to the respective files. XOR encryption is a basic method of encryption that ensures that data is obfuscated and cannot be easily read in the file system.
* **Decryption:** The same XOR cipher is applied when retrieving account information (password and balance). The encrypted data is read from the file, decrypted using XOR, and displayed to the user.

### **5. User Interface**

* The program provides a **simple menu-driven interface** that guides users through the various actions available, including account creation, viewing account details, depositing and withdrawing money, and deleting accounts.
* **User Prompts:** Clear and intuitive prompts guide the user through each operation, ensuring ease of use. Error messages are displayed if the user attempts to perform an invalid action

## **How the System Works**

### **1. Account Creation:**

* The user enters a unique account number, the account holder’s name, and an initial deposit amount.
* A password is set by the user, which is then encrypted using XOR before being stored in a separate password file.
* The account information (account number and name) is saved in a file named account\_<account\_number>.txt, and the balance is saved in an encrypted file named balance\_<account\_number>.txt.

### **2. Account Access:**

* To access an account, the user must enter the account number and the password.
* The password is checked against the encrypted version stored in the password file. If the password matches, the system retrieves and displays the account details and balance (after decrypting the balance).

### **3. Deposit and Withdrawal:**

* When depositing money, the user enters the deposit amount, which is added to the account balance. The updated balance is encrypted and saved to the balance file.
* When withdrawing money, the system checks if the user has sufficient funds. If the balance is sufficient, the withdrawal is processed, and the updated balance is saved.

### **4. Account Deletion:**

* The user can delete their account by entering the account number and password.
* The system removes all associated files (account information, balance, password) and updates the central accounts list by removing the account number.

### **5. Data Security:**

* Passwords and balances are never stored in plaintext. Instead, they are encrypted before being saved, making them unreadable without decryption.

## **File Structure**

The following files are used by the banking system:

1. **accounts.txt:** Contains the list of all active account numbers.
2. **account\_<acc\_no>.txt:** Stores the account number and the account holder's name.
3. **balance\_<acc\_no>.txt:** Stores the encrypted balance for each account.
4. **password\_<acc\_no>.txt:** Stores the encrypted password for each account.

## **Security Considerations**

* **XOR Encryption:** While XOR encryption provides basic protection, it is not a highly secure method for production systems. This encryption is intended for educational use and should be replaced with more robust methods, such as AES, for real-world applications.
* **Password Protection:** Passwords are stored securely in encrypted form. The system ensures that unauthorized users cannot read or modify passwords, as they are encrypted before storage.

## **Conclusion**

This **Console-Based Banking System** demonstrates basic banking operations while ensuring security for sensitive data such as account balances and passwords through XOR encryption. The system provides functionalities for account creation, viewing account details, making deposits and withdrawals, and deleting accounts. With its simple menu interface and secure data handling, this project serves as a strong foundation for understanding file management, encryption, and basic account management.

Name : Mashfikuzzaman Taeen

Roll : 25

Registration Number : 2023415968