Build It Document

General behavior: Init

- Written as a script that creates the following files, where <path>/<init-fname> is the path/filename specified by the user as a command line argument:
  - <path>/<init-fname>.bank
    - This file contains a 32 byte symmetric key for AES 256, in the form of raw bytes, using a cryptographically secure pseudorandom number generator (OpenSSL)
  - <path>/<init-fname>.atm
    - This file is a mere copy of <path>/<init-fname>.bank

General behavior: Bank

- Bank records the time stamps of every message it sends. Only the time stamp of the last message is kept. In a situation where the Bank does not send a reply to ATM, the time stamps of the last message received from the ATM is recorded instead. The time stamps are set to current time at program start.
- Message time stamps are checked, making sure that the received message was sent at a time after the last received or sent message
- Message account/card number is checked when there's an active session or the user is trying to log in and the Bank needs to validate the PIN. Messages not matching the active user's card number are ignored
- Only when the Bank is at the initial 0 state (NO_SESH) that the card numbers aren't checked because in this state, the only thing that the Bank allows is login requests.
- Then the Bank processes the rest of the command.
- Bank account specifications:

- Bank generates the account card number by encrypting the string: “<username>.card<pin>”, then hashing it, resulting in 32 hexadecimal characters.
- If the card number already exists, the bank will continue the above process until a unique number is found.
- These hex characters are what’s in the .card files
- Card numbers are all unique for each user

General behavior: ATM

- ATM records the time stamps of every message it sends. Only the time stamp of the last message, sent or received, is kept. In a situation where the ATM does not send a reply to Bank, the time stamps of the last message would be the one received from the Bank. The time stamps are set to current time at program start.
- Message time stamps are checked, making sure that the received message was sent at a time after the last received message or sent message
- Message account/card number is checked whenever a message is received from the Bank. Messages not matching the active user’s card number are ignored
- Then the ATM processes the rest of the command
- If a message is rejected, the ATM waits until a valid message is received before continuing execution

Structure of a message: IV (16 bytes) + Epoch time (seconds, 8 bytes) + Epoch time (microseconds, 4 bytes) + account/card number (32 bytes) + **command (bolded)**

Protocol start

Initially, both Bank and ATM start at state 0 (INITIAL and NO_SESH respectively).

atm→bank: ATM begins session with a username, sends message if <username>.card file found

- sends command: **login-request**
- ATM changes state → 11 (LOGIN_REQ_WAITING), waiting for confirmation of user account
- ATM stores card number in atm→active_card: a char array of 32 hexadecimal characters
- Bank receives message, stores card number in bank→active_card (same type array as atm)
- Bank checks for account existence

bank→atm: Bank replies to login request after searching for user

- (IF USER NOT FOUND) sends command: **no-user-found**
  - Bank does not change state
  - Bank clears card number out of bank→active_card
  - ATM clears card number out of atm→active_card
  - ATM changes state → 0 (INITIAL)
- (IF USER FOUND) sends command: **user-found**
  - Bank changes state → 99 (AWAIT_PIN)
  - ATM receives message from bank, reads PIN from stdin

    atm→bank: ATM sends result of PIN read to Bank
    - (IF PIN INCORRECT FORMAT) sends command: **unverifiable**
      - ATM clears card number out of atm→active_card
      - ATM change state back → 0 (INITIAL)
      - Bank receives message and clears bank→active_card
      - Bank changes state back to → 0 (NO_SESH)
    - (ELSE) sends command: **verify XXXX** (PIN to verify, 4 chars)
      - ATM changes state → 22 (VERIFY_PIN_WAITING)

- Bank receives PIN in message, checks stored PIN

  bank→atm: Bank sends result of PIN comparison to ATM

  - (IF NOT EQUAL) sends command: **access-denied**
    - Bank clears card number out of bank→active_card
    - Bank changes state back to 0 (NO_SESH)
    - ATM receives message
    - ATM clears card number out of atm→active_card
    - ATM change state back to 0 (INITIAL)
  - (IF EQUAL) sends command: **access-granted**
    - Bank changes state → 11 (OPEN_SESH)
    - Bank saves a pointer to the logged user's file (bank→logged_user)
    - ATM receives message
    - ATM changes state → 33 (ACTIVE_SESSION)

(ALL MESSAGES BELOW ARE ONLY APPLICABLE IN **access-granted** CASE I.E. WHEN THERE IS AN ACTIVE/OPEN SESSION)

atm→bank: ATM sends user's request to withdraw money

- sends command: **withdraw\0(<int> four bytes)**
  - ATM changes state → 44 (WITHDRAW_WAITING)

    bank→atm: Bank sends result of withdraw attempt back to ATM
    - (IF BALANCE ENOUGH) sends command: **dispense\0(<int> four bytes)**
      - Bank changes state to → 33 (WITHDRAW)
      - ATM receives message and dispenses money
      - ATM changes state back → 33 (ACTIVE_SESSION)

atm→bank: ATM sends confirmation of money dispensed

- sends command: **dispensed\0(<int> four bytes)**
  - Bank deducts the dispensed amount from the active user's balance
  - Bank changes state back to 11 (OPEN_SESH)

- (IF BALANCE NOT ENOUGH) sends command: **insufficient**
  - Bank does not change state
  - ATM receives message
  - ATM changes state back → 33 (ACTIVE_SESSION)

atm→bank: ATM sends user's request to check balance to Bank

- sends command: **balance**
  - ATM changes state → 55 (BALANCE_WAITING)
  - Bank receives message and obtains balance

  bank→atm: Bank sends balance back to ATM
  - Bank sends command: **balance\0(<int> four bytes)**
  - Bank does not change state
  - ATM receives message
  - ATM changes state back → 33 (ACTIVE_SESSION)

atm→bank: ATM sends user's request to end session to Bank

- sends command: **end-session**
  - ATM changes state → 0 (INITIAL), clears atm→active_card and atm→curr_user (username of logged in user)
  - Bank receives message
  - Bank changes state → 0 (NO_SESH), sets bank→logged_user (pointer to the user's file) to NULL, and clears bank→active_card

Security Features:

Mitigated attacks:

1. Buffer Overflow:
    - All input from the user is sanitized to be accurate
    - Only DATASIZE bytes of input is allowed at once
    - stdin is cleared of all input outside of DATASIZE
    - DATASIZE is large enough for all valid inputs
2. Card file Forgery
    - Bank generates the card number by encrypting the string: “<username>.card<pin>”, then hashing it, resulting in 32 hexadecimal characters.
    - Encryption is done using AES 256 in CBC mode with a randomly generated IV
    - One-way hashing is done with MD5
    - Card numbers are unique for every user
    - The card number of a user attempting to use the ATM is sent with every message and checked to match for all applicable transactions
3. Message Forgery/Spoofing
    - Messages are encrypted with AES 256 in CBC mode with a randomly generated IV
    - Symmetric secret keys are used to allow proper decryption
    - Symmetric secret keys are generated with a cryptographically secure pseudorandom number generator (OpenSSL)
4. Message Replay
    - Bank and ATM changes state whenever a reply message is expected

- If a message isn’t being expected for a certain state, then it is ignored
- Each message is encrypted with a time stamp that gives the seconds and microseconds since the Epoch time (00:00:00 1970 Jan 1st)
- Messages received are checked to make sure that they were sent from a later time than the previously recorded time stamp

5. Message Splicing
    - Since messages are encrypted in CBC mode, they are no longer valid if spliced
    - IVs are generated with a cryptographically secure pseudorandom number generator (OpenSSL)