Author: rupp,scd
Project:SafeSign
Date: 03/15/12

Reverse-Engineering of SafeSign Applet

Index

Figures

This document describes the work carried on to reverse the safesign applet

Introduction.

We describes the process we use to reverse-Engineer the applet component of AET's SafeSign product.

We reverse the applet against the following documentation:

- Programmers Guide, SafeSign, Version 2.0/Status 05.07.2004
- MICROSOFT_TAU_Guide_SafeSign_v1.2_28940.pdf
- Microsoft Corporation. *Cryptography Reference*
- RSA Laboratories. *PKCS 15*
- RSA Laboratories. *PKCS #11*
- SafeSign Identity Client Middleware White Paper

1. Analysis of Initialization performed by the SafeSign Middleware (APDU Sniffing)

Component to be debugged/sniffed when communicating with the applet:

- winscard.dll (PCSC)

1.1. Deciphering of SCP02 channel

We Capture the Apdu trace during SafeSign Initialization of a Jcop Card.These APDUS are secure-channel SCP02 encrypted. Using the card javacard default key set and analysis of External Authenticate commands, we reconstruct the 3DES session keys:

we are in secure channel mode 2 SCP 02 (chapter E of GP specs)

In the log file :

init-update: 80 50 00 00 08 77 DF F6 6B 1A 1A C2 68
card answer: 00 00 10 77 03 85 47 95 53 12 01 02 00 49 E5 6D F3 D3 97 00 F7 60 84 50 0D 8B 9A 67 90 00

We extract (GP specs for SCP02):

CARD DIVERSIFICATION DATA=00 00 10 77 03 85 47 95 53 12
KEY INFO=01 02 (SCP Versions)
SEQUENCE COUNTER=00 49
CARD CHALLENGE=E5 6D F3 D3 97 00
CARD CRYPTOGRAM=F7 60 84 50 0D 8B 9A 67
HOST CHALLENGE=77 DF F6 6B 1A 1A C2 68

External Authenticate

SCardTransmit (handle 0xEA040001):
transmitted:
84 82 03 00 10 81 AA 1D E5 6A 91 3C 45 E5 4A 14 B5 41 44 AD FE
received:
90 00

P1=03 means C-DECRYPTION and C-MAC

We extract (GP Specs)

HOST CRYPTOGRAM=81 AA 1D E5 6A 91 3C 45 E5 4A 14 B5 41 44 AD FE

CARD S-ENC= 404142434445464748494A4B4C4D4E4F
CARD S-MAC= 404142434445464748494A4B4C4D4E4F
CARD DEK= 404142434445464748494A4B4C4D4E4F

session keys are generated using 3DES in CBC mode

the session encryption key is obtained by applying 3DES CBC encryption to a 16 byte derivation data. The key used for this ciphering is the static S-ENC key

the 16 byte derivation data is obtained by concatenation of a 2 bytes constant , a 2 bytes sequence counter and 12 bytes of 0's

---
u2 constant
u2 sequence counter ===> CBC 3DES ENCRYPTION WITH DEK/S-ENC KEY ==> SESSION ENCRYPTION KEY

u14 0-pad
---

to get the session key we must therefore get the 16 byte derivation data :

the 4 session keys used by the secure channel are computed from the following requirements :

Generating the Secure Channel C-MAC session keys using the Secure Channel base key or MAC key (S-MAC) and the session keys derivation data with a constant of '0101',

• Generating the Secure Channel R-MAC session keys using the Secure Channel base key or MAC key (S-MAC) and the session keys derivation data with a constant of '0102',

• Generating the Secure Channel encryption session keys using the Secure Channel base key or encryption
key (S-ENC) and the session keys derivation data with a constant of '0182',

• Generating the Secure Channel data encryption session keys using the Secure Channel base key or data
encryption key (DEK) and the session keys derivation data with a constant of '0181'.

our derivation data is therefore: 01 82 00 49 00 00 00 00 00 00 00 00 00 00 00 00

the key we will use for the session encryption key is the card S-ENC :
404142434445464748494A4B4C4D4E4F
in our case, the icv used for 3-DES CBC should made with 0's

we get a session key of **1C541E904AB4A28759F4E3F911CACEA2**

1.2. Applet Installation

a) GLOBALPLATFORM Install for Load (card manager)

84 E6 02 00 20

06 A0 00 00 00 63 02 08 A0 00 00 00 03 00 00 00 00 00 00 [80 00 00 00 00 05 1C 77 1D FE 0B 33 E7]

Mandatory 1 Length of Load File AID

06
Mandatory 5-16 Load File AID
A0 00 00 00 63 02
Mandatory 1 Length of Security Domain AID
08
Conditional 0-16 Security Domain AID
A0 00 00 00 03 00 00 00
Mandatory 1 Length of Load File Data Block Hash
00
Conditional 0-n Load File Data Block Hash
Mandatory 1 Length of load parameters field
00
Conditional 0-n Load parameters field
Mandatory 1 Length of Load Token
00
Conditional 0-n Load Token

b) GLOBALPLATFORM Load (card manager)

Load blocks of the Cap Components (from the jcf file)

c) GLOBALPLATFORM Install for Install (card manager)

84 E6 0C 00 40

37411EF1332D35F1E8905145E92A275C2A32A91CC931173C88A35771AF59C40867A1BE5BDAE
EB0E39E12A1ED8779EEDDA2A21191140123AB7A3A9CEB401423B4

key :
1C541E904AB4A28759F4E3F911CACEA2

INSTALL FOR INSTALL IN CLEAR

06 A0 00 00 00 63 02 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 01 00 0C C9 0A 10 77 03 85 47 95 53 12 C1 04 00 [80000000000000E8B200F948FF4C7F]

Mandatory 1 Length of Executable Load File AID
06
Mandatory 5-16 Executable Load File AID
A0 00 00 00 63 02
Mandatory 1 Length of Executable Module AID
0C
Mandatory 5-16 Executable Module AID
A0 00 00 00 63 50 4B 43 53 2D 31 35
Mandatory 1 Length of Application AID

0C
Mandatory 5-16 Application AID
A0 00 00 00 63 50 4B 43 53 2D 31 35
Mandatory 1 Length of Application Privileges
01
Mandatory 1 Application Privileges
00
Mandatory 1 Length of install parameters field
0C
Mandatory 2-n Install parameters field
C9 0A 10 77 03 85 47 95 53 12 C1 04
Mandatory 1 Length of Install Token
00
Conditional 0-n Install Token

1.3. Post-Install Initialization

These commands are executed once the applet is loaded and registered (e.g. when the applet is instantiated)

a) GLOBALPLATFORM *Safesign Pkcs15 Applet selection*

00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00

b) ISO-7816-4 Get Data command (Applet)

→ 00 CA 01 01 0A

← C2 08 19 C1 04 06 01 0D 00 00 90 00

This is ISO7816 -4 Get Data (not GP)

Get Data for Tag 0x0101.

CLA *As defined in ISO7816 -4 5.4.1*
INS *'CA'*
P1-P2 0101
Lc field *Empty*
Data field *Empty*
Le field 0A

Tag 0101 is in the range of application proprietary data.

Unknown answer

c) *ISO-7816-4 Get Data command (Applet)*

→ 00 CA 01 00 08

← 10 77 03 85 47 95 53 12 90 00

This is ISO7816 -4 Get Data (not GP)

Get Data for Tag 0x0100.

CLA *As defined in ISO7816 -4 5.4.1*
INS *'CA'*
P1-P2 0100
Lc field *Empty*
Data field *Empty*
Le field 08

Tag 0100 is in the range of application proprietary data.

Answer is card serial 10 77 03 85 47 95 53 12 90

d) *ISO-7816-4 Get Data command (Applet)*

→ 00 CA 01 05 00

← 03 03 02 04 00 90 00

This is ISO7816 -4 Get Data (not GP)

Get Data for Tag 0x0105.

CLA *As defined in ISO7816 -4 5.4.1*
INS *'CA'*
P1-P2 0105
Lc field *Empty*
Data field *Empty*
Le field 00??

Tag 0105 is in the range of application proprietary data.

Unknown answer

e) Unknown proprietary command 80 AE (applet)

→ 80 AE 00 00 14 4F 03 31 32 33 34 00 00 00 00 00 00 00 00 00 00 00 40 20 0C

command 80AE is generally used in EMV contexts to perform generate Application Cryptogram.Here it does not match generate AC context and format.

Here we see that this command is sent because the previous command 00 CA 01 02 06 (get data for tag 0x0102) has failed with SW 0x6A88 (reference data not found from iso7816 -4) Since the next command is to retry this get data command, 80 AE command must precise to the card where to get this data object.

==> Init Token PKCS11

The role of this command seems to precise location of the resource and/or provide adequate credentials for accessing the resource.

f) *ISO-7816-4 Get Data command (Applet)*

→ 00 CA 01 02 06

← 7F FF 20 20 0C 0C 90 00

This is ISO7816 -4 Get Data (not GP)

Get Data for Tag 0x0102.

CLA *As defined in ISO7816 -4 5.4.1*
INS *'CA'*
P1-P2 0102
Lc field *Empty*
Data field *Empty*
Le field 06

Tag 0102 is in the range of application proprietary data.

Unknown answer.

g) ISO-7816-4 Verify command (Applet)

→ 00 20 00 01 0F 31 32 33 34 00 00 00 00 00 00 00 00 00 00

CLA *As defined in 5.4.1*
INS *'20'*
P1 *00*
P2 **01**
Lc field *0F*
Data field **31 32 33 34 00 00 00 00 00 00 00 00 00 00 00**
Le field *Empty*

P2=01 => Specific reference data (e.g. DF specific password)
We can see the matching of the password/ref data with the data of 80AE command of step e)

h) Unknown Proprietary Command 8036 (applet)

→ 80 36 03 01 07 0A 05 01 03 02 01 01 01

← 02 90 00

TBR (To Be Reversed)

i) Unknown Proprietary Command 8030 (applet)

→ 80 30 83 4F 0A 03 08 01 06 02 01 02 02 01 01 01

← 03 90 00

TBR (To Be Reversed)

j) ISO-7816-4 select File Command (Applet)

→ 00 A4 00 0C 02 3F 00

CLA	As defined in 5.4.1
INS	'A4'
P1	**00**
P2	**0C**
Lc field	**2**
Data field	**3F00**

P1=0 => Select MF, DF or EF
P2=0x0c=> File control information

3F00 indicates MF according to PKCS15 norm (ref 5.6)

k) Unknown Proprietary Command 80 E0 (Applet)

→ 80 E0 50 31 15 10 01 01 11 01 02 12 02 00 30 04 02 01 00 05 05 01 03 02 01 01

→ 80 E0 50 32 15 10 01 02 11 01 02 12 02 00 75 04 02 01 00 05 05 01 03 02 01 01

→ 80 E0 AE 0A 15 10 01 0A 11 01 02 12 02 00 40 04 02 01 00 05 05 01 03 02 01 01

→ 80 E0 AE 0B 18 10 01 0B 11 01 02 12 02 00 40 04 02 01 00 05 08 01 06 02 01 03 02 01 01

→ 80 E0 44 00 18 10 01 10 11 01 02 12 02 01 00 04 02 01 00 05 08 01 06 02 01 03 02 01 01

→ 80 E0 44 01 18 10 01 11 11 01 02 12 02 01 00 04 02 01 00 05 08 01 06 02 01 03 02 01 01

(...)

TBR (To Be Reversed)

l) ISO7816-4 Update Binary Command (Applet)

→ 00 D6 81 00 29 A0 06 30 04 04 02 44 00 A1 06 30 04 04 02 44 01 A4 06 30 04 04 02 44 04 A7 06 30 04 04 02 44 07 A8 06 30 04 04 02 44 08 00

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	**81 00**

Lc field	**29**
Data field	**A0 06 30 04 04 02 44 00 A1 06 30 04 04 02 44 01 A4 06 30 04 04 02 44 04 A7 06 30 04 04 02 44 07 A8 06 30 04 04 02 44 08 00**
Le field	Empty

short EF identifier of the file =01

Offset where the update starts = 00

m) ISO7816-4 Select File Command (Applet)

→ 00 A4 02 00 02 AE 0A [Le=00]

← 6F 07 80 02 00 40 82 01 01 90 00

CLA	As defined in 5.4.1
INS	'A4'
P1	02
P2	**00**
Lc field	**2**
Data field	**AE 0A**

P1=02 => Select EF under current DF (data field=EF identifier)
P2=00=> First record

n) ISO7816-4 Update Binary Command (Applet)

→ 00 D6 00 00 03 04 01 03

CLA	As defined in 5.4.1
INS	'D6'

P1-P2	**00 00**
Lc field	**3**
Data field	**04 01 03**
Le field	Empty

short EF identifier of the file =00

Offset where the update starts = 00

→ 00 D6 82 00 75 30 73 02 01 00 04 08 10 77 03 85 47 95 53 12 0C 12 41 2E 45 2E 54 2E 20 45 75 72 6F 70 65 20 42 2E 56 2E 80 20 42 6C 61 6E 6B 20 54 6F 6B 65 6E 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 03 02 04 50 83 2A 43 6F 70 79 72 69 67 68 74 20 28 63 29 20 31 39 39 37 2D 32 30 30 39 20 41 2E 45 2E 54 2E 20 45 75 72 6F 70 65 20 42 2E 56 2E

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	**82 00**
Lc field	**75**
Data field	**30 73 02 01 00 04 08 10 77 03 85 47 95 53 12 0C 12 41 2E 45 2E 54 2E 20 45 75 72 6F 70 65 20 42 2E 56 2E 80 20 42 6C 61 6E 6B 20 54 6F 6B 65 6E 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 03 02 04 50 83 2A 43 6F 70 79 72 69 67 68 74 20 28 63 29 20 31 39 39 37 2D 32 30 30 39 20 41 2E 45 2E 54 2E 20 45 75 72 6F 70 65 20 42 2E 56 2E**
Le field	Empty

short EF identifier of the file =0x02

Offset where the update starts = 00

o) ISO7816-4 Update Binary Command (Applet)

→ 00 D6 90 00 01 00

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	**90 00**
Lc field	**1**
Data field	**0**
Le field	Empty

short EF identifier of the file =0x10

Offset where the update starts = 00

p) ISO7816-4 Update Binary Command (Applet)

→ 00 D6 91 00 01 00

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	**91 00**
Lc field	**1**
Data field	**0**
Le field	Empty

short EF identifier of the file =0x11

Offset where the update starts = 00

q) ISO7816-4 Update Binary Command (Applet)

→ **00 D6 94 00 01 00**

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	**94 00**
Lc field	**1**
Data field	**0**
Le field	Empty

short EF identifier of the file =0x14

Offset where the update starts = 00

r) ISO7816-4 Update Binary Command (Applet)

→ **00 D6 97 00 01 00**

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	**97 00**
Lc field	**1**
Data field	**0**
Le field	Empty

short EF identifier of the file =0x17

Offset where the update starts = 00

s) ISO7816-4 Select File Command (Applet)

→ **00 A4 02 00 02 44 08 00**

← **6F 07 80 02 00 FF 82 01 01 90 00**

CLA	As defined in 5.4.1
INS	'A4'

P1	**02**
P2	**00**
Lc field	**2**
Data field	**44 08**

P1=02 => Select EF under current DF (data field=EF identifier)
P2=00=> First record

t) ISO7816-4 Read Binary Command (Applet)

→ 00 B0 00 00 80

**← 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00**

CLA	As defined in 5.4.1
INS	'B0'
P1-P2	00 00
Lc field	Empty
Data field	Empty
Le field	80

Read from offset 0

u) ISO7816-4 Update Binary Command (Applet)

→ 00 D6 98 00 80 30 36 30 11 0C 08 55 73 65 72 20 50 69 6E 03 02 06 C0 04 01 01

30 03 04 01 03 A1 1C 30 1A 03 03 04 C4 90 0A 01 01 02 01 04 02 01 0F 80 01 03 04 01 00 30 04 04 02 3F 00 30 31 30 0C 0C 06 53 4F 20 50 69 6E 03 02 06 C0 30 03 04 01 01 A1 1C 30 1A 03 03 04 CF 90 0A 01 01 02 01 04 02 01 0F 80 01 01 04 01 00 30 04 04 02 3F 00 A1 30 30 1F 0C 16 43 68 61 6C 6C 65 6E 67 65 20 52 65 73 70 6F

→ 00 D6 00 80 1E 6E 73 65 20 4B 65 79 03 02 06 C0 04 01 01 30 03 04 01 02 A1 08 30 06 01 01 FF 04 01 02 00

CLA	As defined in 5.4.1
INS	'D6'
P1-P2	**98 00**
Lc field	**80**
Data field	**30 36 30 11 0C 08 55 73 65 72 20 50 69 6E 03 02 06 C0 04 01 01 30 03 04 01 03 A1 1C 30 1A 03 03 04 C4 90 0A 01 01 02 01 04 02 01 0F 80 01 03 04 01 00 30 04 04 02 3F 00 30 31 30 0C 0C 06 53 4F 20 50 69 6E 03 02 06 C0 30 03 04 01 01 A1 1C 30 1A 03 03 04 CF 90 0A 01 01 02 01 04 02 01 0F 80 01 01 04 01 00 30 04 04 02 3F 00 A1 30 30 1F 0C 16 43 68 61 6C 6C 65 6E 67 65 20 52 65 73 70 6F**
Le field	Empty

short EF identifier of the file =0x12

Offset where the update starts = 00

v) unknown command 80 3F

→ 80 3F 00 00

deselect applet / return to card manager ?

w) GLOBALPLATFORM Safesign Pkcs15 Applet selection

→ **00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00**

x) Unknown command 80F6 (Applet)

→ **80 F6 00 01 0A**

y) steps b),c),d)

z) Unknown command 80 54 (applet)

→ **80 54 01 01 00**

aa) step j)

ab) ISO7816-4 MANAGE CHANNEL Command (Card manager)+GP applet selection

→ **00 70 00 00 01**

CLA	As defined in 5.4.1
INS	'70'
P1	P1='00' to open a logical channel
P2	0
Lc field	Empty
Data field	Empty
Le field	'01' if P1-P2='0000'

Open a logical channel

ac) steps e),f),g),h).i).j).k)

ad) ISO-7816-4 select File Command (Applet)

→ 00 A4 00 0C 02 50 31

CLA	As defined in 5.4.1
INS	'A4'
P1	**00**
P2	**0C**
Lc field	**2**
Data field	**5031**

→ 00 A4 02 00 02 AE 0A 00

CLA	As defined in 5.4.1
INS	'A4'
P1	**00**
P2	**02**
Lc field	**2**
Data field	**AE 0A**

ae) ISO7816-4 Read Binary Command (Applet)

→ 00 B0 00 00 40

← 04 01 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00

READ TAG
00 CA 01 02 06
received:
7F FF 20 20 0C 0A 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 02 00 02 50 32 00
received:
6F 07 80 02 00 75 82 01 01 90 00

af) ISO7816-4 Read Binary Command (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 B0 00 00 75
received:
30 73 02 01 00 04 08 10 77 03 85 47 95 53 12 0C 12 41 2E 45 2E 54 2E 20 45 75
72 6F 70 65 20 42 2E 56 2E 80 20 42 6C 61 6E 6B 20 54 6F 6B 65 6E 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 03 02 04 50 83 2A 43 6F 70
79 72 69 67 68 74 20 28 63 29 20 31 39 39 37 2D 32 30 30 39 20 41 2E 45 2E 54
2E 20 45 75 72 6F 70 65 20 42 2E 56 2E 90 00

ag) ISO7816-4 Select FileCommand (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 02 00 02 44 08 00
received:
6F 07 80 02 00 FF 82 01 01 90 00

ah) ISO7816-4 Read Binary Command (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 B0 00 00 80
received:
30 36 30 11 0C 08 55 73 65 72 20 50 69 6E 03 02 06 C0 04 01 01 30 03 04 01 03
A1 1C 30 1A 03 03 04 C4 90 0A 01 01 02 01 04 02 01 0F 80 01 03 04 01 00 30 04
04 02 3F 00 30 31 30 0C 0C 06 53 4F 20 50 69 6E 03 02 06 C0 30 03 04 01 01 A1

1C 30 1A 03 03 04 CF 90 0A 01 01 02 01 04 02 01 0F 80 01 01 04 01 00 30 04 04
02 3F 00 A1 30 30 1F 0C 16 43 68 61 6C 6C 65 6E 67 65 20 52 65 73 70 6F 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 B0 00 80 7F
received:
6E 73 65 20 4B 65 79 03 02 06 C0 04 01 01 30 03 04 01 02 A1 08 30 06 01 01 FF
04 01 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00

ai) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 03 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

aj) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 01 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

ak) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 02 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

al) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 03 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 01 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 02 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

am) ISO7816-4 Get Data (applet)

SCardTransmit (handle 0xEA030000):

transmitted:
00 CA 01 02 06
received:
7F FF 20 20 0C 0A 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

an) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 03 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 01 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 02 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 02 00 02 AE 0B 00
received:
6F 07 80 02 00 40 82 01 01 90 00

ao) iso7816-4 Read Binary (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 B0 00 00 40
received:
69 86

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 02 00 02 AE 0B 00
received:
6F 07 80 02 00 40 82 01 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 B0 00 00 40
received:
69 86

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

ap) iso7816-4 get data (applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 CA 01 02 06
received:
7F FF 20 20 0C 0A 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

aq) iso7816-4 select file (applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 02 00 02 AE 0B 00
received:
6F 07 80 02 00 40 82 01 01 90 00

ar) iso7816-4 Read Binary (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 B0 00 00 40
received:
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 90 00

as) iso7816-4 Verify (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 20 00 01 0F 31 32 33 34 00 00 00 00 00 00 00 00 00 00
received:
90 00

at) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 03 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 01 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 02 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

au) iso7816-4 Verify (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 20 00 01 0F 31 32 33 34 00 00 00 00 00 00 00 00 00 00
received:
90 00

av) iso7816-8 change reference data (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 24 01 03 0F 31 32 33 34 00 00 00 00 00 00 00 00 00 00
received:
90 00

aw) iso7816-4 Update Binary (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 D6 98 01 80 47 30 11 0C 08 55 73 65 72 20 50 69 6E 03 02 06 C0 04 01 01 30
03 04 01 03 A1 2D 30 2B 03 03 04 CC 90 0A 01 01 02 01 04 02 01 0F 80 01 03 04
01 00 18 0F 32 30 31 32 30 32 32 30 30 36 31 34 34 37 5A 30 04 04 02 3F 00 30
31 30 0C 0C 06 53 4F 20 50 69 6E 03 02 06 C0 30 03 04 01 01 A1 1C 30 1A 03 03
04 CF 90 0A 01 01 02 01 04 02 01 0F 80 01 01 04 01 00 30 04 04 02 3F 00 A1 30
30 1F 0C
received:
90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 D6 00 81 2D 16 43 68 61 6C 6C 65 6E 67 65 20 52 65 73 70 6F 6E 73 65 20 4B
65 79 03 02 06 C0 04 01 01 30 03 04 01 02 A1 08 30 06 01 01 FF 04 01 02
received:
90 00

ax) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 03 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 01 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 02 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

ay) iso7816-4 Verify (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 20 00 01 0F 31 32 33 34 00 00 00 00 00 00 00 00 00 00
received:
90 00

az) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 03 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 01 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 02 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

ba) iso7816-4 Verify (Applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 20 00 01 0F 31 32 33 34 00 00 00 00 00 00 00 00 00 00 00
received:
90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 00 0C 02 3F 00
received:
90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00

received:
90 00

bb) get retry counter (8034)?

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 03 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 01 03
received:
03 03 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
80 34 00 02 03
received:
03 03 00 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 04 00 0C A0 00 00 00 63 50 4B 43 53 2D 31 35 00
received:
90 00

bc) ISO7816-4 get data (applet)

SCardTransmit (handle 0xEA030000):
transmitted:
00 CA 01 02 06
received:
7F FF 20 20 0C 0A 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 02 00 02 AE 0B 00
received:
6F 07 80 02 00 40 82 01 01 90 00

bd) iso7816-4 Read Binary

SCardTransmit (handle 0xEA030000):

transmitted:
00 B0 00 00 40
received:
30 73 02 01 00 04 08 10 77 03 85 47 95 53 12 0C 12 41 2E 45 2E 54 2E 20 45 75
72 6F 70 65 20 42 2E 56 2E 80 20 42 6C 61 6E 6B 20 54 6F 6B 65 6E 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 A4 02 00 02 AE 0B 00
received:
6F 07 80 02 00 40 82 01 01 90 00

SCardTransmit (handle 0xEA030000):
transmitted:
00 B0 00 00 40
received:
69 86

2. APDU Scan

2.1. Attempt to load the applet on the javacard simulator

We use the javacard simulator provided by sun/oracle in order to perform full APDU scan. Indeed doing this on real EEprom may destroy the card before the scan ends and reach the maximum values of R/W uses. A full scan may also be very slow in real T=0/T=1 protocol while using the simulator will speed up dramatically the scans (APDUS are sent through local tcp/ip connections).

The script generator tool allows us to get a template for loading all the cap components to the applet installer (The Javacard simulator do not use Global Platform or any other “intermediary” method to install applet but send the cap component directly to the installer applet of the javacard platform).We use this skeleton to send our own cap components (that we retrieved from the jcf files) to the simulator.

```
C:\javacardkit\java_card_kit-2_2_2\bin>scriptgen.bat C:\temp\cap\safesign.cap >
loadSafeSign.apdu
```

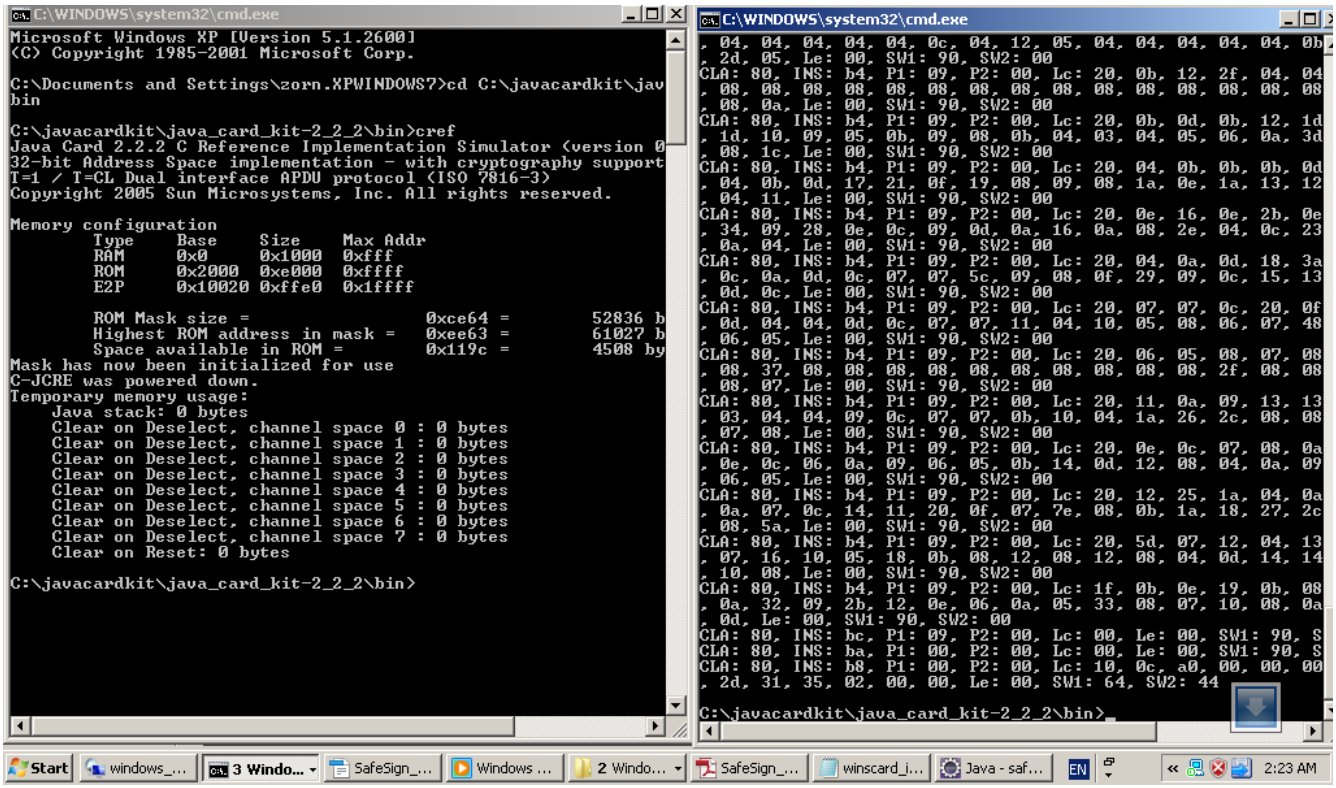

Illustration 1: Loading Applet in Javacard simulator

The result is a 0x64 44 error (after many tries with different configurations) when installing (installing="applet creation command" in terms of the javacard installer applet). This means here that our install parameters may be incorrect. Since the install parameters are the 8byte chipID+2 bytes Manufacturer ID , these parameters may be impossible to get in a javacard simulator which cause applet install to fail during some of the constructor checks.

Thus the apdu scan needs to be done in the card itself.

Apdu scanner will use CLA and INS values as defined by ISO7816-4:

2.2. Values of CLA

Value	Meaning
'0X'	Structure and coding of command and response according to this part of ISO/IEC 7816 (for coding of 'X' see table 9
10 to 7F	RFU
8X, 9X	Structure of command and response according to this part of ISO/IEC 7816. Except for 'X' (for coding, see table 9), the coding and meaning of command and response are proprietary

AX	Unless otherwise specified by the application context, structure and coding of command and response according to this part of ISO/IEC 7816 (for coding of 'X', see table 9)
B0 to CF	Structure of command and response according to this part of ISO/IEC 7816
D0 to FE	Proprietary structure and coding of command and response
FF	Reserved for PTS

0X,8X,9X,AX,B0-CF,D0-FE

This makes 4*16 +32+47=143 possible values of CLA.

2.3. Values of INS

The instruction byte INS of a command shall be coded to allow transmission with any of the protocols defined in part 3 of ISO/IEC 7816. Table 10 shows the INS codes that are consequently invalid.

Table 10 - Invalid INS codes

b8 b7 b6 b5 b4 b3 b2 b1	Meaning
x x x x x x x 1	Odd values
0 1 1 0 x x x x	'6X'
1 0 0 1 x x x x	'9X'

This makes 2^7+2*2^4=160 invalid INS and consequently the possible values of INS will be 255-160=95.

The total amount of combination CLA,INS is therefore 143*95=13585

2.4. Methodology of the scan, building of the apdu scanner

We first TRY CLA values with default valid INS code. If we receive a status word of 6E00 then we consider that this value of CLA is not implemented by the applet. Next we try all values of INS with

the remaining values of CLA and P1,P2 set to 00,00. If we receive a 6D00 this mean that this INS is not supported. We also consider that a command that returns 0x6881 (Logical Channel not supported) is not implemented as well. This will give us a potential list of commands (CLA,INS) implemented by the card.

Status word returned which we consider mean the CLA is not implemented

0x6D00

0x6881

0x6986

Status word returned which we consider mean the command CLA/INS is not implemented

0x6E00

0x6881

0x6986

Of course we assert that the applet follows ISO7816-4 norm.

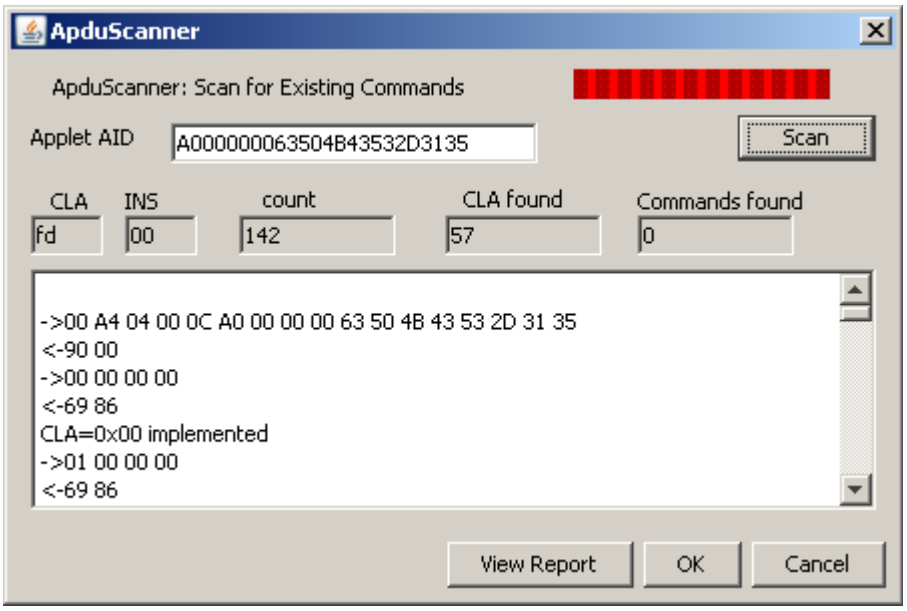

Illustration 2: APDU Scanner

a) Result with SafeSign Applet

The result of the scan is that Safesign applet does not respect ISO7816-4 logic and do not returns systematically 6E00 for missing CLA and 6D00 for missing INS but returns in 90% of the cases a 6985 or 6986 error .

As a result, implemented commands like 0x80 0x36 will return a 0x6986 status which prevent them to be discriminated from other unimplemented commands!

This is potentially a way from the developers to prevent apdu scan and command discovering.

Result of Full Scan:

This scan will reveal only a partial list of implemented commands.

Command implemented:00 20

Command implemented:00 22

Command implemented:00 24

Command implemented:00 2a

Command implemented:00 46

Command implemented:00 82

Command implemented:00 84

Command implemented:00 a4

Command implemented:00 ca

Command implemented:01 20

Command implemented:01 22

Command implemented:01 24

Command implemented:01 2a

Command implemented:01 46

Command implemented:01 82

Command implemented:01 84

Command implemented:01 a4

Command implemented:01 ca

Command implemented:02 20

Command implemented:02 22

Command implemented:02 24

Command implemented:02 2a

Command implemented:02 46

Command implemented:02 82

Command implemented:02 84

Command implemented:02 a4

Command implemented:02 ca

Command implemented:03 20

Command implemented:03 22

Command implemented:03 24

Command implemented:03 2a

Command implemented:03 46

Command implemented:03 82

Command implemented:03 84

Command implemented:03 a4

Command implemented:03 ca

Command implemented:80 2a

Command implemented:80 30

Command implemented:80 34

Command implemented:80 36

Command implemented:80 38

Command implemented:80 3a

Command implemented:80 3c

Command implemented:80 50

Command implemented:80 82

Command implemented:80 e0

Command implemented:80 e2

Command implemented:80 e4

Command implemented:84 50

Command implemented:84 82

3. JCF File reversing

We reverse the jcf files the following way:

After analysis , it appears that jcf file used by safesign middleware to load the applet on the card are almost similar to BIN files : This is the concatenation as bytes streams of the cap file components (Header.cap, etc...). We write a tool that parse the jcf files and extract the components , then jar them into a global cap file, ready to be loaded on the card. **The jcf file do not contains the Descriptor.cap component – which is mandatory component for offcard checks so our cap file will miss that component and will not pass offcard verifier checks performed by most cap loaders (including the Sun/Oracle's loader)**

We may write a custom jcf to cap converter by using the cap component specifications : we simply parse the jcf component tag after component tag and we store each component in a global jar file with the adequate MANIFEST attributes. Of course our descriptor component will be null (which is not allowed theoretically by the off-card verifier).

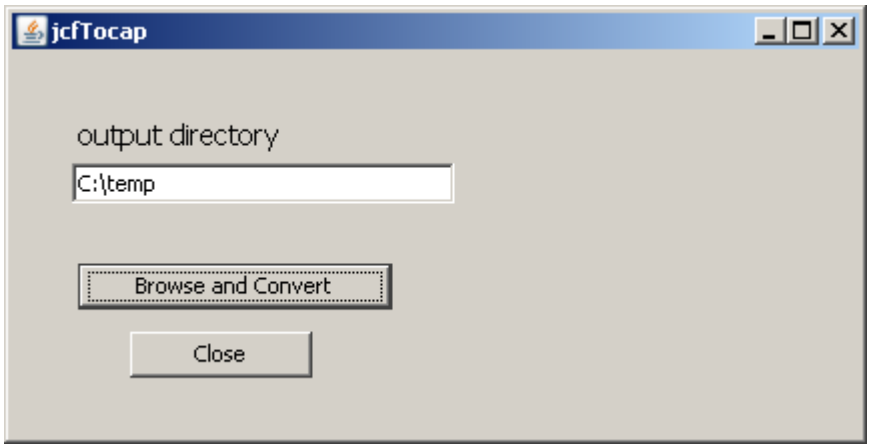

Illustration 3: jct to cap main windows

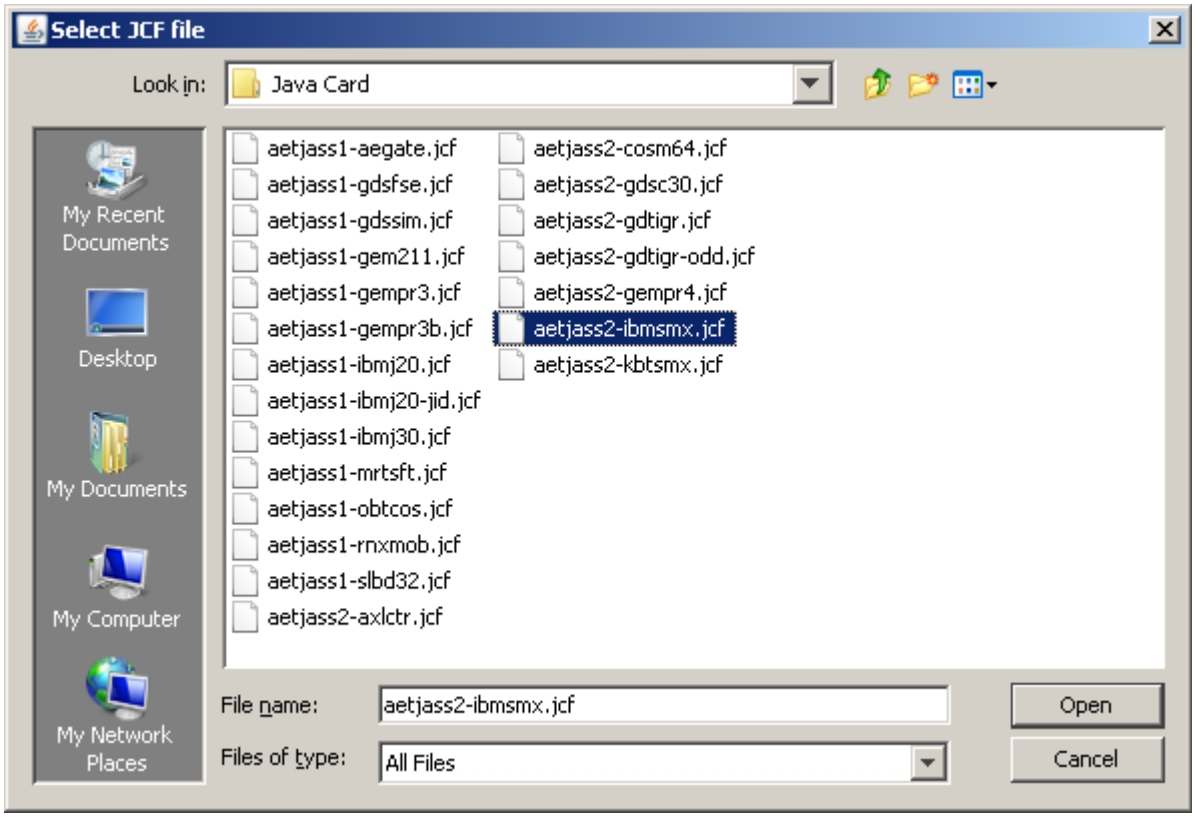

Illustration 4: jcf to cap file selection

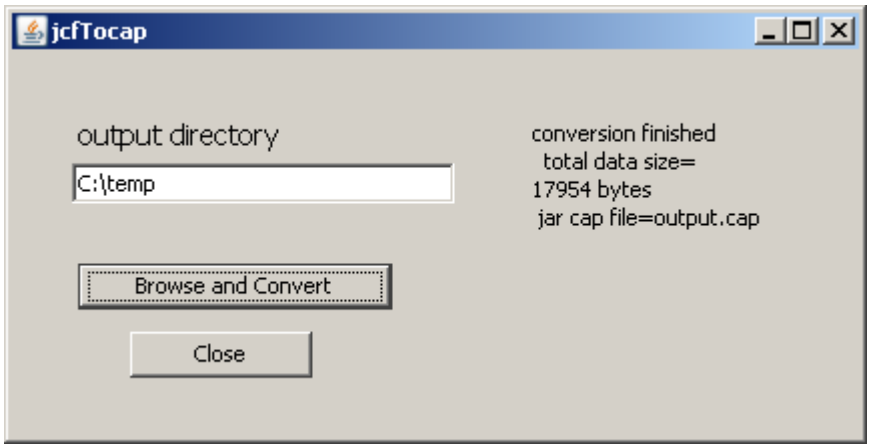

Illustration 5: jcf to cap processing

We must write a custom loader (that works well with our cap file) which provide SCP02 authentication and bypass offcard bytecode verification.

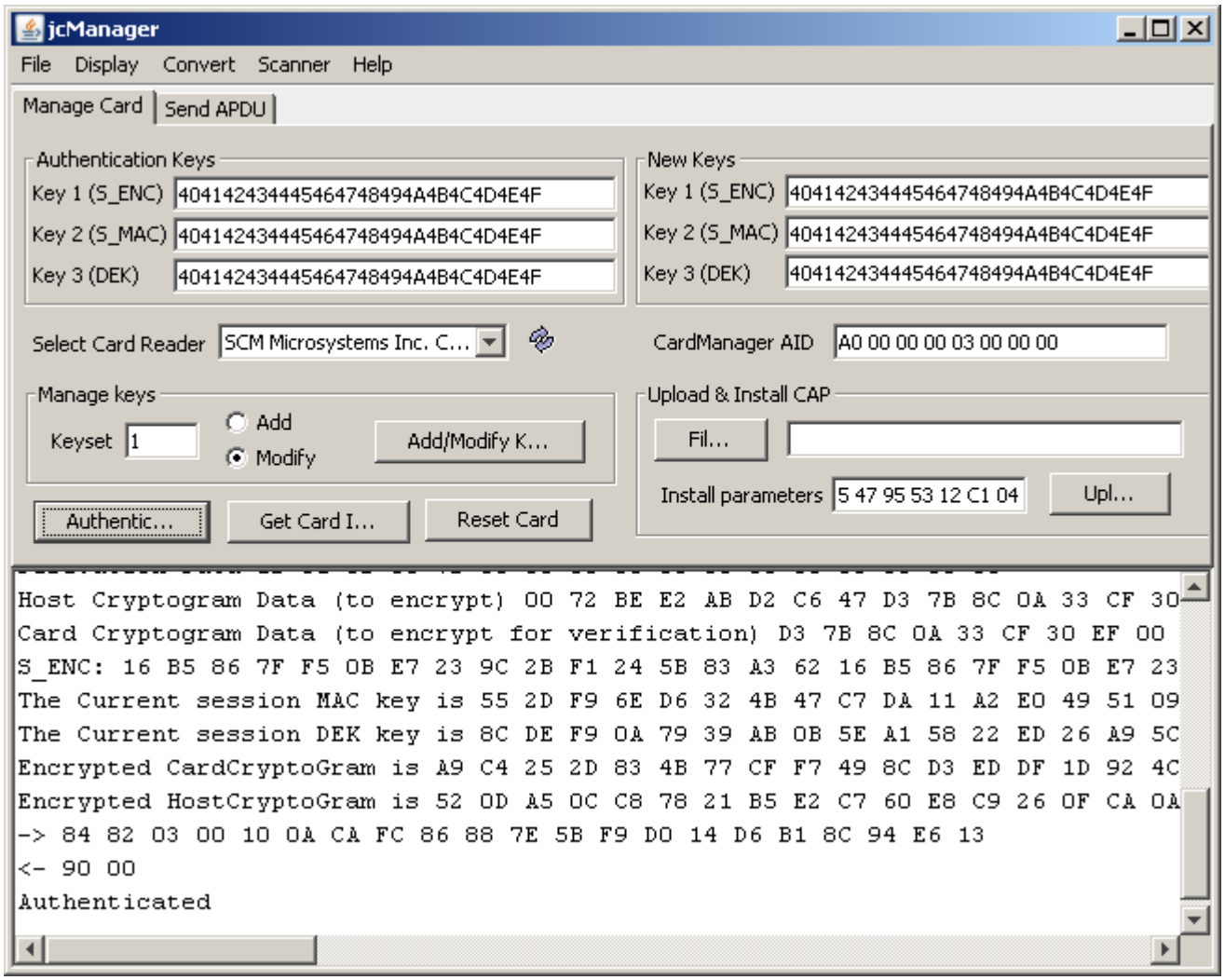

Illustration 6: SafeSign Toolkit (Modified jcmanager) custom cap loader

4. Cap File Decompilation

4.1. Cap file compilation process

We try to reverse the javacard code by decompiling the cap file to a class file and reversing the class to javacard sources.

The way class files are transformed into cap files by the converter is documented in the jcre specifications. The main aspects of class compilation is the tokenization of the data (only the exp file -which is a separated file- contains the namespace, the methods names and the class names) .so in the

cap file original names are lost and replaced by automatically generated tokens such as AA1,AA2,etc... The cap file is the result of a partial pre-execution of the javacard virtual machine (which is a classical way to optimize execution)

It is hard and time costly to build a custom cap decompilator since we must reverse the partial pre-execution done by the converter.

4.2. The normalizer

Fortunately with javacard 3.0 a tool known as the normalizer has been introduced in the jdk toolkit. This tool is intended to migrate cap files generated with previous versions of javacard to javacard3. But what the normalizer does is to *decompile* the cap file to a class then recompile it with the javacard 3 converter.In fact th normalizer uses itself the *capprocessor* component to decompile a cap file.

The core decompilation is done in the following code (reversed by jad)

```
ClassDescriptor cdesc = Cap.Descriptor.firstClassDescriptor();
String clName = null;
for(int i = Cap.Descriptor.classCount(); i > 0;)
{
ClassFile cf = new ClassFile();
int clToken = cdesc.token();
clName = CAPUtil.getRefName(cdesc.thisClass());
if(Cap.Applet != null && Cap.Applet.isAppletClass(cdesc))
{
com.sun.javacard.offcardverifier.AppletInfo applet =
Cap.Applet.getAppletInfoForClass(cdesc);
applets.put(clName, applet);
}
EfClass exportedClass = null;
if(ef != null)
exportedClass = ef.findClassByToken((short)clToken);
cf.setClassName(clName);
ClassConstantPool classConstantPool = new ClassConstantPool(cdesc, clName,
exportedClass);
cf.setConstantPool(classConstantPool);
cf.setAccessFlags(cdesc.flags());
cf.setThisClass((short)classConstantPool.getThisClassIndex());
cf.setSuperClass((short)classConstantPool.getSuperClassRefIndex());
Vector interfaces = classConstantPool.getInterfaceRefs();
int interfaceCount = interfaces.size();
cf.setInterfaceCount((short)interfaceCount);
for(int j = 0; j < interfaceCount; j++)
{
int interfaceRef = ((Integer)interfaces.elementAt(j)).intValue();
```

```
cf.setInterfaceEntry((short)interfaceRef);
}

cf.setMethodTable(new ClassMethodTable(classConstantPool, cdesc, exportedClass));
cf.setFieldTable(new ClassFieldsTable(classConstantPool, cdesc, exportedClass));
String classFileName = (new
StringBuilder()).append(clName.substring(clName.lastIndexOf("/") +
1)).append(".class").toString();
if(outFilePath == null)
outFilePath = (new
StringBuilder()).append(outputPath).append(pathSep).append(pkgName).toString();
String outFileName = (new
StringBuilder()).append(outFilePath).append(pathSep).append(classFileName).toString();
try
{
File outFileDir = new File(outFilePath);
if(!outFileDir.exists())
outFileDir.mkdirs();
File outputFile = new File(outFileName);
FileOutputStream os = new FileOutputStream(outputFile);
cf.write(os);
}
catch(Exception e)
{
System.out.println(e.toString());
}
i--;
cdesc.next();
}
```

at this stage it is clear that the variable cdesc = Cap.Descriptor.firstClassDescriptor() is essential to get the information that will allow the processing and decompilation of the cap components.

Even if we could bypass the checks of the cap integrity (which is done many times at different place of the decompilation process) we cannot get a value for the cdesc variable since we do not have the descriptor component! We must re-create the descriptor component from the javacard runtime specifications – ideally into our jcf to cap converter - which is a very big consuming tasks and need full and detailed understanding of javacard bytecode and jcre behaviour,besides even if we try to re-create this component there is no guarantee that we will not miss critical information for decompilation and to study these issues would draw too much time and energy.

5. Apdu sniffing and Windows Application Sniffing

We combine apdu sniffing and debugging of the client application who communicates with the applet to try to detect the features of the applet. We take a list of user case scenarios as input.

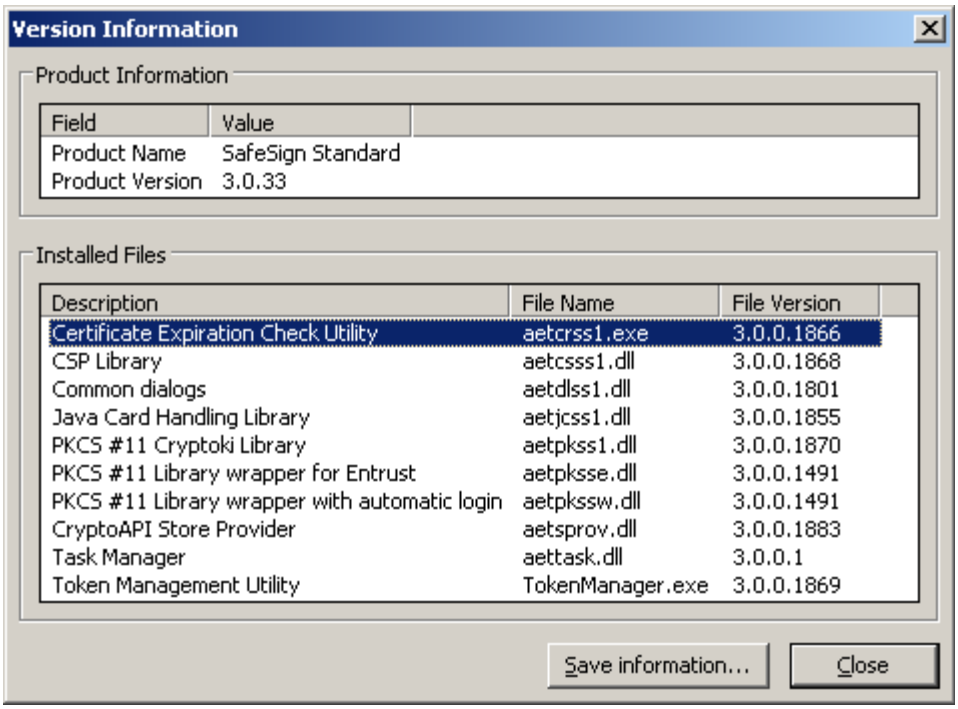

Illustration 7: List of software components used by SafeSign Desktop Client

Component to be debugged/sniffed when communicating with the applet:

- aetcsss1.dll (AET SafeSign CSP)
- aetcsss1.dll (AET SafeSign PKCS#11)
- winscard.dll (PCSC)

6. APDU Tables for SafeSign Applet

These tables are created from apdu scan and apdu sniffing

6.1. ISO7816-4 Commands

INS Value	Command name	Clause
?'0E'	ERASE BINARY	6.4
☑'20'	VERIFY	6.12
☑'70'	MANAGE CHANNEL	6.16
☑'82'	EXTERNAL AUTHENTICATE	6.14
☑'84'	GET CHALLENGE	6.15
?'88'	INTERNAL AUTHENTICATE	6.13
☑'A4'	SELECT FILE	6.11
☑'B0'	READ BINARY	6.1
?'B2'	READ RECORD(S)	6.5
?'C0'	GET RESPONSE	7.1
?'C2'	ENVELOPE	7.2
☑'CA'	GET DATA	6.9
?'D0'	WRITE BINARY	6.2
?'D2'	WRITE RECORD	6.6
☑'D6'	UPDATE BINARY	6.3
?'DA'	PUT DATA	6.10
?'DC'	UPDATE DATA	6.8
☑'E2'	APPEND RECORD	6.7

6.2. ISO7816-8 Commands

INS VALUES

☑ **22 MANAGE SECURITY ENVIRONMENT**

☑ **24 CHANGE REFERENCE DATA**

☑ **2A PERFORM SECURITY OPERATION**

☑ **46 GENERATE PUBLIC KEY PAIR**

6.3. Proprietary Commands

CLA	INS	P1	P2	FORMAT
PKCS11 Init Token				
80	AE	0	0	***This command seems to initialize the token according to PKCS11 specification***
PKCS#11: ***CK_DEFINE_FUNCTION(CK_RV, C_InitToken)(*** CK_SLOT_ID slotID, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen, CK_UTF8CHAR_PTR pLabel); **80 AE 00 00 14 <4F 03 +31 32 33 34 00 00 00 00 00 00 00 00 00 00 00 + 40 20 0C>** **CLA INS P1 P2 LC <SLOTID+PIN+LABEL> ?**				
80	F6			
80	54			
80	84			
80	30			
Get Retry Counter?				
80	34			
80	36			
80	38			
Read/Search Public Key				
80	3A			

Read Public key		keyslot	0x01/0x02	0x01=>returns modulus 0x02=>returns exponent
Search Public key		?	?	?
80	3C			
80	3F			
84	50			
80	E0			
2	84			