ATM-Bank Secure Communication System

Group - 5

Contents

1 System Overview **2**

2 System Components **2**
2.1 Bank Server (`bank`) 2
2.2 ATM Client (`atm`) 2

3 System Architecture **2**
3.1 Client-Server Model 2
3.2 Communication Protocol 2
3.3 Database Management 3

4 Key Components and Modules **3**
4.1 Bank Server Modules 3
4.2 ATM Client Modules 3
4.3 Communication Module 3
4.4 Database Module 4

5 Work Distribution **4**

6 Logging and Testing **5**

7 Tools and Technologies **5**

8 Design Considerations **6**
8.1 Security 6
8.2 Scalability 6
8.3 Reliability 6
8.4 Modularity 6

1 System Overview

The project involves developing two main components: an ATM client (`atm`) and a Bank server (`bank`), which will communicate securely over a network. The system is designed to allow customers to perform transactions such as deposits and withdrawals securely. The ATM client interacts with the Bank server to update or query customer account balances. Security measures are in place to prevent unauthorized access and ensure data integrity during transactions.

2 System Components

2.1 Bank Server (`bank`)

- Maintain customer account information and handle transaction requests from ATMs.
- Authenticate requests using the shared auth file and ensure secure communication.
- Implement concurrency control to manage multiple ATM requests.
- Ensure data integrity and rollback any incomplete transactions in case of errors.

2.2 ATM Client (`atm`)

- Handle user inputs for transaction requests (withdrawals, deposits, account creation, etc.).
- Securely communicate with the Bank server using an authentication file.
- Validate input data and ensure compliance with POSIX standards.
- Implement error handling for communication errors and protocol violations.

3 System Architecture

3.1 Client-Server Model

- **ATM Client (`atm`):** A command-line application that serves as the client in this architecture. It interacts with the bank server to perform account-related operations.
- **Bank Server (`bank`):** A standalone server application that listens for requests from multiple ATM clients and processes them based on account data.

3.2 Communication Protocol

The ATM and bank communicate over TCP/IP. The ATM sends transaction requests to the bank, which processes the requests and responds with the transaction status or account details.

3.3 Database Management

- **Bank Server:** It stores account information temporarily, handles transactions, and doesn't save data between sessions.
- **ATM Client:** Does not maintain state except for using the card file as a means of account authentication.

4 Key Components and Modules

4.1 Bank Server Modules

Functionality	Description
`-p <port>`	The port that the bank should listen on. The default is 3000.
`-s <auth-file>`	The name of the auth file. If not supplied, defaults to `bank.auth`.

4.2 ATM Client Modules

Functionality	Description
`-a <account>`	The customer's account name.
`-n <balance>`	Create a new account with the given balance.
`-d <amount>`	Deposit the amount of money specified.
`-w <amount>`	Withdraw the amount of money specified.
`-g`	Get the current balance of the account.
`-i <ip-address>`	The IP address that bank is running on. The default value is `127.0.0.1`.
`-p <port>`	The TCP port that bank is listening on. The default is 3000.
`-s <auth-file>`	The authentication file that the bank creates for the ATM. If `-s` is not specified, the default filename is `bank.auth`.
`-c <card-file>`	The customer's ATM card file. The default value is the account name prepended to `.card` (`<account>.card`).

4.3 Communication Module

- Handles the communication between the ATM client and the bank server, ensuring that all messages are securely transmitted. Includes functions to set up connections, send and receive messages, and handle any communication errors.

- Socket setup and management
- Data serialization and deserialization
- Error handling for communication failures

- Implements security protocols to ensure the integrity and confidentiality of the data exchanged between the ATM client and the bank server

4.4 Database Module

Implement functions for storing and retrieving account details, transaction histories, and balance updates. Maintain a log of all transactions (deposits, withdrawals, etc.) for auditing and rollback in case of errors. Ensure atomicity and consistency of transactions to prevent issues like double spending or unauthorized access.

- Implement functions for storing and retrieving account details, transaction histories, and balance updates.
- Maintain a log of all transactions (deposits, withdrawals, etc.) for auditing and rollback in case of errors.
- Ensure atomicity and consistency of transactions to prevent issues like double spending or unauthorized access.

5 Work Distribution

Team	Member	Responsibilities
Team 1: Bank Server Development	Aryan Sahu	Developing the core banking logic, including account management and transaction processing. Handles concurrency control and error handling within the Bank server.
	Aashmun Gupta	Implementing secure authentication mechanisms and ensuring data integrity. Ensures the server operates reliably.
Team 2: ATM Client Development	Aayush Parmar	Developing the user interface and validating user inputs. Ensures that transactions are processed correctly and securely.
	Pratyaksh Bhayre	Managing the interaction between the ATM client and the communication module. Addresses error handling and ensures compliance with security protocols.
Team 3: Communication Module and Integration	Arjun Sekar	Developing the communication module that establishes secure and reliable connections between the ATM client and the Bank server.

	Mrugank Patil	Integrating the communication module with the client and server. Oversees testing, debugging, and ensures overall system compatibility and performance.

6 Logging and Testing

- Implement standardized error codes for various failure scenarios.
- Develop comprehensive test cases to validate both `atm` and `bank` functionalities.
- Simulate different scenarios, including concurrent transactions and error conditions.
- Ensure the system meets all specified requirements and performs securely under attack models.

Proposed Structure

The project is organized with the following structure:

```
atm.cpp
atm.h
bank.cpp
bank.h
setup.sql
auth.txt
card.txt
run_atm.sh
run_bank.sh
setup_db.sh
Makefile
config.h
logger.cpp
logger.h
communication.cpp
communication.h
encryption.cpp
encryption.h
decryption.cpp
decryption.h
README.md
```

7 Tools and Technologies

Component	Technology/Tool
Programming Language	C/C++ for both client and server, SQL for MySQL Database management.
Database Management	MySQL for handling relational databases.
Automation	Bash script for automating communications.
Networking	POSIX sockets in C or Boost.Asio in C++ for socket programming.
JSON Processing	JSON web-tokens used for communication.
Threading	Threading will be used for achieving concurrency.

8 Design Considerations

8.1 Security

- Ensure the authentication file is securely generated and validated.
- Ensure the ATM and bank communicate securely over the network.
- Ensure that the application is immune to injection attacks like SQL injections and OS injections.

8.2 Scalability

The bank server should efficiently handle multiple ATM client connections concurrently without degradation in performance.

8.3 Reliability

Robust error handling to manage unexpected client behavior or network failures without crashing the server.

8.4 Modularity

Design the system such that additional features or changes (like additional transaction types or persistent storage) can be integrated without significant overhaul.