The APDU Interface of the PKI Applet

Wojciech Mostowski
Radboud University Nijmegen
Department of Computing Science
Nijmegen, the Netherlands
e-mail: woj@cs.ru.nl

August 10, 2009

1 Introduction

This document briefly describes the features and the APDU interface of the Java Card PKI applet. The applet has been developed according to the ISO7816 specification, for information not included here please refer to [1, 2, 3], 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 from the SourceForge SVN repository at:

`https://javacardsign.svn.sourceforge.net/svnroot/javacardsign`

The SourceForge address of the PKI project is `http://javacardsign.sourceforge.net`. The current code in the project has been developed mainly by Wojciech Mostowski, `woj@cs.ru.nl`.

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 according to the Part 15 of the ISO7816 specification [3]: private key directory, certificate directory, CA and user certificates, etc. It is up to the personalisation software what files will be stored in the applet. 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 PUC user authentication: a 4–20 characters long PIN code, and a 16 characters long PUC code. The PUC code lets the user to unblock a forgotten PIN code.
- The applet does not support any kind of secure messaging for APDU communication.
- The applet only communicates on the contact interface of the card.
- Three different cryptographic operations: signing, decryption, and authentication. Currently, the supported key type and length is RSA 1024 bit.[1] The supported ciphers are the following:

[1] This is an "artificial" limitation. The applet could support keys as long as the underlying Java Card implementation does, but the current personalisation APDU interface limits the amount of data when loading up private keys, which in effect limits the key size to 1024 bits. This limitation will hopefully be lifted in future versions of the applet.

- for signing (perform security operation command): RSA signature with PKCS 1.5 padding and SHA1 or SHA256 digests and RSA signature with PSS padding with SHA1 digest. In all the cases the hashes have to provided ready to the card and in correct format, see APDU interface below, i.e. the card does not do the hashing, as stipulated by the ISO7816-8 [2] specification.[2] The result of the signing operation is the RSA signature. The corresponding Object Identifiers for the supported algorithms are:
  - OID RSA SHA1 = 1.2.840.113549.1.1.5
  - OID RSA SHA256 = 1.2.840.113549.1.1.11
  - OID RSA PSS = 1.2.840.113549.1.1.10
  - OID SHA1 = 1.3.14.3.2.26
  - OID SHA256 = 2.16.840.1.101.3.4.2.1

  The Java Card API involved is:
  - `Cipher.ALG_RSA_PKCS1` `Cipher.ALG_RSA_NOPAD`
- for decryption (perform security operation command): RSA cipher with PKCS 1.5 padding. The input for this operation is a valid (i.e. properly formatted and with matching length) encrypted RSA block. The result is the decrypted plain text. The corresponding Object Identifier for this operation is:
  - OID RSA = 1.2.840.113549.1.1.1

  The Java Card API involved is:
  - `Cipher.ALG_RSA_PKCS1`
- for authentication (internal authenticate command): RSA cipher with PKCS 1.5 padding. The input for this operation is an arbitrary plain text with in the limits of the supported key length. The output is the encrypted RSA block. The corresponding Object Identifier for this operation is:
  - OID RSA = 1.2.840.113549.1.1.1

  The Java Card API involved is:
  - `Cipher.ALG_RSA_PKCS1`

- The AID of the applet is chosen to be `A000000063504B43532D3135`. The applet does not support/provide any FCI information on applet/file selection. It is expected that all such information required for the proper functioning of the host side application is stored in the file system in the form of ISO7816-15 structures (EF.DIR, EF.CIAInfo, EF.OD, EF.CD, etc.). The host library and application provides an example of initialisation of such structures and uploading them to the card. It is also suggested that the PKI is made default selectable on the card.

- During the personalisation phase, the on-card key generation is also possible. Currently, the applet can only generate RSA 1024 bit keys.

3 APDU Interface

Below the APDUs that the applet supports are briefly described.

[2]Technical remark: because the hashes are provided ready to the card, the Java Card API `Signature` API could not be used. In turn, the PSS padding is done by a manually implemented method in the applet and involves the use of `MessageDigest.ALG_SHA`.

3.1 Initialisation

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

1. Select the applet.
2. Set the state of the applet to initial. Applet is in personalisation 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 PUC to the card. The PUC is unchangeable.
9. Set the state of the applet to prepersonalised.
10. At this stage the personalisation should be finished by setting up the user PIN (PUC has to be provided). This will put the card into the personalised state. From this point on the personalised state of the applet is not changeable.

The APDUs needed for that are described in the following. Personalisation 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 9000 with no data, with the notable exception of the key generation command, where the public key data is returned. During the personalisation phase no status words other than 9000 should be accepted.

3.1.1 Select Applet

CLA	INS	P1	P2	Lc	Data	Le
00	A4	04	00	0B	AID=A000000063504B43532D3135	Absent

3.1.2 Put Data – Set Applet State

CLA	INS	P1	P2	Lc	Data	Le
00	DA	68	State	00	Absent	Absent

The P2/state can be one of: 01 initial, 02 prepersonalised. The state is set to personalised 03 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
00	DA	69	00	Var	Data with file structure information	Absent

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	$b_6 \ldots b_n$
80 (−1)	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
00	DA	67	00	Var	Historical bytes	Absent

3.1.5 Put Data – Setup Private Key Identifier

CLA	INS	P1	P2	Lc	Data	Le
00	DA	61/62/63	00	Var	Key identifier bytes	Absent

The P1 byte indicates correspondingly: authentication key, signing key, decryption key. The data field contains a key identifier, maximum length 16 bytes.

3.1.6 Put Data – Upload Private Key

CLA	INS	P1	P2	Lc	Data	Le
00	DA	64/65/66	Key part	Var	Corr. key part raw data	Absent

The P1 byte indicates correspondingly: authentication key, signing key, decryption key. The P2 byte indicates the part of the raw data of the key in CRT format in the data field, in the following way:[3]

Tag	Key part
81	public modulus
82	public exponent
83	prime number P
84	prime number Q
85	prime exponent P
86	prime exponent Q
87	CRT coefficient

3.1.7 Manage Security Environment – Prepare Key Generation

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

CLA	INS	P1	P2	Lc	Data	Le
00	22	41	A4/B6/B8	Var	Key Identifier DO	Absent

The meaning of P2 is the following:

P2	Security Operation
A4	Authentication
B6	Signing
B8	Decryption

The key identifier object is a simple TLV structure with tag 84. The data field of this TLV structure contains the key identifier bytes. The key identifiers have to be already known, see Put Data 3.1.5.

3.1.8 Generate Asymetric Key Pair

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

CLA	INS	P1	P2	Lc	Data	Le
00	46	80	00	Absent	Absent	Absent

The response APDU contains two simple TLV structures containg the RSA public key modulus and exponent, respectively:

81	Public modulus
82	Public exponent

3.1.9 Create File

CLA	INS	P1	P2	Lc	Data	Le
00	E0	00	00	05	File FID, File length, PIN	Absent

The data field has the following format:

FID_{MSB}	FID_{LSB}	Len_{MSB}	Len_{LSB}	PIN byte

where PIN byte indicates whether the file should be PIN protected when reading, 01 means PIN protect, 00 means free read.

[3] It is my intention to simplify this key loading to use just the modulus and private exponent, i.e. the non-CRT form of the key. This will not be part of the current version of the applet though.

3.1.10 Select File

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

CLA	INS	P1	P2	Lc	Data	Le
00	A4	00	00	02/00	FID_{MSB} FID_{LSB} or Absent	00

DF (P1 is 01) or EF (P1 is 02) under the currently selected file, by file identifier *FID*:

CLA	INS	P1	P2	Lc	Data	Le
00	A4	01/02	00	02	FID_{MSB} FID_{LSB}	00

Parent of the current DF:

CLA	INS	P1	P2	Lc	Data	Le
00	A4	03	00	00	Absent	00

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

CLA	INS	P1	P2	Lc	Data	Le
00	A4	08/09	00	Var	The file path	00

3.1.11 Write Binary

CLA	INS	P1	P2	Lc	Data	Le
00	D0	Off_{MSB} or SFI	Off_{LSB}	Var	Data to be written to the file	Absent

The offset should to be less or equal than 7FFF (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.

3.1.12 Change Reference Data – Set PUC

CLA	INS	P1	P2	Lc	Data	Le
00	24	01	00	10	PUC data	Absent

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 PUC number (i.e. from the range 30...39).

3.1.13 Change Reference Data – Set PIN

CLA	INS	P1	P2	Lc	Data	Le
00	24	00	00	14...24	PUC data, PIN data	Absent

The data field should contain the ASCII bytes of the PUC and PIN numbers (i.e. from the range 30...39). The PUC length should be always 16, the PIN length should be between 4 and 20. If not already in the personalised state the applet state is changed to personalised by this command.

3.2 Communication after Personalisation

After the personalisation the following set of ISO7816 commands is available for regular applet operation. In the following, if not stated otherwise the commands return a status word (9000 on success), and no response data.

3.2.1 Select Applet

See 3.1.1.

3.2.2 Select File

See 3.1.10.

3.2.3 Change Reference Data – Change PIN

See 3.1.13.

3.2.4 Read Binary

CLA	INS	P1	P2	Lc	Data	Le
00	B0	Off_{MSB} or SFI	Off_{LSB}	00	Absent	Response length

The offset should to be less or equal than 7FFF (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 specified by the Le field in the command APDU. If there is no Le bytes available in the file, the available bytes will be returned instead and the status word of the response APDU will be 6282 (End of File).

3.2.5 Verify PIN

CLA	INS	P1	P2	Lc	Data	Le
00	20	00	00	04...14	PIN data	Absent

The data field should contain the ASCII bytes of the PIN number (i.e. from the range 30...39).

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
00	22	41	A4/B6/B8	Var	Key Identifier DO, Algorithm Identifier DO	Absent

The meaning of P2 is the following:

P2	Security Operation
A4	Authentication
B6	Signing
B8	Decryption

The data objects are formed as follows. The key identifier object is a simple TLV structure with tag 84. The data field of this TLV structure contains the key identifier bytes. The algorithm identifier object is a simple TLV structure with tag 80. The data field of this TLV structure contains one byte corresponding to the crypto algorithm. Usually, the information about possible values for this byte and corresponding algorithms is included in the EF.CIAInfo file on the card.

3.2.7 Internal Authenticate

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

CLA	INS	P1	P2	Lc	Data	Le
00	88	00	00	Var	The raw data to be signed/encrypted	Absent

The maximum length of the command data is limited by the APDU length (255), the length of the RSA key to be used and the padding algorithm (key size in bytes minus 11). On success, the data in the response APDU is the RSA cipher block.

3.2.8 Perform Security Operation – Sign

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

CLA	INS	P1	P2	Lc	Data	Le
00	2A	9E	9A	Var	Hash to be signed	Absent

Depending on the signing algorithm established by the prior MSE command, the data field is one of the two:

- A matching (SHA1 or SHA256) ASN.1 wrapped hash data object for RSA PKCS 1.5 signing algorithm.
- Raw SHA1 hash data for RSA PSS signing algorithm.

On success, the data in the response APDU is the RSA signature block.

3.2.9 Perform Security Operation – Decipher

This command requires prior successful Manage Security Environment and Verify PIN commands. This is the only command that supports chaining (CLA=10).

CLA	INS	P1	P2	Lc	Data	Le
00/10	2A	80	82/84/86	Var	The cipher block to be decrypted	Absent

The value of P2 can be any of the three, it does not influence the operation of this command. The data field is the (part of) cipher block to be decrypted. If the command is not last in the sequence the response APDU is simply the status word. After the last command (CLA=00), on success, the data in the response APDU is the decrypted plain text without padding.

3.2.10 Get Challenge

CLA	INS	P1	P2	Lc	Data	Le
00	84	00	00	00	Absent	Challenge length

This command returns a random challenge from the card of the required length. Currently this command has no interaction whatsoever with any other crypto commands in the applet and is implemented for possible future use.

References

[1] ISO/IEC. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. Technical report, 2005. ISO 7816-4.

[2] ISO/IEC. Identification cards – Integrated circuit cards – Part 8: Commands for security operations. Technical report, 2004. ISO 7816-8.

[3] ISO/IEC. Identification cards – Integrated circuit cards – Part 15: Cryptographic information application. Technical report, 2004. ISO 7816-15.