Solution to Exercise 4

OO: Applet Design, Java Card RMI, Optimizations

Applet Specification

Applet Functionality:

1. Verify the PIN
2. Update (= change) the PIN
3. Unblock a (blocked) PIN
4. Get the password for a URL
5. Store the password for a URL
6. Remove a URL and password

Command and Response APDUs:

1. Verify PIN:
   Command APDU:
   CLA: any
   INS: 0x00
   P1: --
   P2: --
   CDATA: PIN data
   Response APDU:
   --
   Status Words:
   9000 PIN correct
   6A80 PIN incorrect
   9840 PIN blocked
2. Update PIN:
   Command APDU:
   CLA: any
   INS: 0x02
   P1: --
   P2: --
   CDATA: New PIN data
   Response APDU:
   --
   Status Words:
   9000 PIN successfully updated
   6982 Not authorized
3. Unblock PIN:
   Command APDU:
   CLA: any
   INS: 0x04
   P1: --
   P2: --
   CDATA: PUK data
   Response APDU:
   --
   Status Words:
   9000 PIN successfully unblocked
   9808 PIN not blocked
   6A80 PUK incorrect

4. Get Password:
   Command APDU:
   CLA: any
   INS: 0x10
   P1: --
   P2: --
   CDATA: URL data
   Response APDU:
   Password data (if SW == 9000)
   Status Words:
   9000 Password found
   6A82 Password not found
   6982 Not authorized
5. Store Password:
   Command APDU:
   CLA: any
   INS: 0x12
   P1: --
   P2: --
   CDATA: Lu (1 byte) | URL (Lu bytes) | Lp (1 byte) | Password (Lp bytes)
   Response APDU:
   --
   Status Words:
   9000 Password successfully stored
   6983 Password for this URL already stored
   6A84 Memory full
   6982 Not authorized
6. Remove Password:
   Command APDU:
   CLA: any
   INS: 0x14
   P1: --
   P2: --
   CDATA: URL data
   Response APDU:
   --
   Status Words:
   9000 Password successfully removed
   6A82 Password not found
   6982 Not authorized

The sample implementation uses a simply linked list of “Password Entry” objects which hold two byte arrays, for URL and password data respectively. This list is searched every time a password is requested. I rely completely on the JCRE garbage collection to reclaim space after deletion of an entry. Authorization is performed using two OwnerPIN objects, one for PIN and one for PUK. The implementation supports both an RMI and a traditional APDU interface. The RMI interface is in Vault.java.