**MIDN: Andres, Emmanuel, Petersen**

**General Overview:**

First, the user has to start a session with the router, bank, and ATM. The user will then copy the contents of their card, (ex: "Alice.card”) into a file "inserted.card". The User will go into the ATM interface and enter the command "begin-session". This command will read the contents on their card and prompt them to confirm their identity with their personal four-digit pin. The ATM will then grant access to the user's bank account if it meets the criteria. The user will then be able to enter one of three commands.

1. The balance command does not take any arguments; the user just types it in and the amount of money the user has in their account will be displayed if it meets the criteria.

2. The withdraw command is the only command with a supporting argument. The amount (the other argument with the command) will be the amount of money (in digits) that the user wants to withdraw. The ATM will then talk to the bank to give the money and will display the amount dispensed if it meets the criteria.

3. The end-session command in the ATM prompt will end the communication with the bank and the user's session, therefore causing them to "begin-session" again before any more actions may be taken.

**Card Structure:**

The cards will only contain only the user’s name. All the four-digit PINs will be stored on the secure storage files. This card structure works in harmony with our program’s construction of user verification taking place on the bank system with the ATM system acting only as a pipe in which the bank system and user communicates. The negligible information stored on the cards increases the user’s security because all sensitive information such as their PINs and balances are stored in secure storage. Additionally, all communication pertaining to this will be encrypted.

**Cryptographic Scheme:**

We are going to implement an AES CTR encryption for all communications. This will prevent anyone from eavesdropping on the router and intercepting commands. This also prevents someone from making commands and sending them to the bank’s socket as well. The reason we are choosing an AES CTR is that it is a stream cipher. And because the bank system and ATM system commands and messages vary in length, using a stream cipher would simplify the encryption process.

**Communication between the ATM system and Bank system:**

The messages will be encrypted before being sent using our cryptographic scheme. The messages travel over through the router program while encrypted, and then they are decrypted at the endpoint; either the Bank or ATM. Both of the systems store keys and pins inside of bin files, however, the balances will only be stored in the bank system in a secure fashion.

**Fundamental security principles:**

Our ATM system uses the least privilege mechanism. The user at the atm will not be able to even access the commands the atm has to offer unless they are signed into their account. The whole system changes accounts along with the user, so they can not affect others’ accounts. Our ATM system also adheres to the fail-safe default principle because no withdraws will be attainable if there is no connection with the bank machine.

**Jobs:**

**Andres:**

* Handle local ATM/Handle remote Bank
* Handle local Bank/Handle remote ATM

**Emmanuel:**

* Client information list generation/cards
  + Hard coding information into the bank and generating the format of the information that will be used for user verification

**Petersen:**

* encryption/decryption scheme (bank/atm)
* Ensure local storage of values are secure
* Work on creating extra security features (overdraft protection etc.)