Final Project Documentation

Secure Banking
Version 1.01

Prepared by:
Roberto Perez-Mendoza
perezmendoza_roberto@csu.fullerton.edu

May 25, 2017
CPSC452
Mikhail Gofman

Abstract

This individual proposal research contains brief description of the final project, which was secure banking. It is important to know the concept of how to apply cryptography into this project in order to get an idea of how secure ATM's work in real world. In this case I used both RSA and AES encryption and decryption to provide a secure communication protocol between the ATM1 and ATM2 with the bank's server. In this project, one should be able to handle 2 ATMS to make transactions. The following transactions are as follow: Deposit money, withdraw money, check balance statement, check current bank account records from previous transactions, and finally log off from the user's account. With that being said, this program contains 5 options to used and all these options are being encrypted and decrypted using RSA/AES encryption. 4 accounts have been created for testing purposes. In addition, this program was implemented using python as it was much easier to complete as well as it already offers classes in which I will be discussing later on.

Introduction:

The secure banking project contains two directories. One is for the ATM and the other one is for the server. The purpose of having these to directories is to demonstrate how they both communicate each other using localhost connection. Also, as mentioned previously, the implementation used for this project was RSA and AES encryption, which allows to provide a secure communication protocols between both ATMs and the server. When the program is run, one directory should be created under the atm directory. Inside the created directory contains the public and private keys for both ATM1 and ATM2. These keys were used using RSA encryption and decryption. In addition, if one clicks on the server folder, one should be able to see that 2 directories and 4 files will be created. Inside one directory, one should be able to find the account records list decrypted, which will be saved for later use. The second directory will contain the server public and private keys in which they were

implemented using RSA encryption and decryption. And finally, the 4 files created contain the balance and all files encryption and decryption using AES encryption method. As mentioned previously, the purpose of this project is to provide a secure banking transaction using the above encryptions.

Design:

The implementation of the program was using python, one reason python was used for this project was because of the libraries it already offered, which makes the coding process much easier and also because in my opinion, python is friendly to use. In this paper one should be able to see a use cases as well as a use case diagram, which visually describes the implementation of the program.

Use Cases

1. Check Balance
2. Make Deposit
3. Withdraw Money
4. Record History
5. Log Off

Fully-Dressed Use Cases

USE CASE: Check Balance
UC-ID: #001
Description: Allows the user to check bank balance
Scope: Bank account
Level: primary user goal
Primary actor: end user
Stakeholders/Interests: Application publishers
Preconditions:

- User must be able to successfully enter a correct account number and password
- User must be able to communicate with the server securely

Postconditions:

- If logged in successfully, a menu with 5 options will be display for the user to choose.
- If one is selected, this condition will be satisfied

Main Success Scenario:

1. When 1 is selected, current balance will be displayed.
2. Will go back to the main menu
3. The server will display a message of a successful secure connection from the bank as well as approved selection

Extensions:
N/A
Special Requirements:

N/A

USE CASE: Make Deposit
UC-ID: #002
Description: User should be able to make deposit to the current account logged in. It will update his/her bank account as well as the record history
Scope: ATM machine
Level: basic system goal
Primary actor: end user
Stakeholders/Interests: Application publishers
Preconditions:

- User must be able to successfully enter a correct account number and password
- User must be able to communicate with the server securely
- User must enter a valid value that is non negative nor 0.

Postconditions:

- The record balance will be updated
- It will also display on the screen an updated bank transaction.

Main Success Scenario:

1. Prompts the ATM menu from selection 1-5
2. User selects option 3
3. Securely goes to the server and approves selection
4. User enters a value to deposit to the account being logged in
5. Securely goes to the server and approves is valid selection.

Extensions:
N/A
Special Requirements:
Must enter a valid value to successfully make the deposit.

USE CASE: Withdraw Money
UC-ID: #003
Description: User should be able to withdraw money from the current account logged in. It will update his/her bank account as well as the record history
Scope: ATM machine
Level: basic system goal
Primary actor: end user
Stakeholders/Interests: Application publishers
Preconditions:

- User must be able to successfully enter a correct account number and password
- User must be able to communicate with the server securely
- User must enter a valid value that is non negative nor 0.

Postconditions:

- The record balance will be updated
- It will also display on the screen an updated bank transaction.

Main Success Scenario:

6. Prompts the ATM menu from selection 1-5
7. User selects option 2
8. Securely goes to the server and approves selection
9. User enters a value to deposit to the account being logged in
10. Securely goes to the server and approves is valid selection.

Extensions:
N/A
Special Requirements:
Must enter a valid value to successfully withdraw money from the bank.

USE CASE: Record History
UC-ID: #004
Description: User should be able to see a list of history transaction records from the current account logged in.
Scope: ATM machine
Level: basic system goal
Primary actor: end user
Stakeholders/Interests: Application publishers
Preconditions:
- User must be able to successfully enter a correct account number and password
- User must be able to communicate with the server securely
- A list of recent transactions and time and date should display on the scree
    - It will create a .txt file for later use.

Postconditions:
- It will display a list of transactions made as well as time and date

Main Success Scenario:
11. Prompts the ATM menu from selection 1-5
12. User selects option 4
13. Securely goes to the server and approves selection
14. User then is able to view a list of transactions made with the time and date

Extensions:
N/A
Special Requirements:
Must enter a valid user account in order to access to the history record.

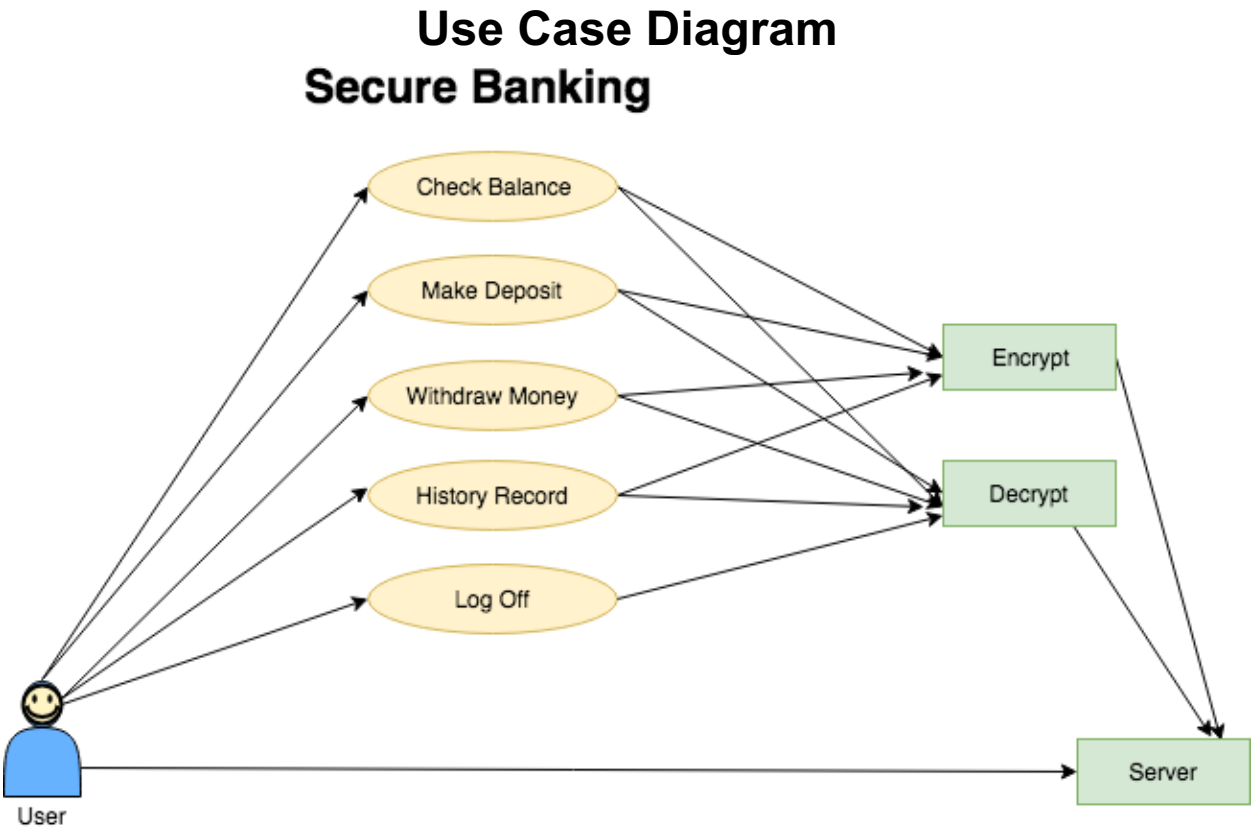
USE CASE: Log Off
UC-ID: #005
Description: User logs off from the account securely to prevent others from accessing. RSA Encryption is then apply.
Scope: ATM machine
Level: basic system goal
Primary actor: end user
Stakeholders/Interests: Application publishers
Preconditions:
- User must select option 5 to successfully log off from the ATM

Postconditions:
- It will display on the screen a log off message.
- Will communicate with the server to start the encryption

Main Success Scenario:
15. Prompts the ATM menu from selection 1-5
16. User makes selection 5
17. A message “log off” will display and user will disconnect from both server and ATM
18. Server will approve the message and start encryption.

N/A

Special Requirements:
N/A



Security Protocols:

I used both RSA and AES encryption. RSA was use for creating public and private keys in both ATMS and the server. RSA encryption was used for encrypting and decrypting bank transactions such as deposit, withdraw, and record history.

Implementation:

The implementation for this project was using python 2.7. Please know that this program won't work with python 3.xx. In addition, in order for this program to work properly, one should be able to install the following libraries:

*python-passlib

-sudo apt-get update

-sudo apt-get install python-passlib

*aragon2_cffi

-sudo pip install argon2_cffi

Python-passlib is being used to create passwords for both ATM1 and ATM2 and also for verification, that way when the user decides to access the server. He/she must create a password for encryption and to decrypt it when accessing to the server securely, he/she must enter the password created. This library is required for bank.py class.

The library argon2_cffi is being used for hashing passwords. One can see that this program uses hashing passwords and this library provides the tools necessary for the program to run successfully.

Moving to the coding portion. One should be able to see two directories, which have been created with the purpose to testing secure communication between the ATMs and the server and also a bank.py class, which uses the pass-lib to ask the user for encryption password creation for both ATM1 and ATM2. Inside the two directories created, one should be able to find the atm.py class, which is located inside the atm directory and the server.py class will be inside the server directory.

Conclusion:

As a result, my main focus was implementing a secure banking program, which provides secure communication protocols in both ATMs and the server. The program was implemented using both RSA and AES encryption. It was written in python programming because python already offers libraries such as python-passlib and argon2. As mentioned previously these libraries are great for password encryption because one takes care of creating password for encryption and decryption and the other

one takes care of hashing password. When the program is being run, one should be able to see two directories. One that contains the atm.py file class and the other directory that contains the server.py file class. In addition, one should be able to see the bank.py file class. This class uses the python-passlib in order for the user to create a encryption password for both ATM1 and ATM2 and to use that password for the server later on so that the user can decrypt it and access to secure connections between the server and the ATMs. When the user follows the procedures correctly, the server and ATM should be able to provide a secure connection. The ATM contains 5 options for the user to use and all these 5 options will connect to the server securely. Please note that 4 users accounts have been created for testing.

References

https://stackoverflow.com/questions/16056135/how-to-use-openssl-to-encrypt-decrypt-files?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

https://passlib.readthedocs.io/en/stable/

https://pypi.org/project/passlib/