ProxRunner Bluetooth RFID scanner

Reference manual

PMA82TP revision AA
03/12/2008

TABLE OF CONTENT

1. INTRODUCTION

This document provides detailed technical information for use of the SpringCard ProxRunner

1.1. AUDIENCE

This reference manual assumes that the reader has expert knowledge of computer configuration and usage. It is designed to be used by system integrators.

1.2. PRODUCT BRIEF

a. Abstract

Developed with **Baracoda Traceability,** a worldwide leader in barcode solutions, ProxRunner Bluetooth RFID scanner is the easiest product for mobile contactless operation.

Thanks to its Bluetooth connection, ProxRunner is integrated easily in PC, PocketPC or SmartPhone based solutions. Lightweight and ruggeddized, the product is ideal for mobile operation even in unfriendly environments.

It reads serial number or data from any standard ISO/IEC 14443 contactless card, including popular NXP MIFARE and DESFire families, and also ISO/IEC 15693 vicinity tags used in RFID systems.

b. Typical applications

This reader is dedicated for loyalty, user identification and tracking and aims to replace barcode scanners where RFID labels may be used instead of barcodes : library or book stores, item management, ….

c. Output modes

This reader uses **Baracoda Manager** to connect to the host (PC, PocketPC, SmartPhone or blackberries) and the data coming from reader is seen as keyboard emulation.

Moreover you can integrate the data collected directly in a software with the SDK included in **Baracoda Manager**.

1.3. RELATED DOCUMENTS

You'll find any details regarding hardware and physical characteristics of each reader in the corresponding datasheet.

Datasheet	Covered products

2. CONFIGURATION DATA

There are two families of data :

- Global settings,
- Card Processing Templates.

Global settings specify output format and timings.

Card Processing Templates specify which kind of cards shall be read (ISO/IEC 14443, Mifare, Desfire, T=CL), how they must be read (serial number, data in file, …), and how the operation is secured (Mifare authentication, Desfire 3-DES secure session, …).

As for Card Processing Templates, ProxRunner is 100% compliant with IWM-K632. This allows using the same card(s) with access control readers and computer-based solutions really easily.

ProxRunner can run 1 to 4 Card Processing Template simultaneously (+ 1 for Master Cards). This means that 4 different kinds of cards can coexist on a single site and can be read by a single ProxRunner.

a. Configuration tags

Each configuration data is recognized by its "tag" and its length. The tag is a one-byte value, that uniquely identify the data.

The list of available tags, and their meaning, is the purpose of this chapter.

Unless specified, each configuration data is exactly one byte (8 bits) long.

b. Non-volatile memory endurance

ProxRunner configuration data are stored in reader's non-volatile memory (flash). They can be changed more than 100 000 times.

2.2. GLOBAL SETTINGS

The following tables enumerate all the data made available when configuring the reader.

2.2.1. General options

Name	Tag	Description	Size
OPT	$_h$60	General options. See table **a** below.	1

a. General options bits

Bit	Value	Meaning
7 – 6		*RFU (set to 00)*
5 – 4		***Anti-collision model :***
	00	*RFU*
	01	*RFU*
	10	When 2 cards are in the field, process the 1st and ignore the 2nd
	11	When 2 cards are in the field, ignore both
3 – 2		***Master Card :***
	00	Master Cards are disabled[1]
	01	*RFU*
	10	*RFU*
	11	Master Cards are enabled all the time
1 – 0		*RFU (set to 00)*

Default value : $_b$00101100

(Master Cards are enabled all the time)

[1] Configuration settings are permanently locked, use this with care !

2.3. CARD PROCESSING TEMPLATES

Each Card Processing Template is configured through a set of 16 tags, from $_{h}$t0 to $_{h}$tF where 't' is the template group ($_{h}1 \le t \le {}_{h}4$).

2.3.1. *Card lookup list*

Name	Tag	Description	Size
LKL	$_{h}$t0	Card lookup list of the template. See table **a** below.	1

a. *Available values for LKL*

Value	Card(s) accepted by the template	Processing template	§
$_{h}$01	ISO/IEC 14443 type A (layer 3)	**ID only**	2.4
$_{h}$02	ISO/IEC 14443 type B (layer 3)		
$_{h}$03	ISO/IEC 14443 A&B (layer 3)		
$_{h}$04	ISO/IEC 15693		
$_{h}$07	ISO/IEC 14443 A&B and ISO/IEC 15693		
$_{h}$08	NXP ICODE1		
$_{h}$0C	NXP ICODE1 and ISO/IEC 15693		
$_{h}$0F	All of the above		
$_{h}$11	ISO/IEC 14443 type A (layer 4 / T=CL)	**7816-4**	2.8
$_{h}$12	ISO/IEC 14443 type B (layer 4 / T=CL)		
$_{h}$13	ISO/IEC 14443 A&B (layer 4 / T=CL)		
$_{h}$22	ST MicroElectronics SR family	**ID only**	2.4
$_{h}$23	ASK CTS256B and CTS512B		
$_{h}$24	Inside Contactless PicoTAG[2]		
$_{h}$61	NXP Mifare Classic 1k & 4k	**Mifare**	2.5
$_{h}$62	NXP Mifare UltraLight	**Mifare UltraLight**	2.6
$_{h}$71	NXP Desfire 4k	**Desfire**	2.7
$_{h}$72	Calypso (Innovatron protocol)	**ID only or 7816-4**	2.9

Other values are *RFU*

The LKL tag is mandatory to enable a template group. If not found, the template group is empty.

[2] Also HID iClass

2.3.2. *Summary of other tags in templates*

Depending of the card lookup list (LKL tag), a specific list of tags controls the behaviour of the Processing Template.

The table below summarize this.

Tag	ID only	Mifare UL	Mifare	Desfire	7816-4	Calypso
$_h$t1	Output format	Output format	Output format	Output format	Output format	Output format
$_h$t2	Output prefix	Output prefix	Output prefix	Output prefix	Output prefix	Output prefix
$_h$t3	Offset	Location of data	Location of data	Location of data	Location of data	Location of data
$_h$t4				T=CL options	T=CL options	C. options
$_h$t5			Auth. method & key	Auth. method & key	1st APDU	1st APDU
$_h$t6			Sign. method & key	Sign. method & key	2nd APDU	2nd APDU
$_h$t7					3rd APDU	3rd APDU

Grey items are *RFU* and must be kept empty.

2.3.3. *Important notice regarding template-ordering*

Be careful that the 4 templates are processed one after the other. The loop is ended after the first successful match.

If a card matches two (or more) templates, it will be handled only by the first one.

Suppose you want to accept both a specific kind of 14443-B T=CL cards, with advanced file reading, and another kind of wired-logic 14443-B cards, where only the ID is significant. You must put the T=CL template *before* the ID template, otherwise the T=CL part will be skipped.

2.4. ID-ONLY PROCESSING TEMPLATE

2.4.1. *Lookup list*

Name	Tag	Description	Size
LKL.IDO	$_h$t0	**ID-only lookup list :** $_h$01 ≤ value ≤ $_h$0F for ISO-compliant cards, $_h$21 ≤ value ≤ $_h$2F for non-ISO cards. See **2.3.1a** for details.	1

2.4.2. *Output format*

Name	Tag	Description	Size
TOF.IDO	$_h$t1	ID-only output format. See table **a** below.	1

a. *Output format bits*

Bit	Value	Meaning
7 – 6		**Byte swapping**
	00	Do not swap ID bytes (ID is transmitted "as is")
	01	*RFU*
	10	Swap bytes for single-size (4 bytes) ISO 14443-A UIDs [3] only ; IDs of any other card is transmitted "as is"
	11	Swap ID bytes for all kind of cards
5		**Padding**
	0	Left-padding with $_h$0
	1	Right-padding with $_h$F
4		**ISO 14443-B specific**
	0	Use ISO 14443-B PUPI (4 bytes) as ID
	1	Use complete ISO 14443-B ATQ (11 bytes) as ID
3 – 0		**Output length**
	0000	Decimal, 4 bytes seen as 10 digits (i.e. 32 → 40 bits expansion)
	0001	Fixed length, 4 bytes [4]
	0010	Fixed length, 8 bytes [5]
	0011	Fixed length, 5 bytes
	0100	Fixed length, 12 bytes [6]
	0101	Fixed length, 7 bytes [7]
	0110	Fixed length, 11 bytes [8]
	0111	*RFU*
	1000	Fixed length, 16 bytes
	1001	*RFU*
	1010	*RFU*
	1011	*RFU*
	1100	Decimal, 5 bytes seen as 12 digits (i.e. 40 → 56 bits expansion)
	1101	Decimal, 5 bytes seen as 13 digits (i.e. 40 → 64 bits expansion)
	1110	Decimal, variable length (maximum 13 digits)
	1111	Variable length (depends on actual size of ID)

Default value : $_b$10000010

(8 bytes fixed length, left padding, swap bytes for short ISO 14443-A UIDs only)

[3] This is the default format in NXP's Mifare Classic related literature.

[4] ISO 14443-A single-size UID, ISO 14443-B PUPI, serial number for ASK CTS256B and CTS512B.

[5] ISO 15693 ID, serial number for NXP ICODE1, Inside Contactless PicoTag, ST MicroElectronics SR family…

[6] ISO 14443-A triple-size UID.

[7] ISO 14443-A double-size UID.

[8] ISO 14443-B complete ATQB.

2.4.3. *Output prefix*

Name	Tag	Description	Size
PFX.IDO	$_h$t2	ID-only output prefix.	Var.

Default value : absent *(no prefix)*

If a non-null ASCII value is specified (either a single character or a string), it will be transmitted before the data (therefore the actual length will be longer than the specified length).

2.4.4. *Offset of data*

Name	Tag	Description	Size
LOC.IDO	$_h$t3	Offset in the ID.	1

Default value : $_b$00000000 ($_d$0)

When TOF.IDO specifies a fixed length output, using LOC.IDO makes it possible to select some bytes in the ID, and not only the first ones. This is principally useful when working with non-ISO cards, see 2.4.5 for details.

2.4.5. *Non-ISO cards*

A few manufacturers offer non standard cards, most of them based on ISO 14443-B bit-level specification, but with a proprietary frame format (protocol) and a proprietary command set.

As those cards don't answer to ISO 14443 standard detection commands, a specific template must be activated to discover them.

a. ST MicroElectronics SR family

When LKL.IDO=$_h$22, the reader performs the lookup sequence for cards in the ST MicroElectronics SR family (SR176, SRX, SRIX).

A 8-byte serial number is returned by the card. Use TOF.IDO and LOC.IDO if you need to truncate it.

b. ASK CTS256B and CTS512B

When LKL.IDO=$_h$23, the reader performs the lookup sequence for cards in the ASK CTS-B family (CTS256B, CTS512B).

A 8-byte identifier is built as follow :

Byte 0	Byte 1	Byte 2	Byte 3	Bytes 4 to 7
Manufacturing code	Product code	Embedded code	Application code	4-byte serial number

- CTS256B's product code is between $_h$50 and $_h$5F,
- CTS512B's product code is between $_h$60 and $_h$6F,
- See ASK's documentation for explanations regarding other bytes.

Define LOC.IDO=$_h$04 (and TOF.IDO=$_h$01) if you need the serial number only (without card type and other data).

c. Inside Contactless PicoTAG[9]

When LKL.IDO=$_h$24, the reader performs the lookup sequence for cards in the Inside Contactless PicoTag family (PicoTag 16KS).

A 8-byte serial number is returned by the card. Use TOF.IDO and LOC.IDO if you need to truncate it.

[9] Also HID iClass

2.5. MIFARE CLASSIC PROCESSING TEMPLATE

Mifare “Classic” refers to NXP’s Mifare 1k and Mifare 4k wired-logic contactless cards.

Mifare 1k is divided into 64 16-byte blocks.

Mifare 4k is divided into 256 16-byte blocks.

Both cards have a 4-byte serial number, located at the beginning of block 0. As those cards are ISO/IEC 14443-3 compliant, you can read the serial number through the generic ID-Only template, instead of using this dedicated template.

2.5.1. *Lookup list*

Name	Tag	Description	Size
LKL.MIF	$_h$t0	Mifare classic lookup list, value = $_h$61. See **2.3.1a** for details.	1

2.5.2. *Output format*

Name	Tag	Description	Size
TOF.MIF	$_h$t1	Mifare output format. See table **a** below.	1

a. *Output format bits*

Bit	Value	Meaning
7	0	Do not swap bytes
	1	Swap bytes
6	0	RAW data
	1	ASCII encoded data [10]
5	0	Left-padding with $_h$0 (RAW) or <SPACE> (ASCII)
	1	Right-padding with $_h$F (RAW) or <SPACE> (ASCII)
4		*RFU*
3 – 0		**Output length** Format depends on bit 6 (RAW or ASCII). See table **b** below for RAW data (bit 6 = 0) See table **c** below for ASCII data (bit 6 = 1)

Default value : $_b$00000010

[10] If data read from the memory card is “31 32 33 43 34 35” (hexadecimal notation), output will be “123C45”. Make sure that only valid digits (values from 31 to 39 and 41 to 46 or 61 to 66) are encoded in every card, otherwise actual reader output will be undefined.

b. Output length when bit 6 = 0

Bit	Value	Meaning
3 – 0	0000	Decimal, 4 bytes seen as 10 digits (i.e. 32 → 40 bits expansion)
	0001	Fixed length, 4 bytes (32 bits)
	0010	Fixed length, 8 bytes (64 bits)
	0011	Fixed length, 5 bytes (40 bits)
	0100	Fixed length, 12 bytes (96 bits)
	0101	Fixed length, 7 bytes (56 bits)
	0110	Fixed length, 11 bytes (88 bits)
	0111	*RFU*
	1000	Fixed length, 16 bytes (128 bits)
	1001	*RFU*
	1010	*RFU*
	1011	*RFU*
	1100	Decimal, 5 bytes seen as 12 digits (i.e. 40 → 56 bits expansion)
	1101	Decimal, 5 bytes seen as 13 digits (i.e. 40 → 64 bits expansion)
	1110	Decimal, variable length (maximum 13 digits)
	1111	Variable length (using $_h$0 and $_h$F as end of string markers)

c. Output length when bit 6 = 1

Bit	Value	Meaning
3 – 0	0000	Max output length = $_d$16
	0001 to 1111	Max output length from $_d$1 to $_d$15

2.5.3. Output prefix

Name	Tag	Description	Size
PFX.MIF	$_h$t2	Mifare output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.5.4. Location of data

Depending on the size, the LOC.MIF tag can either be

- A block number (= address of data in Mifare card) when size = 1,
- An Application Identifier (AID) when size = 2.

a. Fixed block number

Name	Tag	Description	Size
LOC.MIF	$_h$t3	Block number to be read.	1

Default value : $_b$00000100 ($_d$4)

When a Mifare card is found, reader tries to read the block specified in LOC.MIF (16 bytes), and then truncates the data according to the length specified in TOF.MIF.

The block number shall be

- Between 0 and 63 for Mifare 1k cards,
- Between 0 and 255 for Mifare 4k cards.

Note that data must start on a block boundary.

Mifare sector trailers (security blocks) numbered 3, 7, … can be read, but their content is masked (to protect the keys). Using such a block as access control identifier is definitely not a good idea.

b. AID in MAD

Name	Tag	Description	Size
LOC.MIF	$_h$t3	AID to be selected and read.	2

When a Mifare card is found, reader reads the MAD (blocks 1 and 2 of sector 0)[11] and tries to find the specified AID. The location of the AID in the MAD is the pointer onto the actual block to be read.

Note that data must be located at the beginning of the first block marked with the specified AID.

Please refer to NXP application notes for detailed explanations of the MAD.

[11] Sector 0 must be freely readable either with base key A ("A0 A1 A2 A3 A4 A5"), with transport key ("FF FF FF FF FF FF") or with the application key specified in AUT.MIF .

2.5.5. Authentication key

Depending on the size, the AUT.MIF tag can either be

- A pointer to a key located in RC’s secure EEPROM when size = 1.
- The Mifare key itself, when size = 7,
- A master key and its diversification options, when size = 9 or 17

When the AUT.MIF tag is absent, all EEPROM keys are tried out in sequence (this can take a long time…).

Name	Tag	Description	Size
AUT.MIF	$_h$t5	Mifare authentication key.	See below

Default value : absent

a. Size = 1 : pointer to a key in RC’s secure EEPROM

- Values $_h$00 to $_h$0F refer to type A keys $_d$0 to $_d$15, respectively,
- Values $_h$10 to $_h$1F refer to type B keys $_d$0 to $_d$15, respectively.

b. Size = 7 : specified Mifare key

Offset	Length	Content
0	1	Key options. See table **c** below.
1	6	Mifare key value.

c. Key options bits, when size = 7

Bit	Value	Meaning
7	0	Key is an A key
	1	Key is a B key
6 – 0		*RFU*

d. Size = 17 : master key diversification using HMAC-MD5

Offset	Length	Content
0	1	Key options. See table **e** below.
1	16	Master key value.

e. *Key options bits, when size = 17*

Bit	Value	Meaning
7	0 1	Diversified key is an A key Diversified key is a B key
6	0 1	Diversification with card UID and address fixed to $_h$00 Diversification with card UID and address = sector number
5 – 4	10	Diversify the key using HMAC-MD5 algorithm *(see chapter 9)*
3 – 0		*RFU*

f. *Size = 15 or 23 : master key diversification using RC171 algorithm*

Offset	Length	Content
0	1	Key options. See table **g** below.
1	6	Mifare master key.
7	8 or 16	DES or 3-DES diversification key.

g. *Key options bits, when size = 15 or 23*

Bit	Value	Meaning
7	0 1	Diversified key is an A key Diversified key is a B key
6	0 1	Diversification with card UID and address fixed to $_h$00 Diversification with card UID and address = sector number
5 – 4	01	Diversify the key using RC171 algorithm *(see chapter 10)*
3 – 0		*RFU*

2.6. MIFARE ULTRALIGHT PROCESSING TEMPLATE

NXP's Mifare UltraLight is a low-cost wired-logic contactless card. It is divided into 16 4-byte pages. This template reads 4 pages (i.e. exactly 16 bytes) at once.

This card has a 7-byte serial number, located on blocks 0 and 1. As the card is ISO/IEC 14443-3 compliant, you can read the serial number through the generic ID-Only template, instead of using this dedicated template.

2.6.1. Lookup list

Name	Tag	Description	Size
LKL.MFU	$_h$t0	Mifare UltraLight lookup list, value = $_h$62. See **2.3.1a** for details.	1

2.6.2. Output format

Name	Tag	Description	Size
TOF. MFU	$_h$t1	Mifare UltraLight output format.	1

Same as Mifare Classic output format (2.5.2).

2.6.3. Output prefix

Name	Tag	Description	Size
PFX.MFU	$_h$t2	Mifare UltraLight output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.6.4. Location of data

Name	Tag	Description	Size
LOC.MFU	$_h$t3	Number of the first page to be read.	1

Default value : $_b$00000000 ($_d$0)

Remember that this template always reads 4 pages (16 bytes) starting at LOC.MFU.

2.7. DESFIRE CARD PROCESSING TEMPLATE

2.7.1. Lookup list

Name	Tag	Description	Size
LKL.DFR	$_h$t0	Desfire lookup list, value = $_h$71. See **2.3.1a** for details.	1

2.7.2. Output format

Name	Tag	Description	Size
TOF.DFR	$_h$t1	Desfire output format.	1

Same as Mifare Classic output format (2.5.2).

2.7.3. Output prefix

Name	Tag	Description	Size
PFX.DFR	$_h$t2	Desfire output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.7.4. Location of data

Name	Tag	Description	Size
LOC.DFR	$_h$t3	Location of data in Desfire card. See table **a** below.	8

a. Data location bytes

Offset	Length	Content
0	3	Application IDentifier (AID).
3	1	File IDentifier (FID). File must be a "standard data" file.
4	3	Offset of data in file.
7	1	Length of data to be read[12] (1 to 64).

Default value : unspecified.

Values are MSB first.

[12] Data will be truncated to the length specified in TOF.DFR .

2.7.5. T=CL options

Name	Tag	Description	Size
OPT.DFR	$_h$t4	Desfire T=CL options.	1

Same as 7816-4 T=CL options (2.8.5).

2.7.6. Authentication key

Name	Tag	Description	Size
AUT.DFR	$_h$t5	Desfire authentication key. See table **a** below.	9 or 17

Default value : absent

(No authentication is performed, plain read operation is used to fetch the data)

a. Authentication key bytes

Offset	Length	Content
0	1	Desfire key index and options. See table **b** below.
1	8 or 16	Key value (8 bytes for a DES key, 16 bytes for a 3-DES key).

b. Key index and options

Bit	Value	Meaning
7 – 6		**Communication mode for reading**
	00	Plain
	01	MACed with session key
	10	*RFU*
	11	Enciphered with session key
5 – 4		**Key diversification algorithm**
	00	Use the key "as is"
	01	Diversify the key using Desfire SAM algorithm *(see chapter 10)*
	10	Diversify the key using HMAC-MD5 algorithm *(see chapter 9)*
	11	*RFU*
3 – 0		**Index of key in Desfire application**
	0000 to 1110	Index of the key to be used for authentication
	1111	*RFU*

2.8. 7816-4 CARD PROCESSING TEMPLATE

2.8.1. Lookup list

Name	Tag	Description	Size
LKL.TCL	$_h$t0	7816-4 lookup list, $_h11 \le$ value $\le {}_h13$. See **2.3.1a** for details.	1

2.8.2. Output format

Name	Tag	Description	Size
TOF.TCL	$_h$t1	T=CL output format.	1

Same as Mifare Classic output format (2.5.2).

2.8.3. Output prefix

Name	Tag	Description	Size
PFX.TCL	$_h$t2	T=CL output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.8.4. Location of data

Name	Tag	Description	Size
LOC.TCL	$_h$t3	Offset of data in answer to APDU 3[13] (0 to 127).	1

Default value : 0.

2.8.5. T=CL options

Name	Tag	Description	Size
OPT.TCL	$_h$t4	T=CL (ISO/IEC 14443 layer 4) options. See table **a** below.	1

[13] Data will be truncated according to the length specified in TOF.TCL .

a. T=CL option bits

Bit	Value	Meaning
		Card to reader baudrate
7 – 6	00	No PPS, DSI = 106kbit/s
	01	Perform PPS, DSI = 212kbit/s if card allows it
	10	Perform PPS, DSI = 424kbit/s if card allows it
	11	Perform PPS, DSI = 848kbit/s if card allows it
		Reader to card baudrate
5 – 4	00	No PPS, DRI = 106kbit/s
	01	Perform PPS, DRI = 212kbit/s if card allows it
	10	Perform PPS, DRI = 424kbit/s if card allows it
	11	Perform PPS, DRI = 848kbit/s if card allows it
		Card identifier (CID)
3 – 0	0000	Empty CID = $_d$0
	0001 to 1110	CID from $_d$1 to $_d$14
	1111	CID is disabled

This tag exists only if T=CL card is selected in LST.

Default value : $_b$00001111

2.8.6. T=CL APDU 1

Typically this is a Select Application (or Select Applet) command.

May be absent if T=CL APDU 3 is sufficient to fetch the data.

Name	Tag	Description	Size
AU1.TCL	$_h$t5	TCL APDU 1.	Var.

Card's Status Word is checked by the reader. A SW between $_h$9000 and $_h$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between $_h$6100 and $_h$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.8.7. T=CL APDU 2

Typically this is a Select File command.

May be absent if T=CL APDU 3 is sufficient to fetch the data.

Name	Tag	Description	Size
AU2.TCL	${}_h$t6	TCL APDU 2.	Var.

Card's Status Word is checked by the reader. A SW between ${}_h$9000 and ${}_h$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between ${}_h$6100 and ${}_h$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.8.8. T=CL APDU 3

APDU used to actually retrieve the data (typically this is a Read Binary command). Data have to be found in answer at offset specified in LOC.TCL.

Name	Tag	Description	Size
AU3.TCL	${}_h$t7	TCL APDU 3.	Var.

Card's Status Word is checked by the reader. A SW between ${}_h$9000 and ${}_h$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between ${}_h$6100 and ${}_h$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.9. CALYPSO CARD PROCESSING TEMPLATE

This part deals with old Calypso cards, to be accessed only through the legacy Innovatron radio protocol.

New Calypso cards now support ISO/IEC 14443-B, and therefore can be accessed either through ID-Only or ISO/IEC 7816-4 templates.

Working with Calypso cards is subject to a specific licence fee. This function is therefore disabled for out-of-factory readers.

Please contact us to have the Calypso functionality enabled in your readers.

Depending on the specified options, this Calypso card processing template can retrieve :

- A 4-byte serial number (ID-Only template)
- Arbitrary data to be read in Calypso files (7816-4 template)

2.9.1. *Lookup list*

Name	Tag	Description	Size
LKL.CYO	$_h$t0	Calypso/Innovatron lookup list, value = $_h$72. See **2.3.1a** for details.	1

2.9.2. *Output format*

Name	Tag	Description	Size
TOF.CYO	$_h$t1	Calypso/Innovatron output format.	1

Same as Mifare Classic output format (2.5.2).

2.9.3. *Output prefix*

Name	Tag	Description	Size
PFX.CYO	$_h$t2	Calypso/Innovatron output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.9.4. Location of data

Name	Tag	Description	Size
LOC.CYO	$_{h}$t3	Offset of data in answer to APDU 3[14] (0 to 64).	1

Default value : 0.

2.9.5. Calypso APDU 1

Typically this is a Select DF command.

Name	Tag	Description	Size
AU1.CYO	$_{h}$t5	Calypso/Innovatron APDU 1.	Var.

Card's Status Word is checked by the reader. A SW between $_{h}$9000 and $_{h}$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between $_{h}$6100 and $_{h}$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.9.6. Calypso APDU 2

Typically this is a Select EF command.

Name	Tag	Description	Size
AU2.CYO	$_{h}$t6	Calypso/Innovatron APDU 2.	Var.

Card's Status Word is checked by the reader. A SW between $_{h}$9000 and $_{h}$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between $_{h}$6100 and $_{h}$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

[14] Data will be truncated according to the length specified in TOF.CYO .

2.9.7. Calypso APDU 3

Typically this is a Read Binary command.

Name	Tag	Description	Size
AU3.CYO	$_h$t7	Calypso/Innovatron APDU 3	Var.

✋	Card's Status Word is checked by the reader. A SW between $_h$9000 and $_h$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between $_h$6100 and $_h$6FFF) is considered as an error, and the reader will ignore the card. Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.10. SUMMARY OF CONFIGURATION TAGS

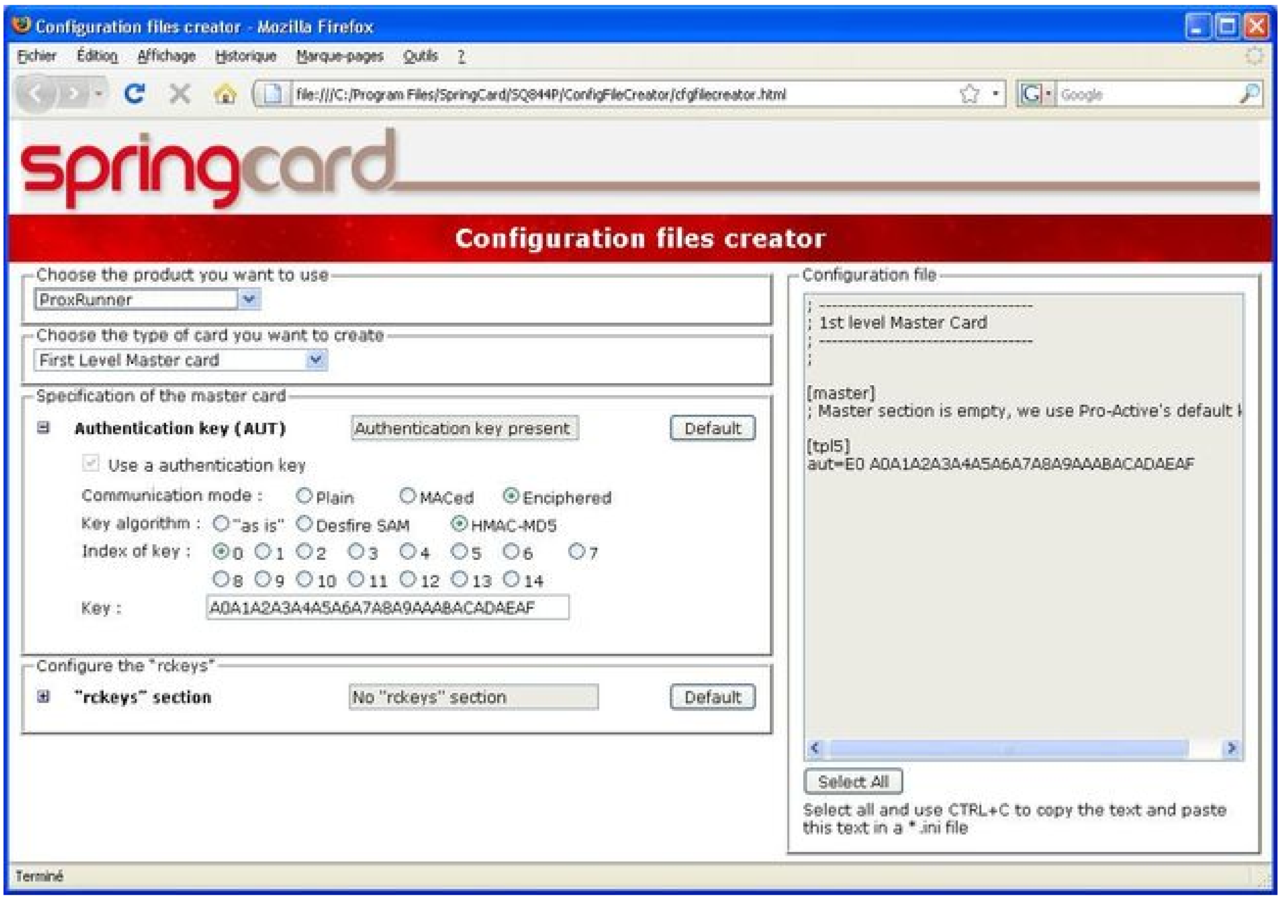
Name	Tag	Content
	h10 h11 … h1F	**Card Processing Template #1** (out of factory : versatile ID-only reader)
	h20 h21 … h2F	**Card Processing Template #2** (out of factory : empty)
	h30 h31 … h3F	**Card Processing Template #3** (out of factory : empty)
	h40 h41 … h4F	**Card Processing Template #4** (out of factory : empty)
	h50 h51 … h5F	**Reserved for Master Cards** (see chapter 7)
OPT	h60	General configuration

3. CONFIGURING PROXRUNNER

ProxRunner is configured through Master Cards, formatted with **cfgfilecreator.exe** software.

cfgfilecreator.exe is a command line software (running on Microsoft Windows) to create Master Cards. **cfgfilecreator.exe** needs a SpringCard Prox'N'Roll PC/SC (or legacy) contactless coupler to program the cards.

cfgfilecreator.html is a standalone web page that helps creating configuration files for **cfgfilecreator.exe** .


Configuration files creator - Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils ?
file:///C:/Program Files/SpringCard/SQ844P/ConfigFileCreator/cfgfilecreator.html
Google
springcard
Configuration files creator

Choose the product you want to use: ProxRunner

Choose the type of card you want to create: First Level Master card

Specification of the master card

Authentication key (AUT): Authentication key present — Default

[✓] Use a authentication key

Communication mode : Plain / MACed / [✓ Enciphered]

Key algorithm : "as is" / Desfire SAM / [✓ HMAC-MD5]

Index of key : [✓ 0] 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Key : A0A1A2A3A4A5A6A7A8A9AAABACADAEAF

Configure the "rckeys"

"rckeys" section: No "rckeys" section — Default

Configuration file

```
; ------------------------------------
; 1st level Master Card
; ------------------------------------
;

[master]
; Master section is empty, we use Pro-Active's default k

[tpl5]
aut=E0 A0A1A2A3A4A5A6A7A8A9AAABACADAEAF
```

Select All

Select all and use CTRL+C to copy the text and paste this text in a *.ini file

3.1. CONFIGURATION FILES

cfgfilecreator.exe uses a configuration file to retrieve configuration data to be written into the Master Card.

Configuration files are written like standard Windows "INI" files. They can be created using Notepad or any other text editor, or using **cfgfilecreator.html** as wizard.

Each line of each section uses the format "name=value" where "name" is either the name or the tag of the configuration variable (e.g. either "opt" or "60"), and "value" its value in hexadecimal.

3.1.1. The “general” section

This section maps to tags $_h60$. Default content is :

```
[general]
opt=2C       ; value for OPT
```

3.1.2. The “rckeys” section

This section holds the Mifare access keys to be written in RC’s secure EEPROM.

Type A keys are named “a0” to “a15”, and type B keys “b0” to “b15”.

Here’s an example of content :

```
[rckeys]
a0=A0A1A2A3A4A5 ; Mifare type A base key (for MAD)
a1=FFFFFFFFFFFF ; NXP transport key
a2=000000000000 ; other transport key
a3=CCCCCCCCCCCC ; unused
(...)
a15=CCCCCCCCCCCC ; unused
b0=B0B1B2B3B4B5 ; Mifare type B base key (for MAD)
b1=FFFFFFFFFFFF ; NXP transport key
b2=000000000000 ; other transport key
b3=CCCCCCCCCCCC ; unused
(...)
b15=CCCCCCCCCCCC ; unused
```

This section (and each line in it) is optional. Only keys listed in this section will be written, other keys will be left unchanged.

3.1.3. Sections for Card Processing Templates

ProxRunner may run from 1 to 4 card accepting templates. Each template is configured by sections “tpl1”, “tpl2”, “tpl3” and “tpl4” respectively.

Mandatory and optional content for each section depends on the card lookup list (LKL field) of the section itself.

a. ID-Only example

This sample section configures template 4 to read any kind of ID. Output format is : 8-byte fixed length, prefixed by the string “ID=” :

```
[tpl4]
lkl=0F                    ; wants any kind of ID
tof=82                    ; 8-byte output, swap 14443 A short IDs
pfx=49443D                ; prefix = “ID=”
```

b. Desfire example

This sample section configures template 1 to read 8 bytes of data from a Desfire card. Output format is : 8-byte fixed length, no prefix :

```
[tpl1]
lkl=71                  ; wants Desfire cards
tof=02                  ; 8-byte output
pfx=                    ; no prefix
loc=123456 01 000100 08 ; 8 bytes of data to be read in application
                        ; 0x123456, field 0x01, at offset 0x000100
aut=00 A0A1A2A3A4A5A7   ; authentication with key 0, plain comm.
                        ; mode, no diversification. Key is a single
                        ; DES key (8 bytes)
```

3.1.4. Master Cards related sections

a. Specifying a new configuration for future Master Cards

The "tpl5" section allows to update the card processing template reserved to Master Cards. See paragraph 8.4.1 for details.

```
[tpl5]
aut=E0 xx...xx        ; 16-byte authentication key
```

This 16-byte authentication key in the "tpl5" section is the one that will be written in the reader(s) by the Master Card.

It is not the key that will be used to create the Master Card itself.

b. Specifying configuration to be used by current Master Card

The "master" section defines how the Master Card shall be created. See paragraph 8.4.2 for details.

```
[master]
aut=E0 xx...xx        ; 16-byte authentication key
```

This 16-byte authentication key in the "master" section is the one that will be used to create the Master Card.

It has no impact on the key written in the reader(s).

3.2. OPERATION INSTRUCTIONS

- Open **Configuration files creator** (**cfgfilecreator.html**)
  (on Windows : Start Menu → All Programs → SpringCard → Configuration Tools),
- Create your configuration file and save it in the directory where **cfgfilecreator.exe** is installed, for instance with the name *siteconf.ini* (on Windows : C:\Program Files\SpringCard\SQ844P),
- Open **Configuration tools directory**
  (on Windows : Start Menu → All Programs → SpringCard → Configuration Tools),
- Plug and power-on your Prox'N'Roll PC/SC (or legacy),
- Put a virgin Desfire card on the Prox'N'Roll PC/SC (or legacy),
- Enter **cfgfilecreator.exe –c *siteconf.ini***,
- Wait until Master Card is written.

If the Desfire card is not virgin, the **software will try to format it** (i.e. erase the whole file structure with all the data) **without prior notification**.

Be sure to put on the reader only a virgin card, or an old Master Card to be overwritten.

You've been warned…

3.3. CHANGING AUTHENTICATION KEY FOR MASTER CARDS

All ProxRunners are shipped with the same out-of-factory authentication key. To secure their site, customers should replace the default key by their own key before installing the readers.

Pro-Active recommends to make (and keep) at least two distinct Master Cards for each customer or site :

- ***1st level Master Card*** alters only the authentication key (replace default key by site specific key).
  - All readers bought for this site shall be configured using this *1st level Master Card* as soon as they are received.
- ***2nd level Master Card*** actually configures the reader (card processing templates, output mode and format, and so on).
  - It uses the site specific key for authentication, but doesn't update the key that is already inside the reader.

- The *2nd level Master Card* shall be used during installation and whenever you wish to change reader configuration.

Note that many *2nd level Master Cards* can be created (one for each kind of output settings, one for each people in charge of installation…) whereas only one *1st level Master Card* should be created and be kept in a secure place[15].

☠ Be sure to remember the new authentication key you put in a reader. If you forget the authentication key, and forget the pin-code (or define pin-code to $_h$FFFF), it will be impossible to change reader configuration again !

You've been warned…

3.3.1. *Creating a first level Master Card*

- Create a configuration file (say, "*master.ini*") with only those 4 lines :

```
[master]
; Master section is empty, we use Pro-Active's default keys

[tpl5]
aut=E0 xx...xx
```

where *xx…xx* is the site specific 16-byte authentication key[16],

- Put a virgin card on the Prox'N'Roll, label it "1st level Master Card",
- Enter **cfgfilecreator.exe –c *master.ini*** ,
- Use this Master Card to write the new authentication key in the reader(s).

3.3.2. *Creating a second level Master Card*

- Create a complete configuration file as seen in § 8.3 .
- Terminate the file with those 4 lines :

```
[master]
aut=E0 xx...xx

[tpl5]
; Template 5 section is empty, we keep current keys in the reader
```

where *xx…xx* is the site specific 16-byte authentication key[16],

[15] That's because *1st level Master Card* has got the authentication key written in it, and anybody may retrieve it using **cfgfilecreator** software, where the authentication key is only used to secure *2nd level Master Cards* and is not written in them.

[16] This is key 0 inside Master Card application ; the key will be diversified using HMAC-MD5 algorithm, so the "E0" header is mandatory.

- Put a virgin card on the Prox'N'Roll, label it "2nd level Master Card",
- Enter **cfgfilecreator.exe –c *siteconf.ini*** ,
- Use this Master Card to write complete configuration in the reader(s).

3.4. REVERTING TO DEFAULT

Sometimes it is necessary to put reader back in "out-of-factory" configuration (for instance when reader goes from one site to another). This is done easily by erasing all tags from reader's memory.

- Create a configuration file (say, "*factory.ini*") with only those 3 lines :

```
[master]
aut=E0 xx...xx
clear=1
```

where *xx…xx* is the site specific 16-byte authentication key

- Put a virgin card on the Prox'N'Roll, label it "Erase all Master Card",
- Enter **cfgfilecreator.exe –c *factory.ini***
- Use this Master Card to put the reader(s) back in out-of-factory configuration.

Erasing all the configuration tags is not really sufficient to put the reader(s) back in out-of-factory configuration, since Mifare keys stored in RC's secure EEPROM are not erased.

Just add an "rckeys" section (as specified in 8.2.2), with dummy keys, to overwrite those keys.

4. SPECIFICATION OF MASTER CARDS

This chapter is provided as a mean for security experts to evaluate ProxRunner Master Card architecture.

Customers do not need to implement this part themselves, since **cfgfilecreator.exe** software is a convenient tool to create Master Cards. See chapter 3 for details.

4.1. BUILDING A MASTER CARD

- The Master Card must be a Desfire 4k,
- Reader tries to fetch configuration data from Desfire cards according to the Master Card template specified in next paragraph. Data are protected by an authentication key that may be changed on a per-customer or per-site basis (i.e. Master Cards belonging to customer X will not work on customer Y's readers),
- Before storing new settings in its non-volatile memory, reader checks that data comes with a valid digital signature. The signing key can't be changed, and is only known by Pro-Active's software. This ensure that only data that has been pre-validated by a genuine software can be loaded in reader's non-volatile memory.

4.2. TEMPLATE FOR MASTER CARDS

4.2.1. *Location of data*

Name	Tag	Description	Size
LOC.MAS	$_h$53	Location of data in master cards. See table **a** below.	5

a. *Data location bytes*

Offset	Length	Content	Specified value
0	3	Application IDentifier (AID).	$_h$504143
3	1	File IDentifier (FID) for configuration data.	$_h$01
4	1	File IDentifier (FID) for digital signature.	$_h$02

4.2.2. Authentication key

Out-of-factory key used for authentication of Master Cards is confidential.

Only Pro-Active genuine software –such as **cfgfilecreator.exe**– is able to create Master Cards with the default authentication key.

To secure their installation, customers should replace this key as soon as they receive the readers, as explained in 8.4 .

This is the same structure as AUT.DFR .

Name	Tag	Description	Size
AUT.MAS	$_h$55	Authentication key. See table **a** below.	17

a. Authentication key bytes

Offset	Length	Content
0	1	Authentication key index and options. See table **b** below.
1	16	***Authentication key for Master Cards*** (this is 3-DES key).

b. Authentication key index and options

Bit	Value	Meaning
7 – 6		**Communication mode in read operation**
	00	Plain
	01	MACed with session key
	10	*RFU*
	11	Enciphered with session key
5 – 4		**Key diversification algorithm**
	00	Use the key "as is"
	01	Diversify the key using Desfire SAM algorithm *(see chapter 10)*
	10	Diversify the key using HMAC-MD5 algorithm *(see chapter 9)*
	11	*RFU*
3 – 0		**Index of key in Desfire application**
	0000 to 1110	Index of the key to be used for authentication
	1111	*RFU*

Specified value : $_h$E0 *(key 0, HMAC-MD5 diversification, ciphered reading)*

4.2.3. Signing key

Name	Tag	Description	Size
SGN.MAS	$_h$56	Signing key. See table **a** below.	17

☠	Key used for digital signature of master cards is confidential. Only Pro-Active genuine software –such as **cfgfilecreator.exe**– is able to sign the Master Cards[17]. Customers shall not try to change this parameter, unless advised to by Pro-Active.

a. Signing key bytes

Offset	Length	Content
0	1	Index and options. See table **b** below.
1	16	***Key data*** (this is 128-bits key).

b. Signing key index and options

Bit	Value	Meaning
7 – 6	00	*Those bits are RFU and must be 00*
5 – 4		**Key diversification algorithm**
	00	Use the key "as is"
	01	Diversify the key using Desfire SAM algorithm *(see chapter 10)*
	10	Diversify the key using HMAC-MD5 algorithm *(see chapter 9)*
	11	*RFU*
3 – 0	0000	*Those bits are RFU and must be 00*

Specified value : $_h$20 *(HMAC-MD5 diversification)*

[17] This choice has been done to ensure that data inside the Master Card have been pre-validated according to reader specifications, and have not been corrupted afterwards.

4.3. DATA STRUCTURE

4.3.1. *Size of file*

File holding configuration data and Mifare keys (offset 3 in LOC.MAS) must be exactly 512-byte long. In case used size is shorter than 512 bytes, file must be padded with $_h$00.

4.3.2. *Configuration data*

The configuration data block uses the T,L,V (tag, length, value) encoding scheme.

- Tag is 1 byte-wide,
- Len is 1 byte-wide,
- Value is 0 to 24 byte-wide.

Items found in T,L,V blocks will overwrite data with the same tag already present in reader's non-volatile memory.

Set Len = 0 to delete an existing tag from the non-volatile memory, without replacing it.

Last T,L,V of the configuration data block must be the digital signature of the whole block, according to the algorithm specified in 7.4.

4.3.3. *Mifare keys to be loaded into RC's secure EEPROM*

Keys to be loaded into RC's secure EEPROM use the T,L,V scheme, as follow :

- Tag (1 byte) = $_h$80 + key index as specified in 2.6.4.a,
- Len (1 byte) = $_h$06,
- Value is the Mifare key (6 bytes exactly).

4.4. DIGITAL SIGNATURE

4.4.1. *Size of file*

File holding the signature (offset 4 in LOC.MAS) must be exactly 16-byte long.

4.4.2. *Algorithm*

This is the signature algorithm when default parameters in SGN.KEY as used :

- Let *Content* be the 512-byte configuration block as written in the card[18],
- Let *SignKey* be the 16-byte key,
- Diversify *SignKey* from card's UID, using HMAC-MD5 diversification algorithm[19] to get *DivKey*,
- Compute *Sign* = HMAC-MD5 (*Block*) using *DivKey* [20].

As specified in 7.2.3, value of *SignKey* is confidential. Customers shall not try to change the key, nor the signature algorithm.

[18] This is the configuration data plus the Mifare keys to be loaded into RC's secure EEPROM. Total size is up to 512 bytes (as required by 7.3.1). Note that signature is computed over the whole file, including its padding, whatever the used length is.

[19] See 8.3.1

[20] See 8.2

5. HMAC SIGNATURE AND KEY DIVERSIFICATION

5.1. HMAC-MD5

5.1.1. *Abstracts*

A message authentication code, or MAC, is a short piece of information used to authenticate a message. A MAC algorithm accepts as input a secret key and a message, and outputs a MAC that protects both message's integrity and authenticity.

An HMAC (or keyed-hash message authentication code) is a type of MAC function were a cryptographic hash function is used to compute the output.

5.1.2. *Algorithm*

$$\mathrm{HMAC}_K(m) = h\Big((K \oplus \mathrm{opad}) \| h\big((K \oplus \mathrm{ipad}) \| m\big)\Big),$$

Where *h* is the hash function, *K* is the secret key padded with extra zeros up to 64 bytes, *m* is the message to be authenticated. *opad* is the value $_h$5C repeated 64 times, and ipad the value $_h$36 repeated 64 times.

HMAC-MD5 is a particular HMAC function where *h* is the MD5 standard function, as defined by RSA laboratories. Size of HMAC is 16 bytes exactly.

5.2. USING HMAC-MD5 FOR SIGNATURE

HMAC protects both message's integrity and authenticity, so it can be considered as a digital signature[21].

ProxRunner implementation allows only 16-byte keys. The key can be used "as is" or be the result of a diversification from a master key.

5.3. USING HMAC-MD5 FOR KEY DIVERSIFICATION

In this particular mode, we name *K* the "master key" and we compute the HMAC over card's identifier to establish a "diversified key" *Ku*.

[21] Literature often reserve the name "digital signature" to public key schemes, where verifier doesn't need to know signer's private key to verify the signature. HMAC is a scheme where signer and verifier must share the same secret key.

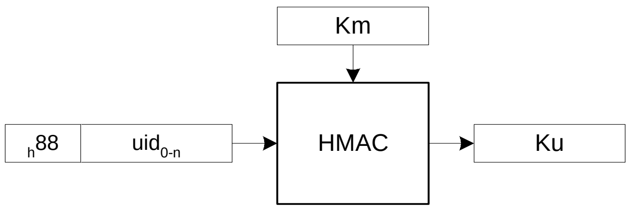
5.3.1. DES & Triple-DES key diversification algorithm

The algorithm takes as inputs :

- A 16-byte master key (Km)
- The card serial number (uid)[22]

It provides as output :

- The 16-byte diversified key specific to this card (Ku).



The diversified key can now be used either for Desfire authentication, or for HMAC-MD5 signature.

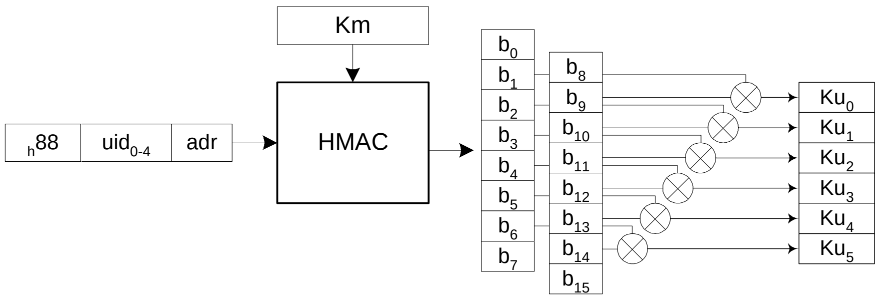
5.3.2. Mifare key diversification algorithm

The algorithm takes as inputs :

- A 16-byte master key (Km)
- The 4-byte card serial number (uid)
- The 1-byte block address (adr)

It provides as output :

- The 6-byte Mifare key specific to the couple card + address (Ku).



See last two paragraphs of chapter 10, for details regarding how the *adr* parameter shall be understood.

[22] The UID is 7-byte long for a Desfire card, 4-byte long for a Mifare card. The same diversification algorithm is usable whatever the length is.

6. DESFIRE SAM & RC171 KEY DIVERSIFICATION

6.1. DES AND 3-DES KEY DIVERSIFICATION

The key diversification algorithm described here is the one provided by Desfire SAM. Please refer to the corresponding datasheet for details.

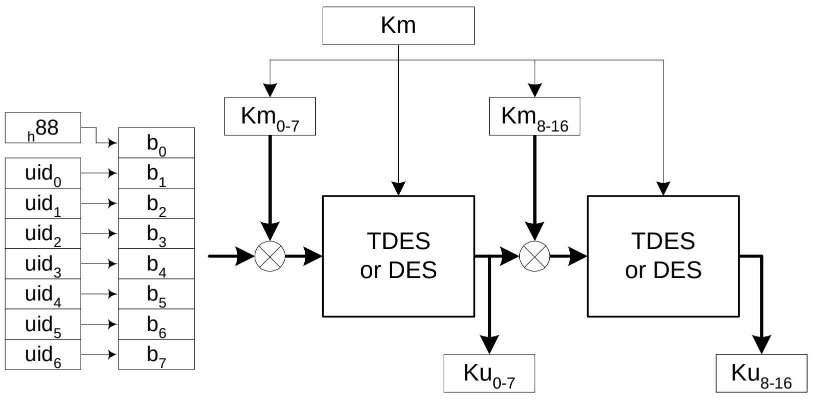
The algorithm takes as inputs :

- A 16-byte Triple-DES master key (Km)[23]
- The 7-byte card serial number (uid)

It provides as output :

- The 16-byte diversified key specific to this card (Ku).

Here's the flowchart :



The diversified key now be used for Desfire authentication.

[23] If both halves are equals, the key maps to a single DES key

6.2. MIFARE KEY DIVERSIFICATION

The Mifare diversification algorithm described here is provided both by Desfire SAM and by RC171 secure coprocessor. Please refer to the corresponding datasheets for details.

6.2.1. *Basis*

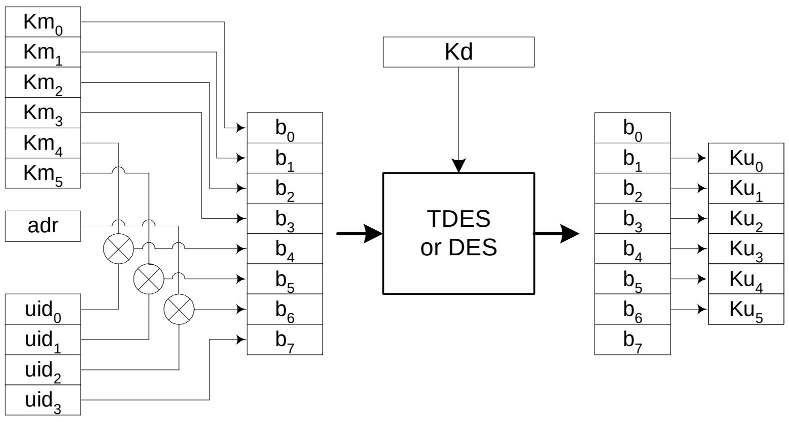
The algorithm takes as inputs :

- A 6-byte master key (Km)
- A 16-byte Triple-DES diversification key (Kd)[24]
- The 1-byte block address (adr)
- The 4-byte card serial number (uid)

It provides as output :

- The 6-byte Mifare key specific to the couple card + address (Ku).

Here's the flowchart :



6.2.2. *Diversification based on UID only*

If this option is selected, the *adr* input parameter is fixed to $_h00$ whatever block to be read is.

[24] If both halves are equals, the key maps to a single DES key

6.2.3. Diversification based on UID and address

If this option is selected, the *adr* input parameter is the <u>Mifare sector number</u>.

Here’s an example with a Mifare 1k card :

- Data is located on block 29,
- Block 29 belongs to sector 7 (29 / 4),
- The diversification algorithm will be fed with adr = 7.

Here’s an example with a Mifare 4k card :

- Data is located on block 231,
- Block 231 belongs to sector 38 (32 + (231-128) / 16),
- The diversification algorithm will be fed with adr = 38.

DISCLAIMER

COPYRIGHT NOTICE

EDITOR'S INFORMATION

Published by **Pro-Active SAS**, 13, voie La Cardon 91120 Palaiseau – France

R.C.S. EVRY B 429 665 482 - APE 26127

For more information, please contact us at info@pro-active.fr .