The purpose of this TP is to develop a Javacard applet that provides similar functionality than a Debit/Credit Card

Use Case

The payment, in an EMV payment ecosystem, is done following this steps (this is a high level description flow):

- The merchant 'enter' the amount to pay into a Payment Terminal
- The payee insert its card in the payment terminal
- The terminal ask the PIN to end user
- The terminal send a command to the card to verify the PIN
- The terminal send a command to the card to 'sign' the transaction,
    - the transaction is represented by a set of information (transaction identifier, merchant identifier, transaction amount, …)
    - The signature is computed using a secret key (symmetric 3DES encryption) known by the payee's bank authorization server and provision in the card, over transaction information
- The terminal send a payment authorization request that is routed through
    - the bank of the merchant
    - the payment network
    - the bank of the payee
- The bank verify the signature against
    - the information of the transaction
    - the payee's card secret key
- The bank verify also if the payee has enough funds in its account
- The bank return an authorization response to the payment terminal through bank/network
- The payment terminal display the merchant the result of the authorization
- Usually, the 'money' is transfer from the payee bank account to the merchant bank account at the end of the day

So, during the TP, you will develop an applet **'PaymentApplet'** that perform the signature computation over the transaction information data in order to let the payment terminal building the payment request.

This applet provides two APDU command:

- 'Personalized' : This command aims to be used in factory, it is used to:
    - Inject the Card signature secret key
    - Set initial PIN value
- 'Compute Signature' : to compute the payment signature

Step 1: Create a new Javacard Project using Netbeans

Applet Design

Here is the different APDU command to develop in order to build our Debit/Credit card applet

Step 2 : implement Personalized Applet APDU command

It used to 'configure' the applet with

- A bank account number (in EMV it is a **P**rimary **A**ccount **N**umber or PAN)
- An initial PIN value
- The PaymentKey used to compute signature
- The initial counter value (cf Compute Signature apdu command bellow)

APDU Header

Parmater	size	Description
CLA	1	A0
INS	1	20
P1	1	00
P2	1	00
LC	1	
Data	Variable	Cf bellow

Data In:

Parmater	size	Description
PIN length	1	Length of the PIN
PIN	PIN length	Value of the PIN to set
Max Amount	2	Maximum transaction amount
accountId	16	User account identifier
paymentKey	16	3DES key 16 bytes length key use to compute payment transaction cryptogram
Counter	2	Initial value of the transaction counter

Data Out: none

- Control to be performed:
  - The applet is not already personalized
  - PIN length is below 8 digits
- Design tips:
  - To create the key, take a look at the class javacard.security.KeyBuilder
  - The type of the key is 'DES' with a length 16 bytes (so the equivalent of two DES KEY)

Test APDU :

0xA0 0x20 0x00 0x00 0x29 0x04 0x31 0x32 0x33 0x34 0x00 0x14 0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x10 0x20 0x30 0x40 0x11 0x21 0x31 0x41 0x12 0x22 0x32 0x42 0x13 0x23 0x33 0x43 0x10 0x23 0x7F;

Step 3 : implement Compute Signature APDU command

Suring a payment, the terminal:

- Generate a challenge (random value)
- Prepare information about the payment transaction
- Send the details and challenge in the Compute signature command

Parmater	size	Description
CLA	1	A0
INS	1	22
P1	1	00
P2	1	00
LC	1	
Data	Variable	Cf bellow

Data In:

Parmater	Size	Description
transactionId	4	Payment Transaction Unique Identifier
Amount	2	Payment Transaction amount
merchantId	1	Indentifier of the merchant where the purchase is done
Date	3	Date if the purchase, the 3 byte of the date represent

Byte 1	Byte 2	Byte 3
Day	Month	Year since 2000

paymentTerminalId	4	Serial number of the payment terminal
Country	2	Country code / ISO 3166 alpha 2 country code
terminalChallenge	16	Random value generated by the payment terminal
PIN length	1	Length of the PIN
PIN	PIN length	Value of the PIN to set

Data Out:

Parmater	Size	Description
Counter	2	Transaction counter value
accountId	16	User account Identifier
cryptogram	8	Cryptogram value

- Security verification:
  - Amount is below mx amount transaction value
  - PIN verification is ok
  - Date is up or equal to the previous transaction date (if any)

Cryptogram computation :

- Use the javacard.security.Signature class and initialize a signature object with a 8 Bytes MAC computation with ISO9797M2 data padding.
- The signature is computed other the following concatenated data: transactionId + amount + merchantId + Date + paymentTerminalId + country + terminalChallenge + counter + accountId
- The Cryptogram/Signature is computed using the 3DES MAC algorigthm using the PaymentKey and the Transaction Data
- After each transaction, the counter is increased of 1

MAC 3DES Computation

The MAC 3DES is a cryptogram computed using the 3DES ciphering algorithm, you are using it everyday when you pay with your 'CB/Credit Card' !

Test:

0xA0 0x22 0x00 0x00 0x25 0x94 0x95 0x57 0x50 0x00 0x0E 0X55 0x11 0x02 0x17 0xAA 0xAA 0xAA 0XAA 0x55 0x53 0x63 0x55 0x59 0x53 0xF0 0xD6 0x32 0x17 0x80 0xE9 0xC8 0x80 0x0F 0x37 0x1A 0x18 0x04 0x31 0x32 0x33 0x34 0x7F;

```
// MUST RETURN : 10, 23, 00, 11, 22, 33, 44, 55, 66, 77, 00, 11, 22, 33, 44, 55, 66, 77, 68, 09, eb, 2d, a3, 1a, af, a1


0xA0 0x22 0x00 0x00 0x25 0x94 0x95 0x57 0x50 0x00 0x0E 0X55 0x11 0x02 0x17 0xAA 0xAA 0xAA 0XAA 0x55 0x53 0x63 0x55
0x59 0x53 0xF0 0xD6 0x32 0x17 0x80 0xE9 0xC8 0x80 0x0F 0x37 0x1A 0x18 0x04 0x31 0x32 0x33 0x34 0x7F;

// MUST RETURN : 10, 24, 00, 11, 22, 33, 44, 55, 66, 77, 00, 11, 22, 33, 44, 55, 66, 77, 84, d6, 71, a3, 22, c6, 8b, 7f
```