AN 073120

mifare Ultralight Features and Hints

Rev. 2.0 — 18 December 2006 Application note

Document information

Info	Content
Keywords	Multiple ticketing, secured data storage, implementation hints
Abstract	This document presents features and hints for a secured and optimized application development using mifare® Ultralight cards.

Revision history

Rev	Date	Description
2.0	18.12.2006	Security features (section 2.2) and example (annex 6.3) added
1.0	15.05.2002	First release

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

1.1 Purpose and Scope

This application note is intended to describe the features of the mifare® Ultralight MF0ICU1, and the comparison to the mifare® Classic MF1ICS50.

It addresses some security mechanisms which may be used to protect the data stored in the card and demonstrates the implementation of mifare® Ultralight into an existing mifare® Classic application and shows the necessary modification over the existing mifare® reader environment. The changes apply to all relevant Philips mifare® readers.

1.2 How to use this document

This document contains a collection of hints and features that could be of interest for those, who plan to use the mifare® Ultralight.

None of this information is intended to replace any of the relevant datasheets or design guides.

1.3 Reference documents

1. [M028630] mifare® Ultralight, Contact-less Single-trip Ticket IC MF0 IC U1, Functional Specification.

2. [M011731] mifare®, (Card) Coil Design Guide.

3. [sl070010] Temperature Management, Inlet Design.

4. [M057432] MIFARE® MF RC530 ISO14443A reader IC.

5. [M056633] MIFARE® MF RC531; ISO 14443 Reader IC.

6. [MC073933] MIFARE® and I Code CL RC632 Multiple protocol contact less reader IC.

7. [ISO/IEC 14443-3] Identification cards- Contactless integrated circuit(s) cards- proximity cards- Part 3: Initialization and anticollision.

8. [FIPS46-3] Data Encryption Standard.

9. [ISO/IEC 9797-1] Information technology Security techniques Message Authentication Codes.

2. Mifare® Ultralight application hints

2.1 Memory features

In addition to the user memory area the mifare® Ultralight offers the features of an OTP[1] area and lock bytes to lock the OTP and user area. The usages of LOCK bits are described in [M028630].

A mifare® Ultralight dedicated 4-byte WRITE-command provides a high transaction speed. All the add-on features are dedicated to support special application functionality e.g. ticketing.

2.1.1 Using OTP memory for multiple ticketing

The 4 OTP bytes, which are pre-set to “0” at the delivery, can be set to “1” only once. This gives the possibility to interpret them as an irreversible counter. Therefore, the number of “1” bits in Page 3 can be considered as counter value. This counter can be incremented by changing a bit from “0” to”1”. The counter cannot be decremented due to a “1” bit of Page 3 cannot be cleared. In this case this 4-byte offers a number of 32 states that could be used to allow a certain number of passing turn-styles.

Example:

An example of a four rides ticket is shown in figure 1. For this application a ticket issuing the OTP memory of the mifare® Ultralight has to be pre-set to “FFFFFFF0hex”.

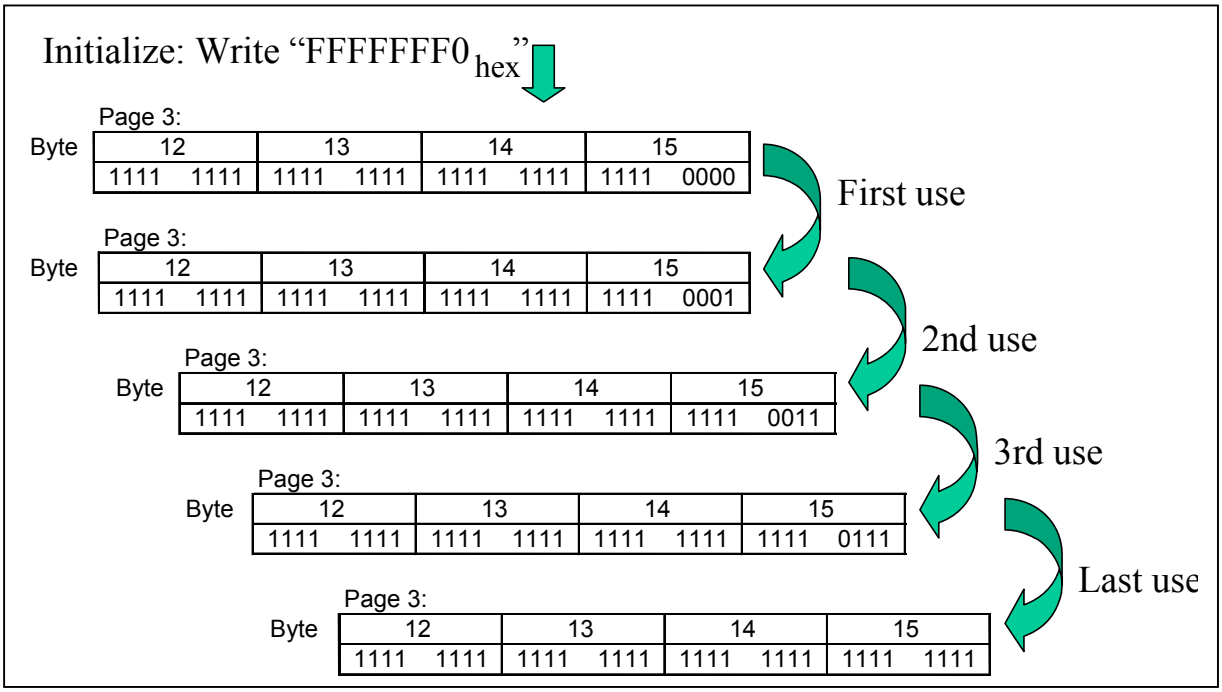


Figure 1: Example of a four rides ticket

[1] **O**ne **T**ime **P**rogramming

For every access, the 4-byte OTP (page 3) has to be checked and updated (as shown in Figure 2) to ensure the validity of the tickets. Using the same command flow the counter might be extended to a number of 31 times just by changing the initial memory content.

E.g. “FFFFFC00hex” or “FC00FC00hex” might be used for a 10 times counter initial value, or “FFF00000hex” for a 20 times counter initial value, which has to be written into the OTP memory while issuing the ticket.

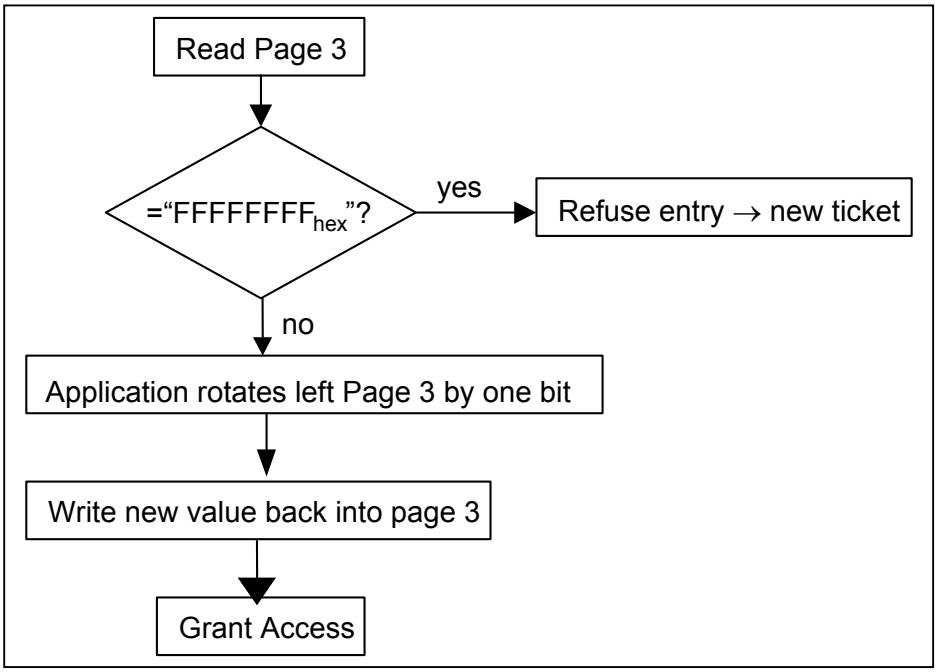


Figure 2: Ticket Counter Command flow

Remark:

With the initial value “00000000hex” for 32 times counter, the rotate left at the very first access check after selling the ticket has to be exchanged into another initial WRITE command.

2.1.2 Transaction Speed

Although the mifare® Ultralight offers the 16-byte Compatibility Write command to be compatible with the mifare® Classic environment, the use of the 4 byte Write command is recommended, if the application requires a fast transaction. The Write saves approximately 20% of the transaction time[2] compared to the Compatibility Write, as can be seen in Table 1.

Command	4 pages		12 pages	
REQA	0.4 ms		0.4 ms	
Anticollision level 1 + 2 & Select	3.7 ms		3.7 ms	
Read	2.0 ms		6.0 ms	
Write	18.7 ms		56 ms	
Compatibility Write	24.0 ms		72 ms	
Halt	0.5 ms		0.5 ms	
∑ (approximately)	25.3 ms	30.6 ms	66.6 ms	82.6 ms

Table 1: Transaction time

The transaction time[2] as shown in Table 1 counts the time needed for

- the communication from the reader to the mifare® Ultralight,
- the response time of the mifare® Ultralight and
- the communication from the mifare® Ultralight back to the reader.

[2] This doesn't include the time requires the host computer for the application itself (e.g. calculating & checking ticket values, displaying results, opening gates, etc.).

2.2 Proposed Security Mechanism

Mifare® Ultralight has been designed to support the faster application with the cheapest solution. Therefore it does not address any security feature except the unique identity (UID). From the application point of view this means, no authentication has to be performed and no key is needed. Performing a mifare® authentication generates an error, and the mifare® Ultralight goes back to the Idle (or Halt) state.

But if requires, smarter secured application using mifare® Ultralight can be created using an intelligent reader with the respective application software. A lot of cryptographic technologies are available to assist you. One successfully implemented approach is demonstrated in the following sections. This approach uses a 16-byte Master key (Mk), which is used to add the confidentiality and integrity of the storage data.

2.2.1 Confidentiality of stored Data

As the mifare® Ultralight pages can be read without any authentication, anyone can read the pages using any standard reader. But if the stored data is encrypted with a secured key then these are just some bytes to one who does not have the secret key and information regarding the encryption method. Therefore by storing the encrypted data in mifare® Ultralight memory, one can add the confidentiality to the data itself.

As an example a 3-DES [FIPS46-3] may be used for this encryption.

$$Data_{stored} = f\left(key, data_{origin}, UID\right)$$

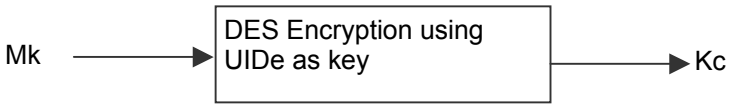
7-byte UID of the mifare® Ultralight has to be incorporated in the security system to confirm the uniqueness of the tickets and a 16-byte Master Key (Mk) has to be defined by the application provider. To decrease the threat on the Master key (Mk), for each card, a Card key (Ck) is derived from the Master Key (Mk) using the well known key diversification. The steps to be followed are:

- Expand the 7-byte UID,(56 (7x8) bits (b_{55} - b_0)) to 64 bits to get 8-byte extended UID (UIDe) by adding the odd parity (p) in the least significant bit position for each 7 bits as shown in table 2: (Byte0 to Byte7, 8 bytes are the bytes of UIDe whereas b0-b55 are the bits of original UID)

$p_7 b_{55} b_{54} b_{53} b_{52} b_{51} b_{50} b_{49} p_7$	$p_6 b_{48} b_{47} b_{46} b_{45} b_{44} b_{43} b_{42} p_6$	$p_5 b_{41} b_{40} b_{39} b_{38} b_{37} b_{36} b_{35} p_5$	.....	$p_1 b_{13} b_{12} b_{11} b_{10} b_9 b_8 b_7 p_1$	$p_0 b_6 b_5 b_4 b_3 b_2 b_1 b_0 p_0$
Byte7	Byte6	Byte5	.....	Byte1	Byte0

Table 2: Expanding of the UID to get UIDe

- Derive the Card key (Ck) as follows:



- Encrypt the plain data using the Card Key (Ck) in CBC (send) mode.
- Use standard initial vector (IV) of all '00's, IV= "0000000000000000_h"
- As DES works with 8-byte block wise, organize the data in multiple of 8 by adding the standard padding [ISO/IEC 9797-1] with all zeros '00_h'. As example ('xx' is the data bytes):

```
2 padding bytes: xx xx xx xx xx xx 00 00
7 padding bytes: xx 00 00 00 00 00 00 00
```

The complete scheme is shown in the following figures (figure 3 & figure 4):

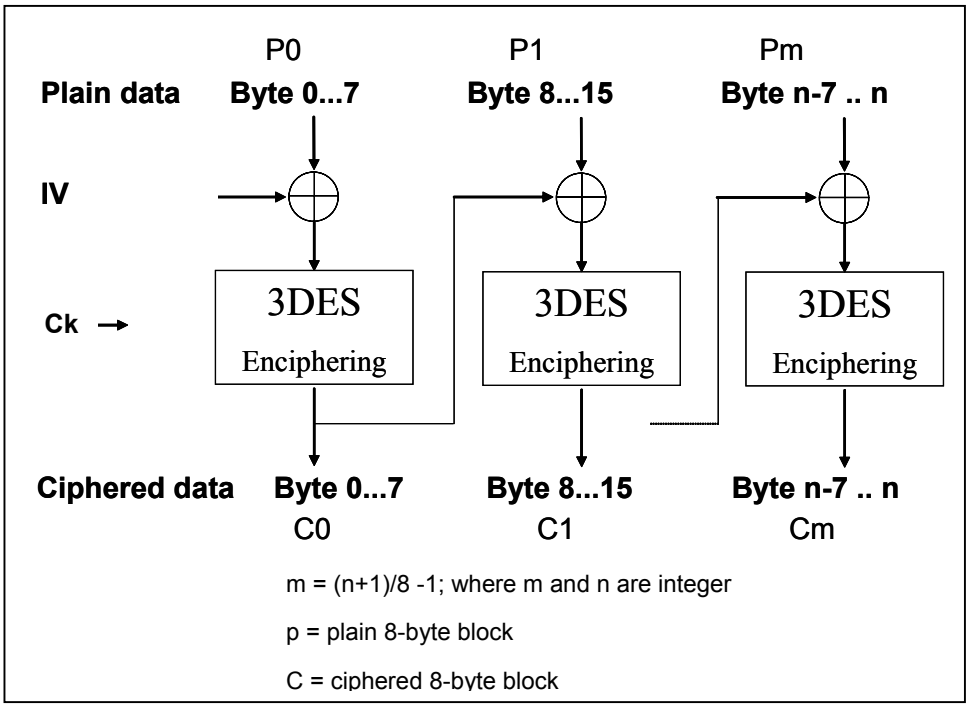


Figure 3: Data encryption scheme

Therefore the data has to be decrypted after reading it from the card.

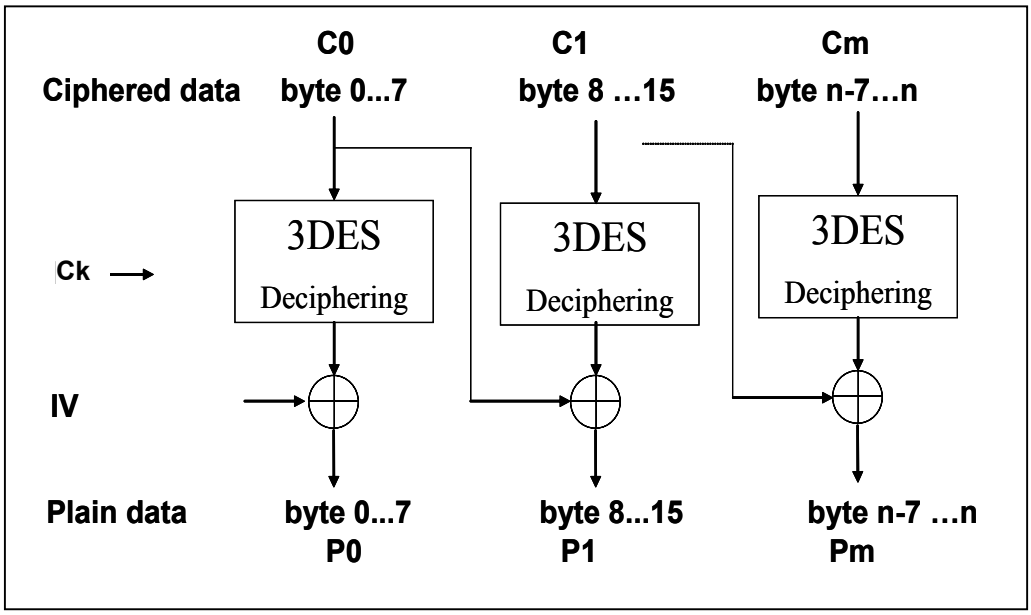


Figure 4: Data decryption scheme

2.2.2 Integrity of stored Data

As the mifare® Ultralight pages can be updated without satisfying any security measures (anyone can easily update the memory), the content of the memory lacks guaranteed integrity. To avoid this inconvenience we propose a security checksum which has to be calculated over the bytes in pages 2 to used end and has to be appended with the data. For this purpose MAC (Message Authentication Code) [ISO/IEC 9797-1] may be a good choice. The complete scheme is shown in figure 5:

$$Checksum = f\left(key, usedmemory(inc.page2\ \&\ 3), UID\right)$$

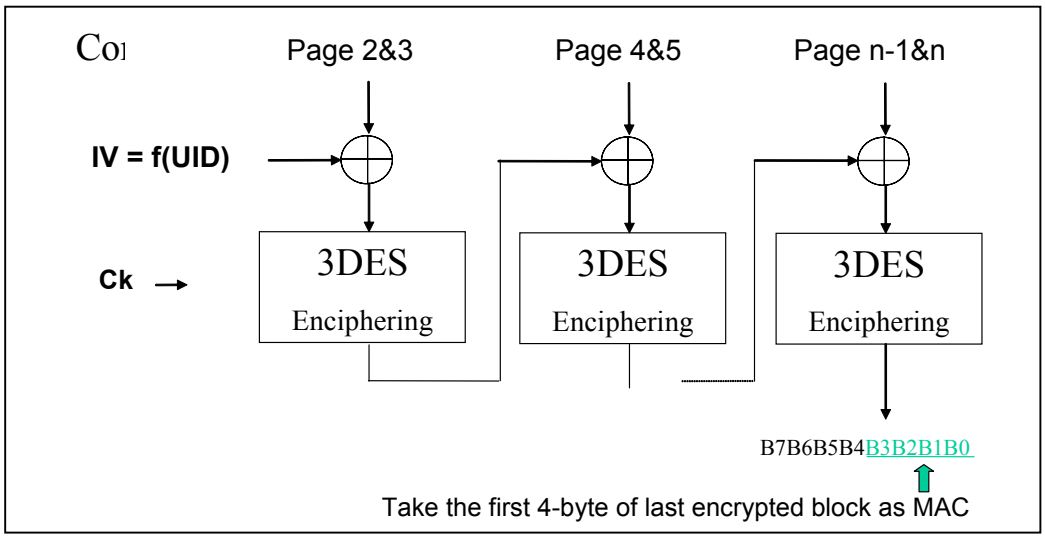


Figure 5: MAC calculation

- Use the same Initial Vector (IV) and Card Key(Ck) used in the previous section

- If number of used pages is odd or number of data bytes is not a multiple of 8-byte, add standard padding with all '00'. As example ('xx' is the data bytes):

```
3 padding bytes: xx xx xx xx xx 00 00 00
6 padding bytes: xx xx 00 00 00 00 00 00
```

Either the encrypted data and/or the checksum can be stored in the mifare® Ultralight user memory. In this case **the data is protected against damage and being copied** from one mifare® Ultralight to another one, as long as the key is kept secret.[3]

If high level security features are required, other members of the mifare® card IC family can be used in the application, e.g. the mifare® Classic or the mifare® ProX.

Please note, the DES operation may be performed using a DESFire® SAM module. This SAM module will facilitate the system in the following ways:

[3] However, a complete security against replay attack can not be provided with the mifare® Ultralight alone, here also an appropriate application software has to ensure reply attack resistance.

- The key can be stored once
- The key cannot be read back
- The module provide one step functions for calculation of DES/ 3DES
- The Cryptographic operations are fast enough for real time operations.

A complete worked out numerical example is added in the appendix 6.3

3. Using mifare® Ultralight in an existing mifare® Classic application

The mifare® Ultralight offers the feature to be used in an existing mifare® Classic application. Therefore the mifare® Ultralight command structure is compatible to the mifare® Classic one.

Because the mifare® Ultralight addresses a different application category (single trip ticketing, fast and cheap transactions), some application changes have to be done anyway, but the existing mifare® transaction command structure and the mifare® Reader may be used with the mifare® Ultralight (as shown in Figure 6).

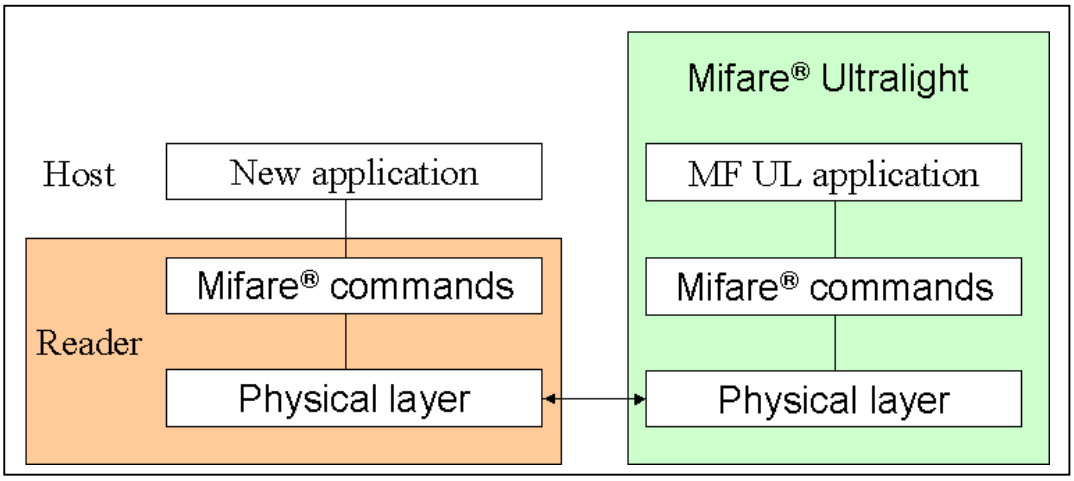


Figure 6: mifare® Ultralight in an existing mifare® Classic application

Differences: mifare® Classic - mifare® Ultralight

The basic differences between mifare® Classic and mifare® Ultralight are:

I) The mifare® Ultralight uses a **7-byte UID** instead of 4-byte (like mifare® Classic). There are 2 possibilities to select a mifare® Ultralight card. It suggests to use the anti-collision cascade level 2 (as specified in the ISO14443A - 3) to get the complete UID and select one mifare® Ultralight (see 3.1.1).

If the reader does not support the ISO cascade level 2 anti-collision and there is no chance to update the reader to do so, a combination of anti-collision cascade level 1 (using the first 3 bytes of the UID) and a read of block 0 after the Select can be used instead. Please note that this workaround has the limitation that there is no chance to fully RESOLVE a collision between two cards in case of the unlikely event, that the

first part of the UID is equal. The collision can only be DETECTED, allowing the reader to inform the user to present only one card to the reader (see 3.1.2).

II) The mifare® Ultralight doesn't need the authentication and no keys, as it uses **no encryption**.

III) The mifare® Ultralight only uses **"Read", "Write" and "C.Write"** (mifare® Classic compatible write command with 16 Bytes).

No Value-commands are used.

3.1 Transaction Command Flows

3.1.1 Transaction flow using Cascade Level 2

The anti-collision cascade level 1 and 2 (as described in the ISO14443-A part 3) should be used to select a mifare® Ultralight. This command sequence gives back the complete 7-byte UID of the mifare® Ultralight, and allows selecting only one card (as shown in Figure 7).

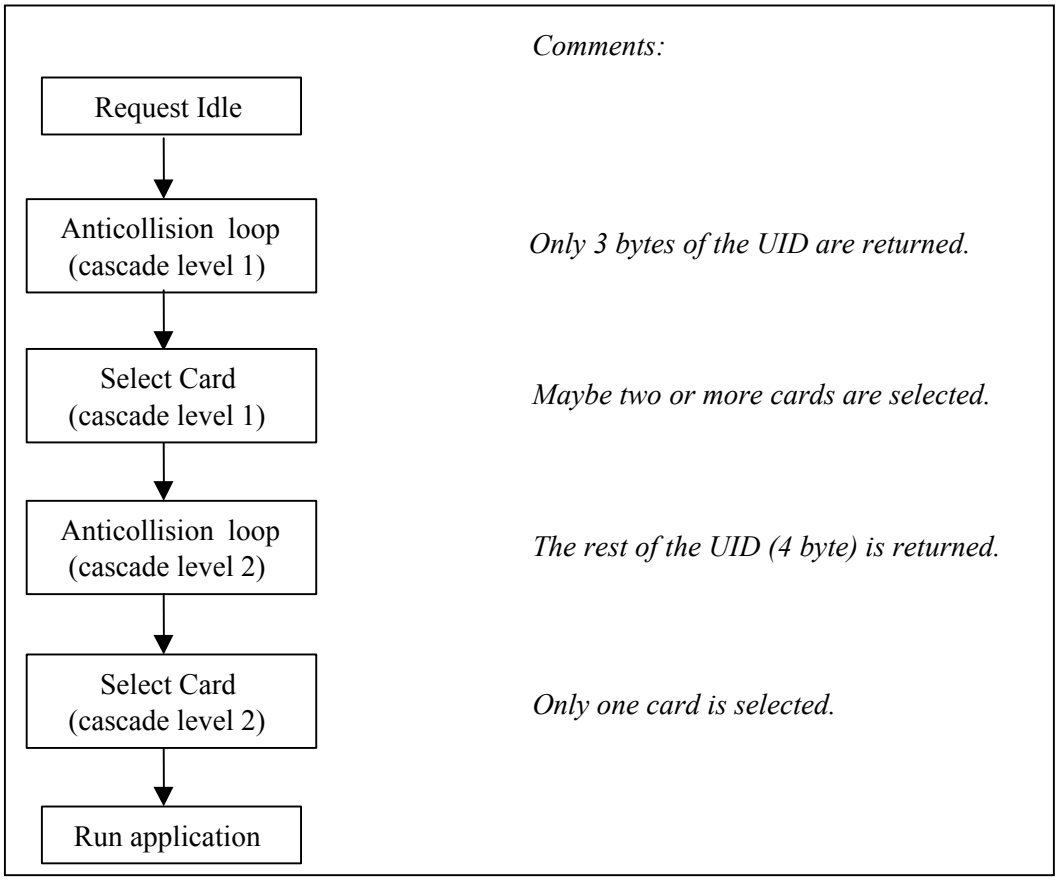


Figure 7: Transaction flow with Cascade Level 2

3.1.2 Transaction flow using Cascade Level 1 & Read Block 0

If the reader does not support the anti-collision cascade level 2, only the anti-collision cascade level 1 (ISO14443A - 3) can be used to select a mifare® Ultralight. This is the "Classic mifare Anti-collision and it returns 3 significant bytes of the UID. In this case the complete UID shall be checked after selection with a read of block 0 to make sure, that only one card is selected. **If a collision is detected** during that read of block 0, the user has to be informed, that **only one card** has to be presented to the reader (see Figure 8).

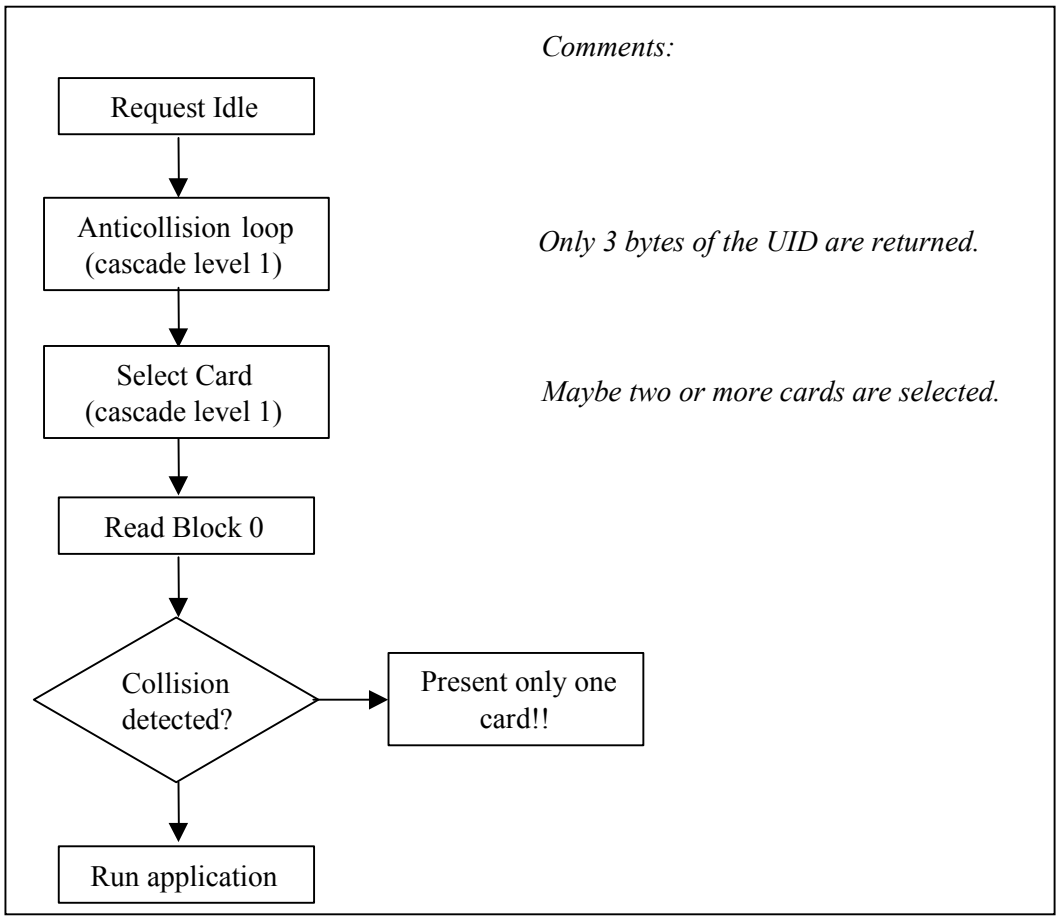


Figure 8: Transaction flow with Cascade Level 1 & Read Block 0

Remark:

This command flow as given in 3.1.2 doesn't follow the ISO14443, and only offers a compatible command flow to work with some old reader environment. If possible, the use of the complete anti-collision cascade level 1 & 2 is recommended.

3.2 mifare® Ultralight and mifare® Reader

The mifare® Ultralight can be selected, read, and written by every mifare® Classic Reader.

The authentication has to be skipped, and the selection of the mifare® Ultralight has to be done as shown either in Figure 7 or Figure 8.

A mifare® Classic Read command can be used. In this case only the first 4 Bytes contain valid data according to the addressed page, the other 12 bytes refer to the next 3 pages (see the related datasheet of the mifare® Ultralight).

To write data into the memory of the mifare® Ultralight, either the (4-byte) WRITE or the COMPATIBILITY WRITE can be used (see the related datasheet of the mifare® Ultralight).

Reader Modules:

Reader	Anti-collision	WRITE	Comment
MF CM200	cascade level 2 possible, but LLL[4] has to be adapted[5]	possible, but LLL has to be adapted	supports mifare® Ultralight
MF CM500	cascade level 2 possible, but LLL has to be adapted[6]	possible, but LLL has to be adapted	supports mifare® Ultralight

Reader Devices:

Reader	Anti-collision	WRITE	Comment
MF RD260	only cascade level 1, no firmware update or extension possible	only COMPATIBILITY WRITE, no firmware update or extension possible	supports mifare® Ultralight only in compatibility mode
MF RD560	only cascade level 1, no firmware update or extension possible	only COMPATIBILITY WRITE, no firmware update or extension possible	supports mifare® Ultralight only in compatibility mode

[4] **L**ow **L**evel **L**ibrary

[5] example see 6.2

[6] example see 6.2

Reader ICs:

Reader	Anti-collision	WRITE	Comment
MF RC171	full cascade level 2 possible, but LLL has to be adapted	possible, but LLL has to be adapted	supports mifare® Ultralight[7]
MFRC500	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	supports mifare® Ultralight
MF RC530	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	supports mifare® Ultralight
MF RC531	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	supports mifare® Ultralight
CL RC632	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	supports mifare® Ultralight
MF RC522	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	supports mifare® Ultralight
MF RC523	BFL contains the full cascade level 2 support	BFL contains the full 4 byte WRITE support	supports mifare® Ultralight

4. mifare® Ultralight Coil design hints

The mifare® Ultralight chip is available in two versions: either the **"standard" version MF0ICU10** with an input capacitance of approximately 17pF or a **high capacitance version MF0ICU11** with approximately 50pF. For a complete coil design please refer to the "mifare® (Card) Coil Design Guide" [M011731].

Using the standard version of the mifare® Ultralight chip it's recommended to use **the same coil design for the mifare® Ultralight as for the mifare® Classic**. Although the mifare® Ultralight has a slightly higher capacitance than the mifare® Classic (by 0.5pF), the same coil design should be used to result in a slightly lower resonance frequency. This lower resonance frequency increases the overall performance of cheap antennas and ensures a similar performance compared to mifare® Classic – but has its limitation, if multiple cards operate simultaneously in the field.

5. delivery types

The mifare® Ultralight is available in a bumped version for flip chip assembly and in the Philips Flip Chip Package FCP2. For coil design issues it's recommended to use the Philips application notes "mifare® (Card) Coil Design Guide" [M011731] and "Temperature Management, Inlet Design" [sl070010].

[7] example see 6.1

6. Appendix

6.1 MF RC171 low level library extension: Cascade Anticollision

```
/*****************************************************************************/
int CALL_CONV MfPiccCascAnticoll (unsigned char select_code,
                                  unsigned char bcnt,
                                  unsigned char *snr)
/*****************************************************************************/
{
  int           status;
  unsigned char snr_chk = 0;
  int           i;

  if (MfAssertMode(select_code,0x93|0x95|0x97))
      return (MI_WRONG_PARAMETER_VALUE);

  MfOutp(ENABLE, _PEN | _PRE);          // CRC-disable, Parity enable
  MfOutp(MODE  , __mode);               // __mode preset
  MfOutp(BCNTS ,(unsigned char)(bcnt + 16));           // 16 + number of
bits
  MfOutp(STACON, (unsigned char)(__stacon|_AC));        // anticollision-
mode
  MfDelay50us(4);                                  // BUS-access not allowed
                                                   // for 35us
  MfOutp(DATA, select_code);            // "SELTYPE" of MIFARE1
  MfOutp(DATA, (unsigned char)(((2 + (bcnt >> 3)) << 4) | (bcnt &
0x07)));
                                                    // bytecount higher nibble
                                                    // bitcount  lower  nibble
                                                    // incl. first 2 bytes!!
  for (i = 0; i < (bcnt + 7)/8; i++)
  {
    MfOutp(DATA, snr[i] );
  }
  MfOutp(TOC, TIMEOUT_14443_3); // set timeout
  while (!((status = MfInp(STACON)) & _DV));
  MfOutp(TOC, 0);          // reset timer

  if ((status = MfInp(STACON)) & (_TE | _BE))       // any error
  {
     if (status & _TE)
       return (MI_NOTAGERR);
     if (status & _BE)
     {
       MfDelay50us(10);             // delay 500us
       return (MI_BITCOUNTERR);
     }
  }
  for (i = 0; i < 4; i++)
  {
     snr[i] = MfInp(DATA);
     snr_chk ^= snr[i];
  }
  snr_chk ^= MfInp(DATA);
  // serialnumber check
  if (snr_chk)
    return (MI_SERNRERR);
  return (MI_OK);
}
```

6.2 MF CM200 / CM500 low level library extension: Cascade Anticollison

```
/*****************************************************************************/
int CALL_CONV MfPiccCascAnticoll (unsigned char select_code,
                                  unsigned char bcnt,
                                  unsigned char *snr)
/*****************************************************************************/
{
  int           status;
  unsigned char snr_chk = 0;
  int           i;

  if (MfAssertMode(select_code,0x93|0x95|0x97))
      return (MI_WRONG_PARAMETER_VALUE);

  MfOutp(ENABLE, _PEN | _PRE);          // CRC-disable, Parity enable
  MfOutp(MODE  , __mode);               // __mode preset
  MfOutp(BCNTS ,(unsigned char)(bcnt + 16));           // 16 + number of
bits
  MfOutp(STACON, (unsigned char)(__stacon|_AC));        // anticollision-
mode
  MfDelay50us(4);                                 // BUS-access not allowed
                                                  // for 35us
  MfOutp(DATA, select_code);            // "SELTYPE" of MIFARE1
  MfOutp(DATA, (unsigned char)(((2 + (bcnt >> 3)) << 4) | (bcnt &
0x07)));
                                                   // bytecount higher nibble
                                                   // bitcount  lower  nibble
                                                   // incl. first 2 bytes!!
  for (i = 0; i < (bcnt + 7)/8; i++)
  {
    MfOutp(DATA, snr[i] );
  }
  MfOutp(TOC, TIMEOUT_14443_3); // set timeout
  while (!((status = MfInp(STACON)) & _DV));
  MfOutp(TOC, 0);          // reset timer

  if ((status = MfInp(STACON)) & (_TE | _BE))       // any error
  {
     if (status & _TE)
       return (MI_NOTAGERR);
     if (status & _BE)
     {
       MfDelay50us(10);             // delay 500us
       return (MI_BITCOUNTERR);
     }
  }

  for (i = 0; i < 4; i++)
  {
     snr[i] = MfInp(DATA);
     snr_chk ^= snr[i];
  }
  snr_chk ^= MfInp(DATA);
  // serialnumber check
  if (snr_chk)
    return (MI_SERNRERR);
  return (MI_OK);
}
```

6.3 Worked out example of proposed security mechanism

Let's take a mifare® Ultralight card with

	UID	= 046EBF11127A00$_h$, (7 byte UID (Byte0…..Byte7))
	UIDe	= 09B8FA1B42843C00$_h$
Master Key	Mk	= 0F1E2D3C4B5A6978FFA83720E1735A0C $_h$
Initial vector	IV	= 0000000000000000$_h$
Card Key	Ck	= 5A35ED688ABDFCB4AAFAF15467DC1DAB$_h$
Say user plain data	Name	= ABC UVWXYZ
	Date of Birth	= 230780
	ID	= P092541
	Status	= VIP

Plain data has to be written = <ABC UVWXYZ 230780 P092541 VIP>

ASCII of the user plain data P=
3C41424320555657585 95A20020300070800205000090205040120564 9503E$_h$

As number of byte = 31, let's put padding '00'$_h$.

So after padding, P =
3C41424320555657585 95A20020300070800205000090205040120564 9503E00$_h$

After encryption, C = 72F20A176F32EC4BDDC3257155A4C85EF4B21B1F545C65467914219ED33F48DD$_h$

So the memory content could be as follows:

Byte	0	1	2	3	Page	
SNR	04	6E	BF	5D	0	
SNR	11	12	7A	00	1	
I/L	79	C8	00	00	2	
OTP	00	00	00	00	3	
R/W	72F5	F2FE	0AFD	1771	4	Encrypted user data
R/W	6F97	3256	ECEE	4B37	5	
R/W	DD1A	C31C	2546	715C	6	
R/W	5534	A471	C896	5E95	7	
R/W	F48F	B271	1BDD	1F77	8	
R/W	54F0	5CC3	653F	4683	9	
R/W	79AC	143D	2137	9E23	10	
R/W	D39F	3F40	480F	DDDC	11	
R/W					12	
R/W					13	
R/W					14	
R/W					15	

Figure 9: Memory with encrypted user data

Now Let's calculate the MAC.

For checksum calculation content of page 2 and page 3 will be considered together with the data to be written.

So the content on which MAC will be calculated = 79C800000000000072F20A176F32EC4BDDC3257155A4C85EF4B21B1F545C65467914219ED33F48DD$_h$

The same key and IV is used as described in section 2.2.2 and the calculated

MAC = **B81DCBDD**$_h$ (4 LSB bytes of last block "**B81DCBDD**B68212DF")

So the final memory will look like as in figure 10.

(If it is necessary page 4 may be used to present the header which can show data storage type, length etc.)

Byte	0	1	2	3	Page	
SNR	04	6E	BF	5D	0	
SNR	11	12	7A	00	1	
I/L	79	C8	00	00	2	
OTP	00	00	00	00	3	
R/W	72F5	F2FE	0AFD	1771	4	Encrypted user data
R/W	6F97	3256	ECEE	4B37	5	
R/W	DD1A	C31C	2546	715C	6	
R/W	5534	A471	C896	5E95	7	
R/W	F48F	B271	1BDD	1F77	8	
R/W	54F0	5CC3	653F	4683	9	
R/W	79AC	143D	2137	9E23	10	
R/W	D39F	3F40	480F	DDDC	11	
R/W	B89E	1D54	CB15	DDA1	12	MAC
R/W	XX	XX	XX	XX	13	Not Used
R/W	XX	XX	XX	XX	14	
R/W	XX	XX	XX	XX	15	

Figure 10: Final memory with encrypted data and MAC

An example application flow diagram is shown in the following:

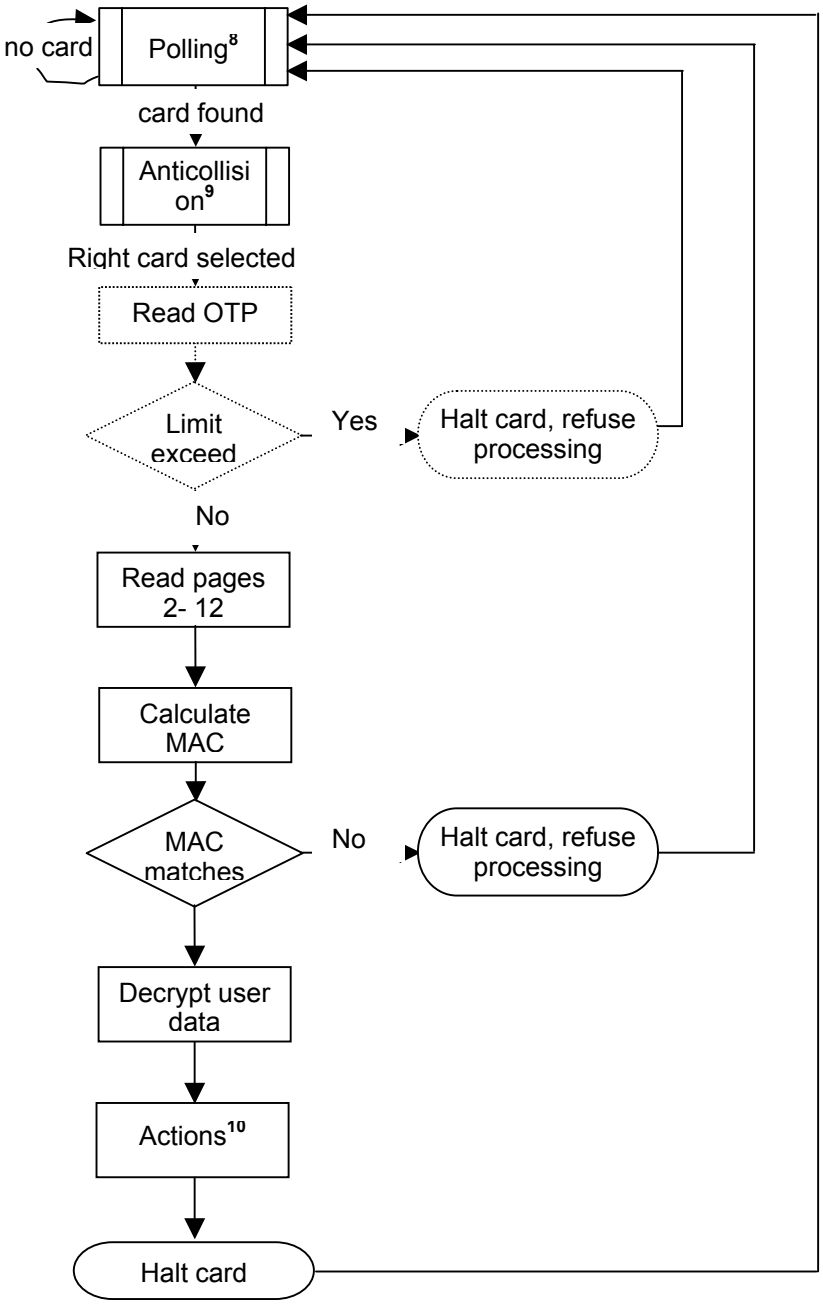


Figure 11: Example application flow diagram

Dotted blocks may be avoided if the OTP bytes are not used

[8] Pre-defined process for card detection, reader sends always REQA and check if there is any answer.

[9] Standard anticollision [ISO/IEC 14443-3], which includes the selection of the right card (also from the multiple cards).

[10] If OTP or any memory content is updated, MAC has to be recalculated and rewritten.

7. Legal information

7.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

7.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

Mifare®— is a trademark of NXP B.V.

8. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, email to: salesaddresses@nxp.com

Date of release:December 2006
Document identifier: <DOC_ID>