Indian Institute Of Technology Gandhinagar

CS431 Computer and Network Security

Secure ATM Project

(Group 11)

Members:

Krupa Chetanbhai Rajani 24310035
Arpan Agarwal 24120002
Chhavi Goyal 24120004
Ishika Jain 24120009
Labish Bardiya 24120011
Prateek Goswami 24120016
Sreyashi Karmakar 24310059

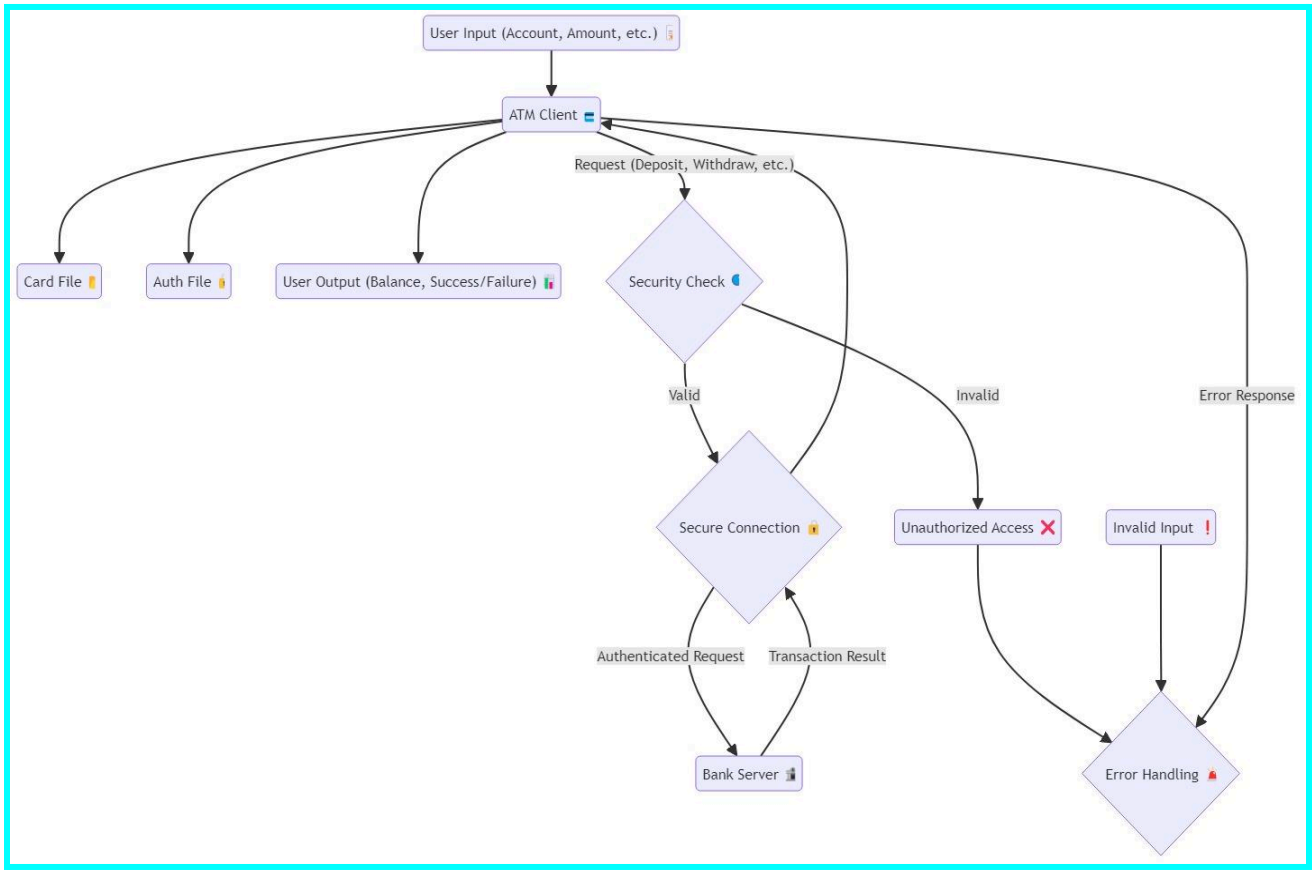
SECURE ATM

Objective

To design a secure C++ communication protocol for ATMs to ensure safe and reliable transactions between the ATM (client) and the bank (server).

Main Entities

ATM: A client application for users to interact with their bank accounts.
BANK: A server application to manage user accounts and process transactions.



Key Features

Authentication Mechanism

- The system uses the auth file for mutual authentication between atm and bank. Both the ATM and bank approve of this file, resulting in the establishment of secure communications.

Secure Transactions

- People with the inaccurate card file do not have any chance to get into their balancing account and improper changes to it.
- The system restricts third parties to execute some actions which are not allowed, as for instance the over-withdrawal.

Man-in-the-Middle Attack Prevention

- One way to protect the safe data is to encrypt the communication between the ATM and the bank so that no one could be able to read or manipulate it by means of man-in-the-middle attack.
- Instituting the checks to confirm that only the authorized messages are transmitted.

Data Integrity

- Making the transactions atomic so that inconsistencies in the account balance can be prevented by the system.

ATM

Flowchart of the ATM Request

1. **Start**

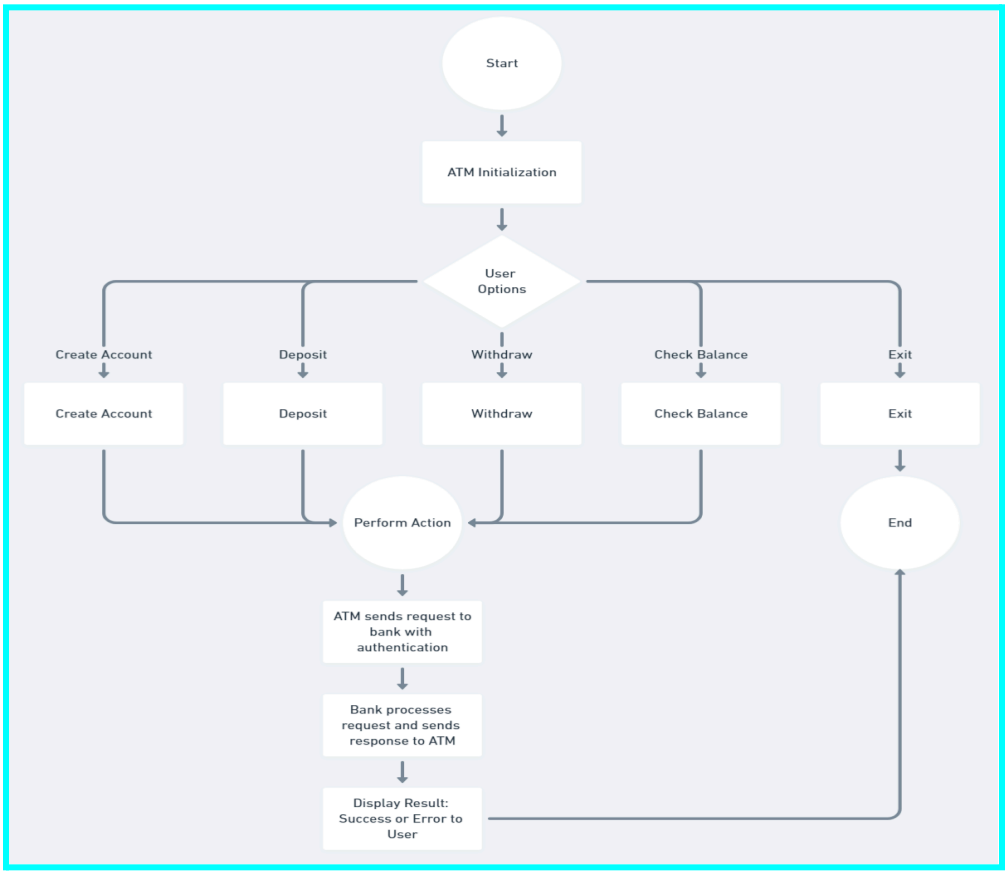
2. **ATM Initialization**
    - The ATM connects to the bank using the IP address and port.
    - ATM reads the auth file for secure communication.
    - User inserts the card (ATM reads the card file).

3. **User Options:**
    - **Create Account**: Users can create a new account by entering account details and an initial balance.
    - **Deposit**: Users can deposit money into their account.
    - **Withdraw**: Users can withdraw money from their account.
    - **Check Balance**: Users can check their account balance.
    - **Exit**: User exits the ATM.

4. **Perform Action:**
    - Depending on the user's choice, the corresponding operation is performed.
    - ATM sends the request to the bank with necessary authentication (using auth file).
    - Bank processes the request and sends a response back to the ATM.

5. **Display Result:**
    - The ATM displays the result (success or error) to the user.

6. **End**



Key Operations of the ATM

1. **Account Creation:**

    - User enters account details and initial balance.
    - An ATM generates a card file (a file that acts as the user's PIN).
    - The ATM sends a request to the bank to create the account.
    - Bank creates the account and returns a confirmation.

2. **Deposit:**

    - User enters the amount to deposit.
    - The ATM sends a deposit request to the bank.
    - Bank updates the account balance and returns a confirmation.

3. **Withdraw:**

    - User enters the amount to withdraw.
    - ATM checks if the balance is sufficient.
    - If sufficient, the ATM sends a withdrawal request to the bank.
    - Bank updates the account balance and returns a confirmation.
4. **Check Balance:**

    - The ATM sends a balance inquiry to the bank.
    - Bank returns the current account balance.

Functionality of the ATM

1. Authentication:

- The ATM authenticates itself to the bank using the auth file, ensuring that communication is secure.

2. User Validation:

- ATM verifies that the card file matches the account information before processing transactions. Creates a new card file in case of Account Creation.

3. Transaction Processing:

- Handles deposits, withdrawals, and balance inquiries securely by communicating with the bank.

Security Aspect of the ATM

1. Auth File:

- Used to secure communication between the ATM and the bank. It ensures that no unauthorized entity can communicate with the bank.

2. Card File:

- Acts as the user's PIN. If compromised, a new account creation process will generate a new card file. Before any transaction, the ATM first checks if the credentials provided by the user (e.g., account name and card file) are correct and match the information stored in the card file. This validation step ensures that only the legitimate account holder can initiate transactions. If the credentials do not match, the ATM will deny access, preventing unauthorized use.

3. Encryption:

- All communication between the ATM and bank is encrypted to prevent man-in-the-middle attacks.

4. Timeouts and Error Handling:

- If communication with the bank fails, the ATM exits securely, preventing unauthorized access.

Features of the ATM

1. Secure Account Creation:

- Users can create new accounts with an initial balance.

2. Deposit and Withdrawal:

- Users can securely deposit and withdraw money.

3. Balance Inquiry:

- Users can check their account balance at any time.

4. Error Handling:

- The ATM handles errors gracefully, ensuring that no sensitive information is leaked.

Communication Protocol

1. Request-Response Model:

- The ATM sends requests to the bank and waits for responses. Each request includes necessary authentication information.

2. JSON Format:

- All communications are in JSON format for simplicity and easy parsing.

3. Timeouts:

- If the bank doesn't respond within a certain timeframe, the ATM considers the request failed and exits securely.

Implementation Details

1. **Programming Language**:

C++

2. Networking:

Use of socket programming in C++ for communication between the ATM and bank.

3. File Handling:

C++ file I/O for managing the auth file and card file.

4. JSON Parsing:

Libraries like nlohmann/json for handling JSON data.

5. Concurrency:

Use of threads if multiple clients (ATMs) need to interact with the bank simultaneously.

BANK

Key Operations

- **Customer Account Management**

    - **Account Creation:** The system allows the creation of new customer accounts with a unique ID and some initial deposit amount.
    - **Balance Inquiry:** The current balance of a customer's account, on request from any ATM client, would be provided by the server.
    - **Balance Update:** On any deposit or withdrawal, the server updates the account balance and instantly writes this new balance to the database.
- **Processing Transactions**
    - **Deposit:** Server deposits an amount into the customer's account and refreshes the balance.
    - **Withdrawal:** It subtracts the amount from the customer's account but never allows it to overdraw.
    - **Transaction Validation:** The server checks if there is adequate money in the account before the withdrawal of cash and that transactions have followed any set rules.
- **Transaction Logging**
    - **Logging transaction detail:** the server records each and every detail of the transaction and it involves a log about the type of transaction, amount, time it occurred and whether it is successful or not.
- **Authentication File Management**
    - **Auth File Creation:** When the server is started, it will create an authentication file used by the ATM clients. The file should not be reused multiple times.
    - **Verify Authentication File:** The server verifies that at any one time, there exists a valid and most recent authentication file being used to communicate with an ATM client. If an older version is found, then it will error out and exit.
- **Database Administration**
    - **Connect to Database and Operations:** The server has to connect to the database in order to create, read, update, and delete customer accounts. It provides logging of all transactions with full support for transactions.
    - **ACID Compliance:** The database operations should be done within the ambit of transactions to ensure that data remains consistent and reliable and follows Atomicity, Consistency, Isolation, and Durability principles.

Communication Protocol

The Bank Server and its ATM clients communicate via a TCP-based protocol with SSL/TLS-based encryption. The communication happens as follows:

- **Connection Establishment:** The ATM client has to connect to the Bank Server, which is listening on a predefined TCP port.
- **SSL/TLS Handshake:** After having the secure channel set up, the actual exchange of data starts-wherein, at the client's end, the requests for transactions are sent to the server; it processes the same and responds accordingly.

- **Connection Termination:** Finally, the connection terminates either after all the transactions are processed or on the request of the client.

Implementation Details for Secure Communication

- **Bank Server:** The Bank Server is a C++ based program, using the OpenSSL library for SSL/TLS communication and authenticates the client using a digital certificate while in a handshake.
- **Transaction Logging:**The use of JSON to persistently log all transactions allows for an audit trail to be determined.
- **User Authentication and Authorization:** Bank server provides functionality for strong authentication, allowing users with authorization to operate on certain things.
- **Scheduled Transactions:**Users schedule a transaction to be executed at some specific future date and time.
- **Card File Authentication and Security:**This such that only the customer who has the correct card file can view or update account balance information.
- **Integration with DBMS:** Bank Server interacts and integrates with DBMS that maintains customer's balances and details of transactions.
- **Authentication File Management:**It provides an authentication file that protects communication by maintaining user credentials.
- **Server-Side Operations:**Bank startup and runtime and its shutdown must securely and effectively be dealt with by the server.

Authentication File (auth file)

Purpose:

- The authentication file has information between the atm and bank that is used to authenticate their communication. It is the assurance of the client and server validity that transaction code is prompted.

Structure:

- The cryptographic variables involved should be disclosed in the auth file by the ATM machine upon startup. It takes place after the bank has securely generated and the ATM has securely transferred it using a reliable channel.

Security Considerations:

- The authentication file should be kept away from unintended eyes.
- Use of a strong cryptographic algorithm to generate it which is difficult for attackers to break.
- It is active only for the length of the session and should be changed occasionally to avoid its use by an intruder.

Technology and Implementation

- **With C++:** We shall use C++ programming language in these two cases as we shall require to write both server end as well as the client logic in C++ technology.

- **Encryption:** The best way to address this problem is the usage of a session key between both the ATM machine and the bank, and this together with use of the bank's certificate with the bank's public key and the ATM machine's public key to create a session key.

- **Authentication:** For the security protocols, the auth file shall be created and exchanged with the help of the key exchange and encryption technique, which is implemented through the cryptographic libraries.

- **Database Interaction:** SQLite or PostgreSQL libraries will connect it; the user accounts, transactions, and other vital records will be stored in it. File Handling: In the C++ code, the secure I/O function to store the card files, and the information on the smartphone, should be added.

- **Rollback Mechanism:** Thus, we will have the transaction rollback feature through development of new C++ classes that will enable the usage of the transaction and the error handling.

Security Properties

- **Confidentiality:**

  Examples of sensitive data are users' PINs, account information and transactional data. This ought to be kept secure by employing means that make the held information comprehensible only if it is decrypted using the correct key. The data should also be secure and in the transfer of the data from the ATM client to the server, it should not be reached easily by hackers, man-in-the-middle attacks among others. Further, there are key exchange models that tend to provide confidentiality to the encryption key.

- **Integrity:**

  This is due to the fact that cryptographic methods allow detecting even minor interference or unauthorized modifications of the data transmitted between the parties. Away from the functionality of each, normalization and veri cation also improve the safeguard of integrity by making it nearly impossible for a log-based attack which can be exploited to alter data.

- **Authentication:**

  It includes for instance, security measures geared towards the verification of the identities of the parties communicating and confirming the identity of entities. Moreover, validation of data request is done adequately meaning that only data which come from legitimate and secure sources are worked on hence the malicious operation cannot be done.

- **Authorization:**

  The authorization processes function so as to implement access controls, thus either permitting or prohibiting an action based on the role of the user. Input validation checks prevent only unauthorized actions from being performed and; therefore, the system is protected from unlawful access or usage.

Project Structure and Groups Allocation

- **Group 1:**

  Bank Server Development (2 members:- Arpan Agarwal, Sreyashi Karmakar)

  Responsibilities:
  - Implement the "bank" server application.
  - Develop secure communication protocols using SSL/TLS.
  - Manage account activities such as creation, deposit, withdrawal, and balance checking.
  - Implement secure authentication with the "auth file".

  Technologies:
  - C++ for the server implementation.
  - Use OpenSSL for SSL/TLS encryption.
  - Access JSON for communication format between "atm" and "bank".

  Timeline:
  - **Week 1:** The bank server overall architecture designing and setting up the development environment will be the key tasks here.

- **Week 2:** Implement basic account management features (creation, deposit, withdrawal, balance checking).
- **Week 3:** Integration of SSL/TLS for the secure communication will be carried out and also the development of the "auth file" authentication mechanism.
- **Week 4:** Implement error handling, transaction rollback, and concurrency control functionalities.
- **Week 5:** Testing, debugging, and finalization.

- **Group 2:**

  ATM Client Development (2 members:- Labish Bardiya, Krupa)
  Responsibilities:
  - Implement the "atm" client application.
  - Develop features for account creation, deposits, withdrawals, and balance inquiries.
  - To have secure communication with the bank, use SSL/TLS and the auth file.
  - Validate the user inputs and handle the card file related to the accounts.

  Technologies:
  - C++.
  - Use OpenSSL for the SSL/TLS encryption.
  - Use the JSON for bank communication.

  Timeline:
  - **Week 1:** Set up the development environment and design the `atm` client architecture.
  - **Week 2:** Implement the basic features: account creation, deposits, withdrawals, and balance inquiries.
  - **Week 3:** Integrate SSL/TLS and handle the `auth file` for secure communication.
  - **Week 4:** Implement input validation, card file management, and error handling.
  - **Week 5:** Testing, debugging, and finalization.

- **Group 3:**

  Security and Testing (3 members:- Chhavi Goyal, Prateek Goswami, Ishika Jain)
  Responsibilities:
  - **Security Focus:** The design of the atm and bank systems has to put a great deal of importance on security to avoid attacks like man-in-the-middle.

  - **Testing Plan:** Develop a comprehensive testing strategy by integrating unit tests, integration tests, and security tests to really validate the robustness of the system.

  - **Auth File Management:** Ensure that auth file distribution is securely done to different environments to preserve the integrity of secure communications.

- **Code Reviews and Audits:** Perform regular code reviews and audits of security for the identification and removal of any potential vulnerabilities.

Technologies:

- C++ which is employed to evaluate and test the code.
- We are testing 'OpenSSL' and 'secure communication'.
- Test frameworks: Use appropriate C++ testing frameworks such as Google Test for automatic tests.

Timeline:

- **Week 1:** Strategy and setup the testing environment. Start security analysis and plan for encryption.
- **Week 2:** Writing unit tests will be begun, and security tests for the `bank` server will be performed.
- **Week 3:** Test the `atm` client by running tests on secure communication and error handling.
- **Week 4:** Perform the integration test between `atm` and `bank`, attack stimulation (e.g., MITM), and improve security measures.
- **Week 5:** The final security audit, code review, and ensure all tests are successful. Write down the findings and prepare for the final submission.

References

- https://www.gleek.io/blog/atm-system-class
- https://www.techtarget.com/searchnetworking/definition/TCP-IP
- https://www.endpointprotector.com/blog/ways-banks-secure-data/

—END—