APDU Interface of the SigAnima JavaCard applet

Version 1.0 RC1
August 6th, 2012

1. Introduction

This document describes the features and the APDU interface of the SigAnima JavaCard applet. This applet and documentation are based on the javacardsign applet from Wojciech Mostowski (http://sourceforge.net/projects/javacardsign/).
The applet has been developed according mainly to the ISO7816 specification, for information not included here please refer to [1], [2], or contact the author. For this applet a certain small selection of options from the ISO standard has been implemented, as explained below. The source code of the applet and the host library is available at: https://github.com/tsenger/SigAnima

2. Applet Specifications

The current version of the applet and the host library implements the following features:

- An ISO7816 file system for storing PKI files. The applet support hierarchical file system including relative to current file selection or selection by path. Reading of each file can be user PIN protected.
- PIN and PUK user authentication: a 4 to 10 characters long PIN code, and a 10 characters long PUK code. The PUK code lets the user change and unblock a forgotten PIN code.
- The applet does not support any kind of secure messaging for APDU communication.
- Signing cryptographic operation with ECC GF(p) curves. The supported key length is up to 320 bit depending on underlying Java Card implementation. Signing (perform security operation command) uses a plain ECDSA signature. No hashes will be calculated before the signing inside the card. The signing function will pad data to sign with leading zero up to the size of the public key if the data is shorter then the public key size. If the data to sign is bigger then the public key, the data will be truncated to the size of the public key.
  The Java Card API involved is *SignatureX.ALG_ECDSA_PLAIN* which is currently only available in cards with JCOP v2.4.1 R2 or later. The result of the signing operation is the ECDSA signature in the following ASN.1 format.

```
ECDSA-Signature ::= SEQUENCE {
  r   INTEGER,
  s   INTEGER }
```

- The AID of the applet is chosen to be `0xD2760001324543534947`. The applet does not support/provide any FCI information on applet/file selection. It is recommended that the SigAnima applet is made default selectable on the card.

- Only during the personalization phase, the on-card key generation is possible. Currently, the applet can generate EC keys up to 320 bits.
- The applet supports the following standardized EC domain parameters:
    - secp224r1
    - BrainpoolP224r1
    - secp256r1
    - BrainpoolP256r1
    - BrainpoolP320r1

3. APDU Interface

Below the APDUs that the applet supports are described.

3.1. Initialization

After a fresh applet is loaded onto the card, the suggested initialization sequence is the following:

1. Select the applet.
2. Set the state of the applet to initial. Applet is in personalization mode.
3. Optionally, change the historical bytes of the card through the applet. For this the applet needs to be default selectable.
4. Load up the file structure information. Note: this step does not create any files, only provides the intended file structure information.
5. Load up the RSA private keys and their identifiers to the card. Three keys are expected in total: for signing, decryption, and authentication.
6. Alternatively to the last point, only the key identifiers are loaded onto the card, and then the card is asked to generate the private keys. The corresponding public keys are returned in response to key generation command.
7. Create the contents of all the ISO7816-15 structures intended to be on the card (including user certificates matching the keys and CA certificate), create the corresponding files in the applet and load up the file contents to the card.
8. Upload the PUK to the card. The PUK is unchangeable.
9. Set the state of the applet to prepersonalized.
10. At this stage the personalization should be finished by setting up the user PIN (PUK has to be provided). This will put the card into the personalized state. From this point on the personalized state of the applet is not changeable.

The APDUs needed for that are described in the following. Personalization APDUs are only available in the initial state. On error conditions the APDUs may return a variety of abnormal termination status words. On success the response APDU is always `0x9000` with no data, with the notable exception of the key generation command, where the public key data is returned. During the personalization phase no status words other than `0x9000` should be accepted.

3.1.1. Select Applet

CLA	INS	P1	P2	Lc	Data	Le
`0x00`	`0xA4`	`0x04`	`0x00`	`0x0A`	`0xD2 0x76 0x00 0x01 0x32 0x45 0x43 0x53 0x49 0x47`	–

3.1.2. Put Data – Set Applet State

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xDA	0x68	State	0x00	–	–

The P2/state can be one of: 0x01 initial, 0x02 prepersonalised. The state is set to personalised (0x03) implicitly by the change reference data command when setting the user PIN.

3.1.3. Put Data – Upload File System Structure Information

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xDA	0x69	0x00	var	Data with file structure information	–

The data field in this APDU contains the file structure information according to the following format. It is a list of concatenated single file information byte strings. For DF file the following bytes should be present:

b_1	b_2	b_3	b_4	b_5	b_6 .. b_n
80	FID_{MSB}	FID_{LSB}	parent index	#children	children indexes

The indexes (parent and children) are relative to the beginning of the whole data field. For EF files the format is this:

b_1	b_2	b_3	b_4	b_5
00	FID_{MSB}	FID_{LSB}	parent index	SFI

The SFI byte should be 00 if no SFI is provided for the file. And example of a valid file system structure information is this (quote from Java code):

```
byte[] fileStructure = {
 -1, // DF
 0x3F, 0x00, // FID, MF
 -1, // no parent
 2, 7, 12, // two children at indexes 7 and 12
 0, // EF
 0x2F, 0x00, // FID, EF.DIR
 0, 0x1E, // parent at index 0, SFI is 1E
 -1, // DF
 0x50, 0x15, // FID, DF.CIA
 0, // parent at index 0
 9, 26, 31, 36, 41, 46, 51, 56, 61, 66,
 // 9 children
 0, // EF
 0x50, 0x32, // FID, EF.CIAInfo
 12, 0x12, // parent at index 12, SFI is 12
 0, 0x50, 0x31, 12, 0x11, // EF.OD
 0, 0x42, 0x00, 12, 0x00, // EF.AOD
 0, 0x40, 0x00, 12, 0x00, // EF.PrKD
 0, 0x41, 0x00, 12, 0x00, // EF.CD
 0, 0x41, 0x01, 12, 0x00, // EF.CACert
 0, 0x41, 0x02, 12, 0x00, // EF.UserCert1
```

```
  0, 0x41, 0x03, 12, 0x00, // EF.UserCert2
  0, 0x41, 0x04, 12, 0x00, // EF.UserCert3
};
```

3.1.4. Put Data – Set Historical Bytes

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xDA	0x67	0x00	var	Historical bytes	–

3.1.5. Manage Security Environment – Prepare Key Generation

This command is used to prepare the upcoming key generation command.

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x22	0x41	0xB6	0x06	Key ID / Domain Parameter ID	–

The key identifier object is a simple TLV structure with tag 0x84 and length 0x01. The value of this TLV structure contains the key identifier byte. There three slots for key pair. The key identifier is simply the index of the key pair and must be a value between 0x00 and 0x02.
The domain parameter identifier is a simple TLV structure with tag 0x80 and length 0x01. The value of this TLV structure contains the ID of the standardized domain parameters as defined in [3]. Valid values are:

- 0x0A for secp224r1
- 0x0B for BrainpoolP224r1
- 0x0C for secp256r1
- 0x0D for BrainpoolP256r1
- 0x0E for BrainpoolP320r1

3.1.6. Generate Asymmetric Key Pair

This command is used to perform the actual key generation. The private key selected with the MSE command will be regenerated. The corresponding public key is returned in the response APDU.

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x46	0x80	0x00	–	–	–

The response APDU is a simple TLV structures with tag 0x86 which contains the uncompressed EC public key as value.

3.1.7. Create File

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xE0	0x00	0x00	0x05	FID, File length, Permission	–

The data field has the following format:

b_1	b_2	b_3	b_4	b_5
FID_{MSB}	FID_{LSB}	Len_{MSB}	Len_{LSB}	Permission

Where “Permission” indicates whether the file should be PIN protected (0x01) for reading or free tor read (0x00).

3.1.8. Select File

Absolute, by file identifier FID (if none provided, selects MF):

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xA4	0x00	0x00	0x00 / 0x02	FID_{MSB}, FID_{LSB}, or absent	0x00

DF (P1 is 0x01) or EF (P1 is 0x02) under the currently selected file, by file identifier FID:

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xA4	0x01 / 0x02	0x00	0x00	FID_{MSB}, FID_{LSB}	0x00

Parent of the current DF:

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xA4	0x03	0x00	0x00	–	0x00

By path from the current DF (P1 is 09) or MF (P1 is 08):

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xA4	0x09 / 0x08	0x00	var	The file path	0x00

3.1.9. Write Binary

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xD0	Off_{MSB} or SFI	Off_{LSB}	var	Data to be written to the file	–

The offset should to be less or equal than 0x7FFF (most significant bit set to 0). If the most significant bit of P1 is set to 1, then the remaining bits of P1 indicate SFI, in which case P2 indicates the offset.

3.1.10. Change Reference Data – Set PUK

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x24	0x01	0x00	0x0A	PUK data	–

This form of this command is only available when the applet is in the initial state. The data field should contain the ASCII bytes of the PUK (i.e. from the range 0x30 .. 0x39). The PUK length should always be 10.

3.1.11. Change Reference Data – Set PIN

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x24	0x00	0x00	0x0E .. 0x14	PUK data, PIN data	–

The data field should contain the ASCII bytes of the PUK and PIN (i.e. from the range 0x30 .. 0x39). The PUK length should always be 10, the PIN length should be between 4 and 10. If not already in the personalized state the applet state is changed to personalized after successful

execution of this command.

3.2. Communication after Personalization

After the personalization the following set of ISO7816 commands is available for regular applet operation. If not stated otherwise all commands shall return a status word (`0x9000` on success) without response data.

3.2.1. Select Applet

See 3.1.1

3.2.2. Select File

See 3.1.8

3.2.3. Change Reference Data – Change PIN

See 3.1.11

3.2.4. Read Binary

CLA	INS	P1	P2	Lc	Data	Le
0x00	0xB0	Off_{MSB} or SFI	Off_{LSB}	0x00	-	Response length

The offset should to be less or equal than `0x7FFF` (most significant bit set to 0). If the most significant bit of P1 is set to 1, then the remaining bits of P1 indicate SFI, in which case P2 indicates the offset.
This command will return the number of bytes as specified by the Le field in the command APDU. If Le bytes is set to `0x00` all available bytes up to the maximum response APDU size will be returned and the status word will be `0x6282` if the end of the file is reached.

3.2.5. Verify PIN

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x20	0x00	0x00	0x04 .. 0x0A	PIN data	-

The data field should contain the ASCII bytes of the PIN (i.e. from the range `0x30` .. `0x39`). If the PIN was incorrect the status word in response will be `0x63Cx` where x will be the remaining tries.

3.2.6. Manage Security Environment

This command is used to prepare the upcoming crypto operation (Perform Security Operation or Internal Authenticate). Just prior to the actual crypto operation PIN verification is required (every time).

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x22	0x41	0xB6	var	Key Identifier DO	-

The key identifier object is a simple TLV structure with tag `0x84` and length `0x01`. The value of this TLV structure contains the key identifier byte. There three slots for key pair. The key identifier is the index of one of the key pairs and must be a value between `0x00` and `0x02`.

3.2.7. Perform Security Operation – Sign

This command requires prior successful Manage Security Environment and Verify PIN commands.

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x2A	0x9E	0x9A	var	Data to be signed	–

The data field may contain the hash value of the data to be signed. The sign command will sign what ever it gets in the data field. It just call the JCOP signature function *SignatureX.ALG_ECDSA_PLAIN*. No hashes will be calculated before the signing inside the card. The signing function will pad data with leading zero up to the size of the public key if the data is shorter then the public key size. If the data is bigger then the public key, the data will be truncated to the size of the public key (most significant bytes will be cut off).

3.2.8. Get Challenge

CLA	INS	P1	P2	Lc	Data	Le
0x00	0x84	0x00	0x00	–	–	Challenge length

This command returns a random challenge from the card of the requested length. This command uses a instance of *RandomData.ALG_SECURE_RANDOM*. Currently this command has no interaction with any other cryptographic commands in the applet and is implemented for possible future use.

I. References

[1] ISO/IEC, Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. Technical report, ISO 7816-4., 2005
[2] ISO/IEC, Identification cards – Integrated circuit cards – Part 8: Commands for security operations. Technical report, ISO 7816-8, 2004
[3] Bundesamt für Sicherheit in der Informationstechnik, Advanced Security Mechanisms forMachine Readable Travel Documents, Part 3 – Common Specifications, 2012