Finding Vulnerabilities in Low-Level Protocols

Nordine Saadouni

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2017

Abstract

Basic example of an abstract (this will be changed)

Smart cards are used commercially and within industry for authentication, encryption, decryption, signing and verifying data. This paper aims to look into how the smart card interacts with an application at the lower level. PKCS#11 (public key cryptography system?) is the standard that is implemented at the higher level and then broken down into command/response pairs sent as APDU traffic to and from the smart card. It is the APDU low-level protocol that will be analysed to see if any vulnerabilities are present with regard to the smart cards tested.

Acknowledgements

Acknowledgements go here.

Table of Contents

Chapter 1

Introduction

Smart-cards are formally known as integrated circuit cards (ICC), and are universally thought to be secure, tamper-resistant devices. They store and process, cryptographic keys, authentication and user sensitive data. They are utilised to preform operations where confidentiality, data integrity and authentication are key to the security of a system.

Smart-cards offer what seems to be more secure methods for using cryptographic operations. (And should still provide the same level of security that would be offered to un-compromised systems, compared to those that are compromised by an attacker). This is partly due to the fact that the majority of modern smart-cards have their own on-board micro-controller, to allow all of these operations to take place on the card itself, with keys that are unknown to the outside world and stored securely on the device. Meaning the only actor that should be able to preform such operations would need to be in possession of the smart-card and the PIN/password. In many industries, for applications such as, banking/ payment systems, telecommunications, healthcare and public sector transport, smart-cards are used due to the security they are believed to provide.

The most common API (application programming interface) that is used to communicate with smart-cards is the RSA defined PKCS#11 (Public Key Cryptography Standard). Also known as 'Cryptoki' (cryptographic token interface, pronounced as 'crypto-key'). The standard defines a platform-independent API to smart-cards and hardware security modules (HSM). PKCS#11 originated from RSA security, but has since been placed into the hands of OASIS PKCS#11 Technical Committee to continue its work (since 2013). [reference wikipedia PKCS#11].

In the previous 10-15 years, literature has shown a great deal of research into the examination of the PKCS#11 API, and the security it provides.

Yet little attention has been paid to that of the lower-level communication (command-response pairs), in which the higher level API is broken down into. It is this area that we wish to research within this paper. The reasoning is simple. If we cannot trust the security of the low-level commands that implement the high level API functions, then in turn we cannot trust the security of the high level API. This is analogous to C code being complied down to binary data to be operated on by the CPU. The addition of two integers cannot be considered correct in C, unless the corresponding binary instructions sent to the CPU are correct as well.

The research of the low-level communication will take two forms.

1. An analysis of the raw communication between PC and Smart-card for all API function calls
2.

Before we move into the above analysis, supporting material must be introduced. This the rest of this paper will be organized as follows.

Chapter 2

Background

In this chapter we give background knowledge on two essential standards that are required for the use of the contact smart-card we analyse in this paper. These two standards are PKCS#11 and ISO 7816.

2.1 PKCS#11

This is the API that each card manufacture implements themselves.

2.1.1 Key Object

2.1.2 Attributes

```
template = (\\
(CKA\_CLASS, LowLevel.CKO\_SECRET\_KEY), [private, public, data, cert (X.5
(CKA\_KEY_TYPE, LowLevel.CKK_DES), [AES,DES,RSA,ECC]
(CKA\_LABEL, label), [name]
(CKA\_ID, chr(id)), [id]
(CKA\_PRIVATE, True), [requires authentication]
(CKA\_SENSITIVE, True), [cannot be extracted unencrypted]
(CKA\_ENCRYPT, True), [can be used for encrypting data]
(CKA\_DECRYPT, True), [can be used for decrypting data]
(CKA\_SIGN, True), [[can be used for signing data]
(CKA\_VERIFY, True), [can be used for verifiying data]
(CKA\_TOKEN, True), [can be stored permanently on the device]
(CKA\_WRAP, True), [can encrypt a key to be extracted]
(CKA\_UNWRAP, True), [can decrypt an encrypted key]
(CKA\_EXTRACTABLE, False)) [is allowed to be extracted from the device]
```

2.1.3 Most Common Functions

2.2 ISO 7816

This is the international standards organization that defines the low level communication protocol between smartcards and the API on the computer.

2.2.1 Command Structure

2.2.2 Response Structure

2.2.3 Inter-Industry/ Proprietary

2.2.4 Most Common Commands

CLA	INS	P1	P2	Lc	Data Field	Le	Description
00	-	-	-	-	-	-	Inter-industry (II)
80	-	-	-	-	-	-	Proprietary (P)
Xc	-	-	-	-	-	-	Secure Messaging (SM)
00	84	00	00	-	-	L	(II) Get Challenge [# bytes = L]
00	b0	X	X	-	-	L	(II) Read Binary [# bytes = L]
00	c0	00	00	-	-	L	(II) Get Remaining Bytes
00	d6	X	X	L	X	-	(II) Update Binary
00	e0	X	X	L	X	-	(II) Create Binary
00	47	00	00	L	params	-	(II) Generate RSA KeyPair
00	e4	00	00	00	-	-	(II) Delete File
00	2a	82	0e	L	X	00	(II) Encrypt Data
00	2a	80	0e	L	X	00	(II) Decrypt Data
00	2a	9e	12	L	X	00	(II) Sign Data
00	2a	80	0a	L	X	00	(II) Unwrap Key (RSA_PKCS)
80	a4	08	00	L	FL	-	(P) Select File [FL = file location; L = len(fl)]
80	a4	00	00	-	FL	-	(P) Select File [append previous path]
80	a4	08	0c	L	FL	-	(P) Read File Control Parameters
80	20	00	00	10	R	-	(P) Verify PIN [R = Response]
80	28	00	00	04	00 00 00 20	-	(P) Clear Security Status
80	30	01	00	00	-	-	(P) List Files
80	48	00	80	00	-	-	(P) Get Cards Public Key (genkey)
80	48	00	00	00	-	-	(P) Get Cards Public Key (genkeypair)
80	86	00	00	L	X	00	(P) Open Secure Messaging
80	86	ff	ff	-	-	-	(P) Close Secure Messaging
90	32	00	03	ff	X	-	Reallocate Binary 256 bytes
80	32	00	03	L	X	-	Reallocate Binary L bytes (changes file)

Table 2.1: Common APDU Commands

2.2.5 File Structure

Chapter 3

Cryptographic Operations

In this chapter we describe different cryptography standards, and give a brief overview of the terminology and maths used behind some of the types of cryptography. This will be used to aid our explainations in later chapters that regard the cards functionality and cryptographic operations used to attempt the secure transfer of sensitive information.

3.1 Hash Function

explain this in an overview term

3.2 Asymmetric Encryption

public and private keys for communicating over an insecure channel

3.2.1 RSA

Chinese Remainder Theorem RSA Keys

3.3 Symmetric Encryption

explain this in an overview term

3.3.1 DES

3.3.2 Triple DES

3.3.3 AES

3.3.4 ECB Mode

3.3.5 CBC Mode

3.4 Message Authentication Codes

explain what a message authentication code is, what its used for
(wikipedia have good diagrams and explanations of this!!)

3.4.1 Hash Based - Message Authentication Codes (HMAC)

3.4.2 Cryptographic Based - Message Authentication Codes (CMAC)

3.5 One Time Passwords

What is a one time password → what forms are there?

3.5.1 Hash Based - One Time Passwords (HOTP)

3.5.2 Time Based - One Time Passwords (TOTP)

Chapter 4

Tools

4.1 PCSC-lite

4.2 Virtual Smart-Card

frank something Explain how this creates a virtual smartcard reader This is then used to run API functions, and the commands are relayed to the actual card reader and therefore card.

4.3 Man in The Middle (MiTM)

Chapter 5

Related Work / Literature Review

This will be a brief chapter and will discuss all of the research I have conducted.
Mainly regarding PKCS#11 API attacks due to the small amount of literature that is available for APDU level attacks I shall also explain why some of the attacks are not able to be conducted on the particular card I am reviewing

Chapter 6

PKCS #11 Functions - APDU analysis

In this chapter we analyse the APDU traces of 9 functions from the API. C_sign and C_verify were not included in this analysis as from analysis of the traces from the previous work on this card it was clear that both of these functions operated in a similar manner to C_encrypt and C_decrypt. C_createObject was also omitted form this analysis, as the function is not used for security operations, and rather for the creation and storing of certificates and data. The analysis will include a step by step guide of how the functions are broken down into command response pairs at the APDU level, and also suggest methods for attacking the card using this knowledge.

All of the traces are provided in appendix B, and have been shortened to only include the parts which we deem to be the most important to the evaluation of how each function is broken down into APDU command response pairs. However, the full traces for each function are given within the project directory. In the analysis to come we will refer to appendix B, with the corresponding command response pair in brackets to allow ease of locating the steps we analyse.

6.1 Initialization

The first process before anything can occur on the card is the initialization. This occurs as soon as a session is opened with the card. The process has the following steps:

1. Open the laser PKI file and select the applet (B.1 - command 1)
2. Open the file and send to the API the cards serial number (B.1 - command 4,5)

3. Open a second file and send to the API another serial number (B.1 - command 8,9)

None of these traces reveal sensitive information. But is required in order for the card to be able to operate and communicate with the API. The API is a generic library that works with many different hardware security modules created by the 'Athena smartcard solutions'. Hence this information is required by the library in order for it to operate correctly for the specific type of card.

6.2 C_login

The login function has 5 main components. The API locates the file control parameters to check if the retry counter is greater than 0. In the APDU trace this is highlighted in bold and has a value of AA. From there the PIN file is selected, a challenge is asked for by the card and a response is calculated from the knowledge of the PIN and the challenge. Once this is completed the security status of the card is cleared. This is done because every security operation requires a new login by the API. While the user only logs in once, at the APDU layer this occurs many times. These steps are listed below:

1. Open file control parameters for the PIN file (B.2 - command 10)
2. Open the PIN file (B.2 command 11)
3. Request challenge from card (B.2 - command 12)
4. Verify PIN (B.2 - command 13)
5. Clear security Status (B.2 - command 20)

From evaluating multiple traces of the login, it is clear that the response calculated seems to have an element of randomness. From studying the traces alone, the method of calculating the response cannot be intuitively determined.

While no immediate sensitive information is revealed in this trace as the challenge-response algorithm used for verifying a users PIN hides its value, successfully reverse engineering the protocol will allow an attacker to calculate the users PIN given successful login trace.

Furthermore, if an attacker can find a method for changing the file control parameter, it might be possible to reset the retry counter at the APDU level to allow an unlimited number of attempts of logging in.

6.3 C_findObject

[For this trace to show meaningful information we pre-loaded a triple des key onto the card. The label of the key was 'des3' with id '01']

The findObjects functions is split into 3 main components. First all the attribute files are listed that are stored on the card. The card has a file named 'cmapfile' that stores these locations. Using the data saved in this file, the API can locate the directory of each attribute file to load it into the API. A final file is opened that is zeroized, which dictates that no more attribute files for keys are present on the card. These steps are listed below:

1. Selects and opens 'cmapfile' [List files command]. This stores all file locations for attributes of the keys (B.3 - command 32, 33, 34)
2. Based on those locations, finds and opens the attribute file for the des3 key (B.3 - command 36)
3. And also opens the last attribute file to determine there are no more keys (B.3 - command 38, 39)

Due to key and attribute files being stored separately, this function call reveals no sensitive information. Only the attributes are opened and listed at the APDU layer. These same attributes can be printed at the API layer as well.

A possibility for an attack is present here. The ISO 7816 'UPDATE BINARY' command can be used to alter a file. This will allow modification of attributes. As mentioned in the literature review, previous work on the same card was undertaken and modified the attributes of CKA_sensitive & CKA_extractable from false to true. The change in this direction is forbidden by the API. However the ability for changing these at the APDU level was achieved. It still yielded no significant results, as keys and attribute are stored in separate files. And the keys could still not be loaded.

6.4 C_generateKey

[For this trace we generate a triple DES key, key length of 24 bytes.]

The generateKey function has 9 main components. 2 different secure messaging sessions are used to generate 24 bytes (same as the key length) on the card and send it to the API, and create the key file. The attribute file is created without the use of secure messaging as it does not contain sensitive information. The steps are listed below:

1. Open secure messaging (B.4 - commands 26, 27, 28)
2. Generate 24 random bytes and send them to the API via secure messaging (B.4 - command 29)
3. Close secure messaging (B.4 - command 30)
4. commands skipped include finding spare file for attributes and opening the file.
5. Update file with key attributes (B.4 - commands 40, 41)
6. Open secure messaging [again] (B.4 - commands 42, 43, 44)
7. Open key file directory via secure messaging (B.4 - command 45)
8. Create key file for triple DES key via secure messaging (B.4 - command 46)
9. Close secure messaging (B.4 - command 47)

From this analysis of the communication trace it seems that the card is requested to generate the key value from a 'get challenge' request (within secure messaging). We assume that these same bytes are used to be stored in the key file of the triple DES key we ask it to generate. We cannot verify this at this time, as two different secure messaging sessions are used, and therefore have different session keys. This causes the encryption of the generated bytes and the storing of them in a file to be different, even if the bytes are actually used as the key value. As stated in the literature review, previous hardware security modules created by 'Athena smartcard solutions' have in the past done this without the use of secure messaging. Thus that vulnerability seems to be patched in this version of the API and smartcard, by using secure messaging when creating these bytes and storing them in the key file.

If this assumption turns out to be correct it posses as quite a significant vulnerability that could be exploited. If the protocol for secure messaging can be reversed engineered, then we can inject our own bytes when the the request challenge is sent to the card. Or simply we can use a man in the middle attack to generate two sets of session keys. One with the attacker and the API, and one with the attacker and the smartcard. The bytes can be sent to the card in the correct format, however the attacker will also be in knowledge of the key value. This should never be the case when the attribute CKA_Extractable is set to false.

6.5 C_generateKeyPair

[For this trace we generate an RSA-1024 public/private key pair]

We find 15 main components in this generation of the RSA public and private key pair. 7 components transfer data that are wrapped within an ASN.1 BER encoding [?]. These are to transfer public key information and what we believe to be file location parameters. The steps listed below give the ordering of the most important commands, and below that we give the decoding of the ASN.1 BER encoding of those 8 commands that we have found.

1. Create public key attribute file (B.5 - command 54)
2. Add public key attributes to file [inc. public exponent] (B.5 - commands 56, 57)
3. Create private key attribute file (B.5 - command 59)
4. Add private key attributes to file (B.5 - commands 61, 62)
5. Create Private CTR RSA Key file (B.5 - command 64)
6. Select temporary file (B.5 - command 65)
7. Generate RSA Key Pair (B.5 - command 66)
8. Select temporary file (B.5 - command 67)
9. Get RSA public key (B.5 - command 68)
10. Select parent folder of private key attribute file (B.5 - command 69)
11. Create new file, with public modulus and additional info (B.5 - command 70
12. Select temporary file (B.5 - command 71)
13. Get RSA public key (B.5 - command 72)
14. Select public attribute file (B.5 - command 73)
15. Add public modulus to attribute file (B.5 - command 74)
16. The rest of the communication we believe to be resetting the temporary file, and adding file location information to file control parameters of parent files. (These commands are not included in the traces within the appendix)

The following are the ASN.1 BER decodings of the commands that have these wrapping. An online tool was used in the aid of decoding these hexadecimal bytes [?]. We provide references to the above numbering of the commands, and the appexidx references as well.

1. Create public key attribute file (B.5 - command 54)
Application 2(4 elem)

[10] (1 byte) 04
[03] (2 byte) 01 40
[00] (2 byte) 01 A7
[06] (8 byte) 00 20 00 20 00 20 00 20

3. Create private key attribute file (B.5 - command 59)
Application 2(5 elem)
[10] (1 byte) 04
[03] (2 byte) 02 00
[00] (2 byte) 01 23
[04] kx s0
[06] (8 byte) 00 00 00 20 00 20 00 20

5. Create private CTR RSA key file (B.5 - command 64)
Application 2(6 elem)
[10] (1 byte) 04
[03] (2 byte) 00 41
[00] (2 byte) 00 80
[05] (5 byte) 05 0C 20 00 A3
[06] (14 byte) 00 00 00 FF 00 FF 00 20 00 20 00 00 00 20
Application 17(0 elem)

7. Generate RSA key pair (B.5 - command 66)
[12] (2 elem)
[00] (1 byte) 06
[01] (3 byte) 01 00 01

9,13. Get RSA public key (B.5 - commands 68, 72)
Application 73(2 elem)
[01] (128 byte) D1 EF 7C A5 06 A1 87 FD 5F 13 5B 25 B7 16..
[02] (3 byte) 01 00 01

11. Create new file with public modulus and additional info (B.5 - command 70)
Application 2(6 elem)
[10] (1 byte) 04
[03] (2 byte) 00 81
[00] (2 byte) 00 80
[05] (5 byte) 05 08 20 00 A3
[06] (14 byte) 00 00 00 FF 00 FF 00 20 00 20 00 00 00 20
Application 17(2 elem)
[16] (3 byte) 01 00 01
[17] (128 byte) D1 EF 7C A5 06 A1 87 FD 5F 13 5B 25 B7...

RSA key pair generation is supported on the card. A dedicated processor is used for this, hence the need to change to temporary files to access the generated key and then store the public information. We see no sensitive information within the traces that are outputted in plain text. We thought that the additional information provided that is wrapped within the ASN.1 BER encoding, especially for **5**, might have exported the private key and the additional parameters required for CRT RSA keys [?]. To test this theory we deleted the generated key and generated another. The public modulus did change, however the remaining parameters did not. With a change in the public modulus, a change in the private exponent and therefore CRT parameters would also occur. Thus we concluded from that the private key is not exported out in plain text.

It seems these parameters that are listed above are used used to save the location of the RSA private key, and the public key for that matter. We have not been able to intuitively decode the proprietary encoding of file directories to locate the private key. As we will see later in the 'unwrap' function when we utilise the private key, this occurs within secure messaging and therefore cannot find the location of it from that either.

Thus without the knowledge of the proprietary encoding of file directories we do not see any vulnerabilities within the function as of yet. Studying multiple rsa key generations (without deleting the first keys) might show differences in these decodings which could lead to finding the location of the private key.

6.6 C_destroyObject

[For this trace we delete/destroy a triple DES key and its attribute file]

In the analysis of the destroyObject function we omitted the finding of the objects as that is the job of the findObject function and the authentication processes. However these are both crucial parts to the functions correct operation. Once the key and attribute files have been located and the user is authenticated with the card, there are 6 main components to the destruction of the object. These are listed below:

1. Select counter file for key attributes (B.6 - command 49)
2. Update counter (B.6 - command 50)
3. Select key file (B.6 - command 53)
4. Delete key file (B.6 - command 54)
5. Select key attribute file (B.6 - command 55)
6. Delete key attribute file (B.6 - command 56)

We believe a counter file keeps track of how many keys have been created and deleted in a certain path within the cards operating system. We are unsure on its specific use as we have only seen it being updating upon deletion of an object, and the new counter value being appended to the new keys attributes (the one created after deleting an object). The original and updated counter value can be seen at B.6 - command 37 and 50 respectively.

Once this has been done the key attribute file and the key file are selected and deleted using the inter-industry command 'DELETE FILE'. This is the first time we have seen the location of a key file been accessed without secure messaging. Thus we thought this might present a vulnerability which would allow us to open and read key values at the APDU level.

Hence we attempted to use the 'SELECT FILE' command and then 'READ BINARY' command to try and read the files data. This resulted in an error message = '69 81', meaning the command is incompatible with the file structure. With this in mind we decided to try the other command which is 'OPEN FILE CONTROL PARAMETERS' which did successfully work. The output was wrapped in an ASN.1 BER encoding, thus we decoded it which gave the following result:

Application 2(6 elem)
[07] (1 byte) 08
[03] (2 byte) 00 C1
[00] (2 byte) 00 18
[10] (1 byte) 04
[06] (14 byte) 00 00 00 FF 00 FF 00 00 00 00 00 00 00 00
[05] (4 byte) 01 0C 10 00

This seemed quite similar to what we noticed with the RSA private key analysis in the previous section. There is not present a 16 byte key that could be used for triple DES. Rather it seems that these are more pointers to more files that could hold the keys value. All security polices that we have read for 'Athena smart-card solutions' suggest that keys are held in a non-readable memory to outside actors, and only when required are encrypted internally (by the cards OS) and stored in the EEPROM for them to be used for the security operation required of them.

Thus we did not find any security vulnerabilities from this function.

6.7 C_encrypt

[For this function we use a triple DES key to encrypt a string 'TestString123456'] using ECB mode]

Just like the destroyObject function we omit the APDU communication for

login authentication and finding the object to use for encryption. These are still essential parts to the correct use of the encrypt function though.

The commands given for this trace are actually repeated and therefore occur twice for one encrpytion of a given string. This is due to PKCS #11 library being programmed in C by the card manufactures. The resulting length of the encryption is thought to be of an undetermined size. (For block ciphers and even RSA this can be pre-calculated given the input length) Therefore the implementation runs the encryption twice, the first run is to calculate the resulting length of the encryption, and the second result will save the result in a buffer that has been pre-allocated the correct amount number of bytes in a char array.

This function only has two main components to it, these are listed below:

1. Select key file (B.7 - command 52)
2. Encrypt data (B.7 - command 53)

The key file is selected and then the encryption APDU command is called with the string in the data field. As stated in destroyObject function analysis, despite the key file location being revealed here. It is not a file that can be opened nor reveals the value of the key. Instead is a pointer to other locations in memory that hold the key value. We assume that only the operating system has access to these locations but are unable to verify this hypothesis.

Due to this we see no vulnerabilities that could be exploited.

6.8 C_decrypt

For this trace we use the same triple DES key to decrypt the message we previously encrypted

Just like the destroyObject and encrypt function we omit the APDU communication for login authentication and finding the object to use for decryption. These are still essential parts to the correct use of the decrypt function though.

The decrypt function operates nearly an identical manner to the encrypt function, just simply changes one byte in command 65. The key is loaded and then the decrypt APDU command is sent, with the encrypted string placed in the data field. The commands are listed below:

1. Select key file (B.8 - command 64)

2. Decrypt data (B.8 - command 65)

As the decrypt function operates so similarly to the encrypt function, we saw no vulnerabilities present here as well. However we did notice the implementation flaw that had been picked up within previous work. For some reason the first 8 bytes are removed from the encrypted data. This therefore only decrypts and returns the second 8 bytes. As this does not present a security vulnerability, and rather is an issue with the functionality of the decrypt function, we do not analyse it any further.

6.9 C_setAttribute

[For this trace we modify the label of the triple DES key from 'des3' to 'changed']

As we did in the previous 3 functions we omit the login authentication and finding of the object. We only present the commands that are used to alter the attribute we tested. There were only 2 main components for this. The first one selected the attribute file, and the following 2 commands update the whole file to include the new name of the label to the key. Command chaining was used as the number of bytes within the attribute file exceeded the limit of 256. Command chaining is indicated by the use of '90' instead of '80' as the CLA byte. The commands are listed below:

1. Select key attribute file (B.9 - command 51)
2. Alter attribute file with the change in the label name (B.9 - commands 52,53)

Here we notice that at the APDU level attributes can be changed as long as a user is logged in. (We have not tested doing this without being logged in, but might be possible) This is a vulnerability that was discovered in previous work. With the main focus on changing the attribute values of CKA_sensitive and CKA_extractable from false to true. The change in this direction is forbidden by the PKCS #11 API, however this can be conducted at the APDU layer. As stated in the literature review, this had been carried out and due to the split between attribute files and key value files, there was no noticeable difference. Despite those attributes being set to True, the key value could still not be revealed.

Thus, will there is a vulnerability present here. It does not cause a major security issue, in the sense that the key value is still stored securely.

6.10 C_unwrap

[For this trace we used an RSA public key to encrypt a tripe DES key value = 12345678. With its encryption saved, we created a template and used the RSA private key to unwrap the key value and template, to save the key on the card]

We omit the RSA encryption of the key, login authentication and finding the objects from the communication traces. We only present the unwrapping of the key and its template to form a new key that will be saved on the card.

There are 10 main components involved in the unwrapping of a key. 2 different secure messaging sessions are used (just like in the generateKey function). These first secure messasging session is used to unwrap the encrypted key into a temporary file. The second secure messaging session is used to save that key into a key file. This comes after the attribute file has been created in between these two. The order of the commands are listed below:

1. Open secure messaging (B.10 - command 92, 93, 94)
2. Select a file [the file location is encrypted and therefore unknown] (B.10 - command 95)
3. Unwrap key and attributes (B.10 - command 96)
4. Close secure messaging (B.10 - command 97)
5. Select key attribute file (B.10 - command 119)
6. Add key attributes to file (B.10 - command 120, 121)
7. Open secure messaging (B.10 - command 122, 123, 124)
8. Select the key file directory (B.10 - command 125)
9. Create key value file (B.10 - command 126)
10. Close secure messaging (B.10 - command 127)

In a scenario were we are not already in the knowledge of the key value, the secure messaging sessions are the main vulnerability we would wish to try and exploit here. While the key is not exported at the APDU layer in plain text. A successful reverse engineering of the secure messaging session, would allow an attacker to exploit it, inject his own session keys and decrypt the key value that is being stored. This was mentioned in the generateKey function analysis.

Furthermore an attacker could also modify the attributes on the fly if the attacker was able to mount a man in the middle attack of the attributes of

the key as they are being created. We did state that this made little difference to exporting the key value once the attributes had been set, however while the key is being generated is different. This would export the key. But is an attack that requires to be mounted as the key is being unwrapped, or generated in the generateKey function. The attacks that would be more useful is being able to extract the key value once it is already saved within the card.

6.11 C_wrap

The Tookan attack was mentioned within the literature review. It is an attack at the API level whereby one key can be given wrapping and decryption functionalities within its attributes. This allowed for any key (no matter the attributes) to be wrapped, exported out of the card, and then decrypted by the same key, and have its key value in plain text. This vulnerability bypassed all controls that the attributes of a key should provide. Due to this vulnerability discovery, we assume that 'Athena smartcard solutions' decided to remove the ability to wrap keys completely. This assumption came from that fact that requesting any key to wrap another key at the API level caused an error: *CKR_Function_Not_Supported*.

From analysing table 2.1 (section 2.24), we noticed a correlation between the P1 and P2 parameters for the APDU commands for encryption, decryption and unwrapping. These values are the following:

P1	P2	Command
82	0E	encrypt
80	0E	decrypt
80	0A	unwrap

The CLA and and INS bytes are identical for all of the commands. From this we inferred that if the wrap APDU command were to still exist and it is only the API that has been modified to refuse to send the wrap command and instead throw an error. That the command for wrapping at the APDU level would be: **00 2a 82 0a L Data 00**. Were L is the length of the data, and the data is the key to be wrapped.

We ran a simple test to see if this command operates on the card:

1. Wrap key (just entered the string '12345678') (B.11 - command 1)

The result of the test was an APDU level error. '6A 80', meaning the parameters within the data field are incorrect. Suggesting that the command does exist and most likely corresponds to the wrap function. The data field of unwrap is encrypted, as it is used within secure messaging. However

if this was reversed engineered and we could understand the data field of the unwrap command, it is highly likely that we would be able to create the correct data field for the wrap command. This would allow us to bypass all attribute controls of keys at the API level, and implement the Tookan attack at the APDU level.

Chapter 7

Attempts To Attack At the APDU Level

In this chapter we explain the design, implementation and results of the attacks we attempted at the APDU level. The two main attacks we focus on are reverse engineering the PIN authentication protocol, and reverse engineering the secure messaging protocol. The motivations behind this come from literature stated in chapter X.

---

* Here give an overview of each functions possible vulnerabilities and present the motivations behind the attacks we choose. **

The smartcard we analyse uses a challenge-response algorithm to prevent the PIN being sent over in plain text. This is an improvement upon other card vendors implementations whereby they do send the PIN over in plain text for it to be verified. Successfully reverse engineering this challenge-response protocol will allow an attacker to calculate the victims PIN given one successful login trace. Furthermore, it will also remove the first dependency on the use of the API. Being able to communicate to the card directly gives an attacker a higher capability. But as of right now, since we are unaware of how the login occurs the API must be used.

The second dependancy on the use of the API, is secure messaging. When the generateKey function from the API is called to create block cipher keys, it is the computer that requests the card to generate the key, send it over using secure messaging, and then use secure messaging again to place the key into a file to be stored on the card. This again is an improvement upon previous hardware security modules from the manufacture 'Athena- Smartcard solutions' (specifically ASE KeyPro), whereby this was completed without the use of secure messaging, and therefore the block cipher keys where needlessly exported over in plain text which would allow an attacker to sniff them. If this protocol is successfully reversed engineered, it will be possible

to sniff the keys by decrypting the messages which contain the keys that are in transit from computer to the smart-card.

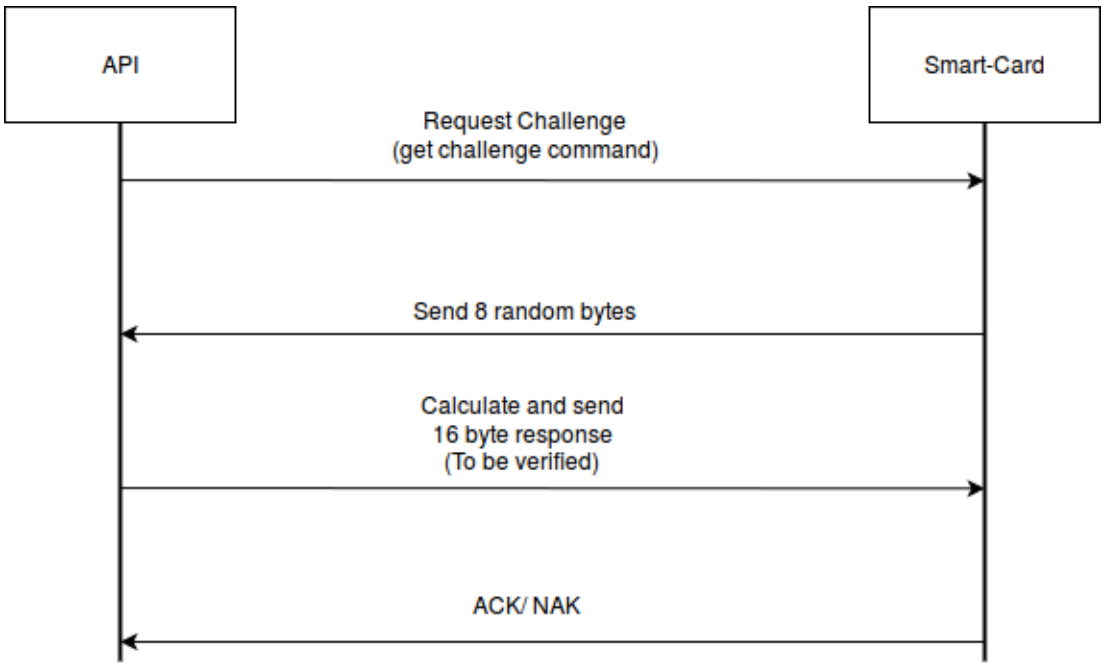
7.1 Reverse Engineering PIN Authentication Protocol

7.2 Reverse Engineering PIN/password Authentication 1st draft

The first attack that I decided to attempt is to reverse-engineer the PIN authentication method. The reasoning behind this is because if this can be successfully done, the PIN can then be inferred from one communication trace sniffed between smart-card and the API (on the computer). The inference comes from the fact that once the method is deduced, an attacker can simply brute force the possible combinations of a PIN, to test if a match is found. (This becomes clearer in the latter sections)

From previous work completed on this card by an MSC student last year [?], and from the analysis conducted in section 6.2, it was quite clear that the card has the following characteristics in terms of PIN authentication. The PKCS#11 API requests a challenge, the smart-card responds with an 8 byte challenge, the API then calculates a 16 byte response (using the 8 byte challenge, and the PIN), the smart-card verifies whether or not the response is correct. There are two response formats to that APDU verification command:

- '90 00' $\rightarrow$ verification succeeded, correct PIN was entered
- '63 CX' $\rightarrow$ verification failed (where X is the number of attempts left before the card is blocked)



The following sections are explanations of the searches that we conducted to try and reverse-engineer the protocol showed above. To give a full understanding of how challenging this part of the project was, we will explain the combinations of possibilities we checked, and the reasoning behind each of them. These will be split up into different 'searches', and increment in terms of new findings and understanding of how the protocol may be implemented.

To aid these explanations, we introduce here 3 key sub-functionalities that most of the conducted searches use. Table 7.1 lists all the hash functions (see section 3.1) that were used, and provides the output length in bits & bytes. Those hash functions were all supported by openSSL and the python package 'hashlib'. Table 7.2 provides the names of the bitwise logical operations that were used to 'join' two bytes together. And table 7.3 provides the description of truncation methods used to reduce the output size of a search down to 16 bytes to match the response provided by the API.

From here on, in the explanation of the searches I will just refer to HASH, JOIN & TRUNCATE which will suggest that all of the elements in the tables have been iterated over and preformed on. For example TUNRCATE(HASH('this is a string')), means the string, 'this is a string', is to be hashed with all the functions in table 7.1, and then truncated to 16 bytes using all the methods listed in table 7.3.

Hash Name	Output Length (bits)	Output Length (Bytes)
MD4	128	16
MD5	128	16
MDC2	128	16
RIPEMD160	160	20
SHA	160	20
SHA1	160	20
SHA224	224	28
SHA256	256	32
SHA384	384	48
SHA512	512	64
WHIRLPOOL	512	64

Table 7.1: Hash Functions

Logical Operations
AND
OR
XOR
NOT AND
NOT OR
NOT XOR

Table 7.2: Bitwise Logical Operations (Joins)

Truncation method	Description?
first_16	Truncates the output by taking the first 16 bytes
last_16	Truncates the output by taking the last 16 bytes
mod_16	Truncates the output by taking modulus 2^{128} [We use 128 because that's the number of bits in 16 bytes]

Table 7.3: Truncation Methods

Before any searches could be conducted, the first task was to extract the values of the 8 byte challenge (denoted X), and the 16 byte response (denoted Y), from a communication trace of C_login. Table 7.4 provides the values for the PIN, X and Y in hexadecimal format.

Data	ASCII	HEX
PIN	'0000000000000000'	30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
Y	N/A	53 17 55 20 F4 30 18 56 80 E6 75 55 E1 91 A7 EC
X	N/A	68 F1 E4 92 85 36 39 A3

Table 7.4

In the very first original experiments, we made assumptions as did previous work on this card [ref Msc] that the PIN number was only numerical characters, only had 4-8 digits and if the PIN was less than 8 characters, it was padded using a special character to form 8 bytes. These assumptions turned out to be false, and thus the elementary experiments were all flawed from the beginning. I have not included a description of those experiments in the following sections, as many of them were in fact very similar to those explained below, just with different assumptions of the PIN. They also used a PIN for the card that was '12345', and hence there was an exponential explosion in the number of experiments for a particular search, due to needing to test for different padding schemes and characters. Hence why in these following experiments the PIN had been set to '0000000000000000' (16 zeros, no need for padding).

7.2.1 Authentication Protocol Search 1.0 (Password Storage)

Assumptions:

- PIN consists of alpha - numeric characters
- PIN is a maximum of 16 bytes
- PIN is encoded in ASCII characters
- For any PIN that is less than 16 bytes long, there is padding character used to pad the PIN to 16 bytes

Search 1 - Hash functions
In this initial stages we thought that there is a large possibility that the 16 byte response was generated by hashing a combination of the PIN and the 8 byte challenge. This was partly due to common practices used in industry

whereby users passwords are often hashed, and in most cases salted (see section 3.1), before storing them in databases. This practice is more secure than storing plain text passwords, as if an attacker were to gain access to the back end databases storing said passwords, the password itself would not be available to see. For authentication the password is just hashed (and salted, if a salt is used), and then compared against to the stored value. The fact that from multiple traces the 16 byte response seemed to be uniformly random, it supported this hypothesis.

Thus the first search that we completed focused on the fact that a hash function was used to produce the 16 byte response. Below we have listed the methods tested in expirments to generate a 16 bytes, given X and the PIN.

We donote ‖ as the concatination function. Thus for two strings ’string1 ’ ‖ ’string2’ = ’string1 string2’.

Methods tested that produced a 16 byte output using X and the PIN

- join(truncate(hash(X)), pin))
- truncate(hash(join(pin, X ‖ X)))
- truncate(hash(pin‖X))
- truncate(hash(X‖pin))
- truncate(hash(pin+X))
- truncate(hash(join(pin, square(X))))

[The methods should be read from the most inner brackets, outward. Therefore this means that the first method dictates that X is first hashed using one of the hash functions listed in the table 7.1. The output of that is then truncated to 16 bytes using one of the functions from table 7.3. All iterations of the functions in the tables were tested.]

The following experiment resulted in 2592 individual tests, but did not find a match to the response generated by the API [Y in table 7.4]. Thus we moved onto search 2.

Search 2 - PBKDF2

Following the failure of search 1, but still assuming there is a large possibility of a hash function being used due to the common practices mentioned in search 1, and the characteristics known so far of the 16 byte response Y. Then we decided to look at the password-based key derivation function (PBKDF2), which was created as part of PKCS #5 by RSA laboratories [?]. It has been mentioned in literature [?], and started to be used for more secure password storage as well as for key derivation. Essentially PBKDF2 takes

as input, a password (the PIN), a salt (the 8 byte challenge X), a hash function, and the number of iterative rounds. If the number of iterative round is set to 10, then the salted password would be hashed once, and the output of that would be the input for the next round of hashing. This would be completed 10 times. Literature [?] has shown that the standard for the number of iterative rounds used to be 10,000, back in 2008 (check this date). Now it is suggested to use as many rounds as is computationally feasible by the device. Due to the processing power of a smart-card I assumed that this would not exceed 100,000 rounds, in the case that PBKDF2 was used.

Hence this search generated experiments that ran through $1 \rightarrow 100{,}000$ rounds of PBKDF2. As the default of PBKDF2 is to truncate the output by taking the first X (X here is a variable) bytes only 'first_16' truncation method was used.

Methods tested that produced a 16 byte output using X and the PIN

- PBKDF2(hash_function, PIN, X, number_of_rounds)

This generated 100,000 experiments per hash function. With 12 hash functions, this resulted in 1.2 million tests being run. Due to the sheer computational power required for this search I decided to parallelize the search based on the hash function, and run them on separate cores of a server. Even by improving the efficiency of this search, it still took 2 weeks to conduct.

Again this unfortunately did not result in a match between the 16 byte responses calculated and Y (the API's response). Hence we move onto search 3.

7.2.2 Authentication Protocol Search 2.0 (One Time Passwords)

Search 3 - OCRA: OATH Challenge-Response Algorithm

With no luck of deducing the method the authentication protocol uses, we decided to look into more complex standards that exist and used in different parts of the computing industry for challenge response protocols, rather than password storage techniques. The international engineering task force (IETF) released a paper in 2011 [?]. Section 7.1 of the paper gives a one-way challenge response algorithm which fitted the characteristics of the authentication that takes between the API and the smart-card.

Section 3.5.1 explains hash based one time passwords. But in essence HTOP is:
$HOTP(K,C) = Truncate(HMAC(K,C))\&0x7FFFFFFF$ [?-wiki]

Still to complete.

Search 4 - TOTP
With the above in mind, I also wanted to rule out TOTP. This was completed by halting communication between the smart-card and API using the man-in-the-middle tool (see section 4.2). I did this for upto 2 hours. We were looking for a failed verifaction, despite having the correct PIN. The failure should have been caused by the delay in the response if TOTP was used. This was not found and therefore ruled out the possibility of TOTP.

This section will be explained better, and will also include reasoning behind why TOTP is not often used (time sync problems).

7.2.3 Authentication Protocol Search 3.0 (Triple DES Encryption)

Need to explain multiple logins and the characteristics found!
That lead me to believe DES3 encryption was used. Tabulate and cross out other possibilites!

Search 5
Need to decrypt two different encryptions with different PINs in order to find out if this is the method used!
Raw ASCII password and MD5 hash (due to output size = 16 bytes)
DES3-ECB $\rightarrow$ encrypt (Na $\|$ Vac)

Search 6

7.3 Block Cipher Injection (MiTM)

7.4 Manually Overriding Attributes

This attack was found by previous literature. We simply tested to see if the same attack was feasiable on the smart card we analyse.

To test this we created a DES3 key on the card that had the encrypt attribute set to False. We asked the card to encrypt the string '12345678'. The API returned an error stating that this function was not supported due to the attribute settings. We then override the API by manually sending the command overselves. This results in the encryption taking place on the card, when in-fact it should not.

Chapter 8

Conclusion / Results

This shall summarise the whole report and my findings in regard to low-level vulnerabilities on the card.

Chapter 9

Future work

1. complete the reverse engineering secure messaging
2. If SM is known get wrap functionality to work from APDU layer
3. Try using ’reallocate binary’ to change retry counter

Bibliography

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Albert Einstein. *Zur Elektrodynamik bewegter Körper*. (German) [*On the electrodynamics of moving bodies*]. Annalen der Physik, 322(10):891âĂŞ921, 1905.

[3] Knuth: Computers and Typesetting,
http://www-cs-faculty.stanford.edu/~uno/abcde.html

[4] Frank Morgner : Creating a Virtual Smart Card
https://frankmorgner.github.io/vsmartcard/virtualsmartcard/README.html

[5] Internet Engineering Task Force (IETF).
OCRA: OATH Challenge-Response Algorithm, June 2011
https://tools.ietf.org/html/rfc6287

Appendices

Appendix A

Attack Traces

A.1 Multiple C_login Traces

A.1.1 Different Pin, Different Nonce

```
----- APDU command/response pair 0 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                        .....

00000000: 00 00 00 00 00 00 00 00  90 00                        ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 C2 29 5F  78 D1 68 29 13 78 BE 7B  . ....)_x.h).x.{
00000010: 8E 61 9B 32 2E                                        .a.2.

00000000: 63 C9                                                 c.

----- APDU command/response pair 1 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                        .....

00000000: 00 00 00 00 00 00 00 01  90 00                        ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 C4 6D 44  03 92 F3 6B EF 13 18 07  . ....mD...k....
00000010: CE 5A A4 B9 27                                        .Z..'

00000000: 63 C8                                                 c.
```

A.1.2 Different Pin, Same Nonce

```
----- APDU command/response pair 0 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                  .....

00000000: 00 00 00 00 00 00 00 00  90 00                  ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 8F 58 1B  91 BC 78 D3 37 A9 D9 FB  . ....X...x.7...
00000010: 9C 20 58 F6 0A                                  . X..

00000000: 63 C9                                           c.

----- APDU command/response pair 1 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                  .....

00000000: 00 00 00 00 00 00 00 00  90 00                  ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 E0 30 33  BB 03 0F 6E 11 08 C0 8D  . ....03...n....
00000010: 1D 9D 85 C4 A6                                  .....

00000000: 63 C8                                           c.
```

A.1.3 Same Pin, Different Nonce

```
----- APDU command/response pair 0 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                  .....

00000000: 00 00 00 00 00 00 00 00  90 00                  ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 6E 78 D4  D5 61 AD 3C 26 D3 89 E8  . ...nx..a.<&...
00000010: 96 B9 92 0D 40                                  ....@

00000000: 63 C9                                           c.

----- APDU command/response pair 1 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                  .....
```

```
00000000: 00 00 00 00 00 00 00 01  90 00                    ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 6E 78 D4  D5 61 AD 3C 26 BC C6 AA  . ...nx..a.<&...
00000010: 72 D3 95 2B 94                                    r..+.

00000000: 63 C8                                             c.
```

A.1.4 Same Pin, Same Nonce

```
----- APDU command/response pair 0 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                    .....

00000000: 00 00 00 00 00 00 00 00  90 00                    ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 EE D2 F1  54 07 18 8A 8F AB A3 F7  . ......T.......
00000010: 3E 64 17 2D 6E                                    >d.-n

00000000: 63 C9                                             c.

----- APDU command/response pair 1 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                    .....

00000000: 00 00 00 00 00 00 00 00  90 00                    ..........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 EE D2 F1  54 07 18 8A 8F AB A3 F7  . ......T.......
00000010: 3E 64 17 2D 6E                                    >d.-n

00000000: 63 C8                                             c.
```

A.1.5 Different Second

```
----- APDU command/response pair 0 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                    .....

00000000: 00 00 00 00 00 00 00 00  90 00                    ..........
```

```
----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 61 50 65  E1 AF 05 7B C3 35 98 0D  . ...aPe...{.5..
00000010: DC 9D C5 42 96                                    ...B.

00000000: 63 C9                                             c.

----- APDU command/response pair 1 -----
(Inter-Industry) Get Challenge
00000000: 00 84 00 00 08                                    .....

00000000: 00 00 00 00 00 00 00 00  90 00                    .........

----- APDU command/response pair 1 -----
(Proprietary) Verify
00000000: 80 20 00 00 10 BF 73 83  F9 30 B1 74 D2 E4 98 83  . ....s..0.t....
00000010: 3A 9F 1F 37 BA                                    :..7.

00000000: 63 C8                                             c.
```

A.2 Successful Login Injection

```
----- APDU command/response pair 12 -----

(Inter-Industry) Get Challenge
COMMAND from API
00000000: 00 84 00 00 08                                    .....

Do you want to automate the injection your own login response? (Y/n)

RESPONSE
00000000: E7 69 60 B5 C8 FC D2 02  90 00                    .i‘.......

----- APDU command/response pair 13 -----

(Proprietary) Verify
COMMAND from API
00000000: 80 20 00 00 10 4A D1 3D  AB 98 7F C5 18 A9 B3 1F  . ...J.=........
00000010: 2F 96 B4 3C AF                                    /..<.

(Proprietary) Verify
COMMAND injected
00000000: 80 20 00 00 10 32 5A 9F  38 CA 4F BE 44 3A CD E1  . ...2Z.8.O.D:..
00000010: C5 03 84 35 DF                                    ...5.

RESPONSE
00000000: 90 00                                             ..
```

A.3 Open Secure Messaging Traces

A.3.1 Generator = 5, [Not modified]

```
----- APDU command/response pair 24 -----

(Proprietary) Get Card Public Key
COMMAND from API
00000000: 80 48 00 80 00                                          .H...

Do you want to to alter command? (y/N)

RESPONSE
00000000: 80 01 05 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{`g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....`.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                             a.

Do you want to alter the response? (y/N)

----- APDU command/response pair 25 -----

(Inter-Industry) Get Remaining Bytes
COMMAND from API
00000000: 00 C0 00 00 09                                          .....

Do you want to to alter command? (y/N)

RESPONSE
00000000: A8 5D D3 30 E3 5C A9 00  39 90 00                       .].0.\..9..

Do you want to alter the response? (y/N)

----- APDU command/response pair 26 -----
```

```
(Proprietary) Open Secure Messaging
COMMAND from API
00000000: 80 86 00 00 80 08 9F EA  A1 DC 8F C3 43 FD FD 4A  ............C..J
00000010: E6 95 7E C0 D3 C6 FE 81  61 59 4B CE 45 21 96 63  ..~.....aYK.E!.c
00000020: 0F AB 19 D8 61 1A B2 6B  00 E2 44 0F 06 A3 5B 60  ....a..k..D...[‘
00000030: 87 76 C0 B7 E9 15 D5 50  DB 17 D6 C1 3C 26 54 47  .v.....P....<&TG
00000040: AA A3 4B DC 2C 14 81 08  84 0D F0 CA FB 49 8B C1  ..K.,........I..
00000050: B1 0B A1 2B 86 20 02 F2  0F 69 F0 56 2C 83 0C 6E  ...+. ...i.V,..n
00000060: A6 6A E9 86 56 47 71 24  0C B7 91 7F 37 85 0A D4  .j..VGq$....7...
00000070: 12 35 1F CE 17 6C D2 52  FB 04 24 CF DD E9 53 BE  .5...l.R..$...S.
00000080: DA 26 EA 54 FB 00                                 .&.T..
Do you want to to alter command? (y/N)

RESPONSE
00000000: 66 56 36 31 16 42 8D 8A  BC 06 BA AC 5D 35 26 F5  fV61.B......]5&.
00000010: BF 58 15 7F 00 4F EF 2F  54 FB C4 F2 10 8F CB D6  .X...O./T.......
00000020: 90 00                                             ..

Do you want to alter the response? (y/N)

----- APDU command/response pair 27 -----

(Proprietary) Get Challenge [SM]
COMMAND from API
00000000: 0C 84 00 00 0D 97 01 20  8E 08 05 E4 4A 19 32 DE  ....... ....J.2.
00000010: 51 CB 00                                          Q..

Do you want to to alter command? (y/N)

RESPONSE
00000000: 87 29 01 BD 69 F3 85 A7  98 2E 08 07 21 88 30 2F  .)..i.......!.0/
00000010: 06 FF 93 E4 2F 31 C5 4A  40 FB 45 3A 45 C1 4A 84  ..../1.J@.E:E.J.
00000020: 7F BA 59 BC 44 8A 70 A0  BC DA FB 99 02 90 00 8E  ..Y.D.p.........
00000030: 08 44 26 95 74 6A 51 A3  72 90 00                 .D&.tjQ.r..

Do you want to alter the response? (y/N)

----- APDU command/response pair 28 -----

(Proprietary) Close Secure Messaging
COMMAND from API
00000000: 80 86 FF FF                                       ....
```

A.3.2 Generator = 1

```
----- APDU command/response pair 24 -----

(Proprietary) Get Card Public Key
```

```
COMMAND from API
00000000: 80 48 00 80 00                                        .H...

Do you want to to alter command? (y/N)

RESPONSE
00000000: 80 01 05 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{`g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....`.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                             a.

Do you want to alter the response? (y/N)
y

00000000: 80 01 01 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{`g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....`.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                             a.
response changed!

----- APDU command/response pair 25 -----

(Inter-Industry) Get Remaining Bytes
```

```
COMMAND from API
00000000: 00 C0 00 00 09                                        .....

Do you want to to alter command? (y/N)

RESPONSE
00000000: A8 5D D3 30 E3 5C A9 00  39 90 00                     .].0....9..

Do you want to alter the response? (y/N)

----- APDU command/response pair 26 -----

(Proprietary) Open Secure Messaging
COMMAND from API
00000000: 80 86 00 00 80 00 00 00  00 00 00 00 00 00 00 00  ................
00000010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 00 00 00 00 01 00                                     ......

Do you want to to alter command? (y/N)

RESPONSE
00000000: 7D 2C 25 47 1C 16 34 51  E9 C3 49 38 C8 79 1E ED  },%G..4Q..I8.y..
00000010: A2 6B 20 D4 54 BD 67 0A  D3 85 3E B9 E0 6E D5 5E  .k .T.g...>..n.^
00000020: 90 00                                             ..

Do you want to alter the response? (y/N)

----- APDU command/response pair 27 -----

(Proprietary) Get Challenge [SM]
COMMAND from API
00000000: 0C 84 00 00 0D 97 01 20  8E 08 08 C6 59 9B 57 E6  ....... ....Y.W.
00000010: B4 4E 00                                          .N.

Do you want to to alter command? (y/N)

RESPONSE
00000000: 69 88                                             i.

Do you want to alter the response? (y/N)

----- APDU command/response pair 28 -----
```

```
(Proprietary) Close Secure Messaging
COMMAND from API
00000000: 80 86 FF FF                                    ....
```

A.3.3 Generator = 0

```
----- APDU command/response pair 24 -----

(Proprietary) Get Card Public Key
COMMAND from API
00000000: 80 48 00 80 00                                 .H...

Do you want to to alter command? (y/N)

RESPONSE
00000000: 80 01 05 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{`g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....`.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                          a.

Do you want to alter the response? (y/N)
y

00000000: 80 01 00 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{`g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....`.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
```

```
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                           a.
response changed!

----- APDU command/response pair 25 -----

(Inter-Industry) Get Remaining Bytes
COMMAND from API
00000000: 00 C0 00 00 09                                  .....

Do you want to to alter command? (y/N)

RESPONSE
00000000: A8 5D D3 30 E3 5C A9 00  39 90 00               .].0....9..

Do you want to alter the response? (y/N)

----- APDU command/response pair 26 -----

(Proprietary) Open Secure Messaging
COMMAND from API
00000000: 80 86 00 00 80 00 00 00  00 00 00 00 00 00 00 00  ................
00000010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 00 00 00 00 00 00                               ......

Do you want to to alter command? (y/N)

RESPONSE
00000000: 7E 4B 40 E6 E3 B1 5D 25  2B 02 48 50 B3 63 CC 9E  ~K@...]%+.HP.c..
00000010: 79 41 34 FC 04 B3 57 1C  06 E3 D1 36 3C 24 45 8D  yA4...W....6<$E.
00000020: 90 00                                           ..

Do you want to alter the response? (y/N)

----- APDU command/response pair 27 -----

(Proprietary) Get Challenge [SM]
COMMAND from API
00000000: 0C 84 00 00 0D 97 01 20  8E 08 99 BD 52 69 31 DD  ....... ....Ri1.
00000010: DB FD 00                                        ...
```

```
Do you want to to alter command? (y/N)

RESPONSE
00000000: 69 88                                                  i.

Do you want to alter the response? (y/N)

----- APDU command/response pair 28 -----

(Proprietary) Close Secure Messaging
COMMAND from API
00000000: 80 86 FF FF                                            ....
```

A.4 Overriding Attribute Controls

A.5 Encrypt_False

Appendix B

API Function Traces

B.1 Initialization

```
----- APDU command/response pair 1 -----
00000000: 00 A4 04 00 0C A0 00 00  01 64 4C 41 53 45 52 00  .........dLASER.
00000010: 01 00                                            ..

00000000: 90 00                                            ..

----- APDU command/response pair 4 -----
00000000: 80 A4 08 00 06 3F 00 30  00 C0 00                .....?.0...

00000000: 90 00                                            ..


----- APDU command/response pair 5 -----
00000000: 00 B0 00 00 00                                   .....

00000000: 49 44 50 72 6F 74 65 63  74 20 20 20 20 20 20 20  IDProtect
00000010: 20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20
00000020: 41 74 68 65 6E 61 20 53  6D 61 72 74 63 61 72 64  Athena Smartcard
00000030: 20 53 6F 6C 75 74 69 6F  6E 73 20 20 20 20 20 20   Solutions
00000040: 49 44 50 72 6F 74 65 63  74 20 20 20 20 20 20 20  IDProtect
00000050: 30 44 35 30 30 30 30 39  32 31 32 32 38 37 39 36  0D50000921228796
00000060: 0D 04 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 10 00 00 00  04 00 00 00 FF FF FF FF  ................
00000080: 00 00 00 00 FF FF FF FF  00 00 00 00 01 00 01 00  ................
00000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000A0: 00 90 00                                         ...


----- APDU command/response pair 8 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 03 40 00          .....?.0.0.@.
```

```
00000000: 90 00                                           ..


----- APDU command/response pair 9 -----
00000000: 00 B0 00 02 64                                  ....d

00000000: 41 54 48 45 4E 41 53 4E  C0 AD AA 78 FC 88 42 0D  ATHENASN...x..B.
00000010: 90 00                                           ..
```

B.2 C_login

```
----- APDU command/response pair 10 -----
00000000: 80 A4 08 0C 04 3F 00 00  20 00                  .....?.. .

00000000: 62 2F 87 01 08 83 02 00  20 80 02 00 10 8A 01 04  b/...... .......
00000010: 86 0E 00 FF C0 30 00 FF  00 10 00 FF 00 10 00 00  .....0..........
00000020: 85 0F 00 01 00 00 AA 00  04 10 00 00 00 00 00 FF  ................
00000030: FF 90 00                                        ...

----- APDU command/response pair 11 -----
00000000: 80 A4 08 00 04 3F 00 00  20                     .....?..

00000000: 90 00                                           ..

----- APDU command/response pair 12 -----
00000000: 00 84 00 00 08                                  .....

00000000: 11 B7 B2 80 4B 17 0D A4  90 00                  ....K.....


----- APDU command/response pair 13 -----
00000000: 80 20 00 00 10 1D ED 9E  47 A8 C9 EA CE 37 82 2C  . ......G....7.,
00000010: 92 CF 07 20 2D                                  ... -

00000000: 90 00                                           ..


----- APDU command/response pair 20 -----
00000000: 80 28 00 00 04 00 00 00  20                     .(......

00000000: 90 00                                           ..
```

B.3 C_findObject

```
----- APDU command/response pair 32 -----
00000000: 80 30 01 00 00                                        .0...

00000000: D1 02 00 03 D2 02 03 40  D2 02 03 46 D2 0A 86 7F  .......@...F....
00000010: 63 6D 61 70 66 69 6C 65  90 00                    cmapfile..


----- APDU command/response pair 33 -----
00000000: 80 A4 08 00 06 3F 00 30  00 30 02                 .....?.0.0.

00000000: 90 00                                             ..


----- APDU command/response pair 34 -----
00000000: 80 30 01 00 00                                        .0...

00000000: D1 02 00 00 90 00                                     ......


----- APDU command/response pair 35 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 03 40           .....?.0.0..@

00000000: 90 00                                             ..


----- APDU command/response pair 36 -----
00000000: 00 B0 00 00 00                                        .....

00000000: 00 03 03 40 01 23 18 00  00 00 00 04 04 00 00 00  ...@.#..........
00000010: 00 01 00 00 01 01 00 02  00 00 01 00 00 03 10 00  ................
00000020: 04 64 65 73 33 FF FF FF  FF FF FF FF FF FF FF FF  .des3............
00000030: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF 00 11 01 00 18 FF FF  FF FF FF FF FF FF FF FF  ................
00000070: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF 01 00  ................
00000080: 00 00 04 15 00 00 00 01  02 10 00 01 01 FF FF FF  ................
00000090: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000A0: FF FF FF FF FF FF FF FF  FF FF FF FF 01 03 30 00  ..............0.
000000B0: 01 01 01 04 00 00 01 01  01 05 50 00 01 01 01 06  ..........P.....
000000C0: 00 00 01 00 01 07 50 00  01 01 01 08 50 00 01 01  ......P.....P...
000000D0: 01 0A 00 00 01 01 01 0C  10 00 01 00 01 10 10 00  ................
000000E0: 00 FF FF FF FF FF FF FF  FF 01 11 10 00 00 FF FF  ................
000000F0: FF FF FF FF FF FF 01 62  50 00 01 00 01 63 00 00  .......bP....c..
00000100: 61 27                                             a'
```

```
----- APDU command/response pair 37 -----
00000000: 00 B0 01 00 00                                        .....

00000000: 01 01 01 64 00 00 01 01  01 65 00 00 01 01 01 66  ...d.....e.....f
00000010: 00 00 04 31 01 00 00 01  70 00 00 01 01 80 10 00  ...1....p.......
00000020: 00 01 00 99 03 99 03 90  00                       .........


----- APDU command/response pair 38 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 03 46           .....?.0.0..F

00000000: 90 00

----- APDU command/response pair 39 -----
00000000: 00 B0 00 00 00                                        .....

00000000: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100: 61 2F                                             a/


----- APDU command/response pair 40 -----
00000000: 00 B0 01 00 00                                        .....

00000000: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 90  ................
00000030: 00                                                .
```

B.4 C_generateKey

```
----- APDU command/response pair 26 -----
00000000: 80 48 00 80 00                                        .H...

00000000: 80 01 05 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ..........p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{‘g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....‘.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                             a.


----- APDU command/response pair 27 -----
00000000: 00 C0 00 00 09                                        .....

00000000: A8 5D D3 30 E3 5C A9 00  39 90 00                     .].0....9..


----- APDU command/response pair 28 -----
00000000: 80 86 00 00 80 84 7F A0  E7 6C 8F AA 50 9C C3 6E  .........l..P..n
00000010: 82 5E 84 B6 E4 F6 77 1C  45 FA AB 06 1B 24 C4 A8  .^....w.E....$..
00000020: 92 03 A9 9C A8 2B BE 1B  28 C4 57 83 A5 5E BB 8D  .....+..(.W..^..
00000030: D2 BF 3F D5 02 8A 7C 13  10 9C 75 06 91 1A 0F 05  ..?...|...u.....
00000040: 55 B4 C9 12 8A 69 59 B6  07 1D 67 F2 8A C9 FA BC  U....iY...g.....
00000050: F3 BE 16 73 51 C0 76 0C  11 E5 0C D3 8C FE 09 E5  ...sQ.v.........
00000060: 1E 52 DE 38 D9 AC 2D EB  C6 A1 C4 8E ED 03 7D 07  .R.8..-.......}.
00000070: 85 B7 FE 66 82 2F 03 65  94 DC 27 77 2B 3A 28 71  ...f./.e..’w+:(q
00000080: 97 08 5D 03 80 00                                 ..]...

00000000: F9 D0 66 F7 48 CB BB E8  CE 93 60 05 99 1B 81 2E  ..f.H.....‘.....
00000010: 73 0B B7 B8 DC 10 A7 84  B3 99 D8 C8 60 D6 48 5A  s...........‘.HZ
00000020: 90 00                                             ..


----- APDU command/response pair 29 -----
00000000: 0C 84 00 00 0D 97 01 18  8E 08 2B 88 7C 0C 8C 24  .........+.|..$
00000010: 00 1F 00                                          ...
```

```
00000000: 87 21 01 69 AB B7 01 F5  F5 8E EA B8 F3 09 D7 5E  .!.i...........^
00000010: F5 26 3C 7F 1D 15 90 B8  40 D4 A1 85 9C 57 3F 27  .&<.....@....W?'
00000020: 87 84 C6 99 02 90 00 8E  08 42 84 88 19 99 3B C2  .........B....;.
00000030: 10 90 00                                          ...


----- APDU command/response pair 30 -----
00000000: 80 86 FF FF                                      ....

00000000: 90 00                                            ..


----- APDU command/response pair 39 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 03 40          .....?.0.0..@

00000000: 90 00                                            ..


----- APDU command/response pair 40 -----
00000000: 00 D6 00 00 FA 01 03 03  40 01 23 18 00 00 00 00  ........@.#.....
00000010: 04 04 00 00 00 00 01 00  00 01 01 00 02 00 00 01  ................
00000020: 00 00 03 10 00 04 64 65  73 33 FF FF FF FF FF FF  ......des3......
00000030: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF FF FF FF FF FF 00 11  01 00 18 FF FF FF FF FF  ................
00000070: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000080: FF FF FF 01 00 00 00 04  15 00 00 00 01 02 10 00  ................
00000090: 01 01 FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000B0: FF 01 03 30 00 01 01 01  04 00 00 01 01 01 05 50  ...0...........P
000000C0: 00 01 01 01 06 00 00 01  00 01 07 50 00 01 01 01  ...........P....
000000D0: 08 50 00 01 01 01 0A 00  00 01 01 01 0C 10 00 01  .P..............
000000E0: 00 01 10 10 00 00 FF FF  FF FF FF FF FF FF 01 11  ................
000000F0: 10 00 00 FF FF FF FF FF  FF FF FF 01 62 50 00     ............bP.

00000000: 90 00                                            ..


----- APDU command/response pair 41 -----
00000000: 00 D6 00 FA 2D 01 00 01  63 00 00 01 01 01 64 00  ....-...c.....d.
00000010: 00 01 01 01 65 00 00 01  01 01 66 00 00 04 31 01  ....e.....f...1.
00000020: 00 00 01 70 00 00 01 01  80 10 00 00 01 00 88 03  ...p............
00000030: 88 03                                            ..

00000000: 90 00                                            ..
```

```
----- APDU command/response pair 42 -----
00000000: 80 48 00 80 00                                        .H...

00000000: 80 01 05 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{`g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....`.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                             a.


----- APDU command/response pair 43 -----
00000000: 00 C0 00 00 09                                        .....

00000000: A8 5D D3 30 E3 5C A9 00  39 90 00                     .].0.\9..


----- APDU command/response pair 44 -----
00000000: 80 86 00 00 80 C3 88 FD  AF 64 0D 35 77 85 D4 20  .........d.5w..
00000010: 57 10 02 F4 1E 38 51 37  40 31 7F 7F 11 E8 4B 8D  W....8Q7@1....K.
00000020: A5 CE C0 50 EB 6B CE E6  E0 DE E8 34 7C FE 0B 6C  ...P.k.....4|..l
00000030: F0 70 9F E3 5D F7 AA 50  BB 1C F6 8C 00 1B 18 EA  .p..]..P........
00000040: BF 73 E4 BE 75 B6 AE 29  B1 A2 A3 B8 1D 52 FD 19  .s..u..).....R..
00000050: C9 CA 20 FB 80 C2 20 A9  E3 A6 15 6C 11 B3 E9 18  .. ... ....l....
00000060: 13 3F 65 02 28 21 74 72  29 EA E2 27 8B DA 3E 45  .?e.(!tr)..'..>E
00000070: 82 A1 B0 D9 A7 1A 3D F3  5D 4D 27 F4 D2 73 ED 0F  ......=.]M'..s..
00000080: A8 88 41 F2 4F 00                                 ..A.O.

00000000: 14 8C 30 9E D5 10 25 B1  F7 AF 07 E7 25 8B 22 3C  ..0...%.....%."<
00000010: 62 61 8F 24 FB 59 E1 63  D7 B1 08 6D 07 7A DD 93  ba.$.Y.c...m.z..
00000020: 90 00                                             ..


----- APDU command/response pair 45 -----
00000000: 8C A4 08 00 15 87 09 01  E5 61 A8 BF 89 AD D7 FF  .........a......
00000010: 8E 08 C2 B3 32 7B D7 83  C9 D1                    ....2{....

00000000: 99 02 90 00 8E 08 E6 37  E6 BE 12 F8 73 6F 90 00  .......7....so..
```

```
----- APDU command/response pair 46 -----
00000000: 0C E0 08 00 4D 87 41 01  41 03 69 5A A4 EE 5F 44  ....M.A.A.iZ.._D
00000010: 2C 4C A9 FE 46 8D 1F 5B  79 D6 89 68 EB 94 CF FB  ,L..F..[y..h....
00000020: 6B A2 55 F6 65 B7 19 66  B3 67 E0 DF 46 F2 27 22  k.U.e..f.g..F.'"
00000030: AC D8 C1 57 C5 54 5B DF  B9 10 87 58 81 2E 9E 65  ...W.T[....X...e
00000040: 07 B1 6E 14 F8 DE 09 AF  8E 08 8C 79 AD C4 3B E2  ..n........y..;.
00000050: D2 84                                            ..


00000000: 99 02 90 00 8E 08 A5 D0  49 2A C0 91 47 68 90 00  ........I*..Gh..

----- APDU command/response pair 47 -----
00000000: 80 86 FF FF                                      ....

00000000: 90 00                                            ..
```

B.5 C_generateKeyPair

```
----- APDU command/response pair 54 -----
00000000: 00 E0 01 00 18 62 81 15  8A 01 04 83 02 01 40 80  .....b........@.
00000010: 02 01 A7 86 08 00 20 00  20 00 20 00 20           ...... . . .

00000000: 90 00                                            ..


----- APDU command/response pair 55 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 02 01 40          .....?.0.0..@

00000000: 90 00                                            ..


----- APDU command/response pair 56 -----
00000000: 00 D6 00 00 FA 01 01 01  40 01 A3 16 00 00 00 00  ........@.......
00000010: 04 02 00 00 00 00 01 00  00 01 01 00 02 00 00 01  ................
00000020: 01 00 03 10 00 03 70 75  62 FF FF FF FF FF FF FF  ......pub.......
00000030: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF FF FF FF FF FF 00 86  32 00 01 00 01 00 00 00  ........2.......
00000070: 04 00 00 00 00 01 01 10  00 00 FF FF FF FF FF FF  ................
00000080: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000090: FF FF FF FF FF FF FF FF  FF FF 01 02 10 00 01 03  ................
000000A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF 01  ................
000000C0: 04 00 00 01 01 01 06 00  00 01 01 01 0A 00 00 01  ................
```

```
000000D0: 01 01 0B 00 00 01 00 01  0C 10 00 01 00 01 10 10  ................
000000E0: 00 00 FF FF FF FF FF FF  FF FF 01 11 10 00 00 FF  ................
000000F0: FF FF FF FF FF FF FF 01  20 00 00 80 A8 FD 0C     ........ ......

00000000: 90 00                                           ..


----- APDU command/response pair 57 -----
00000000: 00 D6 00 FA AD 53 6B 7F  00 00 A8 FD 0C 53 6B 7F  .....Sk......Sk.
00000010: 00 00 B0 AA 47 51 6B 7F  00 00 00 30 00 00 00 00  ....GQk....0....
00000020: 00 00 B0 AA 47 51 6B 7F  00 00 02 30 00 00 00 00  ....GQk....0....
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  00 00 11 01 00 00 00 00  ................
00000050: 00 00 58 FD 0C 53 6B 7F  00 00 58 FD 0C 53 6B 7F  ..X..Sk...X..Sk.
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 00 00 01 21 00 00 04 00  04 00 00 01 22 00 00 03  ...!........"...
00000090: 01 00 01 01 63 00 00 01  01 01 66 00 00 04 00 00  ....c.....f.....
000000A0: 00 00 01 70 00 00 01 01  80 10 00 00 01 00 93 03  ...p............
000000B0: 93 03                                           ..

00000000: 90 00                                           ..


----- APDU command/response pair 58 -----
00000000: 80 A4 08 00 06 3F 00 30  00 30 02               .....?.0.0.

00000000: 90 00                                           ..


----- APDU command/response pair 59 -----
00000000: 00 E0 01 00 1E 62 81 1B  8A 01 04 83 02 02 00 80  .....b..........
00000010: 02 01 23 84 04 6B 78 73  30 86 08 00 00 00 20 00  ..#..kxs0..... .
00000020: 20 00 20                                         .

00000000: 90 00                                           ..


----- APDU command/response pair 60 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 02 02 00         .....?.0.0...

00000000: 90 00                                           ..


----- APDU command/response pair 61 -----
00000000: 00 D6 00 00 FA 01 02 02  00 01 1F 16 00 00 00 00  ................
00000010: 04 03 00 00 00 00 01 00  00 01 01 00 02 00 00 01  ................
00000020: 01 00 03 10 00 04 70 72  69 76 FF FF FF FF FF FF  ......priv......
00000030: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
```

```
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF FF FF FF FF FF 01 00  00 00 04 00 00 00 00 01  ................
00000070: 01 10 00 00 FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000080: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000090: FF FF FF FF 01 02 10 00  01 03 FF FF FF FF FF FF  ................
000000A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000B0: FF FF FF FF FF FF FF FF  FF 01 03 30 00 01 01 01  ...........0....
000000C0: 05 50 00 01 01 01 07 50  00 01 01 01 08 50 00 01  .P.....P.....P..
000000D0: 01 01 09 50 00 01 00 01  0C 10 00 01 00 01 10 10  ...P............
000000E0: 00 00 FF FF FF FF FF FF  FF FF 01 11 10 00 00 FF  ................
000000F0: FF FF FF FF FF FF FF 01  62 50 00 01 01 01 63     ........bP....c

00000000: 90 00                                           ..


----- APDU command/response pair 62 -----
00000000: 00 D6 00 FA 29 00 00 01  01 01 64 00 00 01 00 01  ....).....d.....
00000010: 65 00 00 01 01 01 66 00  00 04 00 00 00 00 01 70  e.....f........p
00000020: 00 00 01 01 80 10 00 00  01 00 93 03 93 03        ..............

00000000: 90 00                                           ..


----- APDU command/response pair 63 -----
00000000: 80 A4 08 00 06 3F 00 30  00 30 02               .....?.0.0.

00000000: 90 00                                           ..


----- APDU command/response pair 64 -----
00000000: 00 E0 08 00 27 62 81 24  8A 01 04 83 02 00 41 80  ....'b.$......A.
00000010: 02 00 80 85 05 05 0C 20  00 A3 86 0E 00 00 00 FF  ....... ........
00000020: 00 FF 00 20 00 20 00 00  00 20 71 00              ... . ... q.

00000000: 90 00                                           ..


----- APDU command/response pair 65 -----
00000000: 80 A4 00 00 02 00 41                            ......A

00000000: 90 00                                           ..


----- APDU command/response pair 66 -----
00000000: 00 47 00 00 0C AC 81 09  80 01 06 81 81 03 01 00  .G..............
00000010: 01                                              .

00000000: 90 00                                           ..
```

```
----- APDU command/response pair 67 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 02 00 41           .....?.0.0..A

00000000: 90 00                                             ..


----- APDU command/response pair 68 -----
00000000: 80 48 00 00 00                                    .H...

00000000: 7F 49 81 88 81 81 80 D1  EF 7C A5 06 A1 87 FD 5F  .I.......|....._
00000010: 13 5B 25 B7 16 B9 BA A7  21 43 3D DB 51 9D C1 D1  .[%.....!C=.Q...
00000020: 5A 3C 95 7C B6 F0 37 57  83 CF 2D 0B 53 66 C7 11  Z<.|..7W..-.Sf..
00000030: D5 6B FD 28 FA A0 EA 50  1E 2B FD B5 09 49 E2 E7  .k.(...P.+...I..
00000040: 51 67 1B 00 B0 9D 52 CD  22 D8 69 8C 36 74 54 41  Qg....R.".i.6tTA
00000050: 6E 40 58 4F 79 52 E4 D9  00 43 9C 2C 79 FE A6 48  n@XOyR...C.,y..H
00000060: B7 31 8A B2 05 04 C4 DD  B3 86 E6 4F 38 A6 5D 2A  .1.........O8.]*
00000070: CD A8 3F 95 E4 FF 7B 05  1E ED 4A B5 99 69 36 F0  ..?...{...J..i6.
00000080: B9 5B 29 C6 EC B3 25 82  03 01 00 01 90 00        .[)...%.......


----- APDU command/response pair 69 -----
00000000: 80 A4 08 00 06 3F 00 30  00 30 02                 .....?.0.0.

00000000: 90 00                                             ..


----- APDU command/response pair 70 -----
00000000: 00 E0 08 00 B0 62 81 AD  8A 01 04 83 02 00 81 80  .....b..........
00000010: 02 00 80 85 05 05 08 20  00 A3 86 0E 00 00 00 FF  ....... ........
00000020: 00 FF 00 20 00 20 00 00  00 20 71 81 88 90 03 01  ... . ... q.....
00000030: 00 01 91 81 80 D1 EF 7C  A5 06 A1 87 FD 5F 13 5B  .......|....._.[
00000040: 25 B7 16 B9 BA A7 21 43  3D DB 51 9D C1 D1 5A 3C  %.....!C=.Q...Z<
00000050: 95 7C B6 F0 37 57 83 CF  2D 0B 53 66 C7 11 D5 6B  .|..7W..-.Sf...k
00000060: FD 28 FA A0 EA 50 1E 2B  FD B5 09 49 E2 E7 51 67  .(...P.+...I..Qg
00000070: 1B 00 B0 9D 52 CD 22 D8  69 8C 36 74 54 41 6E 40  ....R.".i.6tTAn@
00000080: 58 4F 79 52 E4 D9 00 43  9C 2C 79 FE A6 48 B7 31  XOyR...C.,y..H.1
00000090: 8A B2 05 04 C4 DD B3 86  E6 4F 38 A6 5D 2A CD A8  .........O8.]*..
000000A0: 3F 95 E4 FF 7B 05 1E ED  4A B5 99 69 36 F0 B9 5B  ?...{...J..i6..[
000000B0: 29 C6 EC B3 25                                    )...%

00000000: 90 00                                             ..


----- APDU command/response pair 71 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 02 00 41           .....?.0.0..A

00000000: 90 00                                             ..
```

```
----- APDU command/response pair 72 -----
00000000: 80 48 00 00 00                                        .H...

00000000: 7F 49 81 88 81 81 80 D1  EF 7C A5 06 A1 87 FD 5F  .I.......|....._
00000010: 13 5B 25 B7 16 B9 BA A7  21 43 3D DB 51 9D C1 D1  .[%.....!C=.Q...
00000020: 5A 3C 95 7C B6 F0 37 57  83 CF 2D 0B 53 66 C7 11  Z<.|..7W..-.Sf..
00000030: D5 6B FD 28 FA A0 EA 50  1E 2B FD B5 09 49 E2 E7  .k.(...P.+...I..
00000040: 51 67 1B 00 B0 9D 52 CD  22 D8 69 8C 36 74 54 41  Qg....R.".i.6tTA
00000050: 6E 40 58 4F 79 52 E4 D9  00 43 9C 2C 79 FE A6 48  n@XOyR...C.,y..H
00000060: B7 31 8A B2 05 04 C4 DD  B3 86 E6 4F 38 A6 5D 2A  .1.........O8.]*
00000070: CD A8 3F 95 E4 FF 7B 05  1E ED 4A B5 99 69 36 F0  ..?...{...J..i6.
00000080: B9 5B 29 C6 EC B3 25 82  03 01 00 01 90 00        .[)...%.......


----- APDU command/response pair 73 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 02 01 40           .....?.0.0..@

00000000: 90 00                                             ..


----- APDU command/response pair 74 -----
00000000: 00 D6 00 F5 82 00 80 D1  EF 7C A5 06 A1 87 FD 5F  .........|....._
00000010: 13 5B 25 B7 16 B9 BA A7  21 43 3D DB 51 9D C1 D1  .[%.....!C=.Q...
00000020: 5A 3C 95 7C B6 F0 37 57  83 CF 2D 0B 53 66 C7 11  Z<.|..7W..-.Sf..
00000030: D5 6B FD 28 FA A0 EA 50  1E 2B FD B5 09 49 E2 E7  .k.(...P.+...I..
00000040: 51 67 1B 00 B0 9D 52 CD  22 D8 69 8C 36 74 54 41  Qg....R.".i.6tTA
00000050: 6E 40 58 4F 79 52 E4 D9  00 43 9C 2C 79 FE A6 48  n@XOyR...C.,y..H
00000060: B7 31 8A B2 05 04 C4 DD  B3 86 E6 4F 38 A6 5D 2A  .1.........O8.]*
00000070: CD A8 3F 95 E4 FF 7B 05  1E ED 4A B5 99 69 36 F0  ..?...{...J..i6.
00000080: B9 5B 29 C6 EC B3 25                              .[)...%

00000000: 90 00                                             ..
```

B.6 C_destroyObject

```
----- APDU command/response pair 35 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 03 40           .....?.0.0..@

00000000: 90 00                                             ..

----- APDU command/response pair 36 -----
00000000: 00 B0 00 00 00                                    .....

00000000: 00 03 03 40 01 23 18 00  00 00 00 04 04 00 00 00  ...@.#..........
```

```
00000010: 00 01 00 00 01 01 00 02  00 00 01 00 00 03 10 00  ................
00000020: 04 64 65 73 33 FF FF FF  FF FF FF FF FF FF FF FF  .des3............
00000030: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF 00 11 01 00 18 FF FF  FF FF FF FF FF FF FF FF  ................
00000070: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF 01 00  ................
00000080: 00 00 04 15 00 00 00 01  02 10 00 01 01 FF FF FF  ................
00000090: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000A0: FF FF FF FF FF FF FF FF  FF FF FF FF 01 03 30 00  ..............0.
000000B0: 01 01 01 04 00 00 01 01  01 05 50 00 01 01 01 06  ..........P.....
000000C0: 00 00 01 00 01 07 50 00  01 01 01 08 50 00 01 01  ......P.....P...
000000D0: 01 0A 00 00 01 01 01 0C  10 00 01 00 01 10 10 00  ................
000000E0: 00 FF FF FF FF FF FF FF  FF 01 11 10 00 00 FF FF  ................
000000F0: FF FF FF FF FF FF 01 62  50 00 01 00 01 63 00 00  .......bP....c..
00000100: 61 27                                            a'


----- APDU command/response pair 37 -----
00000000: 00 B0 01 00 00                                   .....

00000000: 01 01 01 64 00 00 01 01  01 65 00 00 01 01 01 66  ...d.....e.....f
00000010: 00 00 04 31 01 00 00 01  70 00 00 01 01 80 10 00  ...1....p.......
00000020: 00 01 00 97 03 97 03 90  00                      .........

----- APDU command/response pair 49 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 03 40 01          .....?.0.0.@.

00000000: 90 00

----- APDU command/response pair 50 -----
00000000: 00 D6 00 04 04 98 03 98  03                      .........

00000000: 90 00                                            ..

----- APDU command/response pair 53 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 00 C1          .....?.0.0...

00000000: 90 00                                            ..


----- APDU command/response pair 54 -----
00000000: 00 E4 00 00                                      ....

00000000: 90 00                                            ..


----- APDU command/response pair 55 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 03 40          .....?.0.0..@
```

```
00000000: 90 00                                           ..


----- APDU command/response pair 56 -----
00000000: 00 E4 00 00                                     ....

00000000: 90 00                                           ..
```

B.7 C_encrypt

```
----- APDU command/response pair 52 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 00 C1         .....?.0.0...

00000000: 90 00                                           ..


----- APDU command/response pair 53 -----
00000000: 00 2A 82 05 13 80 81 10  54 65 73 74 53 74 72 69  .*......TestStri
00000010: 6E 67 31 32 33 34 35 36  00                       ng123456.

00000000: 82 10 B4 F0 97 B6 63 E4  68 7A 8B 00 4F DF 3A C1  ......c.hz..O.:.
00000010: 49 9F 90 00                                       I...
```

B.8 C_decrypt

```
----- APDU command/response pair 64 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 00 C1         .....?.0.0...

00000000: 90 00                                           ..


----- APDU command/response pair 65 -----
00000000: 00 2A 80 05 0B 82 81 08  8B 00 4F DF 3A C1 49 9F  .*........O.:.I.
00000010: 00                                                .

00000000: 80 08 6E 67 31 32 33 34  35 36 90 00              ..ng123456..
```

B.9 C_setAttribute

```
----- APDU command/response pair 51 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 03 40         .....?.0.0..@
```

```
00000000: 90 00                                           ..


----- APDU command/response pair 52 -----
00000000: 90 32 00 03 FF 01 27 00  00 62 82 01 27 00 03 03  .2....'..b..'...
00000010: 40 01 23 18 00 00 00 00  04 04 00 00 00 00 01 00  @.#.............
00000020: 00 01 01 00 02 00 00 01  00 00 03 10 00 07 63 68  ..............ch
00000030: 61 6E 67 65 64 FF FF FF  FF FF FF FF FF FF FF FF  anged............
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF 00 11  ................
00000070: 01 00 18 FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000080: FF FF FF FF FF FF FF FF  FF FF FF 01 00 00 00 04  ................
00000090: 15 00 00 00 01 02 10 00  01 01 FF FF FF FF FF FF  ................
000000A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000B0: FF FF FF FF FF FF FF FF  FF 01 03 30 00 01 01 01  ...........0....
000000C0: 04 00 00 01 01 01 05 50  00 01 01 01 06 00 00 01  .......P........
000000D0: 00 01 07 50 00 01 01 01  08 50 00 01 01 01 0A 00  ...P.....P......
000000E0: 00 01 01 01 0C 10 00 01  00 01 10 10 00 00 FF FF  ................
000000F0: FF FF FF FF FF FF 01 11  10 00 00 FF FF FF FF FF  ................
00000100: FF FF FF 01                                     ....

00000000: 90 00                                           ..


----- APDU command/response pair 53 -----
00000000: 80 32 00 03 30 62 50 00  01 00 01 63 00 00 01 01  .2..0bP....c....
00000010: 01 64 00 00 01 01 01 65  00 00 01 01 01 66 00 00  .d.....e.....f..
00000020: 04 31 01 00 00 01 70 00  00 01 01 80 10 00 00 01  .1....p.........
00000030: 00 99 03 99 03                                  .....

00000000: 90 00                                           ..
```

B.10 C_unwrap

```
----- APDU command/response pair 92 -----
00000000: 80 48 00 80 00                                  .H...

00000000: 80 01 05 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
```

```
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{‘g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....‘.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                           a.


----- APDU command/response pair 93 -----
00000000: 00 C0 00 00 09                                  .....

00000000: A8 5D D3 30 E3 5C A9 00  39 90 00               .].0....9..


----- APDU command/response pair 94 -----
00000000: 80 86 00 00 80 D0 7E EE  17 C7 31 DD 53 FB 1F D4  ......~...1.S...
00000010: 36 65 EB 7F 2C B0 A2 34  44 80 D7 F4 31 96 12 DF  6e..,..4D...1...
00000020: C8 AD 3C 41 EE 8F 13 C2  8A 3B 8D 6B 73 18 A6 1B  ..<A.....;.ks...
00000030: 46 3E 10 93 5C 2F 35 1C  A3 FC 48 09 DB E4 BB EA  F>..5...H.....
00000040: 3F 1A 11 7D 85 57 2F 85  75 1D 8B F4 E6 39 2B FA  ?..}.W/.u....9+.
00000050: 19 3D 7A BB E3 75 B2 A2  A9 E4 EE 79 4F A6 3F EE  .=z..u.....yO.?.
00000060: FD BF 4A F8 43 8F DA A9  D1 8D 58 63 12 5D C8 E8  ..J.C.....Xc.]..
00000070: 2C 77 8F 5F 96 C0 51 CA  19 B1 80 D5 80 4E 50 8B  ,w._..Q......NP.
00000080: 88 6B 64 43 0D 00                               .kdC..

00000000: FD 74 3B 38 48 7E 0E D9  4D B0 BF E7 66 3D E4 63  .t;8H~..M...f=.c
00000010: 15 24 EC 7B F3 93 C7 90  85 43 E8 DF D9 E0 60 88  .$.{.....C....‘.
00000020: 90 00                                           ..


----- APDU command/response pair 95 -----
00000000: 8C A4 08 00 1D 87 11 01  65 FD 9A B5 09 70 96 93  ........e....p..
00000010: FB 5D 39 FF B3 24 6B 8E  8E 08 04 D7 B0 58 E0 96  .]9..$k......X..
00000020: E6 01                                           ..

00000000: 99 02 90 00 8E 08 67 C9  1F 50 18 5F 6D 6A 90 00  ......g..P._mj..


----- APDU command/response pair 96 -----
00000000: 0C 2A 80 0A 99 87 81 89  01 1D 7A 97 D8 25 8F 60  .*........z..%.‘
00000010: 52 07 AE DC A9 AC 33 7C  6E 12 A9 79 71 B8 36 1B  R.....3|n..yq.6.
00000020: 29 C3 54 C1 A8 29 A4 4F  75 72 4E C6 C5 71 22 88  ).T..).OurN..q".
00000030: 50 0C 29 9F 75 C7 99 39  E9 B6 5B AF A1 65 51 DE  P.).u..9..[..eQ.
00000040: 56 84 6D 30 B6 2F F3 19  6B 83 82 C4 6B AB 59 E3  V.m0./..k...k.Y.
00000050: 2B FD B1 4B FC 3D BE CD  16 C8 C0 69 80 5C 0E 72  +..K.=.....i...r
00000060: C0 0F 24 0A 3E 8A 88 4A  CA 68 02 5C FA B5 36 33  ..$.>..J.h.:63
```

```
00000070: CB 5A F7 BE 86 21 2F 68  DB 5F 46 1D 67 FA C2 8B  .Z...!/h._F.g...
00000080: A9 58 37 5C F0 34 7E FE  FC 1A 78 46 C7 51 0B 13  .X7\.4~...xF.Q..
00000090: B2 97 01 00 8E 08 F9 2D  53 7C AD 46 EB 79 00     .......-S|.F.y.

00000000: 87 11 01 FC E7 96 A5 B1  96 E9 E3 1D 2D 3A 49 46  ............-:IF
00000010: 8C 97 A7 99 02 90 00 8E  08 41 16 94 D4 58 27 D0  .........A...X'.
00000020: 3F 90 00                                          ?..


----- APDU command/response pair 97 -----
00000000: 80 86 FF FF                                       ....

00000000: 90 00                                             ..


----- APDU command/response pair 119 -----
00000000: 80 A4 08 00 08 3F 00 30  00 30 01 03 41           .....?.0.0..A

00000000: 90 00                                             ..


----- APDU command/response pair 120 -----
00000000: 00 D6 00 00 FA 01 03 03  41 01 23 18 00 00 00 00  ........A.#.....
00000010: 04 04 00 00 00 00 01 00  00 01 01 00 02 00 00 01  ................
00000020: 00 00 03 10 00 04 74 65  73 74 FF FF FF FF FF FF  ......test......
00000030: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000040: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000050: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000060: FF FF FF FF FF FF 00 11  01 00 08 FF FF FF FF FF  ................
00000070: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000080: FF FF FF 01 00 00 00 04  13 00 00 00 01 02 10 00  ................
00000090: 01 10 FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
000000B0: FF 01 03 30 00 01 01 01  04 00 00 01 01 01 05 50  ...0...........P
000000C0: 00 01 01 01 06 00 00 01  00 01 07 50 00 01 01 01  ...........P....
000000D0: 08 50 00 01 01 01 0A 00  00 01 01 01 0C 10 00 01  .P..............
000000E0: 00 01 10 10 00 00 FF FF  FF FF FF FF FF FF 01 11  ................
000000F0: 10 00 00 FF FF FF FF FF  FF FF FF 01 62 50 00     ............bP.

00000000: 90 00                                             ..


----- APDU command/response pair 121 -----
00000000: 00 D6 00 FA 2D 01 00 01  63 00 00 01 00 01 64 00  ....-...c.....d.
00000010: 00 01 00 01 65 00 00 01  00 01 66 00 00 04 FF FF  ....e.....f.....
00000020: FF FF 01 70 00 00 01 01  80 10 00 00 01 00 B0 03  ...p............
00000030: B0 03                                             ..

00000000: 90 00                                             ..
```

```
----- APDU command/response pair 122 -----
00000000: 80 48 00 80 00                                        .H...

00000000: 80 01 05 81 81 80 F7 B5  15 72 07 22 94 6F C4 08  .........r.".o..
00000010: 64 CB BD AF EA 55 7D BD  8F 55 36 B0 01 C2 8B 2E  d....U}..U6.....
00000020: 32 B6 5D 45 F1 74 5D 38  12 0B AD 9D 2C 03 9C 22  2.]E.t]8....,.."
00000030: 46 68 EB 2E A2 8C 20 95  A8 2E 6C A8 E0 6D 47 F2  Fh.... ...l..mG.
00000040: D3 1E D7 01 F8 15 5C AD  DC 05 70 C0 93 B2 6D 74  ......\.p...mt
00000050: B0 9B 95 E6 4D 8C D2 FC  73 3E CD 0F 30 68 79 A5  ....M...s>..0hy.
00000060: B9 35 F2 41 3F 52 AD AD  32 A0 99 1A 18 3D CC 57  .5.A?R..2....=.W
00000070: 7E 39 DA 47 53 1E 67 15  AB 01 70 7F F2 47 96 71  ~9.GS.g...p..G.q
00000080: 44 23 CE 7B 60 67 82 81  80 3C 52 D2 06 89 28 92  D#.{`g...<R...(.
00000090: 2C AB E6 3C 4E E6 DF 0E  D2 29 F1 01 BE 36 C4 F8  ,..<N....)...6..
000000A0: 54 40 56 F3 4A FA 8D 2E  9B 60 F5 07 BC ED B4 44  T@V.J....`.....D
000000B0: 56 68 5D 82 4C C4 EA D7  96 20 F8 C5 46 A6 E0 16  Vh].L.... ..F...
000000C0: B8 AB A5 D8 43 29 58 53  77 17 09 97 AA 70 68 33  ....C)XSw....ph3
000000D0: 9E F1 41 0A 5F 39 D9 75  24 7F 3A 53 63 61 47 87  ..A._9.u$.:ScaG.
000000E0: 87 7F 88 96 BC BB 83 A1  CB D1 42 E0 EB 99 CF 34  ..........B....4
000000F0: 0E CA 56 4F 2C 57 50 6E  7B 1A FC 1F 90 7A E0 C2  ..VO,WPn{....z..
00000100: 61 09                                             a.


----- APDU command/response pair 123 -----
00000000: 00 C0 00 00 09                                        .....

00000000: A8 5D D3 30 E3 5C A9 00  39 90 00                     .].0....9..


----- APDU command/response pair 124 -----
00000000: 80 86 00 00 80 95 3B CF  46 B8 4E 67 E4 6B 97 4B  ......;.F.Ng.k.K
00000010: 70 AD B3 44 22 6A 1B 42  18 4B A9 44 FF 28 FA C0  p..D"j.B.K.D.(..
00000020: 0A EF 44 CD DA C1 28 2B  CF FD 5D 20 48 50 33 59  ..D...(+..] HP3Y
00000030: 7D B7 CB 73 4A EF 28 0A  C7 E4 02 2A 91 A9 F6 55  }..sJ.(....*...U
00000040: 97 D3 A8 DE 21 90 0E 23  0B 9C ED 4B 52 39 46 ED  ....!..#...KR9F.
00000050: 13 1F 7F 9D CB EF 7A DD  7C D7 39 EC 1F BD 2A 3A  ......z.|.9...*:
00000060: 45 48 8F 6C 7E 82 71 E5  14 8F C1 9D F8 E8 53 2B  EH.l~.q.......S+
00000070: D3 AF 3D 7C 11 59 E3 81  F4 0B 08 17 A9 0F 37 69  ..=|.Y........7i
00000080: 90 C1 11 E2 1B 00                                 ......

00000000: B3 0F 6C 66 E6 56 8F 44  55 B2 A6 02 0E 0B 80 01  ..lf.V.DU.......
00000010: FF 89 7A 65 FC 68 25 82  22 C9 97 74 D1 6B 00 AB  ..ze.h%."..t.k..
00000020: 90 00                                             ..


----- APDU command/response pair 125 -----
00000000: 8C A4 08 00 15 87 09 01  82 46 BD FD 60 2D E4 C6  .........F..`-..
00000010: 8E 08 25 35 C0 28 0E E1  20 93                    ..%5.(.. .
```

```
00000000: 99 02 90 00 8E 08 8C D6  A9 A8 99 7F 14 12 90 00  ................


----- APDU command/response pair 126 -----
00000000: 0C E0 08 00 3D 87 31 01  4D 4F 3D AB 31 72 FC F7  ....=.1.MO=.1r..
00000010: B4 84 D1 41 19 1C 22 DF  3F 60 BE 6B 0A 1E 49 5F  ...A.."."?‘.k..I_
00000020: AD 3D 6D 61 5E DA E3 F7  A8 0A 82 EA 65 16 8A 01  .=ma^.......e...
00000030: C5 4F BF 3F 44 73 9C 61  8E 08 A8 A9 A5 4D 55 BB  .O.?Ds.a.....MU.
00000040: E7 B3                                           ..

00000000: 99 02 90 00 8E 08 23 E5  DF 34 11 21 87 1C 90 00  ......#..4.!....


----- APDU command/response pair 127 -----
00000000: 80 86 FF FF                                     ....

00000000: 90 00                                           ..
```

B.11 C_wrap

```
Enter command (spaced integers)
00 2A 82  0A 08 1E 1F 20 22 23 24 25 26 00 (hexadecimals)
00 42 130 10 08 30 31 32 34 35 36 37 38 00 (integers)
command changed!

RESPONSE
00000000: 6A 80                                           j.
```