EMV (Chip-and-PIN) Protocol

Märt Bakhoff

December 15, 2014

Abstract

The objective of this report is to observe and describe a real world online transaction made between a debit card issued by an Estonian bank and a payment terminal issued by a Estonian bank. In this process we can learn how the EMV protocol works and which protocol features are used in a Chip-and-PIN card issued by an Estonian bank.

1 Introduction

The world is slowly but surely moving from cash to using digital banking and card payments. An important part in it is the extra security promised by the chip-and-pin cards. Unfortunately the security of the chip-and-pin protocol (EMV) is difficult to analyse because it requires specialized hardware which is running closed source software.

This report describes an attempt by the author of this paper to verify whether estonian bank cards correctly implement the EMV protocol and gain a better understanding of the protocol by investigating the data moving between a chip-and-pin card and a payment terminal.

> "EMV stands for Europay, MasterCard and Visa, a global standard for interoperation of integrated circuit cards (ICC) and ICC capable point of sale terminals and ATMs, for authenticating credit and debit card transactions." - Wikipedia [1].

Simply put, the EMV standard defines how every compatible card and terminal communicate, from electrical protocol up to the high level crypto operations. It's detailed enough to be useful in decoding a captured transaction, which is exactly what is needed here.

The specification is divided into four "books" by the general topics covered in it. The entire specification [2] is available to anyone free of charge at `http://emvco.com`.

The Smartcard and the terminal communicate using a simple request-response protocol. The requests packets (sent by the terminal) and response packets (sent by the card) are called APDUs (application protocol data units). Each request starts with a instruction code, followed by two parameters and an optional data field (table 1). Each response contains a optional data field followed by a two byte status code (table 2). Status 9000h is the usual "OK" response, 6xxxh means either an error or request for additional processing.

Code	*Description*	*Length*
CLA	Class of instruction	1
INS	Instruction code	1
P1	Parameter 1	1
P2	Parameter 2	1
Lc	Length of command data	0 or 1
Data	Command data	var.
Le	Expected length of response	0 or 1

Table 1: Request APDU

Code	*Description*	*Length*
Data	Response data	var
SW1	Command status	1
SW2	Command qualifier	1

Table 2: Response APDU

2 Capturing The Transaction

To find out what bits and bytes are exchanged between the card and the terminal, the physical communication line between the card and the terminal was tapped. Alternatively, it would have been possible to modify one of the end-points to log commands sent and received. However, there are no commonly available ready-to-use hardware solution to achieve this.

Since a debit card is just a smartcard, it was decided to use the Simtrace development board [3] (see Figure 1) to physically sniff the communication line.

2.1 Simtrace

Simtrace is a standalone electronic device that can be placed between a smartcard and a smartcard reader where it acts as a proxy and forwards data between the card and the reader. It also has a USB port that can be used to view and save all the data going through Simtrace.

Simtrace can be ordered online for 90 EUR and it includes wires that can be used to connect the Simtrace board to a card reader. Simtrace was originally designed for sniffing mobile phone and SIM card communications but since SIM cards are just regular smartcards then it can also be used for sniffing bank cards.

Unfortunately, Simtrace only has a mini-SIM card slot which cannot be used to plug-in a full-size smartcard, therefore, a solution had to be found for connecting a debit card to Simtrace without damaging the card. To achieve this a standard ID card reader was modified by adding a smart card contact interface which could be connected to the wires included in the Simtrace package (Figure 2).

After Simtrace was customized and the software which sends captured data to the computer was installed and configured, the sniffing of real world transaction could begin.

2.2 Captured Transaction

The transaction analyzed in this report was captured using a terminal from a friendly merchant in Tartu and using a Visa Electron debit card issued by SEB Estonia. The amount of transaction was 0.99 EUR. The transaction was performed in September, 2014. The full output (all requests and responses) with annotation can be found from the appendix.

The final setup for the capture can be seen in Figure 3. During the transaction, the computer connected to the Simtrace board produced a stream of APDU requests and responses (Figure 4).

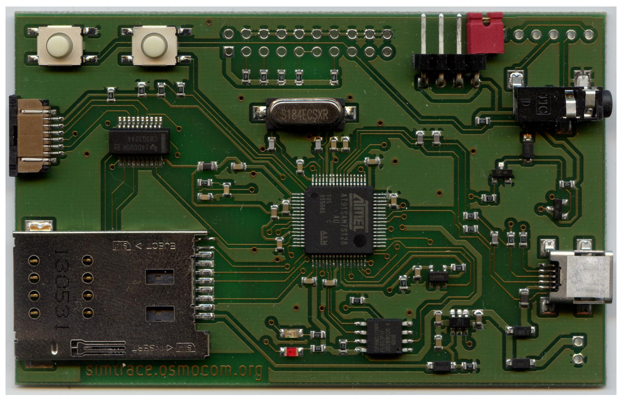

Figure 1: Simtrace development board[1]

2.3 Discrepancies in the data

While the capture of the transaction using Simtrace worked, it wasn't perfect. There were several discrepancies in the captured binary dump all of which are described below. Fortunately, these discrepancies are insignificant and do not prevent us from analysing the transaction. The suspected cause for these discrepancies is software bug in Simtrace

[1]Source `http://bb.osmocom.org/trac/attachment/wiki/SIMtrace/Hardware/simtrace_v13_front.jpg`

Figure 2: Modified ID card reader

Figure 3: Recording the transaction

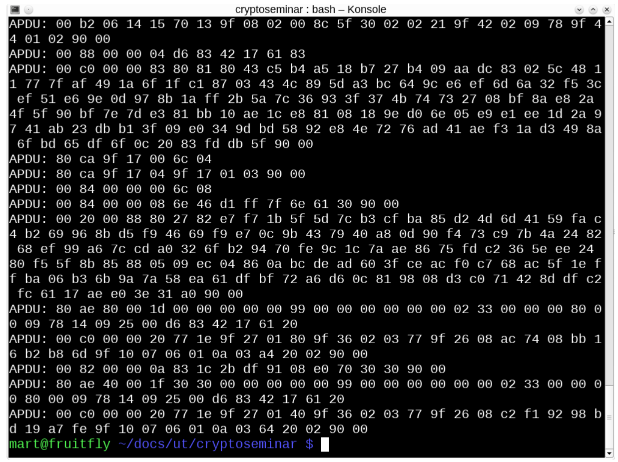

Figure 4: Sniffed APDUs

which fails to capture all data if smart card is communicating too fast [4].

Trailing bytes in request. All request APDUs have two unexpected bytes at the end. This is not specified in EMV and is probably caused by issues with Simtrace. The bytes are usually in the form 6xxxh and the last byte often matches the length of the response. This means that the extra bytes could instead be from the card.

Unexpected header in responses. All response APDUs have five unexpected bytes in the beginning of the packet. This is not specified by EMV and is also probably caused by issues with Simtrace. The first 2 bytes usually match the instruction sent in the request APDU and the fifth byte usually equals the length of the entire response APDU.

Missing packets. There are three packets missing in the captured binary log. The missing packets are not random, but are probably once again related to Simtrace.

Missing request4 follows a response packet with an unusual status code from the card that has special instructions to the terminal.

Missing response16 and response18 should be responses with an empty body that contain only the status code. Oddly the status code 9000h is visible in the end of the preceding request, which again hints that these bytes may be from the card.

3 Analysis of Captured Data

This section goes through captured transaction describing exchanged data and its place in EMV protocol. High level overview is given here with references to the low-level details and captured bytes in the Appendix.

3.1 Application selection

Before starting the payment, the terminal must find and select the payment application on the chip-card. There can actually be several applications on a single card (for example, LHV bank's cards have both debit and credit applications on the same chip) [5]. As the first step, the terminal reads all the application identifiers from the card and presents a selection to the terminal user. An example for terminal and card supporting multiple applications is given in Figure 5.

Figure 5: Card application selection[2]

After card application has been selected, all the following operations until the end of the process are executed in the context of the selected application. The Chip-and-PIN card used in our captured transaction has only single debit card application which is used automatically (requests 1-3).

3.2 Read application data

After selecting the application the terminal proceeds to reading all the data for that application from the card. This includes expiration dates, PIN code options, card authentication options and several crypto keys. All the data is read up front so that later operations won't need to read additional information. This corresponds to requests 4-12.

3.3 Data authentication

After reading all the data in the application, the terminal will verify that the card is authentic and hasn't been tampered with. There are two mechanisms for that (one or both can be used) [6].

In the static data authentication (SDA) mode, the terminal will create a hash of all the important data on the card. It will then read a digital signature from the card that contains the hash for the same data signed by the card's issuer. If the signature is valid, then the data on the card has not been tampered with.

In the dynamic data authentication (DDA) mode, the terminal will generate a unique random number (nonce) and send it to the card. The card will then use its private key to digitally sign the nonce and sends the signature back to the terminal. The terminal can then check the signature using the card's public key to make sure that the card actually contains the right private key.

DDA is much more secure because it's very difficult to extract a private key from a smartcard and it would be impossible to pass this check without the real key. It is also secure agains replay attacks (capture a valid DDA response and use it in another

[2] Source `http://useinability.files.wordpress.com/2014/01/card_terminal_ikea.jpg`

transaction) because the terminal always uses a new nonce.

In case of our transaction the DDA mode is used (see request 13).

3.4 Cardholder verification

Usually a PIN code is used to verify the presence of card's owner. The card will generate a nonce and send it to the terminal. The user must enter the PIN on the pinpad. The pinpad encrypts the PIN code and the nonce using the card's public key and the encrypted PIN is sent to the card. The card can then verify the PIN.

The encryption protects against snooping the PIN codes and the nonce protects against replay attacks (capturing a response for a valid pin entry and using it in another transaction). The process can be seen in requests 14-16.

3.5 Risk and restrictions processing

The terminal will check that the card has not expired and that the card is allowed to be used for the transaction. The card contains a list of flags that restrict it's use, for example: domestic use, international use, use in ATMs.

Additionally the terminal must decide whether to use online mode for the transaction (the card can communicate with the issuing bank over the internet) or the transaction will be done fully offline. This decision is based on the terminal's configuration and the amount of money that is being processed.

The online/offline decision must also be confirmed by the card. If the terminal requests an online payment, then the card can either accept or reject it. If the terminal requests an offline payment, then the card can reject it but request the terminal to switch to online payment. The card can't request an offline payment if the terminal requires an online payment.

3.6 Online processing

The online verification mostly relies on HMAC. HMAC (hash-based message authentication code) is a hash of some data that is mixed with a secret key. As a result, the hash can only be verified and/or created by a party who knows the secret key.

Before confirming the transaction the card will usually verify the payment with the issuing bank. All the data for the payment is sent to the card by the terminal. The card will then create a HMAC of the data and return it to the terminal. The terminal will send the payment data along with the HMAC from the card to the bank to verify (the secret key of the HMAC is known only by the bank and the card). The bank's response in forwarded to the card. This request to the bank is called an ARQC (Authorisation Request Cryptogram). This is visible in captured requests 17-18.

If the connection to the bank fails then the card and the terminal may negotiate an offline payment instead. If the bank rejects the payment then the entire transaction is aborted.

3.7 Final processing

If all the previous steps have succeeded then the terminal will try to authorize the payment. It will send all the necessary data to the card and the card will generate another HMAC that will be used by the merchant to get money from the payment processor (VISA, Mastercard etc). This is the last chance for the card to reject the transaction. The final HMAC that authorizes the payment is called a transaction certificate (TC) (request 19).

4 Conclusion

The author successfully captured a conversation between the card and the terminal. The EMV specification makes it easy to read the captured data and find out what information is sent to the card. The tested SEB card seemed to follow the specification correctly and contained reasonable data.

References

[1] EMV, 2014. `http://en.wikipedia.org/wiki/EMV`.

[2] LLC EMVCo. EMV 4.3 specification, 2011. http://www.emvco.com/specifications.aspx.

[3] Osmocom. Osmocom SIMtrace, 2014. http://bb.osmocom.org/trac/wiki/SIMtrace.

[4] Min Xu. Re: Fast SIM cards loosing bytes, March 2014. http://permalink.gmane.org/gmane.comp.mobile.osmocom.simtrace/126.

[5] Level2Kernel. How EMV (Chip & PIN) Works - Transaction Flow Chart, 2011. https://www.level2kernel.com/flow-chart.html.

[6] Cotignac Consultancy. EMV Offline Data Authentication, 2008. http://cotignac.co.nz/emv-offline-data-authentication/.

[7] javaemvreader project. Collection of CA public keys, revision 20, 2014. https://code.google.com/p/javaemvreader/source/browse/trunk/src/main/resources/certificationauthorities.xml.

Appendix: Captured APDUs with Annotations

This appendix contains the entire captured conversation between the card and the terminal. The messages were decoded by hand using the EMV specification. On the left side are the raw bytes and on the right side is the description of these bytes.

Most of the data is BER encoded which means the data is split into type-length-value triplets. Some of the data is just a concatenation of pieces of data, such as DDOL and CDOL fields.

The description sometimes contains notes such as "B1 x.y.z". This is a reference to the EMV specification, noting the book number and the exact chapter.

4.1 Request 1 (Application selection)

`00 a4 04 00`	B1 11.3.2 SELECT mode: by filename options: select first/next
`0e`	filename length: 14
`31 50 41 59 2e 53 59 53 2e 44 44 46 30 31`	filename: '1PAY.SYS.DDF01'
`61 22`	???

4.2 Response 1

`00 c0 00 00 22`	???
`6f 20`	FCI template (PSE selected)
`84 0e 31 50 41 59 2e 53 59 53 2e 44 44 46 30 31`	directory file name: '1PAY.SYS.DDF01'
`a5 0e`	FCI proprietary template
`88 01 01`	ShortFileIdentifier of directory element: 1
`5f 2d 08 65 74 65 6e 72 75 64 65`	language preference: et,en,ru,de
`90 00`	ok

4.3 Request 2

`00 b2 01 0c`	B1 11.2.2 READ RECORD P1: record number: 1 ShortFileIdentifier: 1; P1 is record number
`00`	data length: 0
`6c 22`	???

4.4 Response 2

`00 b2 01 0c 22`	???
`70 20`	tag+len
`61 1e`	directory entry tag + len
`4f 07 a0 00 00 00 03 20 10`	Application Identifier: VISA electron
`50 10 56 49 53 41 45 4c 45 43 54 52 4f 4e 20 20 20 20`	application label: 'VISAELECTRON '
`87 01 01`	application priority: 1
`90 00`	ok

4.5 Request 3

`00 b2 02 0c`	B1 11.2.2 READ RECORD P1: record number: 2 ShortFileIdentifier: 1; P1 is record number
`00`	data length: 0
`6a 83`	???

4.6 Response 3

`00 a4 04 00 07`	???
`a0 00 00 00 03 20 10`	Application ID of visa electron
`61 35`	read 53 more bytes by GET RESPONSE

4.6.1 command mismatch?

0x00a40400 in the beginning of the response corresponds to Book1 11.3.2 SELECT command (select by name) which is not the same as in the request

4.7 Request 4 (Reading application data)

	MISSING

4.8 Response 4

`00 c0 00 00 35`	???
`6f 33`	template 6f, length 51 bytes
`84 07 a0 00 00 00 03 20 10`	file name: AID of visa electron
`a5 28`	File Control Information (FCI) Proprietary Template, length 40
`50 10 56 49 53 41 45 4c 45 43 54 52 4f 4e 20 20 20 20`	application label 'VISAELECTRON '
`87 01 01`	application priority: 1
`5f 2d 08 65 74 65 6e 72 75 64 65`	language preference: et,en,ru,de
`bf 0c 05 9f 4d 02 0b 14`	issuer url: 0x9f4d020b14
`90 00`	ok

4.8.1 recover request from response?

0x00c00000 in the beginning corresponds to Book1 9.3.1.3 GET RESPONSE command with no parameters

4.9 Request 5

80 a8 00 00	B3 6.5.8.2 GET PROCESSING OPTIONS
02	data len: 2
83 00	get options: empty list
61 0c	???

4.10 Response 5

00 c0 00 00 0c	???
80 0a	tag + length
3c 00	Application Interchange Profile (AIP) bitfield: dynamic data authentication (DDA) supported, cardholder verification supported, perform terminal risk mgmt supported, issuer authentication supported
08 01 01 00	Application File Location: 1, ShortFileIdentifier:1, records to read: range(1..1), offline data authentication records: none
10 01 06 01	Application File Location: 2, ShortFileIdentifier:2, records to read: range(1..6), offline data authentication records: 1
90 00	ok

4.11 Request 6

00 b2 01 0c	B1 11.2.2/B3 6.5.11 READ RECORD P1: record number: 1 ShortFileIdentifier: 1; P1 is record number
00	data length: 0
6c 4f	???

4.12 Response 6

00 b2 01 0c 4f	???
70 4d	tag+len
57 13	track2 equivalent data
49 10 79 21 37 64 61 73 d	card number: 49 10 79 21 37 64 61 73 + terminating 0xD (yes, a nibble)
14 12	expiration date
22 1	service code
15 65 94 29 00 00 0f	"Discretionary Data" (payment system specific)
5f 20 1a	cardholder name
42 41 4b 48 4f 46 46 2f 4d 41 52 54 20 20 20 20 20 20 20 20 20 20 20 20 20 20	'BAKHOFF/MART '
9f 1f 18	Track 1 Discretionary Data
31 35 36 35 39 30 30 30 30 30 30 30 30 30 30 34 32 39 30 30 30 30 30 30	156590000000000429000000?
90 00	ok

4.13 Request 7

00 b2 01 14	B3 6.5.11 READ RECORD P1: record number: 1 ShortFileIdentifier: 2; P1 is record number
00	data length: 0
6c 87	???

4.14 Response 7

00 b2 01 14 87	???
70 81 84	tag + len
5f 25 03 12 10 01	Application Effective Date: (yy/mm/dd) 12 10 01
5f 24 03 14 12 31	Application Expiration Date: (yy/mm/dd) 14 12 31
9f 07 02 ff 00	Application Usage Control bitfield: cash transactions, goods, services, atms, terminal, both domestic and international
5a 08 49 10 79 21 37 64 61 73	Application Primary Account Number (PAN): 4910 7921 3764 6173
5f 34 01 00	Application Primary Account Number (PAN) Sequence Number: 00
8c 15	Card Risk Management Data Object List 1 (CDOL1)
9f 02 06	amount, authorized
9f 03 06	amount, other
9f 1a 02	terminal country code
95 05	Terminal Verification Results
5f 2a 02	transaction currency code
9a 03	transaction date
9c 01	transaction type
9f 37 04	unpredictable number
8d 17	Card Risk Management Data Object List 2 (CDOL2)
8a 02	authorization response code
9f 02 06	Amount, Authorised
9f 03 06	Amount, Other
9f 1a 02	terminal country code
95 05	Terminal Verification Results
5f 2a 02	transaction currency code
9a 03	transaction date
9c 01	transaction type
9f 37 04	unpredictable number
8e 12	Cardholder Verification Method (CVM)
00 00 00 00	amount field
00 00 00 00	second amount field
44 03 01 03 02 03 1e 03 1f 00	cardholder verification rules (2bytes each)
9f 0d 05 b8 60 ac 88 00	Issuer Action Code - Default
9f 0e 05 00 10 00 00 00	Issuer Action Code - Denial
9f 0f 05 b8 68 bc 98 00	Issuer Action Code - Online
9f 4a 01 82	Static Data Authentication Tag List: [Application Interchange Profile]
5f 28 02 02 33	Issuer Country Code: 0x0233
90 00	ok

4.15 Request 8

00 b2 02 14	B1 11.2.2 READ RECORD P1: record number: 2 ShortFileIdentifier: 2; P1 is record number
00	data length: 0
6c e3	???

4.16 Response 8

00 b2 02 14 e3	???
70 81 e0	tag+len
8f 01 08	Certification Authority Public Key Index: 8[7]
90 81 b0	Issuer Public Key Certificate (tag + length)
25 67 fe b4 1a 19 5a 47 69 5b 89 a0 aa 97 3f 7e 8b 69 ab 05 e0 3b c7 e0 5d 10 87 8d fe 6c a3 9b ae 6e 24 96 44 22 98 58 3e ac 91 f5 35 ad 32 8c f3 f6 df ec 3e f5 a4 a8 5a 34 62 ca 4b 28 c6 f7 25 dc 5d 25 bf 39 4c f1 cc 87 1c f9 84 69 85 0d ad 90 c0 32 6e 33 3c 5f ec 1c 7a 2c 6e 1d 4f c1 4e 61 a7 0a 30 6c d5 17 12 a7 c4 01 35 32 67 69 18 b3 4b 71 80 a6 6d a0 d7 ac f2 5d 6f 42 9e fd 33 50 cc 62 7e 15 f2 0e 03 1f 28 62 6f 3d f7 c8 e0 49 bf aa 40 88 b1 c5 74 8c 33 39 f4 3b bc 73 db 89 cb a4 42 33 5c 4c 31 36 27 90 0b 0f b6 43	Issuer Public Key Certificate
9f 32 01 03	Issuer Public Key Exponent: 3
92 24	Issuer Public Key Remainder (tag + length)
8b e3 00 b1 09 ba 1e 63 91 46 82 e1 ad c0 52 c2 5a 16 64 9b d2 d2 b0 0b 85 27 1b 66 c3 e4 77 77 84 c9 ca ad	Issuer Public Key Remainder
90 00	ok

4.16.1 issuer certificate decrypted with VSDC CA Public key

```
6a 02 49 10 79 ff 12 20 03 20 16 01 01 b0 01 89
a6 e7 18 62 67 69 62 9c 4f 02 6a 18 5a 7d 60 f0
32 96 c7 00 06 ba 27 1f 12 e1 c1 b3 c1 72 9b 82
59 d7 dc 04 5b 26 68 12 f4 89 10 e5 78 5f 9a bd
27 e6 df ae 5b e1 7c 5f 7a 97 6b 76 d7 f5 c8 0a
19 1c ec 2e 2b 0d 01 8f 55 a7 20 17 1b e0 8e be
```

```
2b 2c 5f ca df fc bc 06 9a de e0 d1 0e 73 0f ec
db 4f 2e 22 04 5a 6c 08 a7 cc fe ae f9 af 3e c3
27 f8 52 f8 ce fe c0 d6 b9 e4 42 23 49 c7 e8 7d
31 8a 73 97 72 f2 db 2d a7 18 e0 4d 60 3c 23 cc
43 c4 84 fa bb 84 86 80 c8 fb d2 a9 b4 00 e9 bc
```

4.16.2 extracted issuer modulus (Book2 6.3)

```
89 a6 e7 18 62 67 69 62 9c 4f 02 6a 18 5a 7d 60
f0 32 96 c7 00 06 ba 27 1f 12 e1 c1 b3 c1 72 9b
82 59 d7 dc 04 5b 26 68 12 f4 89 10 e5 78 5f 9a
bd 27 e6 df ae 5b e1 7c 5f 7a 97 6b 76 d7 f5 c8
0a 19 1c ec 2e 2b 0d 01 8f 55 a7 20 17 1b e0 8e
be 2b 2c 5f ca df fc bc 06 9a de e0 d1 0e 73 0f
ec db 4f 2e 22 04 5a 6c 08 a7 cc fe ae f9 af 3e
c3 27 f8 52 f8 ce fe c0 d6 b9 e4 42 23 49 c7 e8
7d 31 8a 73 97 72 f2 db 2d a7 18 e0 8b e3 00 b1
09 ba 1e 63 91 46 82 e1 ad c0 52 c2 5a 16 64 9b
d2 d2 b0 0b 85 27 1b 66 c3 e4 77 77 84 c9 ca ad
```

4.17 Request 9

00 b2 03 14	B1 11.2.2 READ RECORD P1: record number: 3 ShortFileIdentifier: 2; P1 is record number
00	data length: 0
6c 0c	???

4.18 Response 9

00 b2 03 14 0c	???
70 0a	tag + len
9f 49 03 9f 37 04	Dynamic Data Authentication Data Object List (DDOL): [Unpredictable Number]
9f 47 01 03	ICC Public Key Exponent: 3
90 00	ok

4.19 Request 10

00 b2 04 14	B1 11.2.2 READ RECORD P1: record number: 4 ShortFileIdentifier: 2; P1 is record number
00	data length: 0
6c b7	???

4.20 Response 10

00 b2 04 14 b7	???
70 81 b4	tag + len
9f 46 81 b0	ICC Public Key Certificate (tag + length)
02 da aa 32 47 c9 76 e4 d4 d0 28 76 4e 1a 09 55 60 54 e5 86 54 17 b0 98 04 fd 70 9a 1e c4 0c 18 69 8f 49 a3 43 c1 01 b6 0c 70 0b 6e 64 55 fe 8c 72 11 c2 8f 47 5b 4c 6f 8c 3d 9d ef 40 bd de a2 bd f6 a5 64 68 06 70 88 a3 63 9c 0a cc 7a 32 48 f7 59 1a 9e c2 12 5f 35 39 94 e9 68 03 10 50 a8 c0 e7 98 0f 43 f5 5b b2 b0 5b c9 ef b7 4e 78 68 fb 57 33 e2 20 55 08 f0 8c 0e 12 e9 8c 3d 36 2d 20 0f 3b 15 00 96 84 c6 8b 88 81 dc 0c 23 ff 71 4e 70 01 10 81 ef ed c2 6e d9 a4 eb fe 3d 90 ab 2a 0a c4 24 82 69 49 09 f3 d5 0b d2 18 23 36 ed	ICC Public Key Certificate
90 00	ok

4.20.1 decrypted icc certificate using issuer public key (Book2 6.4)

```
6a 04 49 10 79 21 37 64 61 73 ff ff 12 14 38 46
34 01 01 80 01 9c 4a c0 dd 6e 40 79 a6 2b 08 d7
45 48 14 26 19 64 3f ca 06 5a 70 14 0b 9a d2 c3
fb 71 c3 4c dc ee 3d f9 ef d5 9d e7 c3 a0 eb 19
17 c9 ba ba de 6d 66 eb 03 9c 77 a4 6c aa 5f 5d
78 c4 9c f2 23 cb e2 71 7d 2f ca f2 97 a2 4e d8
fb 9b d5 21 39 0a d3 d1 be 41 27 c8 7d 03 cd 93
4d dc b6 b1 cb 19 91 1c 7b 89 37 ca 31 fa bc ab
b2 4a 2a f9 c2 32 c3 73 aa e8 fc a8 68 7f 2d ef
b1 07 f6 1d 2d bb bb bb bb bb bb 18 ea d1 99 e4
17 cc ac b2 14 42 65 98 9a c3 d8 37 23 a9 6a bc
```

4.20.2 extracted icc modulus

```
9c 4a c0 dd 6e 40 79 a6 2b 08 d7 45 48 14 26 19
64 3f ca 06 5a 70 14 0b 9a d2 c3 fb 71 c3 4c dc
ee 3d f9 ef d5 9d e7 c3 a0 eb 19 17 c9 ba ba de
6d 66 eb 03 9c 77 a4 6c aa 5f 5d 78 c4 9c f2 23
cb e2 71 7d 2f ca f2 97 a2 4e d8 fb 9b d5 21 39
0a d3 d1 be 41 27 c8 7d 03 cd 93 4d dc b6 b1 cb
19 91 1c 7b 89 37 ca 31 fa bc ab b2 4a 2a f9 c2
32 c3 73 aa e8 fc a8 68 7f 2d ef b1 07 f6 1d 2d
```

4.21 Request 11

00 b2 05 14	B1 11.2.2 READ RECORD P1: record number: 5 ShortFileIdentifier: 2; P1 is record number
00	data length: 0
6c bb	???

4.22 Response 11

00 b2 05 14 bb	???
70 81 b8	tag + len
9f 2d 81 b0	ICC PIN Encipherment Public Key Certificate (tag + length)
2d 54 34 a8 b5 ff 42 53 af fd 9f df 51 74 c3 a7 51 b8 39 cb 6b a9 1f c6 d3 62 9e e9 bd c5 ba 55 a1 3c 91 8c 41 47 08 8c 42 46 1d 76 73 27 d8 a1 88 d3 2f 55 fa b5 21 8d 91 96 35 d3 bd db ed 31 2b 1b e3 aa 9a ea 2b 85 6c 4d 16 52 0b 16 74 fe 14 83 4f f4 29 8b fe 09 a1 82 7f 33 9e a9 d7 42 f7 34 19 5b dc 47 47 c2 8d 78 74 0f 01 bd cf b2 f0 c6 9a 8f af 15 30 76 37 59 af 38 38 95 c3 f0 4f 46 d4 fe f5 d3 1e dc 02 26 dd 48 94 a0 47 dd 6a 6d c0 7b 02 03 d8 b8 4a c6 d5 e6 9b 10 f8 54 78 63 0b cc 06 56 7a eb 55 c3 89 48 69 6e 85 3d	ICC PIN Encipherment Public Key Certificate
9f 2e 01 03	ICC PIN Encipherment Public Key Exponent: 3
90 00	ok

4.22.1 decrypted PIN certificate using issuer public key (Book2 7.1)

```
6a 04 49 10 79 21 37 64 61 73 ff ff 12 14 38 46
34 01 01 80 01 b8 02 c0 05 d0 5e f2 05 85 57 1e
54 ad ab be 02 7a 74 d6 03 22 e2 3d 2a c8 1d 21
8e 21 c8 48 5d 7a 72 64 fe 40 34 ec 9f a2 0a b1
9c ba f7 ec 2b 4e 22 25 17 b8 4f 5a 82 0c b9 76
01 83 34 22 36 1e 54 ee f0 75 0d 76 f6 fb ea 91
bd 3c bf 70 98 51 b7 d4 5b c2 6e 40 80 1f 12 c9
87 62 1b 44 25 1d 9d a1 d0 e0 94 69 de 18 b6 dc
a5 c4 79 b3 b0 cb ac a2 d0 43 e4 c7 f0 db b1 0b
17 30 cc f6 f5 bb bb bb bb bb bb f4 b9 ac aa f0
55 71 7c cf 5b a8 81 89 46 c0 06 2a 3f ac 39 bc
```

4.22.2 extracted PIN modulus

```
b8 02 c0 05 d0 5e f2 05 85 57 1e 54 ad ab be 02
7a 74 d6 03 22 e2 3d 2a c8 1d 21 8e 21 c8 48 5d
7a 72 64 fe 40 34 ec 9f a2 0a b1 9c ba f7 ec 2b
4e 22 25 17 b8 4f 5a 82 0c b9 76 01 83 34 22 36
1e 54 ee f0 75 0d 76 f6 fb ea 91 bd 3c bf 70 98
51 b7 d4 5b c2 6e 40 80 1f 12 c9 87 62 1b 44 25
1d 9d a1 d0 e0 94 69 de 18 b6 dc a5 c4 79 b3 b0
cb ac a2 d0 43 e4 c7 f0 db b1 0b 17 30 cc f6 f5
```

4.23 Request 12

00 b2 06 14	B1 11.2.2 READ RECORD P1: record number: 6 ShortFileIdentifier: 2; P1 is record number
00	data length: 0
6c 15	???

4.24 Response 12

00 b2 06 14 15	???
70 13	tag + len
9f 08 02 00 8c	Application Version Number: 0x008c
5f 30 02 02 21	Service Code: 0x0221
9f 42 02 09 78	Application Currency Code: 0x0978
9f 44 01 02	Application Currency Exponent: 2
90 00	ok

4.25 Request 13 (Dynamic Data Authentication)

00 88 00 00	B3 6.5.9.2 INTERNAL AUTHENTICATE
04	data length: 4
d6 83 42 17	DDOL data: Unpredictable number 0xd6834217
61 83	???

4.26 Response 13

00 c0 00 00 83	???
80 81 80	Signed Dynamic Application Data (tag + length)
43 c5 b4 a5 18 b7 27 b4 09 aa dc 83 02 5c 48 11 77 7f af 49 1a 6f 1f c1 87 03 43 4c 89 5d a3 bc 64 9c e6 ef 6d 6a 32 f5 3c ef 51 e6 9e 0d 97 8b 1a ff 2b 5a 7c 36 93 3f 37 4b 74 73 27 08 bf 8a e8 2a 4f 5f 90 bf 7e 7d e3 81 bb 10 ae 1c e8 81 08 18 9e d0 6e 05 e9 e1 ee 1d 2a 97 41 ab 23 db b1 3f 09 e0 34 9d bd 58 92 e8 4e 72 76 ad 41 ae f3 1a d3 49 8a 6f bd 65 df 6f 0c 20 83 fd db 5f	Signed Dynamic Application Data
90 00	ok

4.26.1 DDA response decrypted with ICC public key (B2 6.5.2)

```
6a 05 01 09 08 8a df fb 90 a8 a9 77 11 bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb
bb bb bb bb bb bb bb bb bb bb bb 29 66 c2 93 45
44 bd 9d e5 03 f2 b4 f2 9a bf 7a cb 0d a4 e0 bc
```

4.27 Request 14 (Cardholder verification)

80 ca 9f 17	B3 6.5.7.2 GET DATA: PIN try counter
00	data len: 0
6c 04	???

4.28 Response 14

80 ca 9f 17 04	???
9f 17 01 03	PIN Try Counter: 3 remaining
90 00	ok

4.29 Request 15

00 84 00 00	B3 6.5.6.2 GET CHALLENGE
00	data len: 0
6c 08	???

4.30 Response 15

00 84 00 00 08	???
6e 46 d1 ff 7f 6e 61 30	8-byte unpredictable number generated by the ICC
90 00	ok

4.31 Request 16

00 20 00 88	B3 6.5.12.2 VERIFY: encrypted PIN (B2)
80	data len: 128
27 82 e7 f7 1b 5f 5d 7c b3 cf ba 85 d2 4d 6d 41 59 fa c4 b2 69 96 8b d5 f9 46 69 f9 e7 0c 9b 43 79 40 a8 0d 90 f4 73 c9 7b 4a 24 82 68 ef 99 a6 7c cd a0 32 6f b2 94 70 fe 9c 1c 7a ae 86 75 fd c2 36 5e ee 24 80 f5 5f 8b 85 88 05 09 ec 04 86 0a bc de ad 60 3f ce ac f0 c7 68 ac 5f 1e ff ba 06 b3 6b 9a 7a 58 ea 61 df bf 72 a6 d6 0c 81 98 08 d3 c0 71 42 8d df c2 fc 61 17 ae e0 3e 31 a0	encrypted PIN
90 00	???

4.32 Response 16

	MISSING

4.32.1 recover status code from request?

The request ends with an unusual 0x9000 - maybe that's the status code of the response

4.33 Request 17 (Online processing)

80 ae 80 00	GENERATE AC: ARQC (B3 6.5.5.2)
1d	data length: 29
00 00 00 00 00 99	amount, authorized
00 00 00 00 00 00	amount, other
02 33	terminal country code
00 00 00 80 00	Terminal Verification Results: transaction exeeds floor limit
09 78	transaction currency code
14 09 25	transaction date
00	transaction type
d6 83 42 17	unpredictable number
61 20	???

4.34 Response 17

00 c0 00 00 20	???
77 1e	tag + len
9f 27 01 80	Cryptogram Information Data: 0x80 (ARQC)
9f 36 02 03 77	Application Transaction Counter (ATC): 0x0377
9f 26 08	Application Cryptogram (tag + length)
ac 74 08 bb 16 b2 b8 6d	Application Cryptogram
9f 10 07	Issuer Application Data (tag + length)
06 01 0a 03 a4 20 02	Issuer Application Data

4.34.1 CDOL1

request data defined in CDOL1

4.35 Request 18

00 82 00 00	EXTERNAL AUTHENTICATE (B3 6.5.4)
0a	data length
83 1c 2b df 91 08 e0 70 30 30	Issuer Authentication Data
90 00	???

4.36 Response 18

	MISSING

4.37 Request 19 (Transaction authorization)

80 ae 40 00	Generate AC: Transaction Certificate (B3 6.5.5.2)
1f	data len: 31
30 30	authorization response code
00 00 00 00 00 99	amount, authorized
00 00 00 00 00 00	amount, other
02 33	terminal country code
00 00 00 80 00	Terminal Verification Results: transaction exeeds floor limit
09 78	transaction currency code
14 09 25	transaction date
00	transaction type
d6 83 42 17	unpredictable number
61 20	???

4.38 Response 19

00 c0 00 00 20	???
77 1e	tag + len
9f 27 01 40	Cryptogram Information Data: 0x40 (TC)
9f 36 02 03 77	Application Transaction Counter (ATC): 0x0377
9f 26 08	Application Cryptogram (tag + length)
c2 f1 92 98 bd 19 a7 fe	Application Cryptogram
9f 10 07	Issuer Application Data (tag + length)
06 01 0a 03 64 20 02	Issuer Application Data
90 00	ok

4.38.1 CDOL2

request data defined in CDOL2