FeliCa Card User's Manual (Unofficial Edition)

KIRISHIKI Yudai[1]

July 24, 2026

[1]Unknown Technologies Inc.

Disclaimer

Every effort has been made to ensure the accuracy of this document. However, please note that, by the nature of reverse engineering, it may contain inaccurate results.

Unless otherwise noted, this document assumes the following.

- Bit order is such that the least significant bit is bit 0.
- Data lengths are given in octets.

Contents

Chapter 1

Introduction

FeliCa is used for many stored-fare (SF) electronic money schemes and identification cards, most notably transit IC cards. Yet the most important part of its specification, the part concerning cryptography, is kept confidential and is not widely known. The security of a cryptographic system is only assured once it has been made public and scrutinised by a broad range of experts. For this reason, this document reveals the hidden specifications of FeliCa. It draws heavily on JIS X 6319-4[1], the FeliCa Card User's Manual Excerpted Edition (hereafter U-MAN)[2], felica-tool[4] and Proxmark3[3]. Readers who want more detail are encouraged to consult these as well.

Chapter 2

What Is FeliCa

FeliCa is a contactless IC card technology developed by Sony Corporation, also known as NFC Type-F. The implementations available on the market are FeliCa Standard and FeliCa Lite-S. This document covers the former.

Chapter 3

Communication Protocol

This chapter describes the communication protocol between a card and a Reader/Writer. The physical layer and the data link layer are omitted because they are fully public; only the application layer is covered.

3.1 Command Packet

The data structure of a command packet is shown below. (Table 3.1)

Table 3.1: Data structure of a command packet

Offset	Length	Item
0x00	0x01	Command Code
0x01	Variable	Command Data

3.1.1 Command Code

A one-byte value that identifies the type of the command.

3.1.2 Command Data

Data that specifies how the command is to be processed. Its format differs from command to command.

3.2 Response Packet

The data structure of a response packet is shown below. (Table 3.2)

Table 3.2: Data structure of a response packet

Offset	Length	Item
0x00	0x01	Response Code
0x01	Variable	Response Data

3.2.1 Response Code

An eight-bit value that identifies the type of the response.

3.2.2 Response Data

Data that specifies the result of processing the command. Its format differs from command to command.

3.3 Command List

An overview of each command, together with its Command Code and Response Code, is shown below. (Table 3.3) In the table, the Command Code is written as CC and the Response Code as RC. Note that commands which are not yet known may exist.

Table 3.3: Command list

Name	CC	RC	Overview
Polling	0x00	0x01	The Reader/Writer captures and identifies a card
Request Service	0x02	0x03	Check the existence of Areas and Services and obtain their Key Versions
Request Response	0x04	0x05	Check the presence and the Mode of a card
Read Without Encryption	0x06	0x07	Read Block Data from a Service whose Service Attribute requires no authentication
Write Without Encryption	0x08	0x09	Write Block Data to a Service whose Service Attribute requires no authentication
Search Service Code	0x0A	0x0B	Obtain Area Codes and Service Codes

Continued on the next page

Table 3.3: Command list (Continued)

Name	CC	RC	Overview
Request System Code	0x0C	0x0D	Obtain the System Codes registered in a card
Request Block Information	0x0E	0x0F	Obtain the number of Blocks allocated to the specified Nodes
Authentication1	0x10	0x11	The Reader/Writer authenticates the card using DES
Authentication2	0x12	0x13	The card authenticates the Reader/Writer using DES
Read	0x14	0x15	Read Block Data with DES from a Service whose Service Attribute requires authentication
Write	0x16	0x17	Write Block Data with DES to a Service whose Service Attribute requires authentication
Request Code List	0x1A	0x1B	Iteratively obtain the list of Nodes under the specified parent Node
Request Block Information Ex	0x1E	0x1F	Obtain the number of Blocks allocated to the specified Nodes and the number of free Blocks
Set Parameter	0x20	0x21	Set card communication parameters (encryption scheme and Node Code size)
Get Container Issue Information	0x22	0x23	Obtain container information such as the format version and the mobile phone model
Get Area Information	0x24	0x25	Obtain information about the specified Area
Get Node Property	0x28	0x29	Obtain Node Property
Get Container Property	0x2E	0x2F	Obtain container properties
Request Service v2	0x32	0x33	Check the existence of Areas and Services and obtain their Key Versions (supports AES)

Continued on the next page

Table 3.3: Command list (Continued)

Name	CC	RC	Overview
Internal Authenticate and Read	0x34	0x35	Perform internal authentication against a Communication-with-MAC-enabled Service and read Block Data
External Authenticate and Write	0x36	0x37	Perform external authentication against a Communication-with-MAC-enabled Service and write Block Data
Get System Status	0x38	0x39	Obtain the configuration state of each System
Request Product Information	0x3A	0x3B	Obtain product information
Request Specification Version	0x3C	0x3D	Obtain the version of the card OS
Reset Mode	0x3E	0x3F	Reset the Mode to Mode0
Authentication1 v2	0x40	0x41	The Reader/Writer authenticates the card using AES
Authentication2 v2	0x42	0x43	The card authenticates the Reader/Writer using AES
Read v2	0x44	0x45	Read Block Data with AES from a Service whose Service Attribute requires authentication
Write v2	0x46	0x47	Write Block Data with AES to a Service whose Service Attribute requires authentication
Delete Key	0x48	0x49	Stop the use of the DES key on a Node that has both an AES key and a DES key
Update Random ID	0x4C	0x4D	Update the random ID (IDr)
Get Container ID	0x70	0x71	Obtain the container ID from a mobile FeliCa device
Set Node Property	0x78	0x79	Set Node Property
Register Issue ID	0x80	0x81	Initialise a System and register its issue ID

Continued on the next page

Table 3.3: Command list (Continued)

Name	CC	RC	Overview
Register Area	0x82	0x83	Register an Area
Register Service	0x84	0x85	Register a Service
Register Issue ID Ex	0x86	0x87	Initialise a System and register its issue ID
Change System Block	0x8E	0x8F	Commit the results of issuance commands

3.4 Manufacture ID and Manufacture Parameter

This section describes the manufacture ID (IDm) and the manufacture parameter (PMm), which are obtained as the response data of the Polling command.

3.4.1 IDm

The IDm is the ID with which a Reader/Writer identifies a card. When a card contains several Systems, each System has a different IDm.

The data structure of the IDm is shown below. (Table 3.4) Note that the upper four bits of the first byte of the manufacturer code indicate the System number within the card.

Table 3.4: Data structure of the IDm

Offset	Length	Item
0x00	0x02	Manufacturer code
0x02	0x06	Card identification number

3.4.2 PMm

The data structure of the PMm is shown below. (Table 3.5) The ROM type and the IC type together are called the IC code.

Table 3.5: Data structure of the PMm

Offset	Length	Item
0x00	0x01	ROM type
0x01	0x01	IC type
0x02	0x06	Maximum response time parameter

Chapter 4

File System

Internally, FeliCa has a hierarchical structure consisting of Systems, Areas, Services and Blocks. A System contains Areas, an Area contains Sub-Areas or Services, and a Service contains Blocks. Any of these may exist more than once on a single card. Of these, Systems, Areas and Services are metadata resembling directories, while Blocks are the regions that hold the actual data. Collectively they are called Nodes. Each Node has a code that represents it. The existence of any Node consumes Blocks.

4.1 System

A System is a logical card unit and sits at the top of the hierarchy. The code that represents a System (as a Node) is 0xFFFF. The following System Definition Information is defined for a System.

- System Code
- Issue ID information
- System Key
- System Key Version
- System property

4.1.1 System Code

A System Code is a two-byte value. Confusingly, the code that represents a System (as a Node) and the System Code are different concepts. Examples of System Codes are shown below. (Table 4.1)

Table 4.1: Examples of System Codes

System Code	Name
0x0003	Japan Railway Cybernetics Area
0x80CD	Free Area
0xFE00	Common Area
0xFE0F	Management Area

4.1.2 Issue ID Information

The Issue ID information consists of the issue ID (IDi) and the Issue Parameter (PMi), each eight bytes long. It can be obtained from the response data once mutual authentication has succeeded with the Authentication2 command or the Authentication2 v2 command. Converting the IDi into a string with a particular algorithm yields the ID number printed at the bottom right of the back of a transit IC card conforming to the Japan Railway Cybernetics standard.

4.1.3 System Key

A System Key is an eight-byte value under DES, and a sixteen-byte value under AES.

4.1.4 System Key Version

A System Key Version is a two-byte value.

4.1.5 System Property

Details unknown.

4.1.6 Switching between Systems

When a card receives a Polling command, or receives a command packet addressed to an IDm other than that of the System currently being operated, the System currently being operated is switched and reset to Mode0. However, when the Authentication1, Authentication1 v2 or Internal Authenticate and Read command succeeds and System switching occurs, the card enters Mode1.

4.2 Area

An Area is contained in a System or in at least one parent Area. The following Area Definition Information is defined for an Area.

- Area Code
- End Service Code
- Number of assigned Blocks
- Area Key
- Area Key Version
- Area property

4.2.1 Area Code

An Area Code is a two-byte value. Bits 6 to 15 give the Area Number and bits 0 to 5 give the Area Attribute (Table 4.2). The Area Code also indicates the logical start point of the Area within the card.

Table 4.2: Area Attributes

Area Attribute	Value
Area in which Sub-Areas can be created	0b000000
Area in which Sub-Areas cannot be created	0b000001

4.2.2 End Service Code

The logical end point of the Area within the card.

4.2.3 Number of Assigned Blocks

The number of Blocks allocated to the Area.

4.2.4 Area Key

An Area Key is an eight-byte value under DES, and a sixteen-byte value under AES.

4.2.5 Area Key Version

An Area Key Version is a two-byte value.

4.2.6 Area Property

Details unknown.

4.2.7 Area 0

A System always contains Area 0, whose range is 0x0000 to 0xFFFE.

4.3 Service

A Service is contained in at least one Area. The following Service Definition Information is defined for a Service.

- Service Code
- Number of assigned Blocks
- Service Key
- Service Key Version
- Service property

4.3.1 Service Code

A Service Code is a two-byte value. Bits 6 to 15 give the Service Number and bits 0 to 5 give the Service Attribute (Table 4.3). The Service Code also indicates the logical position of the Service within the card.

Table 4.3: Service Attributes

Service Attribute	Access control	Value
Random Service	Read/Write Access: authentication required	0b001000
	Read/Write Access: authentication not required	0b001001
	Read Only Access: authentication required	0b001010
	Read Only Access: authentication not required	0b001011
Cyclic Service	Read/Write Access: authentication required	0b001100
	Read/Write Access: authentication not required	0b001101
	Read Only Access: authentication required	0b001110
	Read Only Access: authentication not required	0b001111

Continued on the next page

Table 4.3: Service Attributes (Continued)

Service Attribute	Access control	Value
Purse Service	Direct Access: authentication required	0b010000
	Direct Access: authentication not required	0b010001
	Cashback/Decrement Access: authentication required	0b010010
	Cashback/Decrement Access: authentication not required	0b010011
	Decrement Access: authentication required	0b010100
	Decrement Access: authentication not required	0b010101
	Read Only Access: authentication required	0b010110
	Read Only Access: authentication not required	0b010111

4.3.2 Number of Assigned Blocks

The number of Blocks allocated to the Service.

4.3.3 Service Key

A Service Key is an eight-byte value under DES, and a sixteen-byte value under AES.

4.3.4 Service Key Version

A Service Key Version is a two-byte value.

4.3.5 Service Property

Set only on products equipped with the Value-Limited Purse Service option or the Communication with MAC option.

4.3.6 Overlap Service

Managing the same group of Blocks with more than one Service Code is called overlapping, and a Service that overlaps a Service having the same Service Number within the same System is called an Overlap Service. The

figure showing an example of an Overlap Service, in the ”Overlap Service” section of U-MAN[2], aids understanding.

4.4 Block

A Block is the actual data in the non-volatile memory of a card, divided into sixteen-byte units. Apart from those that hold metadata, a fixed number of Blocks is allocated to each Service, and the Area containing the Service manages the total allocation. The figure showing how Blocks are managed by Area, in the ”Area” section of U-MAN[2], aids understanding.

Chapter 5

Mode

A card has states called Modes, from Mode0 to Mode3. The commands that the card can execute are restricted by the Mode. When power is supplied after a power-off state, the card enters Mode0. Mode1 and above are divided into DES, AES and Communication with MAC; executing the Authentication1, Authentication1 v2 or Internal Authenticate and Read command respectively takes the card to Mode1.

Under DES and AES, executing the Authentication2 or Authentication2 v2 command in Mode1 takes the card to Mode2 of the corresponding scheme. In Mode2 the commands that require mutual authentication (Read, Write, Read v2, Write v2 and so on) can be executed. Executing an issuance command takes the card to Mode3 of the corresponding scheme.

Communication with MAC, on the other hand, has neither Mode2 nor Mode3. The command that can be executed in Mode1 is External Authenticate and Write, and executing it returns the card to Mode0.

Once the card has left Mode0 it no longer accepts the Polling command. However, a Polling command that specifies another System in order to perform Switching between Systems can be executed in any Mode. The Reset Mode command also returns the card to Mode0 from any Mode. The current Mode can be checked with the Request Response command. For further detail, see the ”Mode” section of U-MAN[2].

Chapter 6

Commands

This chapter describes the parameters given to commands and the details of each command. Because of space constraints, the details of some commands are omitted.

6.1 Block List and Block List Element

A Block List is a set of Block List Elements that identify the Service and the Block Number to be accessed. A Block List Element is either two bytes long (Table 6.1) or three bytes long (Table 6.2). Note that the offsets and lengths in these two tables are given in bits.

Table 6.1: Data structure of a Block List Element (two bytes)

Offset	Length	Item
0	1	Length (0b1)
1	3	Access Mode
4	4	Service Code List Order
8	8	Block Number / Key Version

Table 6.2: Data structure of a Block List Element (three bytes)

Offset	Length	Item
0	1	Length (0b0)
1	3	Access Mode
4	4	Service Code List Order

Continued on the next page

Table 6.2: Data structure of a Block List Element (three bytes) (Continued)

Offset	Length	Item
8	16	Block Number / Key Version (little endian)

6.1.1 Access Mode

Specifies how the Node targeted by the Block List Element is accessed. (Table 6.3)

Table 6.3: Access Modes

Access Mode	Value
Other than Cashback Access to a Purse Service	0b000
Cashback access to a Purse Service	0b001
Key change	0b100

6.2 Polling

The Polling command is used by a Reader/Writer to capture and identify a card. The IDm and PMm of the System having the specified System Code can be obtained.

6.2.1 Data Structure of the Command Packet

The data structure of the Polling command packet is shown below. (Table 6.4)

Table 6.4: Data structure of the Polling command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x00
0x01	0x02	System Code (x)	$0x0000 \leq x \leq 0xFFFF$ (0xFF in either byte is a wildcard). Big endian
0x03	0x01	Request Code	0x00: no request, 0x01: System Code request, 0x02: communication performance request

Continued on the next page

Table 6.4: Data structure of the Polling command packet (Continued)

Offset	Length	Item	Value or remarks
0x04	0x01	Time Slot	Specifies the maximum number of Time Slots that may respond. 0x00, 0x01, 0x03, 0x07, 0x0F (for details see the table of Time Slot specifications in the "Polling" section of U-MAN[2])

6.2.2 Data Structure of the Response Packet

The data structure of the Polling response packet is shown below. (Table 6.5)

Table 6.5: Data structure of the Polling response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x01
0x01	0x08	IDm	-
0x09	0x08	PMm	-
0x11	0x02	Request Data	Returned only when the Request Code in the command is other than 0x00 and the specified Request Code is supported by the product

If a Request Code that is not supported is specified, no Request Data is appended to the response packet. Implementations must assume that Request Data may not be returned even when a Request Code is specified.

6.3 Request Service

The Request Service command checks the existence of Areas and Services and obtains their Key Versions.

6.3.1 Data Structure of the Command Packet

The data structure of the Request Service command packet is shown below. (Table 6.6)

Table 6.6: Data structure of the Request Service command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x02
0x01	0x08	IDm	-
0x09	0x01	Number of Node (n)	$0x01 \leq n \leq 0x20$
0x0A	2n	Node Code List	Little endian

6.3.2 Data Structure of the Response Packet

The data structure of the Request Service response packet is shown below. (Table 6.7)

Table 6.7: Data structure of the Request Service response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x03
0x01	0x08	IDm	-
0x09	0x01	Number of Node (n)	$0x01 \leq n \leq 0x20$
0x0A	2n	Node Key Version List	0xFFFF if the Node does not exist. Little endian

6.4 Request Response

The Request Response command checks the presence and the Mode of a card.

6.4.1 Data Structure of the Command Packet

The data structure of the Request Response command packet is shown below. (Table 6.8)

Table 6.8: Data structure of the Request Response command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x04
0x01	0x08	IDm	-

6.4.2 Data Structure of the Response Packet

The data structure of the Request Response response packet is shown below. (Table 6.9)

Table 6.9: Data structure of the Request Response response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x05
0x01	0x08	IDm	-
0x09	0x01	Mode (n)	$0x00 \leq n \leq 0x03$

6.5 Read Without Encryption

The Read Without Encryption command reads Block Data from a Service whose Service Attribute requires no authentication.

6.5.1 Data Structure of the Command Packet

The data structure of the Read Without Encryption command packet is shown below. (Table 6.10)

Table 6.10: Data structure of the Read Without Encryption command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x06
0x01	0x08	IDm	-
0x09	0x01	Number of Service (m)	$0x01 \leq m \leq 0x10$
0x0A	2m	Service Code List	Little endian
-	0x01	Number of Block (n)	$0x01 \leq n \leq$ the maximum number of readable Blocks of the product
-	$2n \leq N \leq 3n$	Block List	-

6.5.2 Data Structure of the Response Packet

The data structure of the Read Without Encryption response packet is shown below. (Table 6.11)

Table 6.11: Data structure of the Read Without Encryption response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x07

Continued on the next page

Table 6.11: Data structure of the Read Without Encryption response packet (Continued)

Offset	Length	Item	Value or remarks
0x01	0x08	IDm	-
0x09	0x01	Status Flag1	0x00: normal
0x0A	0x01	Status Flag2	0x00: normal
0x0B	0x01	Number of Block (n)	0x01 ≤ n ≤ the maximum number of simultaneously readable Blocks of the product
0x0C	16n	Block Data	-

6.6 Write Without Encryption

The Write Without Encryption command writes Block Data to a Service whose Service Attribute requires no authentication.

6.6.1 Data Structure of the Command Packet

The data structure of the Write Without Encryption command packet is shown below. (Table 6.12)

Table 6.12: Data structure of the Write Without Encryption command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x08
0x01	0x08	IDm	-
0x09	0x01	Number of Service (m)	0x01 ≤ m ≤ 0x10
0x0A	2m	Service Code List	Little endian
-	0x01	Number of Block (n)	0x01 ≤ n ≤ the maximum number of simultaneously writable Blocks of the product
-	2n ≤ N ≤ 3n	Block List	-
-	16n	Block Data	-

6.6.2 Data Structure of the Response Packet

The data structure of the Write Without Encryption response packet is shown below. (Table 6.13)

Table 6.13: Data structure of the Write Without Encryption response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x09
0x01	0x08	IDm	-
0x09	0x01	Status Flag1	0x00: normal
0x0A	0x01	Status Flag2	0x00: normal

6.7 Search Service Code

The Search Service Code command obtains Area Codes and Service Codes.

6.7.1 Data Structure of the Command Packet

The data structure of the Search Service Code command packet is shown below. (Table 6.14)

Table 6.14: Data structure of the Search Service Code command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x0A
0x01	0x08	IDm	-
0x09	0x02	Node Index	0x0000 $\leq$ n $\leq$ 0xFFFF. Little endian

6.7.2 Data Structure of the Response Packet

The data structure of the Search Service Code response packet is shown below. (Table 6.15)

Table 6.15: Data structure of the Search Service Code response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x0B
0x01	0x08	IDm	-

Continued on the next page

Table 6.15: Data structure of the Search Service Code response packet (Continued)

Offset	Length	Item	Value or remarks
0x09	0x02 or 0x04	Node Code	Two bytes: a Service Code. Four bytes: an Area Code and an Area End Service Code. Both are little endian. 0xFFFF indicates that no corresponding Node exists

The Nodes in a card can be enumerated by executing this command while incrementing the Node index from 0, treating the point at which the Node code in the response becomes 0xFFFF as the end.

6.8 Request System Code

The Request System Code command obtains the System Codes registered in a card.

6.8.1 Data Structure of the Command Packet

The data structure of the Request System Code command packet is shown below. (Table 6.16)

Table 6.16: Data structure of the Request System Code command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x0C
0x01	0x08	IDm	-

6.8.2 Data Structure of the Response Packet

The data structure of the Request System Code response packet is shown below. (Table 6.17)

Table 6.17: Data structure of the Request System Code response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x0D
0x01	0x08	IDm	-

Continued on the next page

Table 6.17: Data structure of the Request System Code response packet (Continued)

Offset	Length	Item	Value or remarks
0x09	0x01	Number of System Code (n)	-
0x0A	2n	System Code List	Enumerated in order from System 0. Big endian

6.9 Request Block Information

The Request Block Information command obtains the number of Blocks allocated to the specified Nodes. It is supported by mobile FeliCa only.

6.9.1 Data Structure of the Command Packet

The data structure of the Request Block Information command packet is shown below. (Table 6.18)

Table 6.18: Data structure of the Request Block Information command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x0E
0x01	0x08	IDm	-
0x09	0x01	Number of Node Code (n)	-
0x0A	2n	Node Code List	Little endian

6.9.2 Data Structure of the Response Packet

The data structure of the Request Block Information response packet is shown below. (Table 6.19)

Table 6.19: Data structure of the Request Block Information response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x0F
0x01	0x08	IDm	-
0x09	0x01	Number of Block Information (n)	-

Continued on the next page

Table 6.19: Data structure of the Request Block Information response packet (Continued)

Offset	Length	Item	Value or remarks
0x0A	2n	Block Information List	Gives the number of Blocks allocated to each Node

6.10 Authentication1

The Authentication1 command starts the card-side authentication challenge under DES.

6.10.1 Data Structure of the Command Packet

For the data structure of the Authentication1 command packet, see Table 7.1.

6.10.2 Data Structure of the Response Packet

For the data structure of the Authentication1 response packet, see Table 7.3.

6.11 Authentication2

The Authentication2 command completes DES mutual authentication.

6.11.1 Data Structure of the Command Packet

For the data structure of the Authentication2 command packet, see Table 7.2.

6.11.2 Data Structure of the Response Packet

For the data structure of the Authentication2 response packet, see Table 7.4.

6.12 Read

The Read command reads Block Data in a DES session established after mutual authentication.

6.12.1 Data Structure of the Command Packet

For the structure of the internal payload of the Read command before encryption, see Table 7.12.

6.12.2 Data Structure of the Response Packet

For the structure of the internal payload of the Read response before encryption, see Table 7.13.

6.13 Write

The Write command writes Block Data in a DES session established after mutual authentication.

6.13.1 Data Structure of the Command Packet

For the structure of the internal payload of the Write command before encryption, see Table 7.14. For the key change package stored in the Block Data when the Access Mode is 0b100 (key change), see Table 7.16 and the generation algorithm in the same section.

6.13.2 Data Structure of the Response Packet

For the structure of the internal payload of the Write response before encryption, see Table 7.15.

6.14 Get Node Property

The Get Node Property command obtains Node Property. The Node Property of the Value-Limited Purse Service option or of the Communication with MAC option can be obtained. This command is implemented only on some AES card products and AES/DES card products.

6.14.1 Data Structure of the Command Packet

The data structure of the Get Node Property command packet is shown below. (Table 6.20)

Table 6.20: Data structure of the Get Node Property command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x28
0x01	0x08	IDm	-
0x09	0x01	Target	0x00: Value-Limited Purse Service, 0x01: Communication-with-MAC-enabled Service

Continued on the next page

Table 6.20: Data structure of the Get Node Property command packet (Continued)

Offset	Length	Item	Value or remarks
0x0A	0x01	Number of Node (n)	0x01 ≤ n ≤ 0x10
0x0B	2n	Node Code List	Little endian

6.14.2 Data Structure of the Response Packet

The data structure of the Get Node Property response packet is shown below. (Table 6.21)

Table 6.21: Data structure of the Get Node Property response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x29
0x01	0x08	IDm	-
0x09	0x01	Status Flag1	0x00: normal
0x0A	0x01	Status Flag2	0x00: normal
0x0B	0x01	Number of Node (n)	Returned only when Status Flag1 is 0x00
0x0C	mn	Node Property	Returned only when Status Flag1 is 0x00. Enumerated in the order of the Node Code List. m = 10 when the target is 0x00, m = 1 when the target is 0x01

The data structure of the Node Property when 0x00 (Value-Limited Purse Service) is specified as the target is shown below. (Table 6.22)

Table 6.22: Data structure of the Node Property of a Value-Limited Purse Service

Offset	Length	Item	Value or remarks
0x00	0x01	Value-Limited Purse Service flag	0x01: enabled, 0x00: disabled
0x01	0x04	Upper Limit	Little endian, two's complement. All bytes are 0xFF when disabled

Continued on the next page

Table 6.22: Data structure of the Node Property of a Value-Limited Purse Service (Continued)

Offset	Length	Item	Value or remarks
0x05	0x04	Lower Limit	Little endian, two's complement. All bytes are 0xFF when disabled
0x09	0x01	Generation Number	0xFF when disabled

When 0x01 (Communication-with-MAC-enabled Service) is specified as the target, the Node Property is a single byte holding the Communication with MAC flag. A value of 0x01 means enabled and 0x00 means disabled.

6.15 Request Service v2

The Request Service v2 command obtains the Encryption Identifier and Key Version information.

6.15.1 Data Structure of the Command Packet

The data structure of the Request Service v2 command packet is shown below. (Table 6.23)

Table 6.23: Data structure of the Request Service v2 command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x32
0x01	0x08	IDm	-
0x09	0x01	Number of Node (n)	$0x01 \leq n \leq 0x20$
0x0A	2n	Node Code List	Little endian

6.15.2 Data Structure of the Response Packet

The data structure of the Request Service v2 response packet is shown below. (Table 6.24)

Table 6.24: Data structure of the Request Service v2 response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x33
0x01	0x08	IDm	-
0x09	0x01	Status Flag1	0x00: normal
0x0A	0x01	Status Flag2	0x00: normal
0x0B	0x01	Encryption Identifier	Returned only when Status Flag1 is 0x00
0x0C	0x01	Number of Node (n)	Returned only when Status Flag1 is 0x00. $0x01 \leq n \leq 0x20$
0x0D	2n or 4n	Node Key Version List	Returned only when Status Flag1 is 0x00. 4n bytes when the Encryption Identifier is 0x41 or 0x43 (AES first, DES second). 2n bytes otherwise

6.16 Get System Status

The Get System Status command obtains the configuration state of each System.

6.16.1 Data Structure of the Command Packet

The data structure of the Get System Status command packet is shown below. (Table 6.25)

Table 6.25: Data structure of the Get System Status command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x38
0x01	0x08	IDm	-
0x09	0x02	Reserved	0x0000

6.16.2 Data Structure of the Response Packet

The data structure of the Get System Status response packet is shown below. (Table 6.26)

Table 6.26: Data structure of the Get System Status response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x39
0x01	0x08	IDm	-
0x09	0x01	Status Flag1	0x00: normal
0x0A	0x01	Status Flag2	0x00: normal
0x0B	0x01	Flag	Details unknown
0x0C	0x01	Data length (n)	-
0x0D	n	Data	Details unknown

The meaning of the flag and of the data is unknown.

6.17 Request Specification Version

The Request Specification Version command obtains the version of the card OS.

6.17.1 Data Structure of the Command Packet

The data structure of the Request Specification Version command packet is shown below. (Table 6.27)

Table 6.27: Data structure of the Request Specification Version command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x3C
0x01	0x08	IDm	-
0x09	0x02	Reserved	0x0000

6.17.2 Data Structure of the Response Packet

The data structure of the Request Specification Version response packet is shown below. (Table 6.28)

Table 6.28: Data structure of the Request Specification Version response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x3D
0x01	0x08	IDm	-
0x09	0x01	Status Flag1	0x00: normal
0x0A	0x01	Status Flag2	0x00: normal
0x0B	0x01	Format Version	Fixed at 0x00. Returned only when Status Flag1 is 0x00
0x0C	0x02	Basic Version	Returned only when Status Flag1 is 0x00. Little endian
0x0E	0x01	Number of Option (m)	Returned only when Status Flag1 is 0x00
0x0F	2m	Option Version List	Returned only when Status Flag1 is 0x00. Little endian

The assignment of each element of the Option Version List is shown below. (Table 6.29)

Table 6.29: Assignment of the Option Version List

Position	Option
D0-D1	DES option
D2-D3	0x00, 0x00 (mobile FeliCa may set a version of its own here)
D4-D5	Extended Overlap option
D6-D7	Value-Limited Purse Service option
D8-D9	Communication with MAC option
D10-D11	Random ID option

The Basic Version and each option version are two-byte values. The upper four bits are fixed at 0b1000 and the remaining twelve bits give the version value in BCD. For example, version 5.0.0 is 0x500. Which options are present and what their version values are differ from product to product.

6.18 Reset Mode

The Reset Mode command resets the Mode to Mode0.

6.18.1 Data Structure of the Command Packet

The data structure of the Reset Mode command packet is shown below. (Table 6.30)

Table 6.30: Data structure of the Reset Mode command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x3E
0x01	0x08	IDm	-
0x09	0x02	Reserved	0x0000

6.18.2 Data Structure of the Response Packet

The data structure of the Reset Mode response packet is shown below. (Table 6.31)

Table 6.31: Data structure of the Reset Mode response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x3F
0x01	0x08	IDm	-
0x09	0x01	Status Flag1	0x00: normal
0x0A	0x01	Status Flag2	0x00: normal

6.19 Authentication1 v2

The Authentication1 v2 command starts the card-side authentication challenge under AES.

6.19.1 Data Structure of the Command Packet

For the data structure of the Authentication1 v2 command packet, see Table 7.6.

6.19.2 Data Structure of the Response Packet

For the data structure of the Authentication1 v2 response packet, see Table 7.8.

6.20 Authentication2 v2

The Authentication2 v2 command completes AES mutual authentication.

6.20.1 Data Structure of the Command Packet

For the data structure of the Authentication2 v2 command packet, see Table 7.7.

6.20.2 Data Structure of the Response Packet

For the data structure of the Authentication2 v2 response packet, see Table 7.9.

6.21 Read v2

The Read v2 command reads Block Data in an AES session established after mutual authentication.

6.21.1 Data Structure of the Command Packet

For the structure of the internal payload of the Read v2 command before encryption, see Table 7.12.

6.21.2 Data Structure of the Response Packet

For the structure of the internal payload of the Read v2 response before encryption, see Table 7.13.

6.22 Write v2

The Write v2 command writes Block Data in an AES session established after mutual authentication.

6.22.1 Data Structure of the Command Packet

For the structure of the internal payload of the Write v2 command before encryption, see Table 7.14. Key change (Access Mode 0b100) is not valid for Write v2.

6.22.2 Data Structure of the Response Packet

For the structure of the internal payload of the Write v2 response before encryption, see Table 7.15.

6.23 Register Issue ID

The Register Issue ID command is an issuance secure command that registers information related to the issue ID.

6.23.1 Data Structure of the Command Packet

For the structure of the internal payload of the Register Issue ID command before encryption, see Table 7.17.

6.23.2 Data Structure of the Response Packet

In addition to Status Flag1 and Status Flag2, the internal payload of the Register Issue ID response before encryption returns the number of remaining Blocks (two bytes) on success only.

6.24 Register Area

The Register Area command is an issuance secure command that registers an Area.

6.24.1 Data Structure of the Command Packet

For the structure of the internal payload of the Register Area command before encryption, see Table 7.18.

6.24.2 Data Structure of the Response Packet

The internal payload of the Register Area response before encryption is the two bytes of Status Flag1 and Status Flag2.

6.25 Register Service

The Register Service command is an issuance secure command that registers a Service.

6.25.1 Data Structure of the Command Packet

For the structure of the internal payload of the Register Service command before encryption, see Table 7.19.

6.25.2 Data Structure of the Response Packet

In addition to Status Flag1 and Status Flag2, the internal payload of the Register Service response before encryption returns the number of remaining Blocks (two bytes) on success only.

6.26 Change System Block

The Change System Block command is an issuance secure command that commits the results of issuance commands.

6.26.1 Data Structure of the Command Packet

For the structure of the internal payload of the Change System Block command before encryption, see Table 7.20.

6.26.2 Data Structure of the Response Packet

The internal payload of the Change System Block response before encryption is the two bytes of Status Flag1 and Status Flag2.

6.27 Status Flag

Status Flag is the value that represents the result of processing a command. It consists of Status Flag1 (1 byte) and Status Flag2 (1 byte).

6.27.1 Status Flag1

Status Flag1 indicates whether the command succeeded, and the Block position or Service position at which an error occurred. (Table 6.32)

Table 6.32: Values and meanings of Status Flag1

Value	Meaning
0x00	Normal completion
0xFF	An error in a command whose command packet contains no list, or an error that does not depend on a list
Other	The position in the list at which the error occurred

There are two ways in which the position of the error is represented, depending on the product, and the value of Status Flag1 alone does not tell which one is in use. (Table 6.33) For example, when an error occurs at the Node specified tenth in the Block List, the ordinal form returns 0x0A whereas the bitmap form returns 0x02.

Table 6.33: Representations of the error position in Status Flag1

Form	Content
Ordinal form	The position (starting from 1) in the Block List or the Service Code List at which the error occurred is set as is
Bitmap form	Each bit corresponds to a position in the list, where 1 means an error and 0 means no error. Bit 0 corresponds to the first or the ninth entry, bit 1 to the second or the tenth, and so on up to bit 6, which corresponds to the seventh or the fifteenth; bit 7 corresponds to the eighth

6.27.2 Status Flag2

Status Flag2 indicates the detail of the error. (Table 6.34) There are also card-specific values, which are not common to all products. (Table 6.35)

Table 6.34: Values and meanings of Status Flag2

Value	Meaning
0x00	Normal completion
0x01	On a purse decrement the result of the calculation would be less than zero, or on a cashback the result of the calculation would exceed four bytes
0x02	On a purse cashback the specified data exceeds the value of the cashback data
0x03	On a write to a Value-Limited Purse Service the purse data does not fall between the Upper Limit and the Lower Limit
0x70	Memory error (fatal error)
0x71	The number of memory rewrites has exceeded the limit (a warning; the write is still performed)

Because 0x71 is a warning, the write itself is performed. The limit on the number of rewrites differs from product to product, and in this case Status Flag1 is 0x00 on some products and 0xFF on others.

Table 6.35: Values and meanings of Status Flag2 (card-specific)

Value	Meaning
0xA1	The number of Services or the number of Nodes specified in the command is outside the defined range

Continued on the next page

Table 6.35: Values and meanings of Status Flag2 (card-specific) (Continued)

Value	Meaning
0xA2	The number of Blocks specified in the command is outside the range defined for the product
0xA3	The Service Code List Order specified in a Block List Element is outside the range of the number of Services specified in the command or specified at mutual authentication
0xA4	The Area Attribute of the Area Code or the Service Attribute of the Service Code specified in the command is wrong
0xA5	The Area or the Service specified in the command cannot be accessed, or a parameter specified in the command does not satisfy the success conditions
0xA6	The access target identified by the Service Code List Order specified in a Block List Element, or the Node specified in the Node Code List, does not exist
0xA7	The Access Mode specified in a Block List Element is wrong
0xA8	The Block Number specified in a Block List Element exceeds the number of Blocks allocated to the Service
0xA9	A write failed during an issuance command
0xAA	Key change failed
0xAB	The package parity or the package MAC is invalid during an issuance command
0xAC	A parameter is invalid during an issuance command
0xAD	The Service to be registered already exists
0xAE	The System Code is invalid during an issuance command
0xAF	The number of Blocks written simultaneously to the cyclic Service specified in the command exceeds the number of Blocks allocated to the Service
0xC0	The package identifier is invalid during an issuance command
0xC1	The parameters inside and outside the package do not match during an issuance command
0xC2	The issuance command has been disabled
0xC3	The attribute of the Node specified in the command is wrong

Chapter 7

Mutual Authentication and Secure Messaging Algorithms

7.1 Authentication1 / Authentication2 (DES)

7.1.1 Data Structure of the Command Packets

The data structure of the Authentication1 command packet is shown below. (Table 7.1)

Table 7.1: Data structure of the Authentication1 command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x10
0x01	0x08	IDm	-
0x09	0x01	Number of Areas (a)	-
0x0A	2a	Area Code list	Little endian
-	0x01	Number of Service (s)	-
-	2s	Service Code List	Little endian
-	0x08	Challenge 1A	8 bytes

The data structure of the Authentication2 command packet is shown below. (Table 7.2)

Table 7.2: Data structure of the Authentication2 command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x12
0x01	0x08	IDm	-
0x09	0x08	Challenge 2B	8 bytes

7.1.2 Data Structure of the Response Packets

The data structure of the Authentication1 response packet is shown below. (Table 7.3)

Table 7.3: Data structure of the Authentication1 response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x11
0x01	0x08	IDm	-
0x09	0x08	Challenge 1B	8 bytes
0x11	0x08	Challenge 2A	8 bytes

The data structure of the Authentication2 response packet is shown below. (Table 7.4) This response contains no IDm, and everything after the Response Code is encrypted.

Table 7.4: Data structure of the Authentication2 response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x13
0x01	0x20	Encrypted payload	Encrypted with DES-CBC (IV = 0) keyed with the random number R_2. For the structure after decryption, see Table 7.5

The data structure of the payload after decryption is shown below. (Table 7.5)

Table 7.5: Data structure of the decrypted payload of the Authentication2 response

Offset	Length	Item	Value or remarks
0x00	0x02	Transaction number (TN)	Little endian
0x02	0x06	Transaction ID (TID)	Matches the last six bytes of the random number R_1
0x08	0x08	Issue ID (IDi)	-
0x10	0x08	Issue Parameter (PMi)	-
0x18	0x08	MAC	Computed over the preceding 24 bytes

7.1.3 DES Mutual Authentication Algorithm

The principal parameters used in DES mutual authentication are derived as follows.

- Let the System Key be K_{sys}, the sequence of Area Keys be $K_{\text{Area},i}$ and the sequence of Service Keys be $K_{\text{srv},j}$. Then

$$G_0 = K_{\text{sys}}, \quad G_i = \text{DES}_{K_{\text{Area},i}}(G_{i-1})$$

$$K_{\text{group}} = G_m, \quad U_0 = K_{\text{group}}, \quad U_j = \text{DES}_{K_{\text{srv},j}}(U_{j-1}), \quad K_{\text{user}} = U_n$$

  give the group Service Key K_{group} and the user Service Key K_{user}.

- Treating the IDm as an eight-byte vector, compute

$$L = K_{\text{group}} \oplus IDm$$

$$\alpha = \text{DES}_L(K_{\text{user}}), \quad \beta = \text{DES}_\alpha(L)$$

- Random numbers are exchanged with two-key 3DES, in the key order K_1, K_2, K_1. Letting R_1 and R_2 be the random numbers,

$$C_{1A} = \text{3DES}_{\alpha,L}(R_1), \quad C_{1B} = \text{3DES}_{L,\beta}(R_1)$$

$$C_{2A} = \text{3DES}_{L,\beta}(R_2), \quad C_{2B} = \text{3DES}_{\alpha,L}(R_2)$$

- The plaintext of Authentication2 is

$$\text{TN(2)} \parallel \text{TID(6)} \parallel \text{IDi(8)} \parallel \text{PMi(8)}$$

  Here the TID is the last six bytes of R_1. The data used as the Issue ID information is the last sixteen bytes, IDi(8) ‖ PMi(8).

7.2 Authentication1 v2 / Authentication2 v2 (AES)

7.2.1 Data Structure of the Command Packets

The data structure of the Authentication1 v2 command packet is shown below. (Table 7.6)

Table 7.6: Data structure of the Authentication1 v2 command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x40
0x01	0x08	IDm	-
0x09	0x01	Operation Parameter	-
0x0A	0x01	Number of Node (n)	-
0x0B	2n	Node Code List	Little endian
-	0x10	Challenge 1A	16 bytes

The data structure of the Authentication2 v2 command packet is shown below. (Table 7.7)

Table 7.7: Data structure of the Authentication2 v2 command packet

Offset	Length	Item	Value or remarks
0x00	0x01	Command Code	0x42
0x01	0x08	IDm	-
0x09	0x10	Challenge 2B	16 bytes

7.2.2 Data Structure of the Response Packets

The data structure of the Authentication1 v2 response packet is shown below. (Table 7.8)

Table 7.8: Data structure of the Authentication1 v2 response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x41
0x01	0x08	IDm	-
0x09	0x10	Challenge 1B	16 bytes

Continued on the next page

Table 7.8: Data structure of the Authentication1 v2 response packet (Continued)

Offset	Length	Item	Value or remarks
0x19	0x10	Challenge 2A	16 bytes
0x29	0x04	Challenge 3C	4 bytes

The data structure of the Authentication2 v2 response packet is shown below. (Table 7.9) This response contains no IDm either. Only the transaction number is in cleartext; the payload and the MAC are encrypted.

Table 7.9: Data structure of the Authentication2 v2 response packet

Offset	Length	Item	Value or remarks
0x00	0x01	Response Code	0x43
0x01	0x02	Transaction number (TN)	Little endian. Not encrypted
0x03	0x10	Encrypted payload	Encrypted with AES-128-OFB. For the structure after decryption, see Table 7.10
0x13	0x08	Encrypted MAC	Encrypted with the later part of the same OFB stream

The data structure of the payload after decryption is shown below. (Table 7.10)

Table 7.10: Data structure of the decrypted payload of the Authentication2 v2 response

Offset	Length	Item	Value or remarks
0x00	0x08	Issue ID (IDi)	-
0x08	0x08	Issue Parameter (PMi)	-

7.2.3 AES Mutual Authentication Algorithm

The principal parameters used in AES mutual authentication are derived as follows.

- Let the sixteen-byte constant be I_0 = (01 01 00 $\cdots$ 00 80) and the

sequence of Node Keys be $K_{\text{Node},j}$. Then

$$S_0 = I_0, \quad S_j = \text{AES}_{K_{\text{Node},j}}(S_{j-1}), \quad K_{\text{group}} = S_n$$

gives the group key K_{group}.

- With the group key K_{group} and the individualisation code K_{ind}, compute

$$H = K_{\text{group}} \oplus K_{\text{ind}}$$

- Define the sixteen-byte context Block $B(\text{prefix}, IDm)$ as

$$B[0..1] = \text{prefix},\ B[2..5] = 0,\ B[6..13] = IDm,\ B[14..15] = 0x01\ 0x00$$

and obtain

$$\alpha = \text{AES}_H(B([0x01, 0x02], IDm))$$
$$\beta = \text{AES}_H(B([0x02, 0x02], IDm))$$

- From challenge 3C (four bytes), form

$$\beta' = \beta \oplus (C_{3C} \parallel 0x00^{12})$$

and generate and verify the authentication values with

$$C_{1A} = \text{AES}_\alpha(R_1), \quad C_{1B} = \text{AES}_{\beta'}(R_1)$$

$$C_{2A} = \text{AES}_{\beta'}(R_2), \quad C_{2B} = \text{AES}_\alpha(R_2)$$

- The TID used in secure messaging is derived from R_1 as

$$\text{TID} = R_1[2..7]$$

(six bytes, from the third to the eighth byte, counting the first byte as 0).

- The decrypted payload of the Authentication2 v2 response is

$$\text{IDi(8)} \parallel \text{PMi(8)}$$

- After Authentication2 v2 succeeds, the session keys are derived from R_2.

$$K_{\text{enc}} = \text{AES}_{R_2}(01\ 03\ 00 \cdots 00\ 01\ 00)$$
$$K_{\text{mac}} = \text{AES}_{R_2}(02\ 03\ 00 \cdots 00\ 01\ 00)$$

7.3 Secure Read/Write Command Format

Read, Write, Read v2 and Write v2 carry internal payloads of identical form inside the encrypted secure frame. The only difference is the encryption scheme.

Table 7.11: Differences between the four secure Read/Write commands

Command	Scheme	Payload structure
Read	DES	Common to the Read family (Table 7.12 and Table 7.13)
Read v2	AES	Common to the Read family (Table 7.12 and Table 7.13)
Write	DES	Common to the Write family (Table 7.14 and Table 7.15)
Write v2	AES	Common to the Write family (Table 7.14 and Table 7.15)

Table 7.12: Data structure of the internal payload of the Read family commands (Read/Read v2)

Offset	Length	Item	Value or remarks
0x00	0x01	Number of Block (n)	0x01 $\leq$ n $\leq$ the maximum number of simultaneously readable Blocks of the product
0x01	2n to 3n	Block List	Two-byte or three-byte form of the Block List Element

Table 7.13: Data structure of the internal payload of the Read family responses (Read/Read v2)

Offset	Length	Item	Value or remarks
0x00	0x01	Status Flag1	0x00: normal
0x01	0x01	Status Flag2	0x00: normal
0x02	0x01	Number of Block (n)	Returned only when Status Flag1 is 0x00

Continued on the next page

Table 7.13: Data structure of the internal payload of the Read family responses (Read/Read v2) (Continued)

Offset	Length	Item	Value or remarks
0x03	16n	Block Data	Returned only when Status Flag1 is 0x00

Table 7.14: Data structure of the internal payload of the Write family commands (Write/Write v2)

Offset	Length	Item	Value or remarks
0x00	0x01	Number of Block (n)	0x01 ≤ n ≤ the maximum number of simultaneously writable Blocks of the product
0x01	2n to 3n	Block List	Two-byte or three-byte form of the Block List Element
Variable	16n	Block Data	16 bytes × n

Table 7.15: Data structure of the internal payload of the Write family responses (Write/Write v2)

Offset	Length	Item	Value or remarks
0x00	0x01	Status Flag1	0x00: normal
0x01	0x01	Status Flag2	0x00: normal

7.3.1 Generating and Using the DES Key Change Package

To perform a DES key change with the Write command, set the Access Mode of the Block List Element to 0b100 (key change) and store the sixteen-byte key change package in the Block Data.

Table 7.16: Data structure of the key change package (16 bytes)

Offset	Length	Item	Value or remarks
0x00	0x08	Parameter1	Cipher Block containing the new Key Version information

Continued on the next page

Table 7.16: Data structure of the key change package (16 bytes) (Continued)

Offset	Length	Item	Value or remarks
0x08	0x08	Parameter2	Cipher Block containing the new key itself

Let the parent key be K_{parent}, the old key K_{old}, the new key K_{new} and the new Key Version v. First build the eight-byte version Block V as

$$V = 00\ 00\ 00\ 00\ 00\ 00\ v_{\text{LE},0}\ v_{\text{LE},1}$$

then compute the following.

$$P_1 = \text{DES}_{K_{\text{parent}}}\Big(\text{DES}_{K_{\text{old}}}\big(\text{DES}_{K_{\text{new}}}(V)\big)\Big)$$

$$P_2 = \text{DES}_{K_{\text{parent}}}\Big(\text{DES}_{K_{\text{old}}}(K_{\text{new}})\Big)$$

$$\text{KeyChangePackage} = P_1 \parallel P_2$$

In use, the following pair is passed to the Write command for each key to be changed.

- Block List Element: Access Mode = 0b100, Block Number / Key Version = v, Service Code List Order = the target Service
- Block Data (16 bytes): the KeyChangePackage above

When changing several keys at once, arrange the Block List and the Block Data so that their orders match. On failure, Status Flag2 may be, for example, 0xAA (key change failed) or 0xAB (invalid package MAC).

7.3.2 DES Secure Messaging

In a DES session, the payload before encryption is

$$P = \text{TN}(2, \text{LE}) \parallel \text{TID}(6) \parallel \text{CommandPayload}$$

This is padded to an eight-byte boundary by any method to give P', a MAC is appended, and the result is encrypted with DES-CBC (IV = 0).

The MAC is computed as follows.

$$M_0 = [\text{Len},\ \text{Code},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]$$

$$M_i = \text{DES}_{B_i}(M_{i-1})$$

Here B_i is the sequence of eight-byte Blocks of P' and $\text{Len} = 2 + |P'| + 8$. The final value M_n is appended as the MAC (eight bytes).

The response is decrypted, its MAC is verified, the trailing eight-byte MAC is removed and then the padding is removed. The TID must also match and the TN must increase monotonically.

7.3.3 AES-128 Secure Messaging

In an AES session, the transmitted data is

$$\mathrm{TN}(2, \mathrm{LE}) \parallel \mathrm{EncPayload} \parallel \mathrm{EncMAC}(8)$$

The payload itself is encrypted with OFB, and the MAC is encrypted with the later part of the same OFB stream.

The initialisation vector IV (sixteen bytes) is built as follows.

$$IV[0] = 0x01,\ IV[1] = \mathrm{FrameLen},\ IV[2] = \mathrm{Code},$$

$$IV[3..4] = \mathrm{TN},\ IV[5..10] = \mathrm{TID},$$

$$IV[11..13] = \mathrm{C_{3C}}[1..3],\ IV[14..15] = 0$$

The MAC plaintext is computed with AES-CMAC, of which the first eight bytes are used.

$$B_0[0] = 0x19,\ B_0[1..13] = IV[1..13],\ B_0[14..15] = |Payload|_{\mathrm{BE16}}$$

$$MAC = \mathrm{CMAC}_{K_{\mathrm{mac}}}(B_0 \parallel Payload)[0..7]$$

During OFB encryption, if the payload length is not a multiple of sixteen, the keystream is consumed up to the next sixteen-byte boundary before the MAC is encrypted.

7.4 DES Issuance Secure Command Format

Register Issue ID, Register Area, Register Service and Change System Block are sent and received as DES secure commands. The structures of their internal payloads before encryption are shown below.

Table 7.17: Data structure of the internal payload of the Register Issue ID command

Offset	Length	Item	Value or remarks
0x00	0x08	Issue ID	-
0x08	0x08	Issue Parameter	-
0x10	Variable	Package	Issuance package generated with DES

Table 7.18: Data structure of the internal payload of the Register Area command

Offset	Length	Item	Value or remarks
0x00	0x02	Area Code	Little endian
0x02	Variable	Package	Issuance package generated with DES

Table 7.19: Data structure of the internal payload of the Register Service command

Offset	Length	Item	Value or remarks
0x00	0x02	Service Code	Little endian
0x02	Variable	Package	Issuance package generated with DES

Table 7.20: Data structure of the internal payload of the Change System Block command

Offset	Length	Item	Value or remarks
0x00	0x00	None	No payload

The data structures of the internal payloads of the responses are shown below.

Table 7.21: Data structure of the internal payload of the Register Issue ID response

Offset	Length	Item	Value or remarks
0x00	0x01	Status Flag1	0x00: normal
0x01	0x01	Status Flag2	0x00: normal
0x02	0x02	Number of remaining Blocks	Returned only when Status Flag1 is 0x00. Little endian

Table 7.22: Data structure of the internal payload of the Register Area response

Offset	Length	Item	Value or remarks
0x00	0x01	Status Flag1	0x00: normal
0x01	0x01	Status Flag2	0x00: normal

Table 7.23: Data structure of the internal payload of the Register Service response

Offset	Length	Item	Value or remarks
0x00	0x01	Status Flag1	0x00: normal
0x01	0x01	Status Flag2	0x00: normal
0x02	0x02	Number of remaining Blocks	Returned only when Status Flag1 is 0x00. Little endian

Table 7.24: Data structure of the internal payload of the Change System Block response

Offset	Length	Item	Value or remarks
0x00	0x01	Status Flag1	0x00: normal
0x01	0x01	Status Flag2	0x00: normal

The plaintext of an issuance package (before encryption) is sixteen bytes long. Its format for each command is as follows.

Table 7.25: Data structure of the issuance package plaintext for Register Issue ID (16 bytes)

Offset	Length	Item	Value or remarks
0x00	0x02	System Code	Big endian
0x02	0x02	Area 0 Key Version	Little endian
0x04	0x08	Area 0 Key	-
0x0C	0x04	Reserved	0x00000000

Table 7.26: Data structure of the issuance package plaintext for Register Area (16 bytes)

Offset	Length	Item	Value or remarks
0x00	0x02	Service Code start	Little endian
0x02	0x02	Service Code end	Little endian
0x04	0x02	Size	Little endian
0x06	0x02	Key Version	Little endian
0x08	0x08	Area Key	-

Table 7.27: Data structure of the issuance package plaintext for Register Service (16 bytes)

Offset	Length	Item	Value or remarks
0x00	0x02	Service Code	Little endian
0x02	0x02	Reserved	0x0000
0x04	0x02	Size	Little endian
0x06	0x02	Key Version	Little endian
0x08	0x08	Service Key	-

7.4.1 Issuance Package Generation Algorithm

Let the package key be K_{pkg} and the plaintext package be P (where $|P|$ must be a non-zero multiple of eight).

- Build the MAC generation key as

$$K_{\text{macpkg}} = K_{\text{pkg}} \oplus 0xFF \ldots FF$$

- Compute

$$C = \text{DES-CBC}_{IV=0,K_{\text{macpkg}}}(P)$$

  and take the last eight bytes as MAC_{pkg}.

- Compute

$$Package = \text{DES-CBC}_{IV=0,K_{\text{pkg}}}(P \parallel MAC_{\text{pkg}})$$

  and store this ciphertext in the package field of the command.

References

[1] Japanese Standards Association. *JIS X 6319-4:2016 Specification of implementation for integrated circuit(s) cards – Part 4: High speed proximity cards*. 2016.

[2] Sony Corporation. *FeliCa Card User's Manual Excerpted Edition*. Version 2.31. 2026. URL: https://www.sony.co.jp/en/Products/felica/business/tech-support/data/card_usersmanual_2.3e.pdf (visited on 07/23/2026).

[3] C. Herrmann et al. *Proxmark3 – Iceman repo*. https://github.com/RfidResearchGroup/proxmark3.

[4] kormax. *GitHub - kormax/felica-tool: Application for analyzing characteristics of FeliCa cards*. 2025. URL: https://github.com/kormax/felica-tool (visited on 05/28/2025).