Implementation Guide

# Issuer Host implementation Guide For EMV Contact

Version 1.3
Date: 13-June-2019

payhuddle

**Table of Contents**



`

`

`

**Table of Figures**



`

**Document Control**

| Date | version no. | Author | Revision |
|---|---|---|---|
| 02-Mar-2019 | 1.0 | PayHuddle | First Draft |
| 10-Apr- 2019 | 1.1 | PayHuddle | Updated with other sections |
| 15-Apr -2019 | 1.2 | PayHuddle | Script processing, Appendix added |
| 13-June-2019 | 1.3 | PayHuddle | Updated |

`

# 1. Introduction

EMVCo has specified standards to secure payment transactions from counterfeiting card. Every issuer should issue and process EMV card and transaction in order to provide payment security to the consumer. Every payment scheme implements and ensures EMVCo standards for their stakeholders with some specific business requirement changes. Innova solution having issuer host intend to migrate to EMV for chip card transaction processing. This document shall guide them to implement EMV in Innova Solution host.

# 2. Purpose

The purpose of this document is to guide the host developers to implement EMV to process chip transactions.

# 3. Scope

This document shall depict the implementation requirement in issuer host to process EMV transaction in accordance to EMV 4.3 for below payment schemes to process contact transaction.

- Visa
- Mastercard
- RuPay

# 4. Audience

The audience of this document shall be

- Issuer Host Developer
- Issuer service provider

# 5. Overview of EMV Contact Transaction

An EMV transaction is secured and differentiated from normal magstripe transaction because of below reasons

- Card authentication
- Issuer authorization
- Issuer authentication by card.

Issuer plays major role in chip card transaction by controlling chip card with EMV level security and cross authentication of the transaction by both issuer host and card in terminal.

EMV uses cryptography method to secure transaction by generating unique cryptogram for every transaction request and response which gets validated by issuer and card respectively.

`

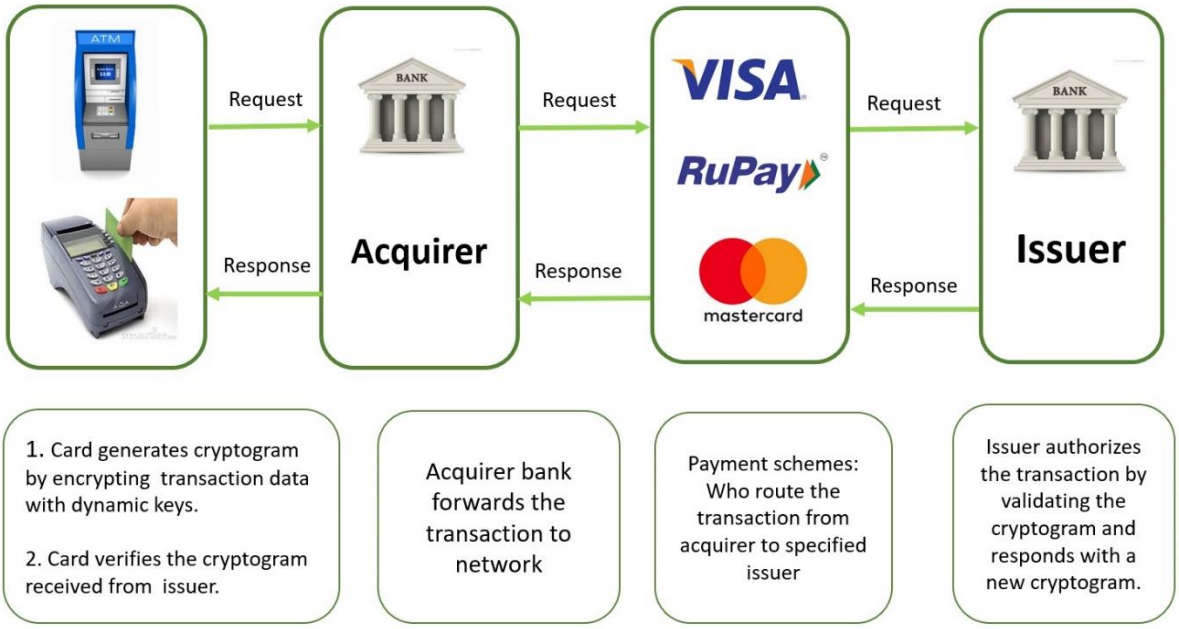



Figure 1: EMV Transaction Flow

## 6. Issuer responsibilities

In an EMV transaction issuer performs authorization and authentication of transactions. In general issuer performs below responsibilities

- Receiving and processing request message (0200/0100)
- Perform PIN validation
- Perform account and balance validation
- Perform risk management
- Perform Authorization request cryptogram validation
- Generate Authorization response cryptogram validation
- Prepare issuer script if any to update data in card for taking action like blocking or unblocking application.
- Prepare issuer authentication data.
- Sends response

`

# 7. Processing Requirements for Host

EMV transaction is processed and authorized by issuer based on the chip information received. The information from card and terminal are transported to issuer by some dedicated fields in ISO 8583 message. Chip data are represented in TLV (Tag Length Value) format. Every payment scheme has set of mandatory tags to be sent in request to issuer in order to enable issuer to verify and authorize transaction after performing EMV security evaluation.

There are three data elements in ISO 8583 carries EMV transaction information. The fields are 22, 23 and 55.
Field 22 helps to identify if it is chip transaction. This field has defined value to identify chip transaction.
Field 23 carries Pan sequence Number
Field 55 carries EMV tags i.e. Chip data.

## 7.1. Visa

### 7.1.1. Field 22- PoS Entry Mode

| Attributes | | | |
|---|---|---|---|
| Length | 4 | | |
| Format | Numeric | | |
| Position | Position 1-2 | Position 3 | Position 4 |
| Subfields | PAN entry mode | PIN Entry capability | Fill |
| Values | 05- Chip read at terminal | 1- Terminal can accept PIN | 0 |
| | 95 – Chip read but data unreliable | 2 – Terminal cannot accept PIN | |
| | 90 – Fallback to magstripe | 8 – Terminal PIN pad down | |

### 7.1.2. Field 23 – PAN Sequence Number

PAN sequence number is used to differentiate card when multiple card has same PAN. PAN sequence number is personalized in card in Tag 5F34 and travels in request message to issuer.
PAN sequence number is also required to derive UDK from ICC Master key to validate ARQC.
Issuer shall send PAN sequence number along with PAN to HSM to validate ARQC for a particular transaction.
If PAN sequence number is absent in request received by Issuer shall fill with zero and send to HSM.

| Attributes | |
|---|---|
| Length | 3 |
| Format | Numeric |
| Values | 000-099 |
| Presence | Mandatory for chip transaction (DE22- 05X / 95X) |

`

### 7.1.3. Field 55 – ICC Related Data

| Attributes | | | | |
|---|---|---|---|---|
| Length | 1-byte Binary + 255 Bytes Variable | | | |
| Format | Hex | | | |
| Position | Byte 1 | Byte 2 | Byte 3 -4 | Byte 5-256 |
| Subfield | Length | Data set ID | Dataset Length | Chip data in TLV format |
| Presence | Mandatory for chip transaction (DE22- 05X / 95X) | | | |

Length (Byte 1) – This is one-byte binary subfield that contains number of bytes present after the length sub field.

Dataset ID – This is one-byte binary identifier given to each data set. The value is hexadecimal 01

Dataset length - This is 2 bytes binary subfield that contains the total length of TLVs.

Chip Data – This maximum 252-byte hexadecimal field where all EMV tags populated in Tag length and value format.

Tag – An EMV tag can be of one byte or two byte long. Whether the tag is of one byte or two bytes can be known from the last five bits of (bits 4 -8) of the first byte. If all the five bits are set to 1 then next byte is part of the tag, else not.

For example: -

Tag 9F06 – Here 9F in binary 10011111. So, bit 1 to 5 is set to 1. So, 06 is part of the tag. So full tag is 9F06

Tag 95 – Here 95 in binary 10010101. So, bit 4 to 8 is not set to 1. So, tag is 95.

Length- Length of the tag is represented in 1 byte or 2 bytes. Most EMV tag length are represented in 1 byte only.
When bit b8 of the most significant byte of the length field is set to 0, the length field consists of only one byte. Bits b7 to b1 code the number of bytes of the value field. The length field is within the range 1 to 127.
When bit b8 of the most significant byte of the length field is set to 1, the subsequent bits b7 to b1 of the most significant byte code the number of subsequent bytes in the length field. The subsequent bytes code an integer representing the number of bytes in the value field. Two bytes are necessary to express up to 255 bytes in the value field

Value- Value of the tag (chip card data) in hexadecimal format.

| Tag | Description | Length | MTI | Presence |
|---|---|---|---|---|
| 9F02 | Amount Authorized | Fixed length<br>12 N, 4-bit BCD (unsigned packed)<br>6 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F03 | Amount Other | Fixed length<br>12 N, 4-bit BCD (unsigned packed);<br>6 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |

`

| | | | | |
|---|---|---|---|---|
| 5F2A | Transaction Currency code | Fixed length<br>3N, 4-bit BCD,<br>2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 82 | Application Interchange profile | Fixed length<br>16-bit string; 2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 95 | Terminal Verification result | Fixed length<br>5 bytes; | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9A | Transaction Date | Fixed length<br>6N, 4-bit BCD; 3 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9C | Transaction Type | Fixed length<br>2N, 4-bit BCD<br>1 byte | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F10 | Issuer Application Data | Variable length<br>Length 1-byte binary +<br>Up to 32 bytes of value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F1A | Terminal country code | Fixed length<br>3N, 4-bit BCD; 2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F26 | Application Cryptogram | Fixed length<br>16 hexadecimals digits; 8 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F33 | Terminal Capability | Fixed length<br>3 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F36 | Application Transaction Counter | Fixed length<br>4 hexadecimal digits,<br>2-byte binary value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F37 | Unpredictable Number | Fixed length<br>8 hexadecimal digits;<br>4 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F27 | Cryptogram Information Data | Fixed length<br>1-byte length +<br>1 byte of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | C |
| 9F34 | CVM results | Fixed length<br>1-byte length +<br>3 bytes of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | C |
| 9F35 | Terminal Type | Fixed length<br>1-byte length +<br>1 byte of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | C |
| 84 | Dedicated file name | Variable length<br>1-byte Binary(length) +<br>5 bytes to 16 bytes value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 91 | Issuer Authentication Data | Variable length<br>1-byte Binary(length) +<br>8 byes to 16 bytes value | 0110 /0210 / 0120 | M |

`

| | | | | |
|---|---|---|---|---|
| 71 | Issuer Script Template 1 | Variable length<br>1 byte +<br>510 hexadecimals digits, maximum 256 bytes | 0110 / 0210 | C |
| 72 | Issuer Script template 2 | Variable length<br>1 byte +<br>510 hexadecimals digits, maximum 256 bytes | 0110 / 0210 | C |
| 9F5B | Issuer script result | Variable length<br>1-byte binary (length) + 40 hexadecimals; Maximum 20 bytes of value | 0400/ 0420 | C |
| C0 | Secondary PIN block | 8 bytes | 0100 / 0200 | C |

Note: Tag 9F03 shall be present for cashback transaction with cashback amount. For other transaction type tag 9F03 may be absent or populated with Zero. If tag 9F03 is absent issuer shall consider tag 9F03 value as zero and send to HSM for ARQC validation.

**Note:**

1. The hexadecimal representation is given here. Every two positions of hexadecimal data are one byte of binary data.
2. Lengths are in binary format.
3. The Total Sub Element Length is the sum of the sub element's ID, length, and data subfields.
4. Issuer shall be able to process any other valid tags coming in request message other than tags mentioned in above table.

### 7.1.4. Issuer Authentication Data

Issuer authentication data is represented in Tag 91 and populated in DE55 in response by issuer. Issuer shall populate tag 91 when issuer authentication is performed by validating ARQC and generating ARPC. There are several ways to build tag 91 after receiving response from HSM with ARPC.

<table>
<tr><th colspan="2">Attributes</th></tr>
<tr><td>Length</td><td>10 bytes</td></tr>
<tr><td>Format</td><td>16 hexadecimal digits + 2 AN EBCDIC (ARC)</td></tr>
<tr><td>Position</td><td>
<table>
<tr><th>Type of VSDC Card</th><th colspan="3">Byte Content</th></tr>
<tr><td>VIS (CVNs: 10, 12, 50–59)</td><td colspan="2">ARPC cryptogram</td><td>ARPC response code</td></tr>
<tr><td rowspan="2">CCD or VIS CVN 18</td><td>ARPC cryptogram<br>Bytes 1–4</td><td>CSU<br>Bytes 5–8</td><td rowspan="2">ARPC response code</td></tr>
<tr><td colspan="2">Bytes 1–8</td><td>Bytes 9–10</td></tr>
</table>
</td></tr>
</table>

This issuer authentication data shall be populated in DE55 Tag 91 and De 139.

`

### 7.1.5. Third bitmap Fields

Other than above fields Visa also transports EMV tags in dedicated ISO fields. Acquirer and Issuer supporting third bitmap sends these fields populating value from DE55 tags.
Issuer shall not decline transaction if transaction request is received without these fields.
But Issuer shall be able to process and store these fields while sent in request messages.

| Field | Format | DE 55 Tag | Mandatory MTI | Optional |
|---|---|---|---|---|
| 130 | Fixed length, 3 bytes | Tag 9F33 | 0100 , 0200, 0220, 0420, 0620 | 0120 |
| 131 | Fixed length, 5 bytes | Tag 95 | 0200,0100, 0220, 0620, 0420 | 0422, 0220 |
| 132 | Fixed Length, 4 Bytes | Tag 9F37 | 0100 | 0120 |
| 134 , Format 1 | variable length<br>1-byte binary +<br>Up to 15 bytes of data; | Tag 9F10 | 0100, 0200, | 0120, 0422 |
| 134, Format 2 | variable length<br>1-byte binary +<br>Up to 32 bytes of data; | Tag 9F10 | 0100, 0400, 0200, 0420 | 0120 |
| 135 | variable length<br>1-byte binary length +<br>Up to 15 bytes of data; | Tag 9F10 Issuer Discretionary data | 0100, 0200 | 0120 |
| 136 | fixed length<br>16 hexadecimal digits; 8 bytes | Tag 9F26 | 0200,0220,0100 | 0120 |
| 137 | fixed length<br>4 hexadecimal digits; a 2-byte binary value | Tag 9F36 | 0100, 0200 | 0120,0422 |
| 138 | fixed length<br>16 hexadecimal digits; 8 bytes | Tag 82 | 0100,0200 | 0120, 0422, 0220 |
| 139 | fixed length<br>16 hexadecimal digits +<br>2 AN EBCDIC; | Tag 91 | 0110, 0210,0220 | |
| 143 | variable length<br>1-byte binary +<br>40 hexadecimal digits; Up to 20 bytes of data | Tag 9F5B | 0420,0620, 0400 | |
| 144 | fixed length<br>2N, 4-bit BCD (unsigned packed); 1 byte | Tag 9C | 0100, 0200, 0220 | 0422 |

`

| | | | | |
|---|---|---|---|---|
| 145 | fixed length 3N, 4-bit BCD; 2 bytes | Tag 9F1A | 0100, 0200, | |
| 146 | fixed length 6N, 4-bit BCD; 3 bytes | Tag 9A | 0100, 0200 | |
| 147 | fixed length 12N, 4-bit BCD (unsigned packed); 6 bytes | Tag 9F02 | 0100, 0200 | |
| 148 | fixed length 3N, 4-bit BCD; 2 bytes | Tag 5F2A | 0100,0200 | |
| 149 | fixed length 12N, 4-bit BCD (unsigned packed); 6 bytes | Tag 9F03 | 0100, 0200 | |
| 152 | fixed length 64 N, bit string; 8 bytes | Tag C0 | 0100,0200 (For PIN change transaction) | |

## 7.2. Mastercard

### 7.2.1. Field 22- PoS Entry Mode

| Attributes | | |
|---|---|---|
| Length | 3 | |
| Format | Numeric | |
| Position | Position 1-2 | Position 3 |
| Subfields | PAN entry mode | PIN Entry capability |
| Values | 05- Chip read at terminal | 1 - Terminal can accept PIN |
| | 80 – Fallback to magstripe | 2 – Terminal cannot accept PIN |
| | 95 – Unreliable CVV (Visa) | 8 – Terminal PIN pad down |
| | | 0 – Unknown Capability |

### 7.2.2. Field 23 – PAN Sequence Number

PAN sequence number is used to differentiate card when multiple card has same PAN. PAN sequence number is personalized in card in Tag 5F34 and travels in request message to issuer.
PAN sequence number is also required to derive UDK /MDK from Issuer/ ICC master key to validate ARQC. Issuer shall send PAN sequence number along with PAN to HSM to validate ARQC for a particular transaction.
If PAN sequence number is absent in request received by Issuer shall fill with zero and send to HSM.

| Attributes | |
|---|---|
| Length | 3 |
| Format | Numeric |
| Values | 000-099 |
| Presence | Mandatory for Chip transaction (DE22 = 05X) |

`

### 7.2.3. Field 55 – ICC Related Data

| Attributes | |
|---|---|
| Format | b..255 LLLVAR |
| Presence | Mandatory for chip transactions (DE 22 = 05X / 95X) |

Length (position 1-3) – This represents DE 55-total length. This comes in ASCII.

Sub element ID or Tag – First sub element ID, in binary representation; the length is either one or two positions depending on the definition of the sub element.

Sub element or Tag length - This represents the length of the tag value. This is 1 byte.

Sub element data – this represents the value of the tag. This is variable based on the defined tag length.

Tag – An EMV tag can be of one byte or two byte long. Whether the tag is of one byte or two bytes can be known from the last five bits of (bits 4 -8) of the first byte. If all the five bits are set to 1 then next byte is part of the tag, else not.

For example: -

Tag 9F06 – Here 9F in binary 10011111. So, bit 1 to 5 is set to 1. So, 06 is part of the tag. So full tag is 9F06

Tag 95 – Here 95 in binary 10010101. So, bit 4 to 8 is not set to 1. So, tag is 95.

Length- Length of the tag is represented in 1 byte or 2 bytes. Most EMV tag length are represented in 1 byte only.
When bit b8 of the most significant byte of the length field is set to 0, the length field consists of only one byte. Bits b7 to b1 code the number of bytes of the value field. The length field is within the range 1 to 127.
When bit b8 of the most significant byte of the length field is set to 1, the subsequent bits b7 to b1 of the most significant byte code the number of subsequent bytes in the length field. The subsequent bytes code an integer representing the number of bytes in the value field. Two bytes are necessary to express up to 255 bytes in the value field

Value- Value of the tag (chip card data) in hexadecimal format.

| Tag | Description | Length | MTI | Presence |
|---|---|---|---|---|
| 9F02 | Amount Authorized | Fixed length<br>12 N, 4-bit BCD (unsigned packed)<br>6 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F03 | Amount Other | Fixed length<br>12 N, 4-bit BCD (unsigned packed);<br>6 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 5F2A | Transaction Currency code | Fixed length<br>3N, 4-bit BCD,<br>2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 82 | Application Interchange profile | Fixed length<br>16 bit string; 2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ | M |

`

| | | | | |
|---|---|---|---|---|
| | | | 0422 | |
| 95 | Terminal Verification result | Fixed length<br>5 bytes; | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9A | Transaction Date | Fixed length<br>6N, 4 bit BCD; 3 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9C | Transaction Type | Fixed length<br>2N, 4 bit BCD<br>1 byte | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F10 | Issuer Application Data | Variable length<br>Length 1 byte binary +<br>Up to 32 bytes of value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F1A | Terminal country code | Fixed length<br>3N, 4 bit BCD; 2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F26 | Application Cryptogram | Fixed length<br>16 hexadecimals digits; 8 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F33 | Terminal Capability | Fixed length<br>1 byte binary length +<br>3 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M, Optional for US |
| 9F36 | Application Transaction Counter | Fixed length<br>4 hexadecimal digits,<br>2 byte binary value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F37 | Unpredictable Number | Fixed length<br>8 hexadecimal digits;<br>4 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F27 | Cryptogram Information Data | Fixed length<br>1 byte length +<br>1 byte of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F34 | CVM results | Fixed length<br>1 byte length +<br>3 bytes of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F35 | Terminal Type | Fixed length<br>1 byte length +<br>1 byte of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | O |
| 84 | Dedicated file name | Variable length<br>1 byte Binary(length) +<br>5 bytes to 16 bytes value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F1E | IFD serial number | Fixed length<br>1 byte length +<br>8 byte of Value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |
| 9F53 | Transaction category code | Fixed length<br>1 byte length +<br>1 byte of Value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |
| 9F09 | Application Version number | Fixed length<br>1 byte length +<br>2 bytes of Value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |

`

| | | | | |
|---|---|---|---|---|
| 9F41 | Transaction sequence counter | Variable length<br>1 byte Binary(length) + 2 bytes to 4 bytes value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |
| 91 | Issuer Authentication Data | Variable length<br>1 byte Binary(length) + 8 byes to 16 bytes value | 0110 /0210 / 0120 | M |
| 71 | Issuer Script Template 1 | Variable length<br>1 byte binary length + 510 hexadecimals digits, maximum 256 bytes | 0110 / 0210 | C |
| 72 | Issuer Script template 2 | Variable length<br>1 byte +<br>510 hexadecimals digits, maximum 256 bytes | 0110 / 0210 | C |
| 9F5B | Issuer script result | Variable length<br>1 byte binary (length) + 40 hexadecimals; Maximum 20 bytes of value | 0400/ 0420 | C |

**Note:**

1. The hexadecimal representation is given here. Every two positions of hexadecimal data are one byte of binary data.
2. Lengths are in binary format.
3. The Total Sub Element Length is the sum of the sub element's ID, length, and data subfields.
4. Issuer shall be able to process any other valid tags coming in request message other than tags mentioned in above table.

### 7.2.4. Issuer Authentication Data

Issuer authentication data is represented in Tag 91 and populated in DE55 in response by issuer. Issuer shall populate tag 91 when issuer authentication is performed by validating ARQC and generating ARPC. There are several ways to build tag 91 after receiving response from HSM with ARPC.

| Attributes | | |
|---|---|---|
| Length | 10 bytes | |
| Format | 20 hexadecimal digits ( 10 bytes binary) | |
| Position | Byte 1- 8 | Byte 9 -10 |
| Values | ARPC | ARPC Response code |

`

## 7.3. RuPay

### 7.3.1. Field 22- PoS Entry Mode

| Attributes | | |
|---|---|---|
| Length | 3 | |
| Format | Numeric | |
| Position | Position 1-2 | Position 3 |
| Subfields | PAN entry mode | PIN Entry capability |
| Values | 05- Chip read at terminal | 1 - Terminal can accept PIN |
| | 80 – Fallback to magstripe | 2 – Terminal cannot accept PIN |
| | 95 – Unreliable CVV (Visa) | 8 – Terminal PIN pad down |
| | | 0 – Unknown Capability |

### 7.3.2. Field 23 – PAN Sequence Number

PAN sequence number is used to differentiate card when multiple card has same PAN. PAN sequence number is personalized in card in Tag 5F34 and travels in request message to issuer.
PAN sequence number is also required to derive UDK or ICC r=derived key from ICC Master key to validate ARQC.
Issuer shall send PAN sequence number along with PAN to HSM to validate ARQC for a particular transaction.
If PAN sequence number is absent in request received by Issuer shall fill with zero and send to HSM.

| Attributes | |
|---|---|
| Length | 3 |
| Format | Numeric |
| Values | 000-099 |
| Presence | Mandatory for Chip transaction (DE22 = 05X / 95X) |

### 7.3.3. Field 55 – ICC Related Data

| Attributes | |
|---|---|
| Formats | b..255 LLLVAR |
| Presence | Mandatory for chip transactions (DE 22 = 05X / 95X) |

Length (position 1-3) – This represents DE 55 total length. This comes in ASCII.

Tag – First sub element ID, in binary representation; the length is either one or two positions depending on the definition of the sub element.

Tag length - This represents the length of the tag value. This is 1 byte.

Tag data – this represents the value of the tag. This is variable based on the defined tag length.

Tag – An EMV tag can be of one byte or two byte long. Whether the tag is of one byte or two bytes can be known from the last five bits of (bits 4 -8) of the first byte. If all the five bits are set to 1 then next byte is part of the tag, else not.

`

For example: -

Tag 9F06 – Here 9F in binary 10011111. So, bit 1 to 5 is set to 1. So, 06 is part of the tag. So full tag is 9F06

Tag 95 – Here 95 in binary 10010101. So, bit 4 to 8 is not set to 1. So, tag is 95.

Length- Length of the tag is represented in 1 byte or 2 bytes. Most EMV tag length are represented in 1 byte only.
When bit b8 of the most significant byte of the length field is set to 0, the length field consists of only one byte. Bits b7 to b1 code the number of bytes of the value field. The length field is within the range 1 to 127.
When bit b8 of the most significant byte of the length field is set to 1, the subsequent bits b7 to b1 of the most significant byte code the number of subsequent bytes in the length field. The subsequent bytes code an integer representing the number of bytes in the value field. Two bytes are necessary to express up to 255 bytes in the value field

Value- Value of the tag (chip card data) in hexadecimal format.

| Tag | Description | Length | MTI | Presence |
|---|---|---|---|---|
| 9F02 | Amount Authorized | Fixed length<br>12 N, 4 bit BCD<br>(unsigned packed)<br>6 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F03 | Amount Other | Fixed length<br>12 N, 4 bit BCD<br>(unsigned packed);<br>6 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 5F2A | Transaction Currency code | Fixed length<br>3N, 4 bit BCD,<br>2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 82 | Application Interchange profile | Fixed length<br>16 bit string; 2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 95 | Terminal Verification result | Fixed length<br>5 bytes; | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9A | Transaction Date | Fixed length<br>6N, 4 bit BCD; 3 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9C | Transaction Type | Fixed length<br>2N, 4 bit BCD<br>1 byte | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F10 | Issuer Application Data | Variable length<br>Length 1 byte binary +<br>Up to 32 bytes of value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F1A | Terminal country code | Fixed length<br>3N, 4 bit BCD; 2 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F26 | Application Cryptogram | Fixed length | 0100 / 0200 /0120/ | M |

`

| | | | | |
|---|---|---|---|---|
| | | 16 hexadecimals digits; 8 bytes | 0220/ 0400/ 0420/ 0422 | |
| 9F33 | Terminal Capability | Fixed length<br>1 byte binary length +<br>3 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M, Optional for US |
| 9F36 | Application Transaction Counter | Fixed length<br>4 hexadecimal digits,<br>2 byte binary value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F37 | Unpredictable Number | Fixed length<br>8 hexadecimal digits;<br>4 bytes | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F27 | Cryptogram Information Data | Fixed length<br>1 byte length +<br>1 byte of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | M |
| 9F34 | CVM results | Fixed length<br>1 byte length +<br>3 bytes of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | O |
| 9F35 | Terminal Type | Fixed length<br>1 byte length +<br>1 byte of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ 0422 | O |
| 84 | Dedicated file name | Variable length<br>1 byte Binary(length) +<br>5 bytes to 16 bytes value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ | O |
| 9F06 | AID Terminal | Variable length<br>1 byte Binary(length) +<br>5 bytes to 16 bytes value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ | O |
| 9F07 | Application Usage control | Fixed length<br>1 byte length +<br>2 bytes of Value | 0100 / 0200 /0120/ 0220/ 0400/ 0420/ | O |
| 9F1E | IFD serial number | Fixed length<br>1 byte length +<br>8 bytes of Value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |
| 9F53 | Transaction category code | Fixed length<br>1 byte length +<br>1 byte of Value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |
| 9F09 | Application Version number | Fixed length<br>1-byte length +<br>2 bytes of Value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |
| 9F41 | Transaction sequence counter | Variable length<br>1 byte Binary(length) +<br>2 bytes to 4 bytes value | 0100 / 0200/0120/ 0220/ 0400/ 0420/ 0422 | O |
| 91 | Issuer Authentication Data | Variable length<br>1 byte Binary(length) +<br>8 byes to 16 bytes value | 0110 /0210 / 0120 | M |
| 71 | Issuer Script Template 1 | Variable length<br>1 byte binary length + | 0110 / 0210 | C |

`

| | | | | |
|---|---|---|---|---|
| | | 510 hexadecimals digits, maximum 256 bytes | | |
| 72 | Issuer Script template 2 | Variable length<br>1 byte +<br>510 hexadecimals digits, maximum 256 bytes | 0110 / 0210 | C |
| 9F5B | Issuer script result | Variable length<br>1 byte binary (length) + 40 hexadecimals;<br>Maximum 20 bytes of value | 0400/<br>0420/0120/0220 | C |

**Note:**

1. The hexadecimal representation is given here. Every two positions of hexadecimal data are one byte of binary data.
2. Total length of DE 55 shall be in ASCII
3. Tag Length Value (Chip Data) coded in BCD.
4. The Total Sub Element Length is the sum of the sub element's ID, length, and data subfields.
5. Issuer shall be able to process any other valid tags coming in request message other than tags mentioned in above table.

### 7.3.4. Issuer Authentication Data

Issuer authentication data is represented in Tag 91 and populated in DE55 in response by issuer. Issuer shall populate tag 91 when issuer authentication is performed by validating ARQC and generating ARPC. There are several ways to build tag 91 after receiving response from HSM with ARPC.

| Attributes | | |
|---|---|---|
| Length | 10 bytes | |
| Format | 20 hexadecimal digits ( 10 bytes binary) | |
| Position | Byte 1- 8 | Byte 9 -10 |
| Values | ARPC | CSU / ARC |

# 8. Fallback Transaction

When a Chip Card is presented at an EMV Chip enabled terminal, the transaction should be completed as a Chip Card Transaction. If the transaction cannot be completed with Chip due to chip malfunction or reader malfunction or unknown AID, the transaction shall "fall back" to a magnetic stripe transaction and is submitted to the Issuer for authorization as a magnetic stripe transaction.

Fallback transactions shall always go online for authorization

- Fall back transactions shall be indicated in the Financial / Authorization request message by setting DE 22 (POS Entry Mode) to '80X'
- DE 55 and DE 23 shall be absent in request message for fallback transaction.

`

** Fallback may not be allowed in few countries. Issuers should contact the Payment System regional representative for the rules in market.

# 9. Key Management System

Key management system plays main role when it comes to perform security evaluation of EMV standard. EMV contact transaction security is managed through symmetric key management involving creation of multiple level of keys which are explained in subsequent sections.

## 9.1. Symmetric Key Management

This section describes the different keys that an Issuer shall generate and manage to process chip card transaction. To process chip card transaction there are various cryptogram like ARQC, ARPC & MAC are derived and validated using symmetric set of keys.
These keys can be securely managed through an HSM, but if issue wish to implement as software HSM they can do so by developing algorithms mentioned in subsequent sections.

## 9.2. Keys Categories

In EMV transaction scenario there are three broad categories of keys

- Application Cryptogram Keys (AC Key)
- MAC Keys or secure messaging for integrity (SMI) key
- Encryption keys or secure messaging for confidentiality (SMC) keys.

Each of the above key has a master key, derived key and session key.

Chip keys can be be divided into three categories, as follows:

- Issuer Master Keys; Length = 16 Bytes
    - Application Cryptogram Master Keys
    - MAC Master keys (SMI)
    - Encryption Master Keys (SMC)
- Derived Unique Keys / ICC Master keys; Length = 16 Bytes
    - ICC AC Master keys
    - ICC MAC Master keys
    - ICC Encryption Master Keys
- Session Keys; Length = 16 Bytes
    - AC Session keys
    - MAC session keys
    - Encryption session keys

Master Keys also referenced as Issuer Master Key (IMK)is property of issuer. Each master key shall have a

`

unique key index which shall be used to identify the key. This key index shall be personalized in the CHIP card & is sent to issuer during the authorization process as part of chip data in data element DE55 in Tag 9F10. Issuer Master Keys are generated by the Issuer are stored in host systems / HSMs for personalization and authorizations.
Each 16 Byte key has two 8 Bytes part and shall be referenced, throughout this document, first part (left most part) as Key A & second part (right most part) as Key B.

Issuer master keys used to derive unique keys, also referenced as ICC Derived Unique Master Keys (RuPay), Unique Derivation key (Visa) and Master Derivation Key (MasterCard), which are the keys personalized on the chip card

- ICC Application Cryptogram (AC) Master Key / UDK / MDK
- ICC MAC Master Key
- ICC Encryption Master Key

Usage of Keys

1. The key derived from Application Master Key is used to generate AC and ARPC & is referenced as ICC Application Master Key (ICCMAC)
2. The key derived from MAC Master Key is used for Secure Messaging for Integrity & is referenced as ICC MAC Master Key (ICCMSMI)
3. The key derived from Encryption Master Key is used for Secure Messaging for Confidentiality & is referenced as ICC ENC Master Key (ICCMSMC)

Using the ICC Derived Unique Master Keys, Session keys are generated during the transaction processing and are unique per transaction. Session keys are further categorized into three types

- Application Session Key
- MAC Session Key
- Encryption Session Key

## 9.2.1. Issuer Master Key Generation

Issuer master key can be generated in HSM or through any software implementation. It is completely Issuer discretion to use any master key and then encrypt the master key under LMK in HSM.
Sample HSM command is given below.

### 9.2.1.1. Application Cryptogram (AC) Master Key Activity

**Creation**
Online-AUTH>fk
Enter key length [1,2,3]: 2
Enter key type: 109
Enter key scheme: u
Enter component type [X,H,T,E,S]: x
Enter number of components [1-9]: 1
Enter component 1: 11111111111111111111111111111111 – Clear Application Cryptogram (AC) Master

`

Key
Encrypted key: output – Encrypted Application Cryptogram (AC) Master Key
Key check value: output

**Exporting**
Online-AUTH>key
Enter key type: 109
Enter key scheme: x
Enter ZMK: Issuer ZMK – Encrypted ZMK
Enter ZMK variant: 00
Enter key: Output of previous step - Encrypted Application Cryptogram (AC) Master Key
Key under ZMK: output – Exported Application Cryptogram (AC) Master Key under ZMK
Key check value: output

#### 9.2.1.2. MAC Master Key

**Creation**
Online-AUTH>fk
Enter key length [1,2,3]: 2
Enter key type: 209
Enter key scheme: u
Enter component type [X,H,T,E,S]: x
Enter number of components [1-9]: 1
Enter component 1: 33333333333333333333333333333333 – Clear MAC Master Key
Encrypted key: output – Encrypted MAC Master Key
Key check value: output

**Exporting**
Online-AUTH>key
Enter key type: 209
Enter key scheme: x
Enter ZMK: Issuer ZMK – Encrypted ZMK
Enter ZMK variant: 00
Enter key: Output of previous step – Encrypted MAC Master Key
Key under ZMK: Output – Exported MAC Master Key
Key check value: output

#### 9.2.1.3. Encryption Master Key

**Creation**
Online-AUTH>fk
Enter key length [1,2,3]: 2
Enter key type: 309Enter key scheme: u
Enter component type [X,H,T,E,S]: x
Enter number of components [1-9]: 1
Enter component 1: 55555555555555555555555555555555 – Clear Encryption Master Key
Encrypted key: output – Encrypted Encryption Master Key
Key check value: output

**Exporting**
Online-AUTH>key
`

Enter key type: 309
Enter key scheme: x
Enter ZMK: Issuer ZMK - Encrypted ZMK
Enter ZMK variant: 00
Enter key: output of previous step – Encrypted Encryption Master Key
Key under ZMK: output – Exported Encryption Master Key
Key check value: output

### 9.2.2. ICC Master Key Derivation

As per EMV 4.3 there are three options to derive ICC master keys from Issuer master key which are used for below functions to perform.

- Application Cryptogram generation,
- Issuer authentication,
- Secure messaging.

The first two methods use the 8-byte block cipher Triple DES in ECB mode and the third method uses the 16-byte block cipher AES in ECB mode. The AES method supports 128-bit, 192-bit and 256-bit keys.

To generate ICC master key from Issuer master key below inputs are required.

| Input | Source |
|---|---|
| PAN | DE 2 |
| PAN Sequence Number | DE 23 |
| Issuer Master Key | Stored in Issuer Host / HSM |

Option A and B methods are only applicable when the block cipher is Triple DES.

#### 9.2.2.1. Option A

This option is applicable in case PAN is equal to or less than 16 decimal digits.

1. Concatenate from left to right the decimal digits of the Application PAN with the PAN Sequence Number (if the PAN Sequence Number is not present, then it is replaced by a '00' byte).

2. If the result X is less than 16 digits long, pad it to the left with hexadecimal zeros in order to obtain an 8-byte number Y in numeric format.

3. If X is greater than 16 digits long, then Y consists of the 16 rightmost digits of X in numeric format.

4. Compute the two 8-byte numbers

    - ZL := DES3(IMK)[Y]

`

- ZR := DES3(IMK)[Y ⊕ ('FF'||'FF'||'FF'||'FF'||'FF'||'FF'||'FF'||'FF')]
- Z := (ZL || ZR)

The 128-bit ICC Master Key MK is then equal to Z, with the exception of the least significant bit of each byte of Z which is set to a value that ensures that each of the 16 bytes of MK has an odd number of nonzero bits (this to conform with the odd parity requirements for DES keys).

#### 9.2.2.2. Option B

This option is applicable in case PAN is greater than 16 decimal digits

1. Concatenate from left to right the decimal digits of the Application PAN and the PAN Sequence Number (if the PAN Sequence Number is not present, it is replaced by a '00' byte).

2. If the result is less than 16 digits long, pad it to the left with hexadecimal zeros in order to obtain an 8-byte number in numeric format.

3. Hash the result of the concatenation using the SHA-1 hashing algorithm to obtain the 20-byte hash result X.

4. Select the first 16 decimal digits (0 to 9) starting from the left side of the 20-byte (40-nibble) hash result Y and use as the value Z. If this does not provide for 16 decimal digits in Y, convert the non-decimal nibbles in Y to decimal digits by means of the following decimalization table:

| Input nibble of X | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| Decimalized nibble | 0 | 1 | 2 | 3 | 4 | 5 |

*Table 1: Decimalization for Master Key Derivation*

5. Add the converted digits starting from the left side of X to the end of Y until Y contains 16 digits.

**Example 1:** Hash result X contains 16 or more decimal digits

X = '12 30 AB CD 56 78 42 D4 B1 79 F2 CA 34 5D 67 89 A1 7B 64 BB'
Y = first 16 decimal digits of X i.e. '12 30 56 78 42 41 79 23'

**Example 2:** Hash result X contains less than 16 decimal digits

X = '1B 3C AB CD D6 E8 FA D4 B1 CD F2 CA D4 FD C7 8F A1 7B 6E BB'
Y = decimal digits from X = '13 68 41 24 78 17 6' plus the required number of converted digits '1 20' (from 'B', 'C', and 'A'), giving: Y = '13 68 41 24 78 17 61 20'

6. Continue with the processing specified for Option A starting at Step 4.

#### 9.2.2.3. Option C

This method is only applicable when the n-byte block cipher is AES.

1. Concatenate from left to right the decimal digits of the Application PAN with the PAN Sequence Number (if the PAN Sequence Number is not present, then it is replaced by a '00' byte). Pad it to

`

the left with hexadecimal zeros in order to obtain a 16-byte number Y in numeric format.

The 16-byte ICC Master Key MK is then equal to

- In the case $k = 8n$:

MK := AES(IMK)[Y]

- In the case 16n ≥ k > 8n:

MK := Leftmost *k*–bits of {AES(IMK)[Y] || AES(IMK)[Y*]}

where Y* = Y ⊕ ('FF'||'FF'||'FF'||'FF'||'FF'||'FF'||'FF'||'FF').

## 9.2.3. Session Key Derivation

There are essentially two methods for session key derivation as per EMV 4.X.

- The EMV CSK method.
- EMV 2000 Method

Other than EMV methods there are two scheme specific methods are also used for MasterCard and JCB.

- The MasterCard Proprietary SKD method
- XOR Session key method

### 9.2.3.1. EMV Common Session Key Method

The common session key derivation option generates a unique session key for each transaction performed by the application. It does this by enciphering an *n*-byte diversification value with the *k*-bit ICC Master Key (MK) to produce a *k*-bit ICC Session Key (SK) using an *n*-byte block cipher algorithm ALG in ECB mode.

The *n*-byte diversification value is represented as
R = R0 || R1 || R2 || … || Rn-1.

For the session key used to generate and verify the Application Cryptogram and the ARPC, the diversification value is the ATC followed by *n*-2 bytes of '00':

R: = ATC || '00' || '00' || … || '00' || '00' || '00'.

| Value | Source | Tag | Length |
|---|---|---|---|
| Application Transaction Counter (ATC) | DE 55 | 9F36’ | 2 |

For the session keys used for secure messaging, the diversification value R is the Application Cryptogram present in authorization request DE55 Tag 9F26 followed by n-8 bytes of '00':

R:= Application Cryptogram … || '00' || '00' || '00'.

| Value | Source | Tag | Length |
|---|---|---|---|
| Application Cryptogram (ARQC) | DE 55 | 9F26’ | 8 |

`

For an *n*-byte block cipher ALG using a *k*-bit key where *k* = 8*n* (AES with *k*=128) the derivation function F is computed as follows:

SK := ALG (MK) [ R ].

For an *n*-byte block cipher ALG using a *k*-bit key where 16*n* ≥ *k* > 8*n* (Triple DES with *k*=128 or AES with *k*=192 or 256) R is used to create two *n*-byte blocks as follows:

F1 = R0 || R1 || 'F0' || … || Rn-1.
F2 = R0 || R1 || '0F' || … || Rn-1.
and
SK := Leftmost *k*-bits of {ALG (MK) [F1] || ALG (MK) [F2] }.

ALG denotes Triple DES or AES.

MK denotes ICC Master key or UDK or MDK

The same session key is used for all commands in a single transaction.

Example:

For the session key used to generate and verify the cryptograms, TC, ARQC, AAC and ARPC, the diversification value is the ATC followed by 6 bytes of '00':

R := ATC || '00' || '00' || '00' || '00' || '00' || '00'

The application computes SK according to the following steps,

1. Compute SKL, the left hand 8 bytes of SK,

SKL := DES3 (MK) [ATC || 'F0' || '00' || '00' || '00' || '00' || '00']

2. Compute SKR, the right hand 8 bytes of SK,

SKR := DES3 (MK) [ATC || '0F' || '00' || '00' || '00' || '00' || '00']

3. SK is the concatenation of SKL and SKR.

SK := SKL || SKR

#### 9.2.3.2. Mastercard Proprietary SKD method

The MasterCard Proprietary SKD method is a special algorithm for deriving session keys for computing Application Cryptograms (ARQC, TC, AAC) and secure messaging cryptograms.

This algorithm follows EMV Common Session key method but with different diversification value R

**For AC session key derivation:**

R:= (ATC || ‘00’ || ‘00’ || UN)

`

SKAC: = SKD(MKAC)[(ATC || ‘00’ || ‘00’ || UN)].

| Value | Source | Tag | Length |
|---|---|---|---|
| Application Transaction Counter (ATC) | DE 55 | 9F36’ | 2 |
| Unpredictable Number (UN) | DE 55 | 9F37 | 4 |

**For Secure Messaging a new session key is derived for each script within a session**:

Secure Messaging session keys SKSMI and SKSMC are derived using an eight-byte number R filled with a new value (RAND) for each script command.

For the first script command processed in the transaction, RAND or the diversification value is the ARQC Received in authorization message De 55 Tag 9F26. For subsequent script commands, RAND is obtained by incrementing the RAND used for the previous command by 1.
Specifically, RAND, as a multi-byte counter, is incremented in the same way as the two-byte ATC. Therefore, incrementing RAND when its least significant byte is 0xFF results in the next least significant byte being incremented and the least significant byte being set to 0x00. In the unlikely event that all the bytes of RAND equal 0xFF, the RAND is incremented by setting all its bytes to 0x00
(i.e. the RAND counter wraps around).

R = RAND

SKSMI: = SKD(MKSMI)[R]

SKSMC: = SKD(MKSMC)[R].

RAND is same for SMI and SMC MAC computation.

### 9.2.3.3. EMV 2000 SKD for MasterCard

The algorithm for deriving session keys (SKAC, SKSMI, SKSMC) used for computing application cryptograms, validating scripts received with secure messaging, and for verifying ARPCs is defined below.

**Structure of the Session Key**

This method uses as derivation data the two-byte Application Transaction Counter (ATC) of the ICC and the Session Key Derivation (EMV2000) function defined in EMV2000:
SK := SKDEMV2000(MK)[(ATC||‘00’||‘00’||‘00’|| ‘00’||‘00’||‘00’)]
Initializing Value
The initializing value IV of the EMV2000 session key derivation is a 16-byte
string of binary zeros:
IV := (‘00’||‘00’||‘00’||‘00’||‘00’||‘00’||‘00’||‘00’||‘00’|| ‘00’||‘00’||‘00’||‘00’||‘00’||‘00’||‘00’)

**Data Elements**

| Value | Source | Tag | Length |
|---|---|---|---|
| Application Transaction Counter (ATC) | DE 55 | 9F36’ | 2 |

**First Session Key Derivation**

`

Depending on the card personalization, the very first session key derivation could have no preceding session key. In this case, this first session key should be derived by the card in the same way as the issuer derives session keys (i.e., from the appropriate ICC Master key and IV). To compute subsequent session keys, the ICC uses the key token (GP, P, $ATC_{SK}$). The value of the new key token must be stored in secure memory. Specifically:

GP := IKH-2, $ATC_{SK}$/b2
P := IKH-1, $ATC_{SK}$/b

where "/" denotes integer division, H is the height of the tree and b is the branch factor. For M/Chip implementation, the height of the tree H is 8 and the branch factor b is 4.

**Other Session Key Derivations**

Depending on the value of the ATC compared to the ATCSK stored on the ICC, the following cases are possible:

• If ATC > $ATC_{SK}$ then the session key can be derived as described in Section 9.2.3.3. When this is done, the values of GP and P stored in the ICC must be updated and $ATC_{SK}$ must be set to ATC.

• If ATC $_{SK}$ = $ATC_{SK}$ then the session key has already been derived and does not require to be derived again.

• If ATC < $ATC_{SK}$ then session key derivation must fail.

**Secure Messaging Session Key**

For Secure Messaging a new session key is derived at the beginning of a new session and is used for all the script commands of the same session provided the session key derivation method is not changed during the session.

The session key is computed as follows:

$SK_{SMI}$ := $SKD_{EMV2000}(MK_{SMI})$ [(ATC || '00' || '00' || '00' || '00' || '00' || '00')]

$SK_{SMC}$ := $SKD_{EMV2000}(MK_{SMC})$ [(ATC || '00' || '00' || '00' || '00' || '00' || '00')]

### 9.2.3.4. XOR Session Key Derivation Method

This section defines the session key generation method for JCB

Derivation of first left most part of session key i.e. key A shall be performed as described below:

- Create an 8-byte data by concatenating 6 bytes of ‘00’s and the latest ATC (2 bytes) in that order.

- Session key A is retrieved with XORing the data created in (1) with the 8-byte unique key A. Here, unique key A refers to the ICC Derived Unique Master Key A (i.e. either Application Cryptogram or MAC or Encryption/Decryption key, depending on the intended use)

Derivation of session key B shall be performed as described below:

- Create an 8-byte data by concatenating 6 bytes of ‘00’ and the inverted data of the latest ATC (2 bytes) in that order. Inversion is performed at the bit level, where each bit with ‘1’ is set to ‘0’ and each bit with value ‘0’ is set to ‘1’

- Session key B is retrieved with performing XOR to the data created in (1) with the 8-byte unique key B. Here, unique key B refers to the Application Unique Key B, the MAC Unique Key B or the

`

Message encryption Unique Key B, depending on the intended use

The same session key, where ever applicable, is used for all commands in a single transaction.

# 10. Crypto Generation and Validation

Cryptography has been used to provide data confidentiality and includes additional functions such as data integrity, authentication.

Cryptography can be categorised in two categories:

- Authorisation Request Cryptogram (ARQC)
- Authorisation Response Cryptogram (ARPC)

Card and Issuer shall be authenticating each other when the online transaction takes place.

- **ARQC** : This is a cryptogram which is generated by the ICC and sent to issuer

This shall be used by issuer to authenticate the ICC

- **ARPC** : This is a cryptogram which is generated by the issuer and sent to ICC.

This shall be used by ICC to authenticate the issuer.

| Crypto | Message Type | Tag | Generated By | Validated By |
|---|---|---|---|---|
| ARQC | Request MSG (0100/0200) | 9F26 | ICC | Issuer HSM |
| ARPC | Response MSG (0110/0210) | 91 | Issuer HSM | ICC |

## 10.1. Overview of Authorisation Request Cryptogram (ARQC)

During the execution of an ICC Transaction, the Chip Card may request the Terminal to forward an authorization request to the card Issuer.
The Issuer shall be able to authenticate the card by validating the ARQC received in request message.
If the verification of the ARQC is successful, the Issuer host confirms that the ICC used for the transaction is genuine else issuer shall decline the transaction.

The recommended minimum set of data elements to be included to generate the Application request Cryptogram, are specified below:

| Data Element | TAG Name | Length in Bytes | Source |
|---|---|---|---|
| **Amount, Authorised (Numeric)** | 9F02 | 6 | DE 55 |
| Amount, Other (Numeric) | 9F03 | 6 | |
| Terminal Country Code | 9F1A | 2 | |
| Terminal Verification Results | 95 | 5 | |
| Transaction Currency Code | 5F2A | 2 | |
| Transaction Date | 9A | 3 | |
| Transaction Type | 9C | 1 | |
| Unpredictable Number | 9F37 | 4 | |
| Application Interchange Profile | 82 | 2 | |
| Application Transaction Counter | 9F36 | 2 | |

`

Table 2: Minimum Set of Data Elements for Application Cryptogram Generation

### 10.1.1. ARQC Algorithm

#### 10.1.1.1. Standard EMV method

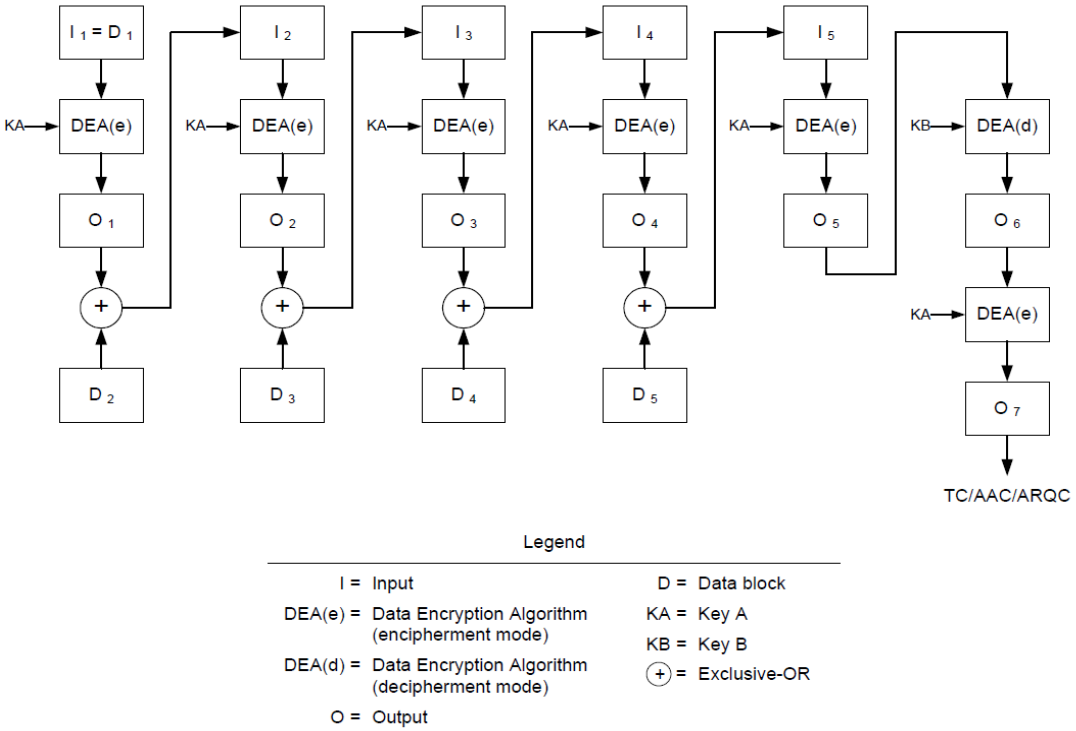



Figure 2 : Standard EMV

- Concatenate the data as per the sequence mentioned in Table 2, to get a data block D.
- Pad the data block D by adding first byte as '80' or ‘00’ byte (as per payment scheme) to the right of D, and then adding the least number of '00' bytes to the right such that the length of resulting data block is a multiple of 8 bytes.
- Divide the result of padding D, into 8-byte blocks D1, D2, . . . , Dn
- Use the 16-byte Application Cryptogram Session Key / Unique Key depending upon the Cryptogram Version Number in use as per respective payment scheme card specification and described in subsequent section.
- Divide the key in two 8-byte parts, called left and right.
- Using these two set of keys (Key A & Key B) generate the 8-byte cryptogram with the data blocks (D1, D2, . . . , Dn) applying data encryption algorithm. There are two types of data encryption algorithm called 3DES and AES.

Note: The algorithm is chosen based on the Cryptogram Version Number in use as per respective payment scheme card specification and described in subsequent section.

`

### 10.1.1.2. Common Core Definition Method

EMV common core definition method takes below data extra appended at the end of Table 2.

| Data Element | TAG Name | Source |
|---|---|---|
| **Issuer Application Data** | 9F10 | DE 55 |

**Preparation of data block.**

- Concatenate the data as per the sequence mentioned in Table 2, to get a data block D.
- Pad the data block D by adding first byte as '80' byte (as per payment scheme) to the right of D, and then adding the least number of '00' bytes to the right such that the length of resulting data block is a multiple of 8 bytes.
- Divide the result of padding D, into 8-byte blocks D1, D2, . . . , Dn
- Derive MAC session key as per method mentioned in section 9.2.3.1 for EMV Common session key derivation.
- Compute the ARQC following algorithm as per section 10.1.1.1

## 10.1.2. Visa

The variation in input data over and above Table 2 and types of DEA used for generating the application cryptograms are dependent on Cryptogram Version Number.
Visa currently supports the following CVNs for generating an Application Cryptogram generation.

| Supported CVN | Defined by |
|---|---|
| CVN 10 | Visa specification |
| CVN 18 | Visa specification |
| CVN 12 & 50 - 59 | Issuer proprietary |

*Table 3 : Supported CVN's*

### 10.1.2.1. ARQC for CVN 10 (HEX = 0A)

The data set needed for to generation of an Application Cryptogram for CVN 10 is the sequence in which they need to be arranged, is provided in Table 2.

Along with the tags mentioned in Table 2, Visa additionally uses CVR for CVN 10 to compute ARQC which will be added at the last of sequence mentioned in Table 2.

| Data Element | TAG Name | Source |
|---|---|---|
| **Card Verification Results** | 9F10 (Bytes 4–7) | DE 55 |

*Table 4: Additional data for ARQC with CVN 10*

To compute ARQC for CVN 10

- The algorithm mentioned in Section 10.1.1 shall be followed to compute ARQC.
- Pad the data block D by adding '00' bytes to the right such that the length of resulting data block is a multiple of 8 bytes.
- Use triple DEA encipherment with Unique DEA key / ICC master key in the algorithm mentioned in section 10.1.1 to compute ARQC.

`

#### 10.1.2.2. ARQC for CVN 18 (HEX = 12)

The data set needed for generation of an Application Cryptogram using CVN 18 is the sequence in which they need to be arranged, is provided in Table 2.

Along with the tags mentioned in Table 2, Visa additionally uses tag 9F10 for CVN 18 to compute ARQC which will be added at the last of sequence mentioned in Table 2.

| Data Element | TAG Name | Source |
|---|---|---|
| **Issuer Application Data** | 9F10 | DE 55 |

*Table 3 : Additional data for ARQC with CVN 18*

To compute ARQC for CVN 18

- The algorithm mentioned in Section 10.1.1.2 shall be followed to compute ARQC.
- For CVN 18, the data is padded with a mandatory '80' byte and the remaining right- most bits in the last data block are zero filled.
- Use triple DEA encipherment with MAC session key in the algorithm mentioned in section 10.1.1.2 to compute ARQC.

### 10.1.3. Mastercard

ARQC generation algorithm for Mastercard is differentiated with combination of input data and session key derivation method. There are 3 session key derivation method mentioned in section 9.2.3.
The combination of input data and session key derivation method is indicated by below CVNs.

| Supported CVN | Meaning |
|---|---|
| CVN 12 | Specified by Mastercard |
| CVN 13 | Specified by Mastercard |
| CVN 10 | Specified by Mastercard |
| CVN 11 | Specified by Mastercard |
| CVN 14 | Specified by Mastercard |
| CVN 15 | Specified by Mastercard |

*Figure 4 : List of Mastercard CVNs*

Please refer section 9.2.3 for respective session key derivation method.

#### 10.1.3.1. ARQC for CVN 10, CVN 12 and CVN 14 (Without counters)

The data set needed for generation of an Application Cryptogram using CVN 10, CVN 12 and CVN 14 is the sequence in which they need to be arranged, is provided in Table 2

Along with the tags mentioned in Table 2, Mastercard additionally uses tag 9F10 (Byte 3 -8) for CVN 10 , CVN 12 and CVN 14 to compute ARQC which will be appended at the last of sequence mentioned in Table 2.

| Data Element | TAG Name | Source |
|---|---|---|
| **Issuer Application Data (CVR)** | 9F10 (Byte 3- 8) | DE 55 |

*Table 5 : Additional data for ARQC with CVN 10, 12 & 14*

`

There are 3 methods for session key derivation for Mastercard to be used in computation of ARQC. The methods are differentiated by CVN.

| Supported CVN | SKD |
|---|---|
| CVN 10 | MasterCard Proprietary SKD |
| CVN 12 | EMV 2000 SKD |
| CVN 14 | EMV CSK |

To compute ARQC for CVN 10, 12 and 14:

- The algorithm mentioned in Section 10.1.1 shall be followed to compute ARQC.
- The input data is padded with a mandatory '80' byte and the remaining right- most bits in the last data block are zero filled to make the string as multiple of 8 bytes.
- Use triple DEA encipherment with session key in the algorithm mentioned in section 10.1.1.1 to compute ARQC.

### 10.1.3.2. ARQC for CVN 11, CVN 13 and CVN 15 (With counters)

The data set needed for generation of an Application Cryptogram using CVN 11, CVN 13 and CVN 15 is the sequence in which they need to be arranged, is provided in Table 2.

Along with the tags mentioned in Table 2, Mastercard additionally uses tag 9F10 Byte 3 -8 and Byte 11 -18 for CVN 13 to compute ARQC which will be added at the last of sequence mentioned in Table 2.

| **Data Element** | **TAG Name** | **Source** |
|---|---|---|
| **Issuer Application Data (CVR +Counters)** | 9F10 (Byte 3- 8) + (Byte 11 – 18, If available) | DE 55 |

*Table 6 : Additional data for ARQC with CVN 11,13, 15*

There are 3 methods for session key derivation for Mastercard to be used in computation of ARQC. The methods are differentiated by CVN.

| Supported CVN | SKD |
|---|---|
| CVN 11 | Mastercard Proprietary SKD |
| CVN 13 | EMV 2000 SKD |
| CVN 15 | EMV CSK |

To compute ARQC for CVN 11, 13 and 15

- The algorithm mentioned in Section 10.1.1 shall be followed to compute ARQC.
- The input data is padded with a mandatory '80' byte and the remaining right- most bits in the last data block are zero filled to make the data string as multiple of 8 bytes.
- Use triple DEA encipherment with session key derived as per above table in the algorithm mentioned in section 10.1.1.1 to compute ARQC.

`

### 10.1.4. RuPay

The variation in input data over and above Table 2 and types of DEA used for generating the application cryptograms are dependent on Cryptogram Version Number.
RuPay currently supports the following CVNs for generating an Application Cryptogram generation.

| Supported CVN | Meaning |
|---|---|
| CVN 1 | Defined by JCB |
| CVN 2 | Defined by JCB |
| CVN 5 | Defined by DPAS |
| CVN 6 | Defined by DPAS |

The minimum set of data elements that shall be included in the calculation of the ARQC, & the sequence in which they need to be arranged, are provided in.

#### 10.1.4.1. ARQC for CVN 5

The data set needed for generation of an Application Cryptogram using CVN 5 is the sequence in which they need to be arranged, is provided in Table 2.
Along with the tags mentioned in Table 2, RuPay additionally uses Tag 9F10 (Issuer Application Data) for CVN 5 to compute ARQC which will be added at the last of sequence mentioned in Table 2.

| **Data Element** | **TAG Name** | **Source** |
|---|---|---|
| **Issuer Application Data** | 9F10 | DE 55 |

*Table 5: Additional data for ARQC with CVN 5*

To compute ARQC for CVN 5

- The algorithm mentioned in Section 10.1.1 shall be followed to compute ARQC.
- Pad the data block D by adding '80' bytes to the right such that the length of resulting data block is a multiple of 8 bytes.
- Use triple DEA encipherment with session key derived from EMV Common session key derivation method (Section 9.2.3.1) in the algorithm mentioned in section 10.1.1 to compute ARQC.

#### 10.1.4.2. ARQC for CVN 6

The data set needed for generation of an Application Cryptogram using CVN 6 is the sequence in which they need to be arranged, is provided in Table 2.

Along with the tags mentioned in Table 2, RuPay additionally uses Tag 9F10 (Issuer Application Data) for CVN 6 to compute ARQC which will be added at the last of sequence mentioned in Table 2.

| **Data Element** | **TAG Name** | **Source** |
|---|---|---|
| **CVR** | 9F10 (Byte 3 – Byte 8) | DE 55 |

*Table 6: Additional data for ARQC with CVN 6*

To compute ARQC for CVN 6

- The algorithm mentioned in Section 10.1.1 shall be followed to compute ARQC.

`

- Pad the data block D by adding '80' bytes to the right such that the length of resulting data block is a multiple of 8 bytes.
- Use triple DEA encipherment with session key derived from EMV Common session key derivation method (Section 9.2.3.1) in the algorithm mentioned in section 10.1.1 to compute ARQC.

### 10.1.4.3. ARQC for CVN 01

The data set needed for generation of an Application Cryptogram using CVN 01 is the sequence in which they need to be arranged, is provided in Table 2.

Along with the tags mentioned in Table 2, RuPay additionally uses Legacy CVR for CVN 01 to compute ARQC which will be added at the last of sequence mentioned in Table 2.

| Data Element | TAG Name | Source |
|---|---|---|
| **CVR** | 9F10 (Byte 4- Byte 7) | DE 55 |

*Table 7: Additional data for ARQC with CVN 01*

To compute ARQC for CVN 01

- The algorithm mentioned in Section 10.1.1 shall be followed to compute ARQC.
- Pad the data block D by adding '00' bytes to the right such that the length of resulting data block is a multiple of 8 bytes.
- Use triple DEA encipherment with Application Unique key / ICC master key in the algorithm mentioned in section 10.1.1 to compute ARQC.

### 10.1.4.4. ARQC for CVN 02

The data input to generation of an Application Cryptogram using CVN 02 is the sequence in which they need to be arranged, is provided in Table 2.

Along with the tags mentioned in Table 2, RuPay additionally uses Legacy CVR for CVN 02 to compute ARQC which will be added at the last of sequence mentioned in Table 2.

| Data Element | TAG Name | Source |
|---|---|---|
| **CVR** | 9F10 (Byte 4- Byte 7) | DE 55 |

Table 8: Additional data for ARQC with CVN 02

To compute ARQC for CVN 02

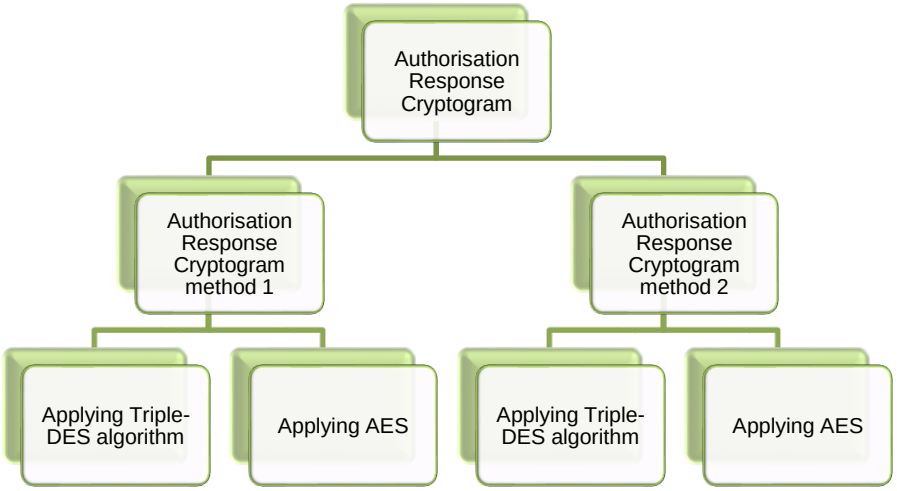
- The algorithm mentioned in Section 10.1.1 shall be followed to compute ARQC.
- Pad the data block D by adding '00' bytes to the right such that the length of resulting data block is a multiple of 8 bytes.
- Use triple DEA encipherment with JCB Session key derived as per section 9.2.3.4 in the algorithm mentioned in section 10.1.1 to compute ARQC.

`

## 10.2. Overview of Authorisation Response Cryptogram (ARPC)

A cryptogram used for a process called Online Issuer Authentication. It is sent to the card in the authorization response. The card validates the ARPC to ensure that it is communicating with the valid issuer.

Two methods are supported for generation of the ARPC used for issuer authentication as per EMV 4.3. Each method can use either Triple DES or AES.




### 10.2.1. ARPC Method 1

In this method generation of an 8-byte ARPC can be achieved by applying the Triple-DES algorithm or AES. The input data required for generation of ARPC are below

| Data | Length |
|---|---|
| ARQC | 8 bytes |
| Authorisation Response Code (ARC) | 2 bytes |

*Table 9: Input Data for ARPC Method 1*

#### 10.2.1.1. Applying the Triple-DES algorithm

Follow the below step

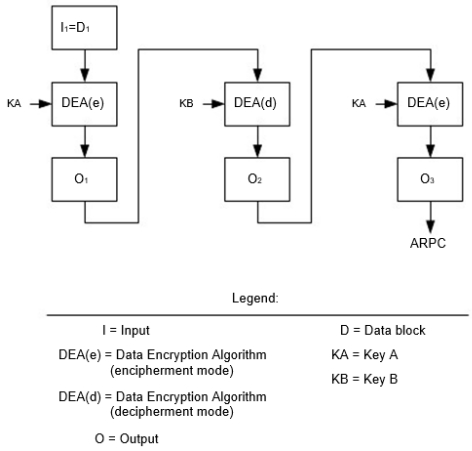
1. Pad the 2-byte ARC with six zero bytes to obtain the 8-byte number
    a. X := (ARC || '00'|| '00' || '00' || '00' || '00' || '00')

2. Compute Y := ARQC ⊕ X.

3. The 8-byte ARPC is then obtained by

ARPC := DES3(SKAC)[Y]

`




SKAC := Session key generated as per CVN.

4. Issuer Authentication Data: = 8 Byte ARPC || 2-byte ARC

### 10.2.1.2. Applying AES

This is identical to the ARPC Method 1 using Triple DES (above) except that in step 3 the 8-byte ARPC is computed using AES and is defined to be the leftmost 8 bytes of:

AES(SKAC)[ Y || Y0 ]

Where

Y0 := ('00' || '00' || '00' || '00' || '00' || '00' || '00' || '00').

## 10.2.2. ARPC Method 2

In this method generation of a 4-byte ARPC can be achieved by applying the Triple-DES algorithm or AES. The input data required for generation of ARPC are below

| Data | Length |
|---|---|
| ARQC | 8 bytes |
| Card Status Update (CSU) | 4 bytes |
| Proprietary Issuer authentication Data | 0-8 byte |

*Table 10: Input data for ARPC Method 2*

The below CSU can be coded by issuer based on validation performed.

`

**CSU Byte 1**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | Proprietary Authentication Data Included |
| | 0 | 0 | 0 | | | | | RFU |
| | | | | x | x | x | x | PIN Try Counter |

**CSU Byte 2**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | Issuer Approves Online Transaction |
| | 1 | | | | | | | Card Block |
| | | 1 | | | | | | Application Block |
| | | | 1 | | | | | Update PIN Try Counter |
| | | | | 1 | | | | Set Go Online on Next Transaction |
| | | | | | 1 | | | CSU Created by Proxy for the Issuer |
| | | | | | | x | x | Update Counters |
| | | | | | | 0 | 0 | Do Not Update Offline Counters |
| | | | | | | 0 | 1 | Set Offline Counters to Upper Offline Limits |
| | | | | | | 1 | 0 | Reset Offline Counters to Zero |
| | | | | | | 1 | 1 | Add Transaction to Offline Counter |

**CSU Byte 3**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RFU |

**CSU Byte 4**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | x | x | x | Issuer-Discretionary |

### 10.2.2.1. Applying the Triple-DES algorithm

Using the Application Cryptogram Session Key SKAC ( Table 2) in the following way:

1. Concatenate the ARQC, the CSU, and the Proprietary Authentication Data.

Y := ARQC || CSU || Proprietary Authentication Data

If issuer proprietary data is not available it shall be considered as 8 byte zero.

`

2. Generate a MAC over the data Y by applying the MAC algorithm specified in 10.1.1.2, to the data defined above using the Application Cryptogram Session Key derived when computing the ARQC.

3. ARPC := MAC := MAC algorithm (SKAC)[Y]

4. The Issuer Authentication Data (tag '91') is formed by concatenating the resulting 4-byte ARPC, the 4-byte CSU, and the Proprietary Authentication Data.

5. Issuer Authentication Data: = ARPC || CSU || Proprietary Authentication Data

## 10.2.3. VISA

### 10.2.3.1. ARPC for CVN 10

Visa uses ARPC method 1 described for CVN 10 mentioned in section 10.2.1.1 with triple DEA encipherment using unique DEA key / ICC master key.

The issuer authentication data is constructed as below:

Issuer Authentication Data: = 8 Byte ARPC || 2-byte ARC

### 10.2.3.2. ARPC for CVN 18

- Visa uses ARPC method 2 described for CVN 18 mentioned in section 10.2.2.1 with triple DEA encipherment using unique DEA key / ICC master key.

- The default value for Proprietary Authentication Data is zero.

- If the ‘Proprietary Authentication Data Included’ bit in the CSU has the value 0b, then the length of Proprietary Authentication Data included in generation and validation of the ARPC shall be 0 bytes.

- If the ‘Proprietary Authentication Data Included’ bit in the CSU has the value 1b, then the Proprietary Authentication Data included in the Issuer Authentication Data shall be used in generation and validation of the ARPC.

- Use session key generated during ATQC validation for CVN 18

- The issuer authentication data is constructed as below:

- Issuer Authentication Data: = 4 bytes ARPC || 4 bytes CSU || 1- 8 bytes Proprietary Authentication Data

`

### 10.2.4. Mastercard

- MasterCard uses ARPC Method 1 mentioned in section 10.2.2 for all CVNs.

- Mastercard Uses 2-byte ARPC response code instead of ARC. The ARPC response code is encoded as per below bits.

**Byte 1**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | x | | | | | Reserved |
| 0 | 0 | 0 | 0 | | | | | Other value RFU |
| | | | | x | x | x | x | *PIN Try Counter* |

**Byte 2**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | | | | | | Reserved |
| 0 | 0 | 0 | | | | | | Other value RFU |
| | | | x | | | | | Approve online transaction |
| | | | 0 | | | | | Do not approve online transaction |
| | | | 1 | | | | | Approve online transaction |
| | | | | x | | | | Update *PIN Try Counter* |
| | | | | 0 | | | | Do not update *PIN Try Counter* |
| | | | | 1 | | | | Update *PIN Try Counter* |
| | | | | | x | | | Set go online on next transaction |
| | | | | | 0 | | | Reset go online on next transaction |
| | | | | | 1 | | | Set go online on next transaction |
| | | | | | | x | x | Update counters |
| | | | | | | 0 | 0 | Do not update offline counters |
| | | | | | | 1 | 0 | Reset counters to zero |
| | | | | | | 0 | 1 | Set counters to upper offline limits |
| | | | | | | 1 | 1 | Add transaction to counter |

- The Session key generated during ARQC generation as per CVN mentioned in section 10.1.3 shall be used for triple DES encipherment.

- The issuer authentication data is constructed as below:

Issuer Authentication Data: = 8 Byte ARPC || 2-byte ARPC Response Code

`

### 10.2.5. RuPay

RuPay uses ARPC Method 1 mentioned in section 10.2.2 for all CVNs.

#### 10.2.5.1. ARPC for CVN 5 & 6

- RuPay uses 2-byte CSU coded as per below bits instead of ARC for CVN 5 & 6.

| ***Byte 1*** | ***Value*** | ***Description*** |
|---|---|---|
| b8 | 0 | RFU |
| b7 | 0 | |
| b6 | 0 | |
| b5 | 0 | |
| b4 | X | PIN Try Counter |
| b3 | X | |
| b2 | X | |
| b1 | X | |
| ***Byte 2*** | ***Value*** | ***Description*** |
| b8 | 0 | RFU |
| b7 | 0 | |
| b6 | 1 | DO NOT Reset Script Counter |
| b5 | 1 | Issuer Approves Online Transaction |
| b4 | 1 | Update PIN Try Counter. If the PTC value in CSU B1b4-b1 > PTL then PTC is updated with PTL value |
| b3 | 1 | Set Go Online on next transaction |
| b2 b1 | X | Update Counters |

| **X** | **X** | |
|---|---|---|
| **0** | **0** | Do Not Update Offline Counters |
| **0** | **1** | Set Offline Counters to Upper Limits |
| **1** | **0** | Reset Offline Counters to Zero |
| **1** | **1** | Reset Offline Counters in all profiles to Zero |

- Use application session key derived as per EMV common session key derivation method mentioned in section 9.2.3.1.

`

- The issuer authentication data is constructed as below:

  Issuer Authentication Data: = 8 Byte ARPC || 2-byte CSU

#### 10.2.5.2. ARPC for CVN 1 & 2

- For CVN 1 use ARPC method 1 with ICC master key / Unique DEA key.
- For CVN 2 use ARPC method 1 with JCB session key derived as mentioned in section 9.2.3.4.
-
- The issuer authentication data is constructed as below:

  Issuer Authentication Data: = 8 Byte ARPC || 2-byte ARC

## 10.3. HSM Commands Interaction

Issuer can manage all ARQC validation and ARPC generation through an HSM. An HSM aka Host Security Module is a hardware which comes with inbuilt key management and cryptogram generation facility with utmost security in place.

Issuer using HSM shall send the transaction data as per HSM format after required data padding as per payment scheme CVN for ARQC validation and ARPC generation.
In this section all the required commands for ARQC validation and ARPC generation are described as per payment scheme requirement.

It is considered that Issuer is using Thales HSM

### 10.3.1. ARQC validation & ARPC generation for EMV 4.X

The validation of ARQC and generation of ARPC involves various algorithms for derivations of keys and crypto methodologies. These different methods are communicated to HSM through some designated command along with required input data.
The command structures for every CVN is described in subsequent section.

**Format notations**

| Notation | Description |
|---|---|
| L | Encrypted PIN length. Set at installation. |
| m | Message header length. Set at installation. |
| n | Variable length field. |
| A | Alphanumeric (can include any non-control type) characters. |
| H | Hexadecimal character ('0'...'9', 'A'...'F'). |
| N | Numeric Field ('0'...'9'). |
| C | Control character. |
| B | Binary data (byte) (X'00...X'FF). |
| D | Binary coded decimal (BCD) character ('0'...'9'). |

`

### 10.3.2. Mastercard

#### 10.3.2.1. Command Message for Mastercard CVN (14,15,12,13)

**Request Command**

| Field | Length & Type | Details | |
|---|---|---|---|
| Message header | m A | Message header length | |
| Command code | 2 A | **KW** | |
| Mode Flag | 1 H | '1': Perform ARQC verification and EMV 4.x Method 1 ARPC generation | |
| Scheme ID | 1 N | CVN 14 & 15 | CVN 12 & 13 |
| | | 2' : Option A ICC Master Key Derivation and EMV Common Session Key Derivation | '0' : Option A ICC Master Key Derivation and EMV2000 Session Key Derivation |
| MK-AC | 32 H or 1 A + 32 H or 1 A + n A | For a Variant LMK, the 'MK-AC' must be encrypted under LMK pair 28-29 variant 1. | |
| IV- AC | 16B | Only present for Scheme ID = '0'. 16 bytes zero ''00''<br>IV for EMV 2000 Session Key Derivation | |
| PAN length | 2N | NA | |
| PAN+PAN Sequence number | 8 B or n B | For Scheme ID = '0' or '2', this field will be fixed at 8 bytes, and should contain the pre-formatted PAN/PAN Sequence No<br>It is the responsibility of the host system to ensure that the PAN/PAN Sequence Number is appropriately padded. | |
| Delimiter | 1A | NA | |
| Branch / Height Parameters | 1 N | Only present for Scheme ID = '0'<br>'0' : Branch factor 2; Tree Height 16 | |
| Application transaction counter | 2 B | Received from DE55.tag 9F36 | |
| Transaction data length | 2 H | Present | |
| Transaction data | n B | Present with 80 padding | |
| Delimiter | 1 A | Present to indicate end of transaction data | |
| ARQC/TC/AAC | 8 B | ARQC received in DE 55.Tag 9F26 | |
| ARC | 2 B | ARPC response code | |
| CSU | 4 B | NA | |
| Proprietary authentication data length | 1 N | NA | |
| Proprietary authentication data | 0 ... 8 B | NA | |
| Delimiter | 1 A | Value '%'. Optional; if present, the following field must be present. | |
| LMK identifier | 2 N | LMK identifier; min value = '00'; max value is defined by issuer; must be present if the above Delimiter is present | |
| End message delimiter | 1 C | Must be present if a message trailer is present. | |

`

| | | |
|---|---|---|
| | | Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

**Response Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Response code | 2 A | **KX** |
| Error Code | 2 A | '00' : No error<br>'01' : ARQC/TC/AAC verification failure<br>'04' : Unrecognized Mode Flag<br>'05' : Unrecognized Scheme ID<br>'10' : MK parity error<br>'52' : Invalid Branch/Height<br>'68' : Command disabled<br>or a standard error code. |
| ARPC | 8 B | Calculated ARPC |
| Diagnostic data | 8 B | Calculated ARQC/TC/AAC returned only if the error code is '01' and the HSM is in Authorized State |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

### 10.3.2.2. Command Message for Mastercard CVN 10 & 11

**Request Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Command code | 2 A | **KQ** |
| Mode Flag | 1 H | '1' : Perform ARQC verification and ARPC generation |
| Scheme ID | 1 N | '1' : Mastercard |
| MK-AC | 32 H or 1 A + 32 H or 1 A + n A | For a Variant LMK, the 'MK-AC' must be encrypted under LMK pair 28-29 variant 1. |
| PAN+PAN Sequence number | 8 B or n B | Preformatted PAN and PAN sequence number.<br>It is the responsibility of the host system to ensure that the PAN/PAN Sequence Number is appropriately padded. |
| Application transaction counter | 2 B | Received from DE55.tag 9F36 |
| Unpredictable Number | 4 B | Received in DE55.Tag 9F37 |
| Transaction data length | 2 H | Present |
| Transaction data | n B | Present with 00 padding |

`

| | | |
|---|---|---|
| Delimiter | 1 A | Present to indicate end of transaction data |
| ARQC/TC/AAC | 8 B | ARQC received in DE 55.Tag 9F26 |
| ARC | 2 B | 2-byte ARC |
| Proprietary authentication data length | 1 N | Present if any |
| Proprietary authentication data | 0 ... 8 B | Present if length is not zero |
| Delimiter | 1 A | Value '%'. Optional; if present, the following field must be present. |
| LMK identifier | 2 N | LMK identifier; min value = '00'; max value is defined by issuer; must be present if the above Delimiter is present |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

**Response Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Response code | 2 A | **KR** |
| Error Code | 2 A | '00' : No error<br>'01' : ARQC/TC/AAC verification failure<br>'04' : Unrecognized Mode Flag<br>'05' : Unrecognized Scheme ID<br>'10' : MK parity error<br>'68' : Command disabled<br>or a standard error code. |
| ARPC | 8 B | Calculated ARPC |
| Diagnostic data | 8 B | Calculated ARQC/TC/AAC returned only if the error code is '01' and the HSM is in Authorized State |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

`

### 10.3.3. Visa

#### 10.3.3.1. Command Message for Visa CVN 18

**Request Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Command code | 2 A | **KW** |
| Mode Flag | 1 H | '3' : Perform ARQC verification and EMV 4.x Method 2 ARPC generation |
| Scheme ID | 1 N | 3' : Option B ICC Master Key Derivation and EMV Common Session Key Derivation |
| MK-AC | 32 H or 1 A + 32 H or 1 A + n A | For a Variant LMK, the 'MK-AC' must be encrypted under LMK pair 28-29 variant 1. |
| IV- AC | 16B | NA |
| PAN length | 2N | present |
| PAN+PAN Sequence number | 8 B or n B | As per length mentioned in PAN length. |
| Delimiter | 1A | Present |
| Branch / Height Parameters | 1 N | NA |
| Application transaction counter | 2 B | Received from DE55.tag 9F36 |
| Transaction data length | 2 H | Present |
| Transaction data | n B | Present with 80 padding |
| Delimiter | 1 A | Present to indicate end of transaction data |
| ARQC/TC/AAC | 8 B | ARQC received in DE 55.Tag 9F26 |
| ARC | 2 B | NA |
| CSU | 4 B | 4-byte CSU |
| Proprietary authentication data length | 1 N | Present if any |
| Proprietary authentication data | 0 ... 8 B | Present if length is not zero |
| Delimiter | 1 A | Value '%'. Optional; if present, the following field must be present. |
| LMK identifier | 2 N | LMK identifier; min value = '00'; max value is defined by issuer; must be present if the above Delimiter is present |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

`

**Response Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Response code | 2 A | **KX** |
| Error Code | 2 A | '00' : No error<br>'01' : ARQC/TC/AAC verification failure<br>'04' : Unrecognized Mode Flag<br>'05' : Unrecognized Scheme ID<br>'10' : MK parity error<br>'52' : Invalid Branch/Height<br>'68' : Command disabled<br>or a standard error code. |
| ARPC | 8 B | Calculated ARPC |
| Diagnostic data | 8 B | Calculated ARQC/TC/AAC returned only if the error code is '01' and the HSM is in Authorized State |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

### 10.3.3.2. Command Message for Visa CVN 10

**Request Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Command code | 2 A | **KQ** |
| Mode Flag | 1 H | '1' : Perform ARQC verification and ARPC generation |
| Scheme ID | 1 N | '0' : Visa |
| MK-AC | 32 H or 1 A + 32 H or 1 A + n A | For a Variant LMK, the 'MK-AC' must be encrypted under LMK pair 28-29 variant 1. |
| PAN+PAN Sequence number | 8 B or n B | Preformatted PAN and PAN sequence number.<br>It is the responsibility of the host system to ensure that the PAN/PAN Sequence Number is appropriately padded. |
| Application transaction counter | 2 B | Received from DE55.tag 9F36 |
| Unpredictable Number | 4 B | Received in DE55.Tag 9F37 |
| Transaction data length | 2 H | Present |
| Transaction data | n B | Present with 00 padding |
| Delimiter | 1 A | Present to indicate end of transaction data |
| ARQC/TC/AAC | 8 B | ARQC received in DE 55.Tag 9F26 |
| ARC | 2 B | 2-byte ARC |
| Proprietary authentication data | 1 N | Present if any |

`

| | | |
|---|---|---|
| length | | |
| Proprietary authentication data | 0 ... 8 B | Present if length is not zero |
| Delimiter | 1 A | Value '%'. Optional; if present, the following field must be present. |
| LMK identifier | 2 N | LMK identifier; min value = '00'; max value is defined by issuer; must be present if the above Delimiter is present |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

**Response Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Response code | 2 A | **KR** |
| Error Code | 2 A | '00' : No error<br>'01' : ARQC/TC/AAC verification failure<br>'04' : Unrecognized Mode Flag<br>'05' : Unrecognized Scheme ID<br>'10' : MK parity error<br>'68' : Command disabled<br>or a standard error code. |
| ARPC | 8 B | Calculated ARPC |
| Diagnostic data | 8 B | Calculated ARQC/TC/AAC returned only if the error code is '01' and the HSM is in Authorized State |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

### 10.3.4. RuPay

#### 10.3.4.1. Command Message for RuPay CVN 05 & 06

**Request Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Command code | 2 A | **KW** |
| Mode Flag | 1 H | '5' : Perform ARQC verification and DPAS Method 1 ARPC generation |
| Scheme ID | 1 N | 2' : Option A ICC Master Key Derivation and EMV Common Session Key Derivation |
| MK-AC | 32 H or 1 A + 32 H or 1 A + n A | For a Variant LMK, the 'MK-AC' must be encrypted under LMK pair 28-29 variant 1. |

`

| | | |
|---|---|---|
| IV- AC | 16B | NA |
| PAN length | 2N | NA |
| PAN+PAN Sequence number | 8 B or n B | For Scheme ID = '0' or '2', this field will be fixed at 8 bytes, and should contain the pre-formatted PAN/PAN Sequence No<br>It is the responsibility of the host system to ensure that the PAN/PAN Sequence Number is appropriately padded. |
| Delimiter | 1A | NA |
| Branch / Height Parameters | 1 N | NA |
| Application transaction counter | 2 B | Received from DE55.tag 9F36 |
| Transaction data length | 2 H | Present |
| Transaction data | n B | Present with 80 padding |
| Delimiter | 1 A | Present to indicate end of transaction data |
| ARQC/TC/AAC | 8 B | ARQC received in DE 55.Tag 9F26 |
| ARC | 2 B | NA |
| CSU | 4 B | For D-PAS (CVN 5 and 6), the CSU is a 2-byte value and must be right-padded with 0s. |
| Proprietary authentication data length | 1 N | NA |
| Proprietary authentication data | 0 ... 8 B | NA |
| Delimiter | 1 A | Value '%'. Optional; if present, the following field must be present. |
| LMK identifier | 2 N | LMK identifier; min value = '00'; max value is defined by issuer; must be present if the above Delimiter is present |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

**Response Command**

| Field | Length & Type | Details |
|---|---|---|
| Message header | m A | Message header length |
| Response code | 2 A | **KX** |
| Error Code | 2 A | '00' : No error<br>'01' : ARQC/TC/AAC verification failure<br>'04' : Unrecognized Mode Flag<br>'05' : Unrecognized Scheme ID<br>'10' : MK parity error<br>'52' : Invalid Branch/Height<br>'68' : Command disabled<br>or a standard error code. |
| ARPC | 8 B | Calculated ARPC |

`

| | | |
|---|---|---|
| Diagnostic data | 8 B | Calculated ARQC/TC/AAC returned only if the error code is '01' and the HSM is in Authorized State |
| End message delimiter | 1 C | Must be present if a message trailer is present. Value X'19. |
| Message trailer | n A | Optional. Maximum length 32 characters |

### 10.3.5. Sample HSM Message

| **Command request to HSM** | | |
|---|---|---|
| 0000 | 00 00 0c 07 ac 11 18 03 73 a5 ff 17 08 00 45 00 | ........s.....E. |
| 0010 | 00 9a 11 39 40 00 80 06 00 00 c0 a8 e2 e2 c0 a8 | ...9@..... |
| 0020 | b3 2c c6 0c 05 dc a2 ea f6 2b 6c 85 f7 84 50 18 | .,.....+l...P. |
| 0030 | fd 5c 17 ed 00 00 00 70 30 30 30 32 4b 57 31 32 | .\.....p0002KW12 |
| 0040 | 55 36 35 30 37 44 38 37 45 37 35 41 44 37 30 44 | U6507D87E75AD70D |
| 0050 | 42 45 39 42 30 46 44 30 39 41 30 39 30 31 43 45 | BE9B0FD09A0901CE |
| 0060 | 42 00 00 00 00 00 00 19 01 00 22 33 30 00 00 00 | B......"30... |
| 0070 | 04 00 00 00 00 00 00 00 00 03 56 40 00 04 80 00 | .....V@.... |
| 0080 | 03 56 13 05 28 00 f6 ba 2c e2 58 00 00 22 01 05 | .V..(...,.X.".. |
| 0090 | 80 00 03 04 00 00 80 00 00 00 00 00 00 3b a6 21 | .....;.! |
| 00a0 | 46 2a 4e c7 3b ec 00 14 | F*N.;... |
| **Command Response from HSM** | | |
| 0000 | 18 03 73 a5 ff 17 cc ef 48 26 90 3f 08 00 45 00 | ..s.....H&.?..E. |
| 0010 | 00 3a 40 a3 00 00 1a 06 48 bb c0 a8 b3 2c c0 a8 | .:@.....H....,.. |
| 0020 | e2 e2 05 dc c6 0c 6c 85 f7 84 a2 ea f6 9d 50 18 | ......l.....P. |
| 0030 | 0f a0 f2 20 00 00 00 10 30 30 30 32 4b 58 30 30 | ... ....0002KX00 |
| 0040 | 05 6f 5d d7 08 f1 84 eb | .o]..... |

# 11. Authorization Processing

## 11.1. Overview

Issuer process EMV transactions of the cards issued by itself. Issuer receives online transaction in the form ISO 8583 message format defined by respective payment schemes.
Transaction performed offline in the terminal does not come to issuer in the form of ISO 8583. Issuer received those transaction during clearing from acquirer bank through payment scheme.
The authorization and authentication decision parameters for offline transactions are pre-configured in card at the time of personalization.

## 11.2. Online Card Authorization

Issuer performs online authorization for transaction coming online in ISO 8583 format with all required fields. Once the transaction is received issuer performs below activity to authorize the transactions.

- PIN validation
- Account validation
- Balance validation

`

- iCVV validation
- Risk management

After performing all the above validation issuer performs EMV level validations by validating request cryptogram and generating response cryptogram.

## 11.3. Issuer Risk Management

Issuer performs risk management by validating terminal verification result and card verification result against pre-decided denial bytes of Issuer action code and card action code.
This process is optional and based on the arrangement with respective payment scheme.

The Terminal Verification Results (TVR) is a mandatory data element forwarded in authorization messages in DE 55. Tag 95. The TVR may be used by Chip Card Issuers, in their authorization decision process, to check transaction processing results from the Terminal. Some of the checking are

- Offline Data Authentication processing results (not performed, failed)
- Whether the card was on the Terminal exception file
- Processing restrictions results
- Cardholder Verification processing results
- Reasons the Transaction was sent online for authorization
- Processing results during the previous Transaction (Issuer authentication failed; script failed)

The Card Verification Result (CVR) is a mandatory data element forwarded in authorization messages in DE 55. Tag 9F10. The CVR may be used by Chip Card Issuers, in their authorization decision process, to check transaction processing results from the card. Some of the checking are

- Offline PIN versification performed
- Offline PIN verification failed
- PIN try limit exceeded
- Issuer authentication performed but failed
- DDA failed
- Application blocked, because PIN try limit exceeded.

Issuer should maintain individual denial bits for Terminal Action Code and Card Action Code for each payment scheme.
Each bit of Terminal action code is compared with terminal verification result (TVR) received in DE55.Tag 95.
Each bit of Card Action code is compared with CVR received in Tag 9F10.
If a particular bit in TVR and TAC is equals to 1, TVR validation fails.

The pseudo logic would be
if ***RISK ANALYSIS*** = TRUE
{
Match all bits of TVR with TAC - Denial
If any MATCH found (both bit == 1)
Set response code == Decline
`

}

Match all bits of CVR Byte 4 and Byte 5 with 2 Byte CAC- Denial

If any MATCH found (both bit == 1)
Set response code == Decline

}

## 11.4. ARQC verification

Issuer performs ARQC validation after validating PIN, account, balance and risk if any. Based on these validation performed, Issuer prepares 2 byte ARC or 4 byte CSU as mentioned in section 10.2 and sends to HSM for ARQC validation along with the cryptogram received in DE55.Tag 9F26.

If ARQC validation is failed Issuer updates ARC or CSU bits accordingly before preparation of Issuer authentication data with the default cryptogram returned by HSM.

## 11.5. Issuer Authentication Data

Once every validation is performed issuer has to prepare issuer authentication data to be sent in DE55. Tag 91.
Issuer authentication data can be prepared in two ways based on ARPC method used as per defined CVNs in respective payment schemes.

- Issuer Authentication Data: = 8 Byte ARPC || 2-byte ARC
- Issuer Authentication Data: = ARPC || CSU || Proprietary Authentication Data

## 11.6. Script Processing

Issuer can control the chip card performing some action without getting the physical card back from customer. EMV allows issuer to do so by sending Issuer scripts.
An issuer may provide command scripts to be sent to the ICC by the terminal in order to perform functions that are not necessarily relevant to the current transaction but are important for the continued functioning of the application present in the ICC. Multiple scripts may be provided with an authorization response & each tag may contain multiple number of Issuer Script Commands.

Tag 71 is used when the Issuer sends Issuer Script Commands to the card in the response to be applied to the current transaction.

Tag 72 is used when the Issuer sends Issuer Script Commands to the card in the response to be applied to the next consecutive transaction.

The total length of all the Issuer Scripts in the response shall be less than or equal to 128 bytes.
In the case where only one Issuer Script is sent & no Issuer Script Identifier is used, the maximum length of the Issuer Script Command (Tag ‘86’) is limited to 124 bytes.

`

The below two tables illustrate the structure of Issuer script commands:

| Tag | Length | Tag | Length | Script ID | Commands |
|---|---|---|---|---|---|
| '71' or '72' | L(Σdata, including Script ID, tags, and lengths) | '9F18' | '04' | Identifier (4 Bytes) | See below table |

| T1 | L1 | V1 | T2 | L2 | V2 | T3 | L3 | V3 |
|---|---|---|---|---|---|---|---|---|
| '86' | L(V1) | Command | '86' | L(V2) | Command | '86' | L(V3) | Command |

Script identified Tag 9F18 is optional in case single script command is present.

It is possible for multiple Issuer Scripts to be delivered with a single authorisation response with following rules

- Issuer Script Commands shall be separated using the BER-TLV coding of the data objects defining the commands (tag '86').
- Each command shall be constructed as a command APDU in the sequence in which it appeared in the Issuer Script.

### 11.6.1. Command Structure

Each command is sent as Application protocol data unit (APDU). The command Application Protocol Data Unit (APDU) consists of a mandatory header of four bytes followed by a conditional body of variable length.

| CLA | INS | P1 | P2 | Lc | Data | Le |
|---|---|---|---|---|---|---|
| Mandatory Header | | | | Conditional Bytes | | |

CLA = Class Byte of the Command Message
INS = Instruction Byte of Command Message
P1 = Parameter 1
P2 = Parameter 2
Lc = Length of command data field
Le = Length of expected data

**Example 1:** Issuer Script Command Format with single script command
710F860D8424000008792480EAF02285B8

- 71 – Script tag
- 0F – total length
- 86 – Script command
- 0D – Script length
- 84 - CLA
- 24 - INS
- 00 - P1
- 00 – P2
- 08 - Lc

`

- 792480EAF02285B8 – Data (MAC)

**Example 2:** *Issuer Script Command Format with three script command:*
72CE9F18044142434486841E0000087E3DC7B967D2E4B586841E0000080A853270D75DCC18
86841E000008F7D01DD065C8575786841E0000086FFF0CC88E34A6C386841E000008F1B6628D20FA2
6F786841E00000841DFE039B9F39CC286841E000008AAFB2E0E58CE8448

- 72- Tag
- CE – Total Length
- 9F18 – Script identifier tag
- 04 – Identifier length
- 41424344 – Identifier
- 841E0000087E3DC7B967D2E4B5 – Script Command with Issuer scripts
- 841E0000080A853270D75DCC18 – Script Command with Issuer scripts
- 841E000008F7D01DD065C85757– Script Command with Issuer scripts
- 841E0000086FFF0CC88E34A6C3 – Contd..

### 11.6.2. Message Authentication Code (MAC)

A symmetric cryptographic transformation of data that protects the sender and the recipient of the data against forgery by third parties.

MAC is computed as mentioned in the section 10.1.1.2 along with the MAC Session Key derived as mentioned in section 9.2.3.1.

| Sl no | Parameters | VISA | Mastercard | RuPay |
|---|---|---|---|---|
| 1 | Secure Messaging Format | Format 2 | Format 2 | Format 2 |
| 2 | Input data | CLA, INS, P1, P2, Lc \|\| ATC | CLA, INS, P1, P2, Lc \|\| ATC \|\| An eight-byte data field RAND[21] | CLA, INS, P1, P2, Lc \|\| ATC \|\| An 8-byte unpredictable value R[21] |
| 3 | MAC Length | 4 or 8 bytes | 8 bytes | 8 bytes |
| 4 | Session Key | XOR session key (section 9.2.3.4) | MasterCard Proprietary SKD approach or EMV CSK approach (Section 9.2.3) based on CVN | EMV CSK (Section 9.2.3.1) |

[21] An 8-byte **RAND** (Random Number) or unpredictable value '**R**' is used to generate a number as explained below. For this explanation, let's call the number as 'R'.
For the 1st Issuer Script Command, **R** is the Application Cryptogram provided by the card in response to the 1st Generate AC command. If more than one Issuer Script command is sent, then all of the next consecutive Issuer Script commands generated & sent in this session uses the previous value of **R + 1**.

For Example, if 'R' is the Application Cryptogram = '1234 1234 1234 1234', then,

- For the 1st Issuer Script in this session, R = **'**1234 1234 1234 1234'
- For the 2nd Issuer Script in this session, R = **'**1234 1234 1234 1235'

`

- For the 3rd Issuer Script in this session, R = ‘1234 1234 1234 1236’

### 11.6.3. List of Issuer Script Commands

List of Issuer Script Commands supported by EMV is as described below:

1. Application Block Command
2. Application Unblock Command
3. Card Block Command
4. PIN Change/Unblock Command

#### 11.6.3.1. Application Block Command

The Application Block Command is a post-issuance command that invalidates or blocks the currently selected application.

| Code | Value |
|---|---|
| CLA | ‘8C’ or ‘84’; coding according to the secure messaging (Format 1 or Format 2) |
| INS | ‘1E’ |
| P1 | ‘00’; all other values are RFU |
| P2 | ‘00’; all other values are RFU |
| Lc | Number of data bytes |
| Data | Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command: 841E0000 + Lc + MAC Data = 841E000008F9F2F1A6B581490D

#### 11.6.3.2. Application Unblock Command

The Application Unblock Command is a post-issuance command that rehabilitates or restores the currently selected application.

| Code | Value |
|---|---|
| CLA | ‘8C’ or ‘84’; coding according to the secure messaging (Format 1 or Format 2) |
| INS | ‘18’ |
| P1 | ‘00’; all other values are RFU |
| P2 | ‘00’; all other values are RFU |
| Lc | Number of data bytes |

`

| | |
|---|---|
| Data | Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command: 84180000 + Lc + MAC Data = 8418000008F9F2F1A6B581490D

### 11.6.3.3. Card Block Command

The Card Block Command is a post-issuance command that permanently disables all applications in the ICC.

| Code | Value |
|---|---|
| CLA | '8C' or '84'; coding according to the secure messaging (Format 1 or Format 2) |
| INS | '16' |
| P1 | '00'; all other values are RFU |
| P2 | '00'; all other values are RFU |
| Lc | Number of data bytes |
| Data | Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command: 84160000 + Lc + MAC Data = 8416000008F9F2F1A6B581490D

### 11.6.3.4. PIN Change/Unblock Command

The PIN Change/Unblock Command is a post-issuance command that provides Issuer the capability either to unblock the PIN or to simultaneously change & unblock the reference PIN.

Note: The reference PIN is the PIN which is stored in the card & which is associated with the application & which the card uses to check the Transaction PIN Data transmitted within the Verify Command.

| Code | Value |
|---|---|
| CLA | '8C' or '84'; coding according to the secure messaging (Format 1 or Format 2) |
| INS | '24' |
| P1 | '00' |
| P2 | '00', '01', or '02' |
| Lc | Number of data bytes |
| Data | Enciphered PIN data component, if present, and Message Authentication Code (MAC) data component; coding |

`

| | |
|---|---|
| | according to the secure messaging |
| Le | Not present |

Structure of the Command: 84240000 + Lc + Enciphered PIN data (If present) + MAC Data = 8424000008F9F2F1A6B581490D

If P2 is equal to ’00’, the reference PIN is unblocked & the PIN Try Counter is reset to the PIN Try Limit & there is no PIN update, since the command does not contain a new PIN value.
The usage of P2 equal to ‘01’ or ’02’ is reserved for payment systems.

### 11.6.4. VISA

Apart from list of Issuer Script Commands mentioned in section 11.6.3, as specified by EMV, VISA supports below mentioned Issuer Script Commands:

A. PUT Data Command
B. Update Record Command

Apart from PUT Data & Update Record Command, rest all the commands are used by VISA as per EMV specified Issuer Scripting command formats. The PUT Data & Update Record Command Issuer Scripts used by VISA are explained further below.

#### 11.6.4.1. PUT Data Command

The PUT Data Command is a post-issuance command that updates specific primitive & constructed data objects which are stored in the card. The data objected can be updated with this command only if the command has a tag associated with it.

| Code | Value |
|---|---|
| CLA | ‘04’ |
| INS | ‘DA’ |
| P1 P2 | If ‘0000’ then the data fields contain one or more primitive data objects (in BER-TLV format) to be updated<br>All other values indicate the tag of the primitive data object or template tag of the constructed data object to be updated |
| Lc | Number of data bytes |
| Data | Data to be updated (Explained below) and Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command: 04DA0000 + Lc + Data to be updated (Explained below) + MAC Data = 04DA00000DCA06DF0104B0FFFCE67C9E63B4

Data:

`

If P1P2 contains the tag of a primitive data object, Data will be;

- The new value for a primitive data object
- Followed by a 4 or 8-Byte MAC. The MAC value is not preceded by a tag or length

If P1P2 contains 0000, Data will be;

- One or more primitive data objects to be updated, each in BER-TLV format
- Followed by a 4 or 8-Byte MAC. The MAC value is not preceded by a tag or length

If P1P2 contains the template tag for a constructed data object, Data will be;

- One or more primitive data elements, each in BER-TLV format & identified by a tag
- Followed by a 4 or 8-Byte MAC. The MAC value is not preceded by a tag or length

#### 11.6.4.2. Update Record Command

The Update Record Command is a post-issuance command used to update a record in a file with the data provided in the command data field.

| Code | Value |
|---|---|
| CLA | '04' |
| INS | 'DC' |
| P1 | Record number to be updated |
| P2 | Reference control parameter |
| Lc | Number of data bytes |
| Data | Record data and Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command: 04DCP1P2 + Lc + Record Data + MAC Data =
DC03141A70108E0E0000000000000000000440342031F0392C354B8847E467

The command data consists of the new record contents followed by 4- or 8-byte MAC generated using Format 2

### 11.6.5. MasterCard

MasterCard allows the Issuer to provide a post-issuance command to be sent to the ICC.
Apart from list of Issuer Script Commands mentioned in section 11.6.3 as specified by EMV, MasterCard supports below mentioned Issuer Script Commands:

`

A. PUT Data Command

B. Update Record Command

Apart from PUT Data & Update Record Command, the rest all the commands are used by MasterCard as per EMV specified Issuer Scripting command formats. The PUT Data & Update Record Command Issuer Scripts used by MasterCard are explained below.

### 11.6.5.1. PUT Data Command

| Code | Value |
|---|---|
| CLA | '84' |
| INS | 'DA' |
| P1 P2 | Tag |
| Lc | Number of data bytes |
| Data | Data to be updated and Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command = 84DA + Tag + Lc + Data to be updated + MAC Data =

84DA00820A6400DF0104B0FFFCE67C

Single byte tags are preceded with a leading '00' byte to fill P1/P2

Below mentioned tag values are supported for PUT Data:

| P1/P2 | Data Element | Length |
|---|---|---|
| '00D5' | Application Control | 2 |
| '00D6' | Default ARPC Response Code | 2 |
| '9F14' | Lower Consecutive Offline Limit | 1 |
| '9F23' | Upper Consecutive Offline Limit | 1 |
| '00CA' | Lower Cumulative Offline Transaction Amount | 6 |
| '00CB' | Upper Cumulative Offline Transaction Amount | 6 |
| '00C4' | Card Issuer Action Code – Default | 3 |
| '00C5' | Card Issuer Action Code – Online | 3 |
| '00C3' | Card Issuer Action Code – Decline | 3 |
| '00C9' | CRM Currency Code | 2 |
| '00D1' | Currency Conversion Table | 25 |
| '00C8' | CRM Country Code | 2 |
| '0094' | Application File Locator | var[27] |

`

| | | |
|---|---|---|
| ‘0082’ | Application Interchange Profile | 2 |
| ‘00C7’ | CDOL 1 Related Data Length | 1 |
| ‘00D3’ | Additional Check Table | 18 |
| ‘DF01’ | Security Limits | 6 |
| [27] = A memory space of at least 32 bytes must be available for the Application File Locator. | | |

### 11.6.5.2. Update Record Command

| Code | Value |
|---|---|
| CLA | ‘84’ |
| INS | ‘DC’ |
| P1 | Record Number |
| P2 | Reference Control Parameter |
| Lc | Number of data bytes |
| Data | Data to be updated and Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command = 84DC + P1P2 + Lc + Data to be updated + MAC Data

= 84DC03141A70108E0E0000000000000000440342031F0392C354B8847E467

Below table specifies the coding of the Reference Control Parameter:

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | | | | SFI |
| | | | | | x | x | x | |
| | | | | | 1 | 0 | 0 | P1 is a record number |

`

### 11.6.6. RuPay

RuPay allows the Issuer to provide a post-issuance command to be sent to the ICC. RuPay supports Issuer Script Template 2 (Tag ’72)

Apart from list of Issuer Script Commands mentioned in section 11.6.311.6.3 as specified by EMV, RuPay supports below mentioned Issuer Script Commands:

A. PUT Data Command
B. Update Record Command
C. Card Block Command (Platform Optional)

Apart from PUT Data & Update Record Command, the rest all the commands are used by RuPay as per EMV specified Issuer Scripting command formats. The PUT Data & Update Record Command Issuer Scripts used by RuPay are explained below.

#### 11.6.6.1. PUT Data Command

| Code | Value |
|---|---|
| CLA | ‘84’ or ‘80’ |
| INS | ‘DA’ |
| P1 P2 | Data Object (See Table below) |
| Lc | Number of data bytes |
| Data | Data Object Value and Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command = **84DA + Data Object + Lc + Data Object Value + MAC Data = 84DA00D00DCA06DF0104B0FFFCE67C9E63B4**

Below are the supported Data Objects in PUT Data:

| P1P2 | Data Element | Length |
|---|---|---|
| ‘0082’ | Application Interchange Profile | 2 |
| ‘0094’ | Application File Locator | Var. |
| ‘00C1’ | Application Configuration Options | 2 |
| ‘00C3’ | Currency Conversion Code 1 | 5 |
| ‘00C4’ | Currency Conversion Code 2 | 5 |
| ‘00C5’ | Card Action Code – Denial | 2 |
| ‘00C6’ | Card Action Code – Default | 2 |
| ‘00C7’ | Card Action Code – Online | 2 |

`

| | | |
|---|---|---|
| ‘00C8’ | Lower Cumulative Offline Amount (LCOA) | 6 |
| ‘00C9’ | Upper Cumulative Offline Amount (UCOA) | 6 |
| ‘00CA’ | Single Transaction Amount (STA) limit | 6 |
| ‘00CB’ | Lower Consecutive Offline Limit (LCOL) | 1 |
| ‘00CC’ | Upper Consecutive Offline Limit (UCOL) | 1 |
| ‘00D0’ | Issuer Application Data Object List | Var. |
| ‘00D2’ | CRM Country Code | 2 |
| ‘00D3’ | CRM Currency Code | 2 |
| ‘DF01’ - ‘DF0A’ | Issuer Defined Data Tag (IDDT) 0 – 9 | Var. |
| ‘DF10’ | PDOL Check Table – Denial | Var. |
| ‘DF11’ | PDOL Check Table – Profile | Var. |
| ‘DF12’ | PDOL Check Table – Online | Var. |
| ‘DF20’ – ‘DF2F’ | Transaction Profile Objects (0 – 15) | 62 bytes each |

### 11.6.6.2. Update Record Command

| Code | Value |
|---|---|
| CLA | ‘84’ |
| INS | ‘DC’ |
| P1 | Record Number |
| P2 | Reference Control Parameter |
| Lc | Number of data bytes |
| Data | Data Object Value and Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command = 84DC + P1P2 + Lc + Data to be updated + MAC Data
= 84DC03141A70108E0E0000000000000000440342031F0392C354B8847E467

Below is the table for Reference Control Parameter Definition

| Bits | Value |
|---|---|
| b8-b4 | SFI |
| b3-b1 | 100 (binary value) |

`

#### 11.6.6.3. Card Block Command

| Code | Value |
|---|---|
| CLA | '84' |
| INS | '16' |
| P1 | '00' |
| P2 | '00' |
| Lc | Number of data bytes |
| Data | Message Authentication Code (MAC) data component; coding according to the secure messaging |
| Le | Not present |

Structure of the Command: **84160000 + Lc + MAC Data**

= 841600001031B2C75ED3128854 A019FD61B4A564BE

# 12. Failure Scenarios

There are different failure scenarios in EMV transaction where transaction is declined based EMV validations. An EMV transaction can be declined for below reasons

- ARQC validation failed by Issuer
- Risk Management failed by Issuer
- Transaction declined by card
- Card has not received response.

## 12.1. ARQC validation failed by Issuer

When ARQC validation is failed at issuer end, the transaction shall be declined by Issuer with valid response code.

| Scheme | MTI | Field | Code | Description | Action |
|---|---|---|---|---|---|
| VISA | 0210/0110 | DE 39 | 05 | Do not honour | D |
| Mastercard | 0210/0110 | DE 39 | 57 | Cryptographic failure | D |
| RuPay | 0210/0110 | DE 39 | E3 | ARQC validation failed by Issuer | D |

`

## 12.2. TVR Validation Failed

Terminal verification results are sent in Tag 95 to issuer, and the transaction can be rejected by Issuer based on the TVR validation.

TVR Values may be used by Issuers during processing the transactions, to check results from the Terminal, including:

- Offline Data Authentication processing results
- If the card appears in the Terminal exception file
- Processing restrictions
- Cardholder Verification results
- TRM
- Script information.

The issuer however, may or may not decline only on the basis of the details available in the TVR. This is optional.

| Scheme | MTI | Field | Code | Description | Action |
|---|---|---|---|---|---|
| VISA | 0210/0110 | DE 39 | 05 | Do not honour | D |
| Mastercard | 0210/0110 | DE 39 | 05 | Do not honour | D |
| RuPay | 0210/0110 | DE 39 | E4 | Transaction Declined by Issuer basis TVR Validation | D |

## 12.3. Transaction Declined by Issuer basis CVR Validation

Card verification results are sent in Tag 9F10 to issuer, and the transaction can be rejected by Issuer based on the CVR validation.

| Scheme | MTI | Field | Code | Description | Action |
|---|---|---|---|---|---|
| VISA | 0200/0100 | DE 39 | 05 | Transaction Declined by Issuer basis CVR Validation | D |
| Mastercard | 0200/0100 | DE 39 | 05 | Transaction Declined by Issuer basis CVR Validation | D |
| RuPay | 0200/0100 | DE 39 | E5 | Transaction Declined by Issuer basis CVR Validation | D |

## 12.4. Card did not receive AAC / TC

This scenario occurs if the card is removed from the terminal before completion of the transaction. In this case card terminal will not find card to pass the final decision command after receiving response from issuer. Terminal generates a reversal message in this scenario with below response code in DE-39, if issuer has approved the transaction.

| Scheme | MTI | Field | Code | Description | Action |
|---|---|---|---|---|---|
| VISA | 0400/0420 | DE 39 | 22 | Card rejected after online Issuer Approval | D |
| Mastercard | 0400/0420 | DE 39 | 17 | Card rejected after online Issuer Approval | D |
| RuPay | 0400/0420 | DE 39 | E2 | Card rejected after online Issuer Approval | D |

`

## 12.5. Card rejected after online Issuer Approval

In an EMV transaction card has the ultimate authority to take decision on approving or rejecting transaction. So, card may decline the transaction based on the validation of ARPC received in response.

Terminal generates a reversal message in this scenario if issuer has approved the transaction, but card has rejected the transaction.

| Scheme | MTI | Field | Code | Description | Action |
|---|---|---|---|---|---|
| VISA | 0400/0420 | DE 39 | 22 | Card rejected after online Issuer Approval | D |
| Mastercard | 0400/0420 | DE 39 | 17 | Card rejected after online Issuer Approval | D |
| RuPay | 0400/0420 | DE 39 | E1 | Card rejected after online Issuer Approval | D |

# 13. Appendix A: Issuer Application Data

The Issuer application data is represented in tag 9F10 of DE 55. Issuer application data contains below components with variable length up to 32 bytes.

- Derivation Key Index (DKI) – 1 byte
- Cryptogram Version Number (CVN) – 1 byte
- Card Verification Results (CVR) – 4 Bytes – 6 bytes.
- Issuer Discretionary Data (Optional) – rest of the bytes.

## 13.1. Visa, JCB, CUP

The issuer application data structure for Visa consists of below

- Visa Discretionary Data
    - Length Indicator – 1 byte: This includes total length of DKI+CVN+4-byte CVR equals to 06.
    - Derivation Key Index (DKI) – 1 byte
    - Cryptogram Version Number (CVN) – 1 byte
    - Card Verification Results (CVR) – 4 Bytes
- Issuer Discretionary Data (Optional) – 8th Byte to 32nd Byte.

Example - 9F100706010A03A41000

- 9F10 – Tag
- 07 – Total length
- 06 – Length Indicator
- 01 – DKI
- 0A – CVN
- 03A41000 - CVR

`

## 13.2. Mastercard, RuPay

The issuer application data structure for Mastercard and RuPay consists of below

- Derivation Key Index (DKI) – 1 byte
- Cryptogram Version Number (CVN) – 1 byte
- Card Verification Results (CVR) – 6 Bytes
- Issuer Discretionary Data (Optional) – 9th Byte to 32nd Byte.

Example - 9F1008010503A410000000

- 9F10 – Tag
- 08 – Total length
- 01 – DKI
- 05 – CVN
- 03A410000000 – CVR

`

# 14. Appendix B: Terminal Verification Results

The Terminal Verification results bits as per EMV depicted below. The same is considered as Issuer action code.

**TVR Byte 1: (Leftmost)**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | X | X | Offline data authentication was not performed |
| X | 1 | X | X | X | X | X | X | SDA failed |
| X | X | 1 | X | X | X | X | X | ICC data missing |
| X | X | X | 1 | X | X | X | X | Card appears on terminal exception file [22] |
| X | X | X | X | 1 | X | X | X | DDA failed |
| X | X | X | X | X | 1 | X | X | CDA failed |
| X | X | X | X | X | X | 0 | X | RFU |
| X | X | X | X | X | X | X | 0 | RFU |

**TVR Byte 2:**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | X | X | ICC and terminal have different application versions |
| X | 1 | X | X | X | X | X | X | Expired application |
| X | X | 1 | X | X | X | X | X | Application not yet effective |
| X | X | X | 1 | X | X | X | X | Requested service not allowed for card product |
| X | X | X | X | 1 | X | X | X | New card |
| X | X | X | X | X | 0 | X | X | RFU |
| X | X | X | X | X | X | 0 | X | RFU |
| X | X | X | X | X | X | X | 0 | RFU |

`

**TVR Byte 3:**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | X | X | Cardholder verification was not successful |
| X | 1 | X | X | X | X | X | X | Unrecognised CVM |
| X | X | 1 | X | X | X | X | X | PIN Try Limit exceeded |
| X | X | X | 1 | X | X | X | X | PIN entry required and PIN pad not present or not working |
| X | X | X | X | 1 | X | X | X | PIN entry required, PIN pad present, but PIN was not entered |
| X | X | X | X | X | 1 | X | X | Online PIN entered |
| X | X | X | X | X | X | 0 | X | RFU |
| X | X | X | X | X | X | X | 0 | RFU |

**TVR Byte 4:**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | X | X | Transaction exceeds floor limit |
| X | 1 | X | X | X | X | X | X | Lower consecutive offline limit exceeded |
| X | X | 1 | X | X | X | X | X | Upper consecutive offline limit exceeded |
| X | X | X | 1 | X | X | X | X | Transaction selected randomly for online processing |
| X | X | X | X | 1 | X | X | X | Merchant forced transaction online |
| X | X | X | X | X | 0 | X | X | RFU |
| X | X | X | X | X | X | 0 | X | RFU |
| X | X | X | X | X | X | X | 0 | RFU |

`

**TVR Byte 5 (Rightmost):**

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Meaning |
|---|---|---|---|---|---|---|---|---|
| 1 | X | X | X | X | X | X | X | Default TDOL used |
| X | 1 | X | X | X | X | X | X | Issuer authentication failed |
| X | X | 1 | X | X | X | X | X | Script processing failed before final GENERATE AC |
| X | X | X | 1 | X | X | X | X | Script processing failed after final GENERATE AC |
| X | X | X | X | 0 | X | X | X | RFU |
| X | X | X | X | X | 0 | X | X | RFU |
| X | X | X | X | X | X | 0 | X | RFU |
| X | X | X | X | X | X | X | 0 | RFU |

# 15. Appendix C: Perso File format

Issuer shall communicate personalization profile in below format to personalization bureau to create personalization file for EMV chip card. A profile either can be a standard profile based on respective payment scheme defined profiles or Issuer can have their own customized profile.

In personalization file all data to be personalized shall be given to personalization bureau. The bureau then creates a personalization file based on his proprietary format or in XML format which needs to be sent to the card.

**Magnetic Stripe data**

| Data Element | Value |
|---|---|
| Track 1 data | %B6174841230000011^BASIC CARD/IMAGE 01 ^2612620990A616468616172000000000?0 |
| Track 2 data | ;6174841230000011=261262099000000000000?0 |
| CVD2 | 123 |

`

**SFI 01 REC 01**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Track 2 Equivalent Data | 57 | 12 | 60 74 84 12 30 00 00 23 D2 61 26 20 19 10 00 00 00 0F |
| Cardholder Name | 5F20 | 13 | 52 55 50 41 59 20 43 41 52 44 2F 49 4D 41 47 45 20 30 31 (BASIC CARD/IMAGE 01) |
| Track 1 Discretionary Data | 9F1F | 18 | 31 39 31 41 36 31 36 34 36 38 36 31 36 31 37 32 30 30 30 30 30 30 30 30 |

**SFI 02 REC 01**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Issuer Public Key Certificate | 90 | F8 | 99 9C B6 A2 33 FD 95 B1 52 AA 03 4D 20 50 E1 6E 51 C3 81 6A 82 E2 C4 F2 AD 32 AE 70 63 0C 15 21 05 10 69 8B 54 DC 6E EF 00 4C F8 73 F9 AC F9 7D 13 26 3A 2D 4B 23 57 46 9D 26 B5 CF EA 79 45 1A B2 BD 04 C5 71 E7 E8 1B 91 4D 6D A9 A3 B2 C7 4C 5A 52 1C B0 1B B8 63 1F 79 46 FF B8 44 E6 96 45 1D FF 12 B3 16 AB E9 D4 3E C8 8E 2D C8 FC E9 D8 59 70 7C 7B 07 A8 88 37 07 D0 4D 1E 94 99 C0 98 68 34 14 48 09 5E 75 4D 1C 16 06 9E 56 70 05 B8 1D 29 A3 53 A7 91 60 E2 63 22 A8 2E 2D A8 B7 69 9A 88 4D F2 83 FE CF 52 8C 52 64 7F FA D3 C4 05 76 9D 08 88 DD 12 11 D6 9C 81 00 A2 52 89 CC C0 D5 9D 7A 7C 14 8C 2A DF 3F C4 94 8A 76 85 C1 98 5D 90 6A 1D 7A 51 45 81 F5 03 C4 86 FE E8 98 FA 9A 66 85 F6 02 59 F0 95 67 D8 FD 96 78 98 16 A6 EF 3F EC 48 49 AD 87 4E |

`

**SFI 02 REC 02**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Issuer Public Key Exponent | 9F32 | 01 | 03 |
| Issuer Public Key Remainder | 92 | 22 | FB F8 61 48 D4 18 3F C5 AD 8E 29 D7 19 E9 49 A7 1C D4 CB 5A 62 CA 13 E4 16 0A A0 17 E2 1E 64 1F 1D FB |
| CA Public Key Index | 8F | 01 | 6D |

**SFI 02 REC 03**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Signed Static Application Data | 93 | F6 | 02 38 EC C7 D7 75 8F D0 12 76 87 22 04 B4 00 FC 63 DE A1 AC 06 AE 51 CB 83 11 8C DA 87 E3 F8 F5 04 B6 A1 02 DF 1E 7C 13 D6 4F 12 6A A3 8E 22 71 63 A7 30 4C ED F1 13 81 56 3C 36 01 4C E5 42 50 AC<br>88 4A 1B 75 28 3C 02 D8 13 37 C4 0A 40 8F D2 95 79 81 D5 B6 A9 34 E0 41 28 D7 2C 36 4F 7A 48 35 A6 15 D6 E3 24 CE 20 2C 7D AB C2 6F FF 4D 14 93 F9 08 00 30 7C 22 FA 12 CA 5B 29 00 7B B3 AE 02 24 76 2A 16 F8 EF 7F 52 3E 1D 77 38 07 C7 EF 3B A8 57 7D B1 CB E3 7E 5E AB BF FA 76 9F D9 62 F7 49 14 17 2B A5 97 CF BB 7C F9 8B 9A CC 3B 8B 1D D5 A5 10 9D 7E 55 DB 34 FB 59 1A DA D5 C8 FE AF 0C 0A<br>58 06 E4 B1 D4 40 8B F5 8C A1 5E 34 4D 86 42 E4 A9 C4 84 60 F6 96 9D 3A 87 30 22 DA 57 3F A6 A2 D7 86 F6 CB 69 FE D3<br>11 A3 24 0D AD 35 A5 CF CA AD EF CA |

**SFI 03 REC 01**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Application Primary Account Number | 5A | 08 | 61 74 84 12 30 00 00 11 |
| Application PAN Sequence Number | 5F34 | 01 | 01 |
| Application Effective Date | 5F25 | 03 | 13 01 01 |
| Application Expiration Date | 5F24 | 03 | 26 12 31 |

`

| | | | |
|---|---|---|---|
| Application Usage Control | 9F07 | 02 | FF 80 |
| Issuer Country Code | 5F28 | 02 | 03 56 |
| CVM List | 8E | 10 | 00 00 00 00 00 00 00 00 42 01 42 03 5E 03 1F 03 |
| CDOL1 | 8C | 1B | 9F 02 06 9F 03 06 9F 1A 02 95 05 5F 2A 02 9A 03 9C 01 9F 37 04 9F 35 01 9F 34 03 |
| CDOL2 | 8D | 09 | 91 0A 8A 02 95 05 9F 37 04 |
| IAC Default | 9F0D | 05 | F0 40 C4 28 00 |
| IAC Denial | 9F0E | 05 | 00 10 00 00 00 |
| IAC Online | 9F0F | 05 | F0 68 DC F8 00 |

**SFI 03 REC 02**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Application Currency Exponent | 9F44 | 01 | 02 |
| Application Currency Code | 9F42 | 02 | 03 56 |
| SDA Tag List | 9F4A | 01 | 82 |
| Service Code | 5F30 | 02 | 06 20 |
| Application Version Number | 9F08 | 02 | 00 01 |

**Card Internal Data Elements**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| CRM Country Code | D2 | 02 | 03 56 |
| CRM Currency Code | D3 | 02 | 03 56 |
| Application Configuration Options | C1 | 02 | 05 00 |

`

| | | | |
|---|---|---|---|
| Log Format | 9F4F | 1A | 9F 02 06 5F 2A 02 9A 03 9F 36 02 9F 34 03 9F 52 06 9F 1A 02 95 05 9C 01 8A 02 |

**Issuer Application Data (IAD)**

| Data Element | Length | Value |
|---|---|---|
| Derivation Key Index (DKI) | 01 | 01 |
| Cryptogram Version Number (CVN) | 01 | 05 |
| Card Verification Results (CVR) | 06 | xx xx xx xx xx xx(xx is the dynamically generated value by card during the transaction.) |

**Security limits**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Failed MAC limit | - | 01 | FF |
| Lifetime MAC Limit | - | 03 | FF FF FF |
| Session MAC Limit | - | 01 | FF |

**Default 'Profile 0' Resources**

| Data Element | Tag | Length | Value |
|---|---|---|---|
| Profile identifier (Profile 0) | DF20 | - | - |
| AIP | 82 | 02 | 58 00 |
| AFL | 94 | 0C | 08 01 01 00 10 01 03 00 18 01 02 01 |
| CAC-Default | C6 | 02 | 03 6B |
| CAC-Denial | C5 | 02 | 01 00 |
| CAC-Online | C7 | 02 | B2 7F |
| LCOL (Lower Consecutive Offline Limit) | CB | 01 | 07 |
| UCOL (Upper Consecutive Offline Limit) | CC | 01 | 09 |
| LCOA (Lower Cumulative Offline Amount) limit | C8 | 06 | 00 00 00 01 00 00 (₹100.00) |

`

| | | | |
|---|---|---|---|
| UCOA (Upper Cumulative Offline Amount) limit | C9 | 06 | 00 00 00 01 50 00 (₹150.00) |
| STA (Single Transaction Amount) limit | CA | 06 | 00 00 00 01 00 00 (₹100.00) |

**3DES keys**

| Data Element | Length | Value |
|---|---|---|
| Master Key for Cryptogram Generation | 10 | 0E F2 29 68 6E 46 FD F4 4C 26 A4 97 C2 2F E9 91 |
| Master Key for Secure Messaging INTEGRITY | 10 | B5 2A F7 7A 1C A7 9E BC 86 91 38 A7 20 13 25 04 |
| Master Key for Secure Messaging ENCRYPT | 10 | B0 15 57 A7 15 34 EC 7A 3E 43 04 CB 79 AE 04 29 |

**SELECT Command Response (FCI)**

| Tag | | | | Description | Length | Value |
|---|---|---|---|---|---|---|
| 6F | | | | FCI template | - | - |
| | 84 | | | DF Name (Main Application) | 07 | A0 00 00 05 24 10 10 |
| | A5 | | | FCI Proprietary Template | - | - |
| | | 50 | | Application Label | 0B | 52 75 50 61 79 20 44 65 62 69 74 (Basic Debit) |
| | | 87 | | Application Priority Indicator | 01 | 01 |
| | | BF0C | | FCI Issuer Discretionary Data | - | - |
| | | | 9F4D | Transaction Log Entry | 02 | 0B04 |

`

## 16. References

| SL No | Scheme | Document name | Version |
|---|---|---|---|
| 1 | EMV | EMV_v4.3_Book_2_Security_and_Key_Management_20120607061923900 | 4.3 |
| 2 | EMV | EMV_v4.3_Book_3_Application_Specification_20120607062110791 | 4.3 |
| 3 | Visa | Visa Integrated Circuit Card Specification | 1.5 |
| 4 | | Base 1 Technical specification Vol- 1 | 2018 |
| 5 | | SMS ATM technical specification Vol 1 | 2018 |
| 6 | Mastercard | M/Chip Card Application Cryptographic Algorithms | 2010 |
| 7 | | M/Chip 4 Version 1.1 Security & Key Management | 1.1 |
| 8 | | M/Chip 4 Version 1.1 Card Application Specifications for Debit and Credit | 1.1 |
| 9 | | Customer Interface Specification | 2018 |
| 10 | | Single Message system specification 2018 | 2018 |
| 11 | RuPay | RuPay-Online Switching Interface Specification_V1.8.5 | 1.8.5 |
| 12 | | D-PAS Card Specification v1.1 | 1.1 |

## 17. Glossary

| Acronyms | Abbreviation |
|---|---|
| AAC | Application Authorization Cryptogram |
| AC | Application Cryptogram |
| AES | Advanced Encryption Standard |
| ASCII | American Standard Code for Information Interchange |
| ARQC | Authorization Request Cryptogram |
| ARPC | Authorization Response Cryptogram |
| BER | Basic Encoded Rules |
| BCD | Binary Coded Decimal |
| CVR | Card Verification result |
| DES | Data Encryption Standard |
| DKI | Derivation Key Index |
| DPAS | Discover Payment Application Specification |
| EMV | Europay Mastercard Visa |
| EBCDIC | Extended Binary Coded Decimal Interchange Code |
| HSM | Hardware Security Module |
| IMK | Issuer Master Key |
| JCB | Japan Credit Bureau |
| MAC | Message Authentication Code |
| MDK | Master Derivation Key |
| SMI | Secure Messaging for Integrity |
| SMC | Secure Messaging for confidentiality |
| TC | Transaction Certificate |
| TVR | Terminal Verification Result |
| UPI | Union Pay International |
| UDK | Unique Derivation Key |

------- End of Document ------

`