CMSC414 Project 4 Design Document

For our implementation of an ATM and bank, we built out the basic functionality of the interactions between the ATM, bank, and user along with security mechanisms to prevent adversaries from disrupting our ATM service. First, the init program is executed, which generates a random key and initialization vector and writes them to the initial files for the ATM and bank (<path/filename>.atm and <path/filename>.bank, respectively) to read from. The random functions from the C standard library were seeded with the current time and used for the key and IV generation.

When the ATM is started, it can begin a user's session, check the balance of the current user, and withdraw money from the user's account at the bank. When a user session is begun, the ATM requires the user's card (<user-name>.card) to be present. If the user's card is found, the ATM will prompt the user for their PIN that they originally set with the bank. The PIN the user gives as input must exactly match the PIN stored with the bank in order for the user to log in. If both of these requirements are filled, the user will be logged in to their account and can choose to check their current balance and withdraw money from the bank. If three unsuccessful attempts are made, the user is locked out of the ATM permanently.

When the bank is started, the bank is able to create a user, check a user's balance, and deposit money for a user. The bank also receives messages from the ATM if a user at the ATM wants to perform some action. Communication between the bank and ATM is secured with 128-bit AES encryption in Cipher Block Chaining (CBC) mode so that any message sent to either the ATM or bank has to be encrypted or decrypted using the random key and IV set in the init program.

There are four types of messages that the ATM can send to the bank. The first type is of the format Encrypt("u <user-name>"), which prompts the bank to check if a user exists. If the user exists, the bank replies with Encrypt("yes") and if not, the bank replies with Encrypt("nouser"). The second type is of the format Encrypt("p <user-name> <PIN>"), which prompts the bank to check if the PIN inputted for <user-name> at the ATM matches the PIN stored for <user-name> in the bank. If the PIN matches, the bank replies with Encrypt("yes") and if not, the bank replies with Encrypt("no"). The third type is of the format Encrypt("w <user-name> 0 <amt>"), which prompts the bank to withdraw <amt> from <user-name>'s account. If withdrawal is successful, the bank sends back Encrypt("yes") and if not, the bank replies with Encrypt("nofunds"). The fourth type is of the format Encrypt("b <user-name>"), which prompts the bank to send back the encrypted balance of <user-name>.

Here is a list of attacks considered and the countermeasures against them:

1. A possible attack on this system is the attacker listening in on the channel and forging messages to and from the ATM and bank. Our system counters this attack by using 128-bit AES CBC encryption for messages between the ATM and bank. Any message

encrypted with anything other than the secret key and IV generated from the init program will result in meaningless text.

2. A possible attack on this system is a brute force attack on a user's pin, where the attacker tries all possible PIN combinations. Our system protects against this attack by checking if more than 3 unsuccessful login attempts for a user has been logged at the ATM, which would result in that user being unable to use the ATM permanently.
3. A possible attack on this system is an attacker forging a user's card to be used at the ATM. Our system counters this attack by encrypting the user's PIN with 128-bit AES CBC encryption and storing it in the card. When the attacker forges a card, the attacker must also know the secret key and IV used to encrypt the user's PIN to use the forged card.
4. A possible attack on this system is a known plaintext attack, where the adversary has access to the format of messages exchanged between the ATM and bank and employs a brute force tactic to decipher the key and IV. Our system protects against this attack by using 128-bit AES CBC encryption, which requires an infeasible amount of computation to compute the key and IV.
5. A possible attack on this system is a buffer overflow attack, where the adversary passes in malicious code as input in an attempt to overflow a buffer used in the system. Our system protects against this attack by reading in a fixed length of the input and ensuring that this length does not surpass the maximum length of the buffer.

There were some additional threats that we considered, but ultimately decided to ignore because we concluded that it was not very important. For instance, we were thinking about using a Diffie Hellman approach where the bank and atm would each generate a unique secret key. This would have prevented an attacker that had the ability to observe what the init program generated for the bank and atm files from knowing the key and iv. Using this approach would have added more security, but it became unimportant since the attacker cannot observe the bank and atm files. Also, we thought of an attacker that could possibly create an extremely large amount of users to overload the bank's hash table. However, we ignored it because it would be difficult to do so given the amount of users that the attacker would have to create.

There were some threats that we would have liked to address but were unable to do so. For instance, the attacker can modify the packets being transferred through the router. We couldn't implement MAC that would have resolved this issue. In addition, we lock out a user if he makes three unsuccessful login attempts, however, an attacker could potentially lock out every user using a dictionary attack with a list of names, resulting in a dysfunctional atm. Although we would have liked to unlock a user by doing further authentication similar to real banks, the bank does not store any security questions and answers for the purposes of this project.