PMA82TP-BB
08/04/2009

ProxRunner Bluetooth RFID Scanner

Reference manual

Headquarters, Europa

SpringCard
13 voie la Cardon
Parc Gutenberg
91120 Palaiseau
FRANCE

Phone : +33 (0) 164 53 20 10
Fax : +33 (0) 164 53 20 18

Americas

SpringCard
964 Fifth Avenue
Suite 235
San Diego, CA 92101
USA

Phone : +1 (619) 544 1450
Fax : +1 (619) 573 6867

www.springcard.com

DOCUMENT INFORMATION

Category : Manual
Group : ProxRunner
Reference : PMA82TP
Version : BB
Status : Approved

Keywords :
RFID Scanner, Reader, Bluetooth, Configuration

pma82tp-bb.doc
saved 08/04/09 - printed 08/04/09

Abstract :

REVISION HISTORY

Ver.	Date	Author	Valid. by Tech.	Valid. by Qual.	Approv. by	Remarks :
BB	02/04/09	LTX	LTX	ECL	ECL	Add buzzer control options
BA	16/02/09	LTX	JDA	JDA	JDA	SpringCard branding. Added reference to “RFID Scanner” family. Common chapters now shared with other products along the same family. Add "template = 0xFF". Allows ID reading of all cards supported by reader.
AA	15/01/09	LTX	ECL	LTX	ECL	Initial release, Pro Active branding

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 in partnership 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.

Reference	Document title
PFL8S0P	ProxRunner Product Information sheet
PMU84QP	ProxRunner Quick Start guide

2. Configuration attributes

There are two families of configuration attributes :

- Product specific Global Configuration Attributes,
- Card Acceptance Templates.

The Card Acceptance Templates are common to all products in the **SpringCard RFID Scanner family**, and are exposed in detail in the next chapter.

In this chapter, we'll introduce configuration tags and detail the **ProxRunner**'s specific configuration attributes.

2.1. Principles

a. Configuration tags

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

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

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

b. Non-volatile memory endurance

ProxRunner configuration attributes are stored in reader's non-volatile memory (flash). They can be changed up to 100 times.

Changing any configuration attribute more than 100 times may permanently damage your **ProxRunner** reader.

2.2. Global Configuration Attributes

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

Default value : $_b$00101100

(Master Cards are enabled all the time)

2.2.2. Buzzer control options

Name	Tag	Description	Size
CBZ	$_h$64	Buzzer control. See table **a** below.	1

a. Buzzer control bits

Bit	Value	Meaning
7 – 5		*RFU (set to 000)*
4	0 1	No action on buzzer when a valid card has been processed Short sound when a valid card has been processed
3	0 1	No action on buzzer for unsupported cards Long sound when an unsupported card has been processed
2 – 0		*RFU (set to 000)*

Default value : $_b$00011000

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

3. CARD ACCEPTANCE TEMPLATES

Products in the **SpringCard RFID Scanners** family are able to manage different types of cards, and different sources of data on each card.

A **Card Acceptance Template** defines how the reader will recognize the card to be read, and how it would get the actual data (serial number, block reading, file selection and reading, authentication keys to be used for Mifare or Desfire, etc).

The template also defines which formatting is to be applied to the data when sending them to the target device (translation to ASCII or to Decimal, constant prefix or suffic, etc).

This product is able to run up to 4 Card Acceptance Templates simultaneously.

3.1. BASIS

Each Card Acceptance Template is configured through a set of configuration attributes, each attribute having its own tag.

- Template 1 uses Configuration tags $_h$10 to $_h$1F
- Template 2 uses Configuration tags $_h$20 to $_h$2F
- Template 3 uses Configuration tags $_h$30 to $_h$3F
- Template 4 uses Configuration tags $_h$40 to $_h$4F

In the following pages, we use the convention " Template t uses Configuration tags $_h$t0 to $_h$tF ". Replace t by the current template number.

3.1.1. 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**	3.2
$_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**	3.6
$_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**	3.2
$_h$23	ASK CTS256B and CTS512B		
$_h$24	Inside Contactless PicoTAG[2]		
$_h$61	NXP Mifare Classic 1k & 4k	**Mifare Classic**	3.3
$_h$62	NXP Mifare UltraLight	**Mifare UltraLight**	3.4
$_h$71	NXP Desfire 4k	**Desfire**	3.5
$_h$72	Calypso (Innovatron protocol)	**ID only** or **7816-4**	3.2 or 3.7
$_h$FF	All cards supported	**ID only**	3.2

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

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

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

3.1.3. 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.

3.2. ID-only Acceptance Templates

Use an ID-only Acceptance Templates when you want to read the serial number and/or the protocol-related constant bytes from a contactless card, or a group of contactless cards.

Depending on the settings you define in the Lookup List attribute (tag LKL.IDO), the reader may either

- Find any supported contactless card,
- Find only a specific family of contactless cards,
- Find ISO compliant contactless cards.

As you may have more than one ID-only Acceptance Template (up to 4 in fact), you may easily display different types of cards with a different format.

Including card's type in the returned ID is also an interesting option (see 3.2.6.b), as for instance there's no rule to prevent an ISO 14443-B card to have a different serial number than any ISO 14443-A ones.

3.2.1. 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, value = $_{h}$FF all the supported cards. See **3.1.1.a** for details.	1

3.2.2. 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.

3.2.3. 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).

3.2.4. 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, as shown in the following paragraphs.

3.2.5. Role of LOC.IDO with non-ISO cards

A few manufacturers still 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 only the serial number (and don't care for 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

3.2.6. Miscellaneous options

Name	Tag	Description	Size
OPT.IDO	ht4	ID-only miscellaneous options. See table **a** below.	1

a. Miscellaneous option bits

Bit	Value	Meaning
7 – 4		*RFU*
3 – 2		**Position of card's type in the output**
	00	Card type is sent before the prefix[10]
	01	Card type is sent after the prefix and before the ID[11]
	10	Card type is sent after the actual ID[12]
	11	*RFU*
1 – 0		**Send card's type in the output**
	00	Do not send card's type
	01	Send card's type on one byte (2 hex digits) *(see table **b** below)*
	10	Send card's type as a string *(see table **b** below)*
	11	*RFU*

Default value : b00000000

b. Values for card's type byte or string

When OPT.IDO is configured to send card's type in the output, the possible values are :

"Physical" card's type	One byte value	String value	Remark
ISO/IEC 14443 A	h01	" A "	Card must be compliant with Layer 3 or layer 4
ISO/IEC 14443 B	h02	" B "	
ISO/IEC 15693	h04	" V "	
NXP ICODE1	h08	" I "	
Inside Contactless PicoTAG	h10	" i "	Also HID iClass
ST MicroElectronics SR family	h20	" s "	
ASK CTS256B and CTS512B	h40	" a "	
Calypso (Innovatron protocol)	h80	" C "	

[10] The actual frame is <card type><PFX.IDO><card id> (PFX.IDO may be empty)

[11] The actual frame is <PFX.IDO><card type><card id> (PFX.IDO may be empty)

[12] The actual frame is <PFX.IDO><card id><card type> (PFX.IDO may be empty)

3.3. Mifare Classic Acceptance Template

Mifare “Classic” refers to NXP Mifare 1k (MF1ICS50) and Mifare 4k (MF1ICS70) 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.

3.3.1. Lookup list

Name	Tag	Description	Size
LKL.MIF	$_h$t0	Mifare classic lookup list, value = $_h$61. See **3.1.1.a** for details.	1

3.3.2. 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 1	Do not swap bytes Swap bytes
6	0 1	RAW data ASCII encoded data [13]
5	0 1	Left-padding with $_h$0 (RAW) or <SPACE> (ASCII) Right-padding with $_h$F (RAW) or <SPACE> (ASCII)
4	 0 1	**Long string reading option**[14] Disable long string reading option Enable long string reading option
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

[13] 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.

[14] This option is only available on Prox’N’Roll RFID Scanner, RDR-K632 and ProxRunner. If working with IWM-K632 or FunkyGate, please ignore this configuration tag.

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

3.3.3. Output prefix

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

Same as ID-only output prefix (see 3.2.3).

3.3.4. 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, the 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)[15] 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.

[15] 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 .

3.3.5. 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$80 to $_h$8F 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 1	Key is an A key 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
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
3 – 0		*RFU*

3.3.6. Reading a long string from a Mifare Classic card

Note : This option is only available on Prox'N'Roll RFID Scanner, RDR-K632 and ProxRunner.

When bits 4 and 6 in TOF.MIF are set (ASCII output, long string reading extension enabled), the reader behaves as follow :

- The output length (bits 0 to 3 of TOF.MIF) is ignored,
- The reader reads sequentially all Mifare data blocks starting at address specified in LOC.MIF (absolute address or pointer found in MAD), until one of those events occurs :
  - The end-of-string character ('\0' i.e. $_h$00) is read,
  - The end of the card is reached,
  - The authentication failed (see note below),
  - 4 blocks (64 bytes) have been read.

Doing so, the reader is able to fetch ASCII strings up to 64 characters.

Note : in this mode, the reading may cross a sector boundary (64 bytes is 4 blocks, where sectors below 32 are 3-block wide). In this case, the two sectors to be read must be formatted with the same Mifare key and the same access mode.

3.4. Mifare UltraLight Acceptance Template

NXP Mifare UltraLight is a low-cost wired-logic contactless cards. 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.

3.4.1. Lookup list

Name	Tag	Description	Size
LKL.MFU	$_h$t0	Mifare UltraLight lookup list, value = $_h$62. See **3.1.1.a** for details.	1

3.4.2. Output format

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

Same as Mifare Classic output format (see 3.3.2).

3.4.3. Output prefix

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

Same as ID-only output prefix (see 3.2.3).

3.4.4. 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.

3.4.5. Reading a long string from a Mifare UltraLight card

Note : This option is only available on Prox'N'Roll RFID Scanner, RDR-K632 and ProxRunner.

When bits 4 and 6 in TOF.MIF are set (ASCII output, long string reading extension enabled), the reader behaves as follow :

- The output length (bits 0 to 3 of TOF.MIF) is ignored,
- The reader reads sequentially all Mifare data blocks starting at address specified in LOC.MIF (absolute address or pointer found in MAD), until one of those events occurs :
  - The end-of-string character ('\0' i.e. $_h$00) is read,
  - The end of the card is reached,
  - 16 pages (64 bytes) have been read.

Doing so, the reader is able to return ASCII strings up to 64 characters[16].

[16] Well, not really, as Mifare UltraLight currently features only 64 bytes of data, with only 48 bytes actually usable to store data.

3.5. Desfire Acceptance Template

Desfire Acceptance Template has been designed for the first version of NXP Desfire 4k cards (MF3ICD40).

It should work with new Desfire versions (MF3ICD21, MF3ICD41 and MF3ICD81) as long as they are configured to remain compatible with the earlier version (DES or two-key Triple-DES authentication, same ATQ/SAK as MF3ICD40).

3.5.1. Lookup list

Name	Tag	Description	Size
LKL.DFR	$_h$t0	Desfire lookup list, value = $_h$71. See **3.1.1.a** for details.	1

3.5.2. Output format

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

Same as Mifare Classic output format (see 3.3.2).

3.5.3. Output prefix

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

Same as ID-only output prefix (see 3.2.3).

3.5.4. 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[17] (1 to 64).

Default value : unspecified.

Values are MSB first.

[17] Data will be truncated to the length specified in TOF.DFR, unless the long string reading extension is enabled.

3.5.5. T=CL options

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

Same as 7816-4 T=CL options (see 3.5.5).

3.5.6. 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
	10	Diversify the key using HMAC-MD5 algorithm
	11	*RFU*
3 – 0		**Index of key in Desfire application**
	0000 to 1110	Index of the key to be used for authentication
	1111	*RFU*

3.5.7. Reading a long string from a Desfire card

Note : This option is only available on Prox'N'Roll RFID Scanner, RDR-K632 and ProxRunner.

When bits 4 and 6 in TOF.DFR are set (ASCII output, long string reading extension enabled), the reader behaves as follow :

- The output length (bits 0 to 3 of TOF.DFR) is ignored,
- The reader reads the data up to the length specified in LOC.DFR (64 bytes max.),
- The reader returns those bytes as an ASCII string, truncated at the correct length when the end-of-string character ('\0' i.e. $_h00$) is reached.

Doing so, the reader is able to fetch ASCII strings up to 64 characters.

3.6. ISO 7816-4 Acceptance Template

3.6.1. Lookup list

Name	Tag	Description	Size
LKL.TCL	$_h$t0	7816-4 lookup list, $_h11 \leq$ value $\leq {}_h13$. See **3.1.1.a** for details.	1

3.6.2. Output format

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

Same as Mifare Classic output format (see 3.3.2).

3.6.3. Output prefix

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

Same as ID-only output prefix (see 3.2.3).

3.6.4. Location of data

Name	Tag	Description	Size
LOC.TCL	$_h$t3	Offset of data in answer to APDU 3[18] (0 to 127).	1

Default value : 0.

3.6.5. T=CL options

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

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

a. T=CL option bits

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

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

Default value : $_{b}00001111$

3.6.6. 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.

3.6.7. 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.

3.6.8. 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.

3.6.9. Reading a long string from a T=CL card

Note : This option is only available on Prox'N'Roll RFID Scanner, RDR-K632 and ProxRunner.

When bits 4 and 6 in TOF.TCL are set (ASCII output, long string reading extension enabled), the reader behaves as follow :

- The output length (bits 0 to 3 of TOF.TCL) is ignored,
- The reader fetches the data from offset LOC.TCL up to the length of the response to APDU 3 (64 bytes max.),
- The reader returns those bytes as an ASCII string, truncated at the correct length when the end-of-string character ('\0' i.e. $_h$00) is reached.

Doing so, the reader is able to fetch ASCII strings up to 64 characters.

3.7. Calypso Acceptance 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 in our readers, unless you order them with the Calypso option.

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)

3.7.1. Lookup list

Name	Tag	Description	Size
LKL.CYO	$_h$t0	Calypso/Innovatron lookup list, value = $_h$72. See **3.1.1.a** for details.	1

3.7.2. Output format

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

Same as Mifare Classic output format (see 3.3.2).

3.7.3. Output prefix

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

Same as ID-only output prefix (see 3.2.3).

3.7.4. Location of data

Name	Tag	Description	Size
LOC.CYO	$_h$t3	Offset of data in answer to APDU 3[19] (0 to 64).	1

Default value : 0.

3.7.5. Calypso APDU 1

Typically this is a Select Application, or 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.

3.7.6. 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.

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

3.7.7. 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.

4. CONFIGURING IWM-X

ProxRunner RFID Scanner must be configured using a Master Card.

The Master Card must be formatted thanks to **cfgfilecreator.exe** software. See chapters 5 and 6 for details.

When the Master Card has been processed the reader, it sends its firmware version (in the keyboard emulation stream), then restarts.

5. Creating Master Cards using SQ844P software

5.1. Overview

Master Cards for **SpringCard RFID Scanners** are NXP Desfire 4k (MF3ICD40 or MF3ICD41). You may buy them from **SpringCard** or any other NXP reseller.

SpringCard SQ844P is a software package featuring :

- A command line utility, that creates the Master Cards from a Master Configuration File, and using a SpringCard contactless reader/writer[20]
- A wizard (HTML page) that helps authoring the Master Configuration File.

SpringCard SQ844P also includes various configuration files, that show typical configuration for Prox'N'Roll RFID Scanner, IWM-K632, FunkyGate, RDR-K632, ProxRunner, etc.

SpringCard SQ844P is available only for Microsoft Windows systems.

a. Downloading and installing

Go to **www.springcard.com/download/sdks.html** and download latest version of package **sq884p**.

Double-click the downloaded file to launch the installer, and follow the wizard.

b. The cfgfilecreator.exe command line utility

cfgfilecreator.exe is a Windows command line software.

Enter **cfgfilecreator.exe -h** to read the complete list of command line switches and options, and the complete list of sections and variables for configuration files.

cfgfilecreator.exe software comes with various sample configuration files that show typical configurations of IWM-K632, FunkyGate, Prox'N'Roll RFID Scanner, etc.

[20] **SpringCard Prox'N'Roll PC/SC** (or Legacy) typically. CSB4 or any product in the CSB6 family may be used to create Master Cards too.

c. The cfgfilecreator.exe web page

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

5.2. 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** .

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.

5.2.1. The "general" section

This section maps to tags $_h60$ to $_h6F$. Default content is :

```
[general]
opt=05        ; value for OPT
odl=02        ; value for ODL
rdl=0A        ; value for RDF
cld=0F        ; value for CLD
cbz=13        ; value for CBZ
wgd=0A        ; value for WGD
dtc=0A        ; value for DTC
```

```
ser=C5        ; value for SER
shd=00        ; value for SHD
pin=0000      ; value for PIN
```

5.2.2. 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.

5.2.3. Sections for Card Processing Templates

SpringCard RFID Scanners run 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)
```

5.2.4. 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 5.4.1 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 5.4.2 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).

5.3. 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…

5.4. Changing authentication key for Master Cards

All **SpringCard** products ship 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.

SpringCard 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).
    - o 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).
    - o It uses the site specific key for authentication, but doesn't update the key that is already inside the reader.
    - o The *2nd level Master Card* shall be used during installation and whenever you wish to change reader configuration.

Note that more than one *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[21].

> 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…

5.4.1. 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 SpringCard's default keys

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

where *xx…xx* is the site specific 16-byte authentication key[22],

- 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).

5.4.2. Creating a second level Master Card

- Create a complete configuration file as seen earlier .
- 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,

- 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).

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

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

5.5. 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, with dummy keys, to overwrite those keys.

6. Specification of Master Cards

☺

This chapter is provided as a mean for security experts to evaluate the Master Card architecture of **SpringCard RFID Scanners**.

Customers do not need to implement this part themselves, since **cfgfilecreator.exe** software is a convenient tool to create Master Cards. See chapter 5 for details.

6.1. Building a Master Card

- The Master Card must be a Desfire 4k,
- The 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, the reader checks that data comes with a valid digital signature. The signing key can't be changed, and is only known by **SpringCard**'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.

6.2. Template for Master Cards

6.2.1. 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

6.2.2. Authentication key

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

Only **SpringCard** 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 5.4 .

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
	10	Diversify the key using HMAC-MD5 algorithm
	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)*

6.2.3. 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 **SpringCard** genuine software –such as **cfgfilecreator.exe**– is able to sign the Master Cards[23].

Customers shall not try to change this parameter, unless advised to by **SpringCard**.

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
	10	Diversify the key using HMAC-MD5 algorithm
	11	*RFU*
3 – 0	0000	*Those bits are RFU and must be 00*

Specified value : $_{h}$20 *(HMAC-MD5 diversification)*

6.3. Data structure

6.3.1. 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.

6.3.2. 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.

[23] 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.

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 (valid) signature of the whole block, according to the HMAC-MD5 digital signature algorithm specified in next chapter.

6.3.3. 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 (see chapter "Mifare Classic Card Acceptance Template"),
- Len (1 byte) = $_h$06,
- Value is the Mifare key (6 bytes exactly).

6.4. Digital signature

6.4.1. Size of file

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

6.4.2. Algorithm

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

- Let *Content* be the 512-byte configuration block as written in the card[24],
- Let *SignKey* be the 16-byte key,
- Diversify *SignKey* from card's UID, using HMAC-MD5 diversification algorithm[25] to get *DivKey*,
- Compute *Sign* = HMAC-MD5 (*Block*) using *DivKey* [26].

The value of *SignKey* is confidential. Customers shall not try to change the key, nor the signature algorithm.

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

[25] See next chapter "Security algorithms"

[26] See next chapter "Security algorithms"

7. SECURITY ALGORITHMS

7.1. HMAC SIGNATURE AND KEY DIVERSIFICATION

7.1.1. 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.

a. HMAC 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.

b. HMAC-MD5

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.

In the **SpringCard RFID Scanners** family, we use HMAC-MD5 for both signature and key diversification.

7.1.2. HMAC-MD5 for digital signature

HMAC protects both message's integrity and authenticity, so it can be considered as a digital signature[27].

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

7.1.3. 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*.

[27] 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.

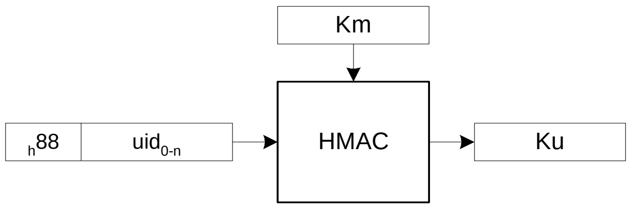
a. DES or Triple-DES key diversification

The algorithm takes as inputs :

- A 16-byte master key (Km)
- The card serial number (uid)[28]

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.

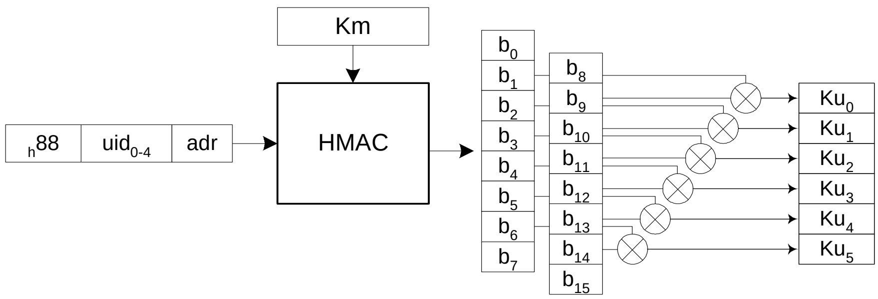
b. Mifare key diversification

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).



Note : the *adr* parameter is the either the sector number (not the block) number) or fixed to $_h00$, depending on the configuration in the Mifare Classic Card Acceptance Template.

[28] 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.

7.2. DESFIRE SAM / RC171 KEY DIVERSIFICATION

7.2.1. DES or Triple DES key diversification

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

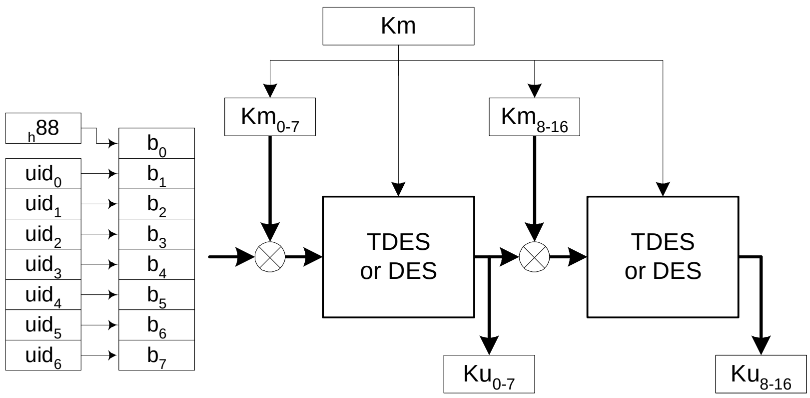
The algorithm takes as inputs :

- A 16-byte Triple-DES master key (Km)[29]
- 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.

7.2.2. Mifare key diversification

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

a. Basis

The algorithm takes as inputs :

- A 6-byte master key (Km)
- A 16-byte Triple-DES diversification key (Kd)[30]

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

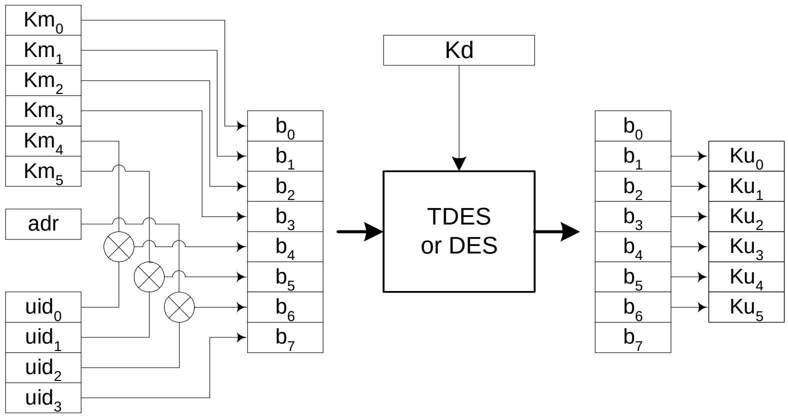
[30] If both halves are equals, the key maps to a single DES key

- 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 :



b. Diversification based on UID only

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

c. Diversification based on UID and address

If this option is selected, the *adr* input parameter is the Mifare sector number (not the block).

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

This document is provided for informational purposes only and shall not be construed as a commercial offer, a license, an advisory, fiduciary or professional relationship between PRO ACTIVE and you. No information provided in this document shall be considered a substitute for your independent investigation.

The information provided in document may be related to products or services that are not available in your country.

This document is provided "as is" and without warranty of any kind to the extent allowed by the applicable law. While PRO ACTIVE will use reasonable efforts to provide reliable information, we don't warrant that this document is free of inaccuracies, errors and/or omissions, or that its content is appropriate for your particular use or up to date. PRO ACTIVE reserves the right to change the information at any time without notice.

PRO ACTIVE does not warrant any results derived from the use of the products described in this document. PRO ACTIVE will not be liable for any indirect, consequential or incidental damages, including but not limited to lost profits or revenues, business interruption, loss of data arising out of or in connection with the use, inability to use or reliance on any product (either hardware or software) described in this document.

These products are not designed for use in life support appliances, devices, or systems where malfunction of these product may result in personal injury. PRO ACTIVE customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify PRO ACTIVE for any damages resulting from such improper use or sale.

COPYRIGHT NOTICE

EDITOR’S INFORMATION

PRO ACTIVE SAS company with a capital of 227 000 €
RCS EVRY B 429 665 482
Parc Gutenberg, 13 voie La Cardon
91120 Palaiseau – France