Relaying EMV Contactless Transactions using Off-The-Shelf Android Devices

Jordi van den Breekel
KPMG the Netherlands

BlackHat Asia
March, 2015

vandenBreekel.Jordi@KPMG.nl

Abstract

Dutch banks introduced contactless payments in April 2014, and have been promoting the use of contactless cards since then. Contactless payments are based on the EMV specification, the worldwide standard for contact and contactless transactions. EMV Contact is a well-researched field and many vulnerabilities have been found. Although EMV Contactless is newer and less researched, a few vulnerabilities have already been identified. All known EMV Contactless vulnerabilities exist in legacy modes that EMV provides. These modes, however, are not supported in the Netherlands and thus these vulnerabilities are not applicable to Dutch cards or terminals.

We present the first vulnerabilities in EMV Contactless that are applicable to Dutch cards and terminals. We show that a relay attack can be performed with very limited resources and widely available off-the-shelf hardware. Our proof-of-concept relay attack proves that a criminal can pay at a Point-of-Sale terminal, using the card inside a wallet of a victim, while the victim is arbitrary far away from the terminal. All Dutch cards are vulnerable to our proof-of-concept relay attack.

1 Introduction

Several major banks in the Netherlands have enabled contactless payments for small transactions with NFC-supporting terminals for their customers in 2014. NFC stands for near field communication and it is the next development in payment methods after EMV Contact cards and magnet stripes. EMV is a global standard that strives for compatibility between all bank cards and terminals in the world and is defined and managed by the private corporation EMVCo. EMVCo is named after its founders Europay, MasterCard and Visa and manages the standards for both contact cards and contactless cards.

EMV Contact cards have been introduced to reduce the exponential growth of skimming damages of magnet stripes. The roll-out of contactless cards is not necessary for security reasons but research suggests that by improving user experience, customers are likely to spend 30% more money[1]. Questions arise about the security and reliability with almost every introduction of new techniques, especially if they involve financial transactions. A

[1] `http://newsroom.mastercard.com/press-releases/new-mastercard-advisors-study-on-contactless-payments-shows-almost-30-lift-in-total-spend-within-first-year-of-adoption/`

new step in technology must not mean a step back in security properties. Intuitively, a protocol for contactless transactions requires additional security measures than a protocol for contact chip transactions, since the contactless nature introduces new attack factors.

Scope This paper focuses on all EMV Contactless cards available in the Netherlands. EMV Contactless implementations in mobile phones are not considered.

Outline The remainder of this thesis is structured as follows: Section 2 gives an explanation of EMV Contactless transaction including the transaction communication. Section 3 describes the relay setup that we used, and how we optimized the performance of the relay attack. Section 4 discusses the results and performance of our relay setup. Section 5 gives the conclusions of our research and this paper.

2 EMV Contactless

EMV Contactless is the latest standard of bank cards. It is not placed as the successor of EMV Contact, but as a faster alternative. The core of the protocol suite of EMV Contactless is still the same as EMV Contact but there are some differences in options, possibilities, supported features and of course a different interface. While EMV Contact uses the contact chip in bank cards, EMV Contactless uses the Radio-Frequency IDentification (RFID) chips available in the newer bank cards. These RFID chips can communicate and carry out transactions with Near Field Communication (NFC) enabled Point-Of-Sale (POS) Terminals.

Bank cards are typically sized according to the ISO/IEC 7810 ID-1 specifications [16]. The contact chips on these cards are based on the ISO/IEC 7816 specifications [21] and the 'contactless integrated circuit' is based on the ISO/IEC 14443 1-4 specifications [17–20].

The EMV Contactless specification is described by the EMV Contactless books [6–8], kernel specifications [9–15], additional documents published by Maestro [22] and to some extent the EMV Contact books [2–5].

The MasterCard contactless implementation PayPass is extensively described in Book C-2 [10]. The PayPass transactions are not too different from standard EMV Contact transactions and the terminology coincides for a large extent. An excellent and detailed description of EMV Contact can be found in the work of De Ruiter and Poll [1]. We will now discuss the basics of EMV Contactless transactions.

EMV Contactless transactions An EMV Contactless transaction begins with the selection of a combination of an application on the card and a kernel on the terminal. An application on the card is a software program that supports the commands, data items and communication protocol for cards as defined by its corresponding kernel in one of the seven kernel books. A kernel on the terminal is a software program that supports the commands, data items and communication protocol for terminals as defined by one of the seven kernel books. Cards can have multiple applications and terminals can have support for multiple kernels. Typically a terminal has support for more than one kernel for compatibility reasons (e.g. in Europe it is very common to support at least Visa and MasterCard), while a card typically has one or two applications.

Contactless transactions performed with Maestro and Visa cards follow slightly different protocols. The following description is based on a contactless transaction performed with a Maestro card.

The 'Handshaking/Negotation' part of a contactless transaction is shown in more detail in Figure 1. Every contactless transaction begins with the card sending its Answer To Reset (ATR) (message 1). The terminal then selects the Proximity Payment System Environment (PPSE) (message 2). The card responds with its list consisting of an Application Identifier (AID) with a priority indicator per supported payment application (message 3). Of the applications that are supported by the terminal, the terminal selects the one with the highest priority (message 4). The card then responds whether the AID was selected correctly (message 5). The terminal requests the processing options (message 6) and the card responds with the Application Interchange Profile (AIP) and Application File Locator (AFL) (message 7). The terminal requests the record indicated by the AFL (message 8) and the card returns these records (message 9).

After the initialization phase, the terminal decides (based on risk assessment) whether to perform an online transaction (Figure 2, action 1), and requests a cryptogram from the card (message 2). The card also performs risk management (action 3), and sends the cryptogram to the POS terminal (message 4). The POS terminal forwards all transaction data and the cryptogram to the bank (message 5), and the bank will either approve or decline the transaction (message 6). Dutch cards and terminals, however, are only allowed to perform online transactions. As a result, actions 1 and 3 are omitted in the Netherlands and message 2 is always a 'Generate AC(ARQC)' message. A complete trace of a Dutch Maestro EMV Contactless transaction (between card and POS terminal) is shown in Appendix A.

3 Relay Attack

We have used the following equipment for our relay attack (as shown in Figure 3): an EMV Contactless card, a Nexus 7 (2012) as card reader, a Nexus 7 (2013) as card emulator, an Omnikey 5321v2 contactless USB reader[2] and a laptop. The card reader together with the laptop are programmed to emulate a POS terminal. This way, transactions can be emulated between card and terminal. From the point of view of a card, the emulated transactions are identical to real transactions. However, the transaction details are not sent to a bank by the emulated reader, as there is no need for that. The card has already left the proximity of the reader, so it would never receive information

[2]https://www.hidglobal.com/sites/hidglobal.com/files/resource_files/omnikey-5321-v2-usb-reader-en-ds.pdf

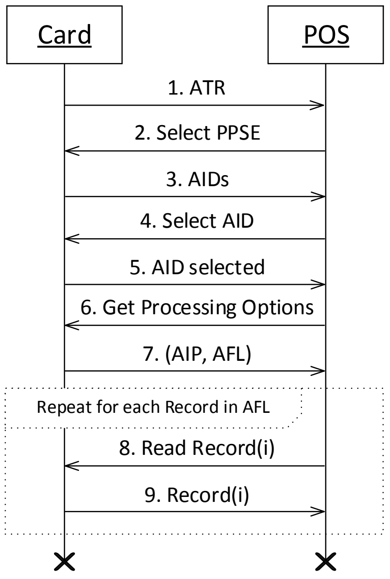

Figure 1. Message sequence chart of the beginning of a Maestro contactless transaction

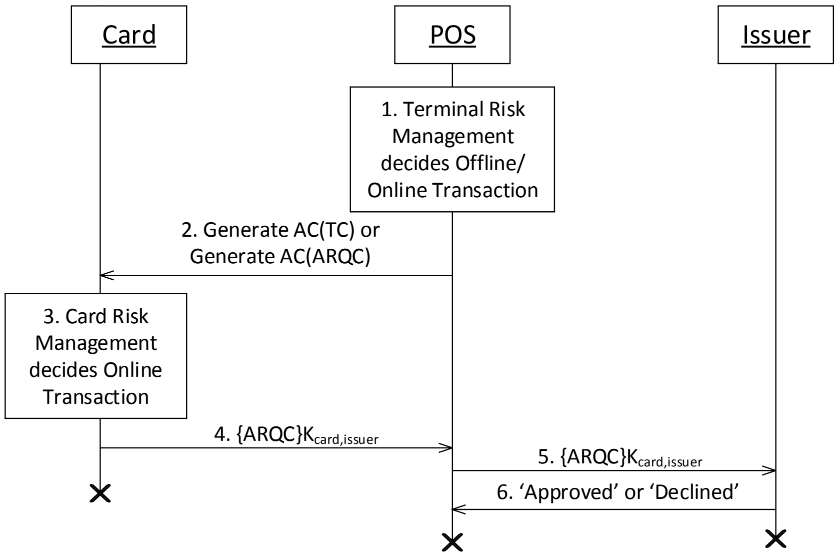
Card
POS
Issuer
1. Terminal Risk Management decides Offline/ Online Transaction
2. Generate AC(TC) or Generate AC(ARQC)
3. Card Risk Management decides Online Transaction
4. {ARQC}K$_{card,issuer}$
5. {ARQC}K$_{card,issuer}$
6. 'Approved' or 'Declined'

Figure 2. Message sequence chart of a Maestro contactless transaction after initialization, with communication to and from the issuer

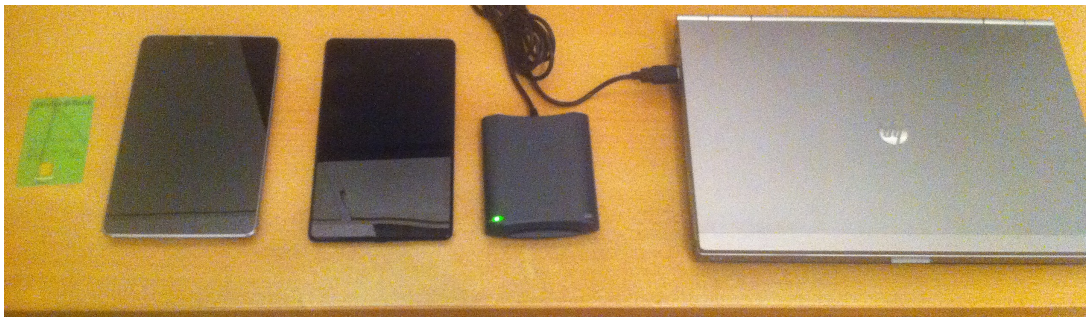

Figure 3. Our test relay setup: (from left to right) bank card, Mole Device, Relay Device, emulated POS terminal)

from the bank even during real transactions. The card emulating device needs Android 4.4 (KitKat) or higher, because since that version Host-based Card Emulation (HCE)[3] is supported.

Figure 4 shows the complete overview of the communication of our relay attack. Intuitively, relaying all communication between the POS terminal and the card adds additional delays. Figure 5 shows the timings of normal transactions, performed directly with a card and a POS terminal (shown in green), and the timings of the relayed transactions with the same cards (shown in red).

Optimizing performance The most intuitive countermeasure against relay attacks is to determine the upper limit of a normal, direct transaction and reject all transactions that take longer, as they are most likely relayed transactions. Therefore, we performed performance optimization to show that this will probably not be a effective countermeasure for Dutch cards.

We discovered that many of the messages in a transaction are static, i.e., they are the same for each transaction with the same card. In Figure 1, all the messages are static and could be cached by the Relay device. In practice, cards in the Netherlands have four records, so messages 8 and 9 are both sent four times each transaction. The only dynamic messages, i.e., messages that are different each transaction, are messages 2 and 4 in Figure 2. Using this information, we created a relay attack that uses less relayed messages but caching instead (Figure 6). This way, we reduced the number of relayed messages from 16 to 2.

Furthermore, some commands can be omitted to be sent to the card in order to perform a transaction. In Figure 1, these commands are messages 2 and 8. As there typically are four records, the total number of commands sent to the card can therefore be reduced from 8 to 3. There is one additional 'Get ready' message to signal the Mole Device that it can already start the initialization of the transaction with the card.

[3]`http://developer.android.com/about/versions/kitkat.html#44-hce`

Additionally, we discovered that there is an aggressive power saving function active in Android for the network adapter. After 100ms of inactivity, it enters some sort of 'sleep mode'. This increases the delay of approximately 40ms per sent and received transaction. This power saving function was activated four times in our model (sending and receiving the 'Generate Cryptogram' command and sending and receiving the actual Cryptogram). We created an asynchronous task in our app that sends (and receives) a 'keep alive' message over the network adapter every 80ms to circumvent the power saving functions. This circumvention led to a reduction of approximately 160ms per relayed transaction.

4 Results

The timing results of our optimized relay attack are shown in Figure 5. All of the transaction times of the optimized relay setup are lower than the transaction time of a normal, direct transaction performed with the slowest card (the ABN 1st generation card). A transaction with the 1st generation ABN card is even faster when the optimized relay setup is used compared to a normal, direct transaction. To the best of our knowledge, this is the first relayed transaction that is actually faster than a direct transaction performed with the same card.

As a POS terminal cannot possibly determine what the normal transaction time for a specific card would be, the upper limit in general cannot be lower than the transaction time needed for a transaction performed directly with the slowest card. As a result, even with a competitive transaction time upper limit, the POS terminal would be unable to detect our optimized relay attack.

5 Conclusions

We have shown that it is very well possible to perform a relay attack with only two off-the-shelf Android devices, and a simple relay app that could be developed in two days. With EMV Contactless and Android specific optimizations to the protocol and the power saving functions respectively,

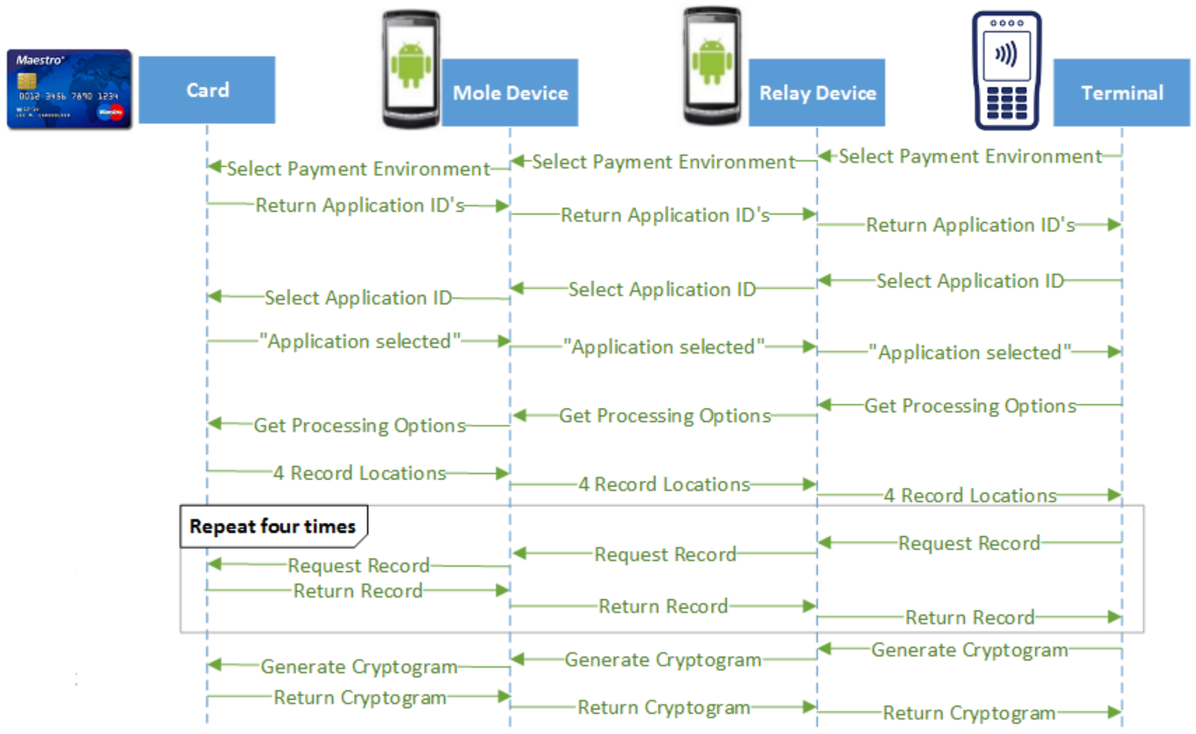

Figure 4. Complete overview of the communication of our relay attack performed on a Maestro card

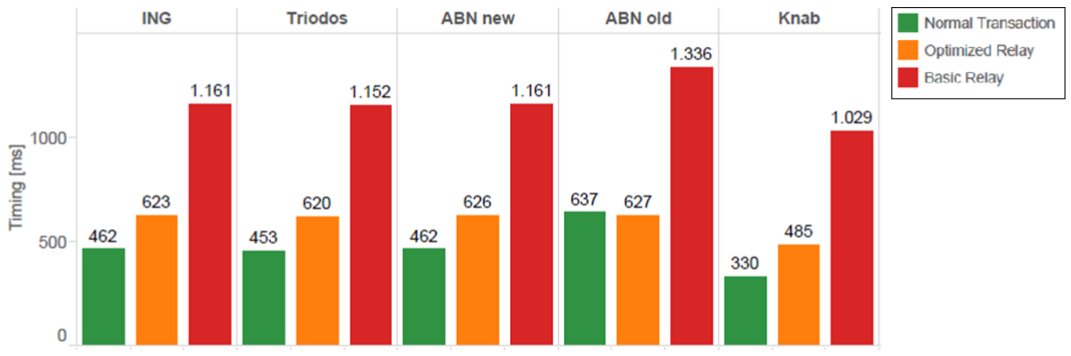

Figure 5. Timings of transactions with Dutch cards, performed directly, with the basic relay setup and with the optimized relay setup

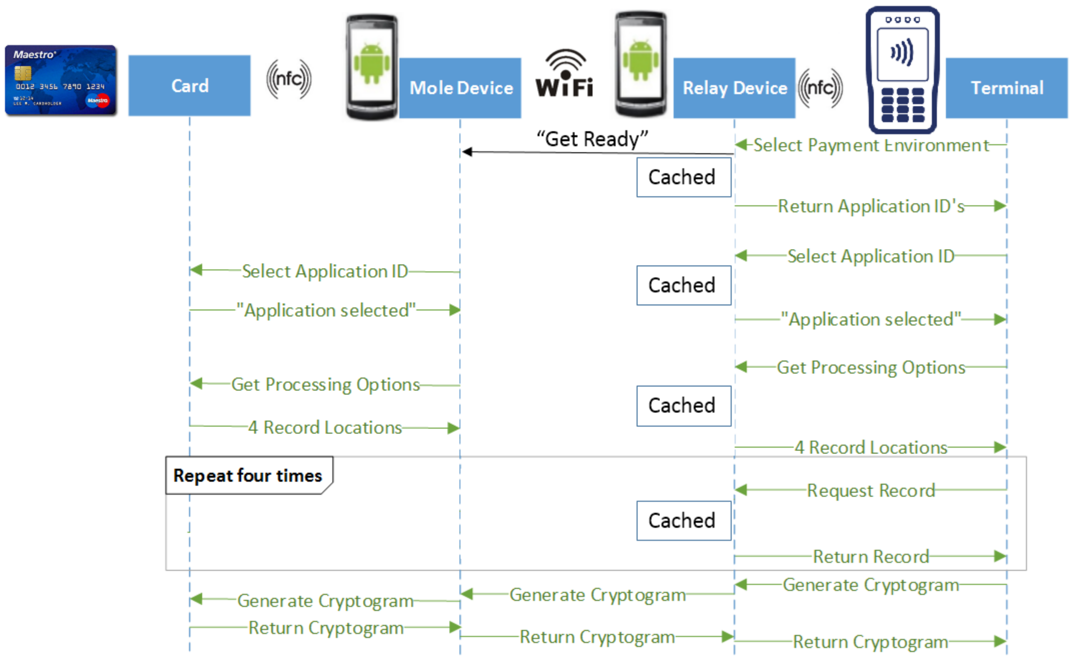

Figure 6. Complete overview of the communication of our optimized relay attack

we successfully created the first relay setup that performs transactions faster than direct transactions with the same card. Furthermore, all relayed transactions are faster than a transaction performed directly with the slowest card. Therefore, relay detection and prevention based on measuring transaction times is not likely to be effective at all.

References

[1] De Ruiter, Joeri and Poll, Erik. Formal analysis of the EMV protocol suite. In *Theory of Security and Applications*, pages 113–129. Springer, 2012.

[2] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 1: Application Independent ICC to Terminal Interface Requirements v4.3*. EMVCo, 2011.

[3] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 2: Security and Key Management v4.3*. EMVCo, 2011.

[4] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 3: Application Specification v4.3*. EMVCo, 2011.

[5] EMVCo. EMV Contact Specifications for Payment Systems. In *Book 4: Cardholder, Attendant, and Acquirer Interface Requirements v4.3*. EMVCo, 2011.

[6] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book A: Architecture and General Requirements v2.4*. EMVCo, 2014.

[7] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book B: Entry Point Specifications v2.4*. EMVCo, 2014.

[8] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book D: Contactless Communication Protocol v2.4*. EMVCo, 2014.

[9] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-1 Kernel 1 Specification v2.4*. EMVCo, 2014.

[10] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-2 Kernel 2 Specification v2.4*. EMVCo, 2014.

[11] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-3 Kernel 3 Specification v2.4*. EMVCo, 2014.

[12] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-4 Kernel 4 Specification v2.4*. EMVCo, 2014.

[13] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-5 Kernel 5 Specification v2.4*. EMVCo, 2014.

[14] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-6 Kernel 6 Specification v2.4*. EMVCo, 2014.

[15] EMVCo. EMV Contactless Specifications for Payment Systems. In *Book C-7 Kernel 7 Specification v2.4*. EMVCo, 2014.

[16] ISO. Identification cards – Physical characteristics. ISO 7810:2003, International Organization for Standardization, Geneva, Switzerland, 2003.

[17] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 1: Physical characteristics. ISO 14443-1:2008, International Organization for Standardization, Geneva, Switzerland, 2008.

[18] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 2: Radio frequency power and signal interface. ISO 14443-2, International Organization for Standardization, Geneva, Switzerland, 2008.

[19] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 4: Transmission protocol. ISO 14443-4:2008, International Organization for Standardization, Geneva, Switzerland, 2008.

[20] ISO. Identification cards – Contactless integrated circuit cards – Proximity cards – Part 3: Initialization and anticollision. ISO 14443-3:2011, International Organization for Standardization, Geneva, Switzerland, 2011.

[21] ISO. Identification cards – Integrated circuit cards – Part 1: Cards with contacts – Physical characteristics. ISO 7816-1:2011, International Organization for Standardization, Geneva, Switzerland, 2011.

[22] MasterCard. PayPass M/Chip Requirements, July 2013. [Available online at https://www.paypass.com/PP_Imp_Guides/PayPass-MChip-Requirements-2013.pdf; accessed 24th June 2014].

Appendices

A Complete Trace of an EMV Contactless Transaction

Bytes that indicate the length of the content of a tag are underlined and not shown in the explanation of the command. Some bytes are anonymized with ‘p’ to hide privacy sensitive information. Some bytes are replaced with ‘....’ where there are many bytes that do not add information (e.g. encrypted data or public keys). Commands from the terminal to the card are (briefly) explained in natural language, as it is more interesting what is commanded rather than how it is commanded. Responses from the card, however, are extensively described.

Terminal → Card (message 2 in Figure 1):
Select 2PAY.SYS.DDF01
```
00A4 04 00 0E 325041592E5359532E4444463031 00
```
Card → Terminal (message 3 in Figure 1):
```
9000 6F 2C 84 0E 325041592E5359532E4444463031 A5 1A BF0C 17 61 15 4F 07
A0000000043060 50 07 4D41455354524F 87 01 01
```

```
9000: Status OK
6F: File Control Information
  84: Dedicated File Name
    325041592E5359532E4444463031: 2PAY.SYS.DDF01
  A5: File Control Information Proprietary Template
    BF0C: File Control Information Issuer Discretionary Data
      61: Application Template
        4F Application Identifier
          A0000000043060
        50: Application Label
          4D41455354524F: MAESTRO
        87: Application Priority Indicator
          01
```

Terminal → Card (message 4 in Figure 1):
Select application with AID: A0000000043060
```
00A4 04 00 07 A0000000043060 00
```

Card → Terminal (message 5 in Figure 1):

```
9000 6F 1E 84 07 A0000000043060 A5 13 50 07 4D41455354524F 87 01 01 5F2D 04
6E6C656E
```

```
9000: Status OK
6F: File Control Information
   84: Dedicated File Name
      A0000000043060
   A5: File Control Information Proprietary Template
      50: Application Label
         4D41455354524F: MAESTRO
      87 Application Priority Indicator
         01
      5F2D Language Preference
         6E6C656E: nlen (Dutch, English)
```

Terminal → Card (message 6 in Figure 1):
Get Processing Options

```
80A8 00 00 02 83 00 00
```

Card → Terminal (message 7 in Figure 1):

```
9000 77 0E 82 02 1980 94 08 0802030018010201
```

```
9000: Status OK
77: Response Message Template Format 2
   82: Application Interchange Profile
      1980
   94: Application File Locator
      08020300: SFI 1, Records 2 and 3, not for offline authentication
      18010201: SFI 3, Records 1 and 2, use Record 1 for offline
                authentication
```

Terminal → Card (message 8 in Figure 1):
Read Record SFI 1, Record 2
```
00B2 02 0C 00
```

Card → Terminal (message 9 in Figure 1):
```
9000 70 1F 9F42 02 0978 9F08 02 0002 57 13 6734000531570525297D19022010000010664F
```

```
9000: Status OK
70: EMV Proprietary Template
   9F42: Application Currency Code
      0978: Euro
   9F08: Application Version Number
      0002
     57: Track 2 Equivalent Data
          67340005ppppppp5297D19022010000010664F: Account number anonymized
```

Terminal → Card (message 8 in Figure 1):
Read Record SFI 1, Record 3
```
00B2 03 0C 00
```
Card → Terminal (message 9 in Figure 1):
```
9000 70 81E0 8F 01 05 9F32 01 03 92 24 5F[34 additional bytes]9F 90 81B0
A0[174 additional bytes]0B
```

```
9000: Status OK
70: EMV Proprietary Template
   8F: Certification Authority Public Key Index
      05
   9F32: Issuer Public Key Exponent
      03
   92: Issuer Public Key Remainder
      5F....9F: 36 bytes total
   90: Issuer Public Key Certificate
      A0....0B: 176 bytes total
```

Terminal → Card (message 8 in Figure 1):
Read Record SFI 3, Record 1
```
00B2 01 1C 00
```

Card → Terminal (message 9 in Figure 1):

```
9000 70 8181 5F25 03 140514 5F24 03 190228 9F07 02 3D00 5A 0A
67340005ppppppp5297F 5F34 01 01 8E 0C 000000000000000042031F03 9F0D 05
B450848000 9F0E 05 0000180000 9F0F 05 B470848000 8C 21
9F02069F03069F1A0295055F2A029A039C019F37049F35019F45029F4C089F3403 8D 0C
910A8A0295059F37049F4C08 5F28 02 0528 9F4A 01 82
```

`9000`: Status OK
`70`: EMV Proprietary Template
  `5F25`: Application Effective Date
    `140514`: 14th May 2014
  `5F24`: Application Expiration Date
    `190228`: 28th February 2019
  `9F07`: Application Usage Control
    `3D00`
  `5A`: Application Primary Account Number (PAN)
    `67340005ppppppp5297F`: PAN anonymized with ‘`p`’
  `5F34`: Application Primary Account Number Sequence Number
    `01`
  `8E`: Cardholder Verification Method List
    `000000000000000042031F03`
  `9F0D`: Issuer Action Code Default
    `B450848000`
  `9F0E`: Issuer Action Code Denial
    `0000180000`
  `9F0F`: Issuer Action Code Online
    `B470848000`
  `8C`: Card Risk Management Data Object List 1
    `9F02069F03069F1A0295055F2A029A039C019F37049F35019F45029F4C089F3403`
  `8D`: Card Risk Management Data Object List 2
    `910A8A0295059F37049F4C08`
  `5F28`: Issuer Country Code
    `0528`: The Netherlands
  `9F4A`: Static Data Authentication Tag List
    `82`

Terminal → Card (message 8 in Figure 1):
Read Record SFI 3, Record 2

```
00B2 02 1C 00
```

Card → Terminal (message 9 in Figure 1):

```
9000 70 81BE 9F49 03 9F3704 9F47 01 03 9F46 81B0 2C[174 additional bytes]11
```

`9000`: Status OK
`70`: EMV Proprietary Template
- `9F49`: Dynamic Data Authentication Data Object List
  - `9F3704`: Unpredictable Number with length 4
- `9F47`: Integrated Circuit Card Public Key Exponent
  - `03`
- `9F46`: Integrated Circuit Card Public Key Certificate
  - `2C....11`: 176 bytes total

Terminal → Card (message 2 in Figure 2):
Generate Application Cryptogram of type Transaction Certificate, with Combined Dynamic Data Authentication (CDA) performed

```
80AE 50 00 2B 000000000001 000000000000 0528 8000000080 0978
140626 00 nnnnnnnn 22 0000 0000000000000000 3F0000 00
```

`80AE`: `GENERATE AC`
- `90`: Application Cryptogram (AC) type: Authorization Request Cryptogram (ARQC) with CDA
  - `000000000001`: Amount, Authorized: 0.01
  - `000000000000`: Amount, Other: 0.00
  - `0528`: Terminal Country Code: The Netherlands
  - `0000000080`: Terminal Verification Results
  - `0978`: Transaction Currency Code: Euro
  - `140626`: Transaction Date: 26th June 2014
  - `00`: Transaction Type: Default value
  - `nnnnnnnn`: Unpredictable Number
  - `21`: Terminal Type: Attended, merchant operated, online only
  - `0000`: Data Authentication Code
  - `0000000000000000`: ICC Dynamic Number
  - `1F0302`: Cardholder Verification Method Results

Card → Terminal (message 4 in Figure 2):

```
9000 77 81A2 9F27 01 80 9F36 02 0070 9F4B 81B0 81[126 additional bytes]74 9F10
12 2B11A04003223000000000000000000000FF
```

`9000`: Status OK
`77`: Response Message Template Format 2
  `9F27`: Cryptogram Information Data
    `80`: AC type: ARQC
  `9F36`: Application Transaction Counter
    `00C5`: 197th transaction
  `9F4B`: Signed Dynamic Application Data
    `1B....6E`: 128 bytes total
  `9F10`: Issuer Application Data
    `2B11A04003220000000000000000000000FF`