Cryptology and Network Security 1

Fall 2014

Bank/ATM Group Project

Luke Jones, Anthony Barbieri, Daniel Ben-Chitrit, Mike Perkins

**Overview/Assumptions**

In order to facilitate our Bank/ATM program we made a few assumptions based on the real world use of a Bank/ATM system rather than a simulation. Anytime a real ATM would require a user to communicate with a Bank Official due to some logon problem or difficulty with their account, we simulate this by freezing the account in question. This requires a reboot of the bank server to unlock the account. We also visualized the card file to be a physical card. If the adversary is in possession of the card they have access to all information on it. Thus it is irrelevant to attempt to mask the stored account number as those are printed on the physical card.

**Library**

We used Crypto++ to implement our security protocols. In order to correctly run the program, a working copy of Crypto++ must be present. The specific libraries that we use are:

#include "includes/cryptopp/sha.h"

#include "includes/cryptopp/hex.h"

#include "includes/cryptopp/aes.h"

#include "includes/cryptopp/ccm.h"

#include "includes/cryptopp/gcm.h"

#include "includes/cryptopp/osrng.h"

**Running the Program**

The Bank program takes one command argument in order to run, the port number it is to listen on. The Proxy program takes two command arguments, the port to listen to and the port to forward to. The ATM takes two command arguments, the port to communicate on and a numerical ID for the ATM (1-50). We added this second argument in order to limit the number of simultaneous ATM connections and help prevent DOS attacks. We also use that ID to create a KEY file which stores that ATM’s private AES key. In order to properly run the system, each program needs to be run in a separate command window using the following example:

./bank 5000

./proxy 7000 5000

./atm 7000 1

The Bank is already set up with three user accounts:

|  |  |  |
| --- | --- | --- |
| Name | PIN | Balance |
| Alice | 123456 | 100 |
| Bob | 456789 | 50 |
| Eve | 654321 | 0 |

**Transactions**

The ATM can complete the following transactions: Login, Balance, Withdraw, Transfer, and Logout. The bank provides the following commands: Deposit and Balance.

**Communications Security**

Communications between the ATM and Bank are encrypted using AES with reliance on Nonces created at every step in the communication. Hashing is achieved through SHA512.

Key: E is AES encryption, L is logon Information, N is described Nonce, H is Hash function, C is command, M is a message.

|  |  |  |  |
| --- | --- | --- | --- |
| STEP | ATM | DIRECTION | BANK |
| 1 | Ask for session |  | Generate Nbank1 + start session |
| 2 | Decrypt and verify |  | E(Nbank1) |
| 3 | E(L, Natm1,H(Nbank1+Natm1) |  | Store Natm1 and check Nbank1, verify Logon Information |
| 4 | Decrypt and verify |  | If Login info good, generate  E(Mlogon\_good, Nbank2,H(Natm1+Nbank2) |
| 5 | Generate  E(C, Natm2, H(Natm2 + Nbank2)) |  | Decrypt and verify, run command |
| 6 | Decrypt and display status |  | E(Mcommand status, Nbank3, H(Natm2+Nbank3)  Kill connection |

If the user wishes to conduct another transaction, they must login again with their credentials.

**Attack Defenses**

In addition to the AES encryption, this program uses several other features to deter an adversary. The use of continually generated Nonces helps thwart replay attacks as each communications step requires a new randomly generated Nonce. By adding the new Nonce and previous Nonce together and passing the Hash value it requires an adversary to collect and decrypt multiple transactions in order to submit a false command.

The Bank software will only accept 50 connections of new ATMs, this will prevent an adversary from causing a DOS attack by generating large numbers of ATM connection requests.

The input PIN will be masked with asterisks to prevent an over-the-shoulder adversary from viewing the pin.

Transfer/Withdrawl/Deposit requests will be checked before being applied by the bank to ensure that there is sufficient money to cover the transaction as well as ensuring that the transaction will not cause a rollover. Thus the maximum allowed amount in any account is $1,000,000.

All Packets are padded to 984 bytes to ensure no information leakage. 984 was selected as it provides the largest encrypted block smaller than 1024 in AES. The packets will be sent at random intervals not to exceed 1 second in order to help thwart timing attacks.