

AN12196

NTAG 424 DNA and NTAG 424 DNA TagTamper features and hints

Rev. 1.8 — 17 November 2020
507218

Application note
COMPANY PUBLIC

Document information

Information	Content
Keywords	NTAG 424 DNA, NTAG 424 DNA TagTamper, Configuration, Personalization
Abstract	Guidelines for personalization, configuring and backend calculations of NTAG 424 DNA



Revision history

Rev	Date	Description
v 1.8	20201117	Typo corrected in [Table 19]
v 1.7	20201112	Typo corrected in [Table 19]
v 1.6	20201022	Typo corrected in [Table 17] , [Table 18] . Details added for [Section 7.4]
v 1.5	20190730	Editorial changes
v 1.4	20190612	More details added on Asymmetric Originality Check procedure
v 1.3	20190322	Typos corrected in chapters Section 4.4.2 and Section 7.2
v 1.2	20190211	Personalization steps updated, Originality Signature check added
v 1.1	20190129	Security status changed into "Company Public"
v 1.0	20181031	Initial version

1 Abbreviations

Table 1. Abbreviations

Acronym	Description
AES	Advanced Encryption Standard
AID	Application IDentifier
APDU	Application Protocol Data Unit
DF-Name	ISO7816 Dedicated File Name
C-APDU	Command APDU
CMAC	MAC according to NIST Special Publication 800-38B
CRC	Cyclic Redundancy Check
IC	Integrated Circuit
KDF	Key derivation function
LRP	Leakage resilient primitive
LSB	Lowest Significant Byte
LSb	Lowest Significant bit
MAC	Message Authentication Code
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
NVM	Non-volatile memory
PCD	Proximity Coupling Device
PICC	Proximity Integrated Circuit Card
PRF	Pseudo Random Function
R-APDU	Response APDU (received from PICC)
SDM	Secure Dynamic Messaging
SSM	Standard Secure Messaging
SUN	Secure Unique NFC Messaging
UID	Unique IDentifier
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

2 Introduction

NTAG 424 DNA introduces a feature called Secure Dynamic Messaging (SDM), which returns a unique secure dynamic response at each tap. NFC forum devices, which have built-in NFC hardware (e.g. NFC mobile phones, tablets), open the link without any dedicated application installed in the device. The "tap unique" NDEF message offers to the backend system (e.g. cloud) a unique tag identifying.

2.1 About this document

This document addresses developers who are developing application based on NTAG 424 DNA.

This application note is a supplementary document for implementations using the NTAG 424 DNA. This document shall be used in addition to NTAG 424 DNA data sheet [\[1\]](#). The best use of this application note is achieved by reading the mentioned data sheet in advance.

Note: This application note does not replace any of the relevant functional specifications, data sheets, or design guides.

2.2 Key benefits using NTAG 424 DNA

- More advanced security, through cryptographic authentication and unique authentication data mirror with each tap
- Stronger protection of goods and documents, with tap-to-check content originality, integrity, authenticity
- Enhanced user engagement, with unique content experiences served in real time (e.g. cloud)
- Easy user adoption, through automatic tag connection to web services –no dedicated app needed

2.3 Target applications

NTAG 424 DNA is attractive for many applications. To name few in the list below:

- **Advanced anti-counterfeiting**
Verify authenticity of physical goods and identify sales outside authorized markets.
- **Secured exclusive user experiences**
Reward customers with truly exclusive and personalized content, offers, and privileges.
- **Secured sensitive data applications**
Protect sensitive product and user data, or trigger an action upon a verified incidence.g. payment.
- **Document Authentication**
Authenticate originality and track provenance of documents that bear credentials.
- **Protected monetary offers**
Confer trust to proximity transactions such as coupons, promotions, or loyalty points.
- **Secure authentication and configuration of closed loop devices**
Authenticate consumables and parts, and enable automated transfer of device settings.
- **Verified physical visitor presence**
Enable secure visitor authentication, with proof of live presence and service records.

- **Secure log-in credentials**

Protect web services using two-factor authentication logons to sensitive content sites.

2.4 Standards compliancy

2.4.1 ISO 14443

NTAG 424 DNA is fully compliant to all layers (1, 2, 3, 4) of ISO/IEC 14443 [\[3\]](#).

2.4.2 ISO 7816-4

NTAG 424 DNA is fully compliant to ISO/IEC 7816-4 [\[6\]](#).

2.4.3 NFC Forum compliancy

NFC tag is a contactless tag capable of storing NDEF data, which interoperates with ISO 14443 infrastructure (or other) and NFC devices as defined by the NFC Forum specifications. NFC Forum defines logical data structure for storing NDEF message on a Tag.

The file structure on NTAG 424 DNA complies to NFC Forum Tag 4 Type [\[5\]](#). There are two (2) required files:

- CC file is 32 bytes large, generally used for defining NDEF structure, info on access rights for NFC device, optionally presence of Proprietary Files. It is pre-personalized as NFC Forum Tag 4 Type, NDEF V2.0.
- NDEF file is 256 bytes large. On delivery, it is empty and all type of NDEF messages/ records can be programmed.

3 Definition of variables used in examples

The following symbols are used to abbreviate operations in the examples:

Symbol	Description
"="	Preparation of data by SAM, PICC, or host
"<" or ">"	Direction of communication
	The concatenation operation
\oplus	exclusive-OR operation
$X \ll 1$	The bit string that results from discarding the leftmost bit of the bit string X and appending a '0' bit on the right
0^s	The bit string that consists of s '0' bytes
$E_{AES}(Kx, M)$	AES-128 encipher in CBC mode, IV all 0x00, using key - K of number x, M is cipher input
$D_{AES}(Kx, M)$	AES-128 decipher in CBC mode, IV all 0x00, using key - K of number x, M is cipher input
$E_{LRP}(Kx, M)$	LRP encipher using key - K of number x, M is cipher input
$D_{LRP}(Kx, M)$	LRP decipher using key - K of number x, M is cipher input
$MAC(K,M)$	Message authentication code of message M using secret key K
$MAC_t(K,M)$	Truncated message authentication code of message M using secret key K. Truncated to 8 bytes, using S14 S12 S10 S8 S6 S4 S2 S0. Even-numbered bytes shall be retained in MSB first order.
$KDF: PRF(key, message) = CMAC(Kx, message)$	A NIST recommended key derivation using pseudorandom functions. Pseudo random function: CMAC algorithm

3.1 Byte order

3.1.1 LSB representation

Represented least significant byte (LSB) first are:

- plain command parameters consisting of multiple bytes
- ISO/IEC 14443 parameters during the activation

3.1.2 MSB representation

Represented as most significant byte (MSB) first are:

- cryptographic parameters
- keys
- random numbers exchanged during authentication
- TI (Transaction Identifier)
- computed MACs

4 Secure Dynamic Messaging (SDM)

Allows for confidential and integrity protected data exchange, without requiring a preceding authentication.

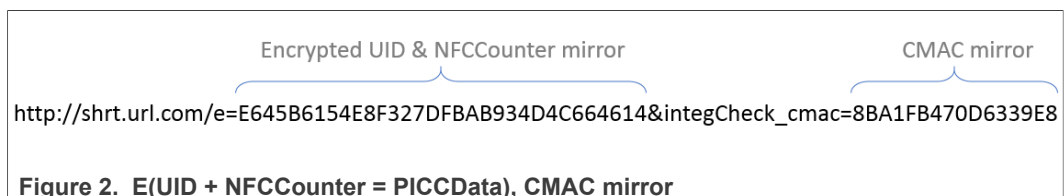
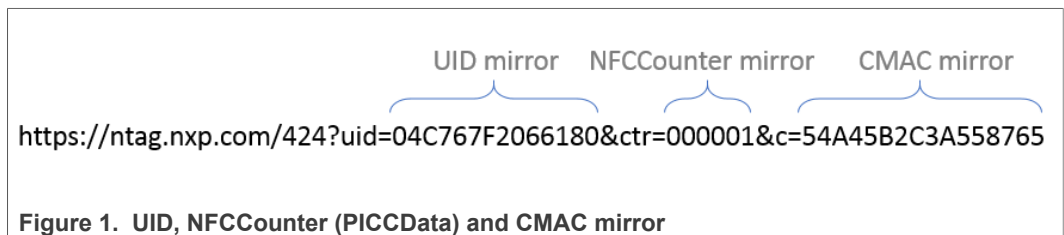
Secure Unique NFC message (SUN) enables user experience in a secure and convenient way. It applies to NDEF file only. Configured static or dynamic values are mirrored as text (ASCII encoded) into the NDEF message (e.g. URL) on each NFC tap [Section 4.2].

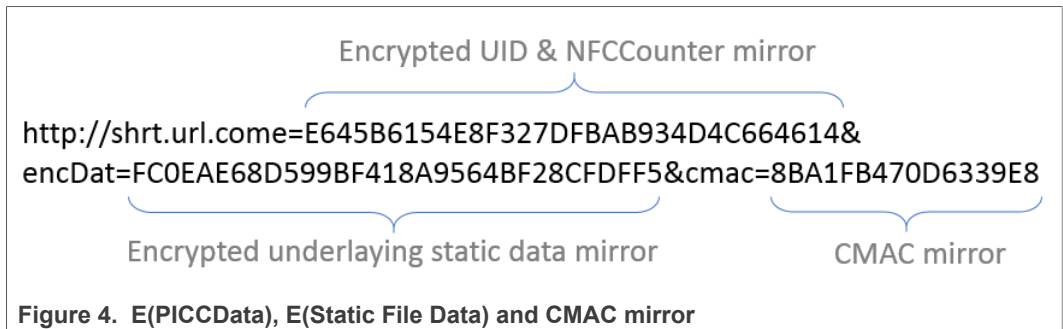
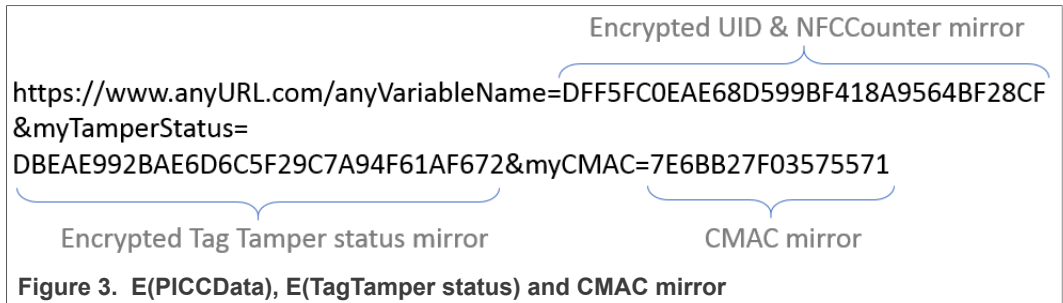
NTAG 424 creates the SUN at power-up procedure, within ISO/IEC 14443 time and H_{MIN} limits.

4.1 Mirroring commons

1. Content is mirrored within the NDEF, only in non-authenticated state
2. Mirrored content (dynamic data) overlays below "place holding" content (static data)
3. Independently configurable what to mirror (UID, Counter, Part of static data, Tag Tamper status, CMAC)
4. Independently configurable where to mirror
5. ASCII encoded (to represent 1 byte – 2 characters are needed)
6. Any separator of any length between mirrors can be set
7. For each mirror following has to be defined:
 - starting offset
 - length
8. Mirror starting position (**offset**) + mirror **length** must not overlap with any other enabled mirror

Few mirroring examples:

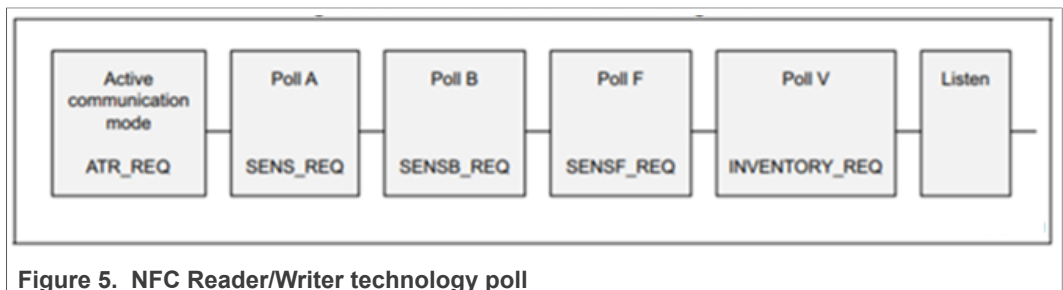




4.2 SUN generation procedure

For detailed NFC Activity procedure, refer to [10]. A high-level description of SUN generation:

1. On the non-locked Home screen, an NFC device (aka Reader/Writer) turns on NFC reader IC
2. NFC reader does the polling cycle for NFC technologies [Figure 5](#)
3. NTAG is tapped. During NFC field present, NTAG boots up
4. Reader/Writer does Tag detection, ISO-14443 anti-collision, device activation [10]
5. NTAG prepares all the mirrors (generates session keys, does encryption, does mirroring, CMAC etc.) which are configured
6. Reader/Writer reads the NDEF with ISOReadBinary command. NFC counter is increased by one (1), any subsequent read within the same session does not increase the counter



4.3 SDM Session Key Generation

Pseudo random function as per CMAC algorithm according to NIST SP 800-38B [2]. These keys are used only for Secure Dynamic NDEF Messaging - SUN.

Note: These are not the same session keys as Secure Standard Messaging ones, which are generated during AuthenticateFirst or AuthenticateNonFirst [Section 5.1].

Prerequisites: CMAC with AES-128 cipher core

Key used: SDMFileReadKey

Length [bytes]: 16

Algorithm:

1. $K_{\text{SesSDMFileReadENC}} = \text{MAC}(K_{\text{SDMFileRead}}; \text{SV1})$
2. $K_{\text{SesSDMFileReadMAC}} = \text{MAC}(K_{\text{SDMFileRead}}; \text{SV2})$

Output:

1. $K_{\text{SesSDMFileReadENC}}$
2. $K_{\text{SesSDMFileReadMAC}}$

Table 2. SDM Session Key Generation

Step	Command		Data
1	Is UID mirrored?	=	If YES, it must be included in SV calculation
2	Is SDMReadCtr mirrored?	=	If YES, it must be included in SV calculation
3	UID	=	04C767F2066180
4	SDMReadCtr	=	010000 (LSB first as per [Section 3.1])
5	$K_{\text{SDMFileRead}}$	=	5ACE7E50AB65D5D51FD5BF5A16B8205B
6	$\text{SV1} = \text{C33C } 0001 \text{ } 0080 \text{ [UID] [SDMReadCtr] [ZeroPadding]}^{[1]}$	=	C33C0001008004C767F2066180010000
7	$\text{SV2} = \text{3CC3 } 0001 \text{ } 0080 \text{ [UID] [SDMReadCtr] [ZeroPadding]}$	=	3CC30001008004C767F2066180010000
8	$K_{\text{SesSDMFileReadENC}} = \text{MAC}(K_{\text{SDMFileRead}}; \text{SV1})$	=	66DA61797E23DECA5D8ECA13BBADF7A9
9	$K_{\text{SesSDMFileReadMAC}} = \text{MAC}(K_{\text{SDMFileRead}}; \text{SV2})$	=	3A3E8110E05311F7A3FCF0D969BF2B48

[1] In case of encrypting file data - PICCENData, mirroring of UID and SDMReadCtr is mandatory. Therefore, both are always included in SV1 calculation.

4.4 SUN Mirroring

4.4.1 PICCData mirror

PICCData consists of UID and SMDCounter. UID, SMDCounter and CMAC are always co-existing, meaning that by enabling/disabling PICCData mirror, all are mirrored or not. Their mirror offsets within NDEF can be individually chosen for UID and SMDCounter. CMAC shall be appended to the end of NDEF.

- Prerequisites:* • n/a
- Offset name:* • UIDOffset,
• SDMReadCtrOffset,
• CMACinputOffset
- Length [bytes]:* • UIDOffsetLength: 14,
• SDMReadCtrOffsetLength: 6,
• CMACOffsetLength: 16

Example SUN mapping:

<https://ntag.nxp.com/424?uid=04C767F2066180&ctr=000001&c=54A45B2C3A558765>

How to verify the CMAC of the SUN is described in chapter [\[Section 4.4.4.2\]](#) - different data is used for computation.

4.4.2 PICCData Encrypted mirror

Note: With encryption of PICCData, we encrypt UID and NFCCounter. Therefore verification side does not have immediate info on UID, which is usually used as input for key derivation function. In this case, $K_{SDMMetaRead}$ key shall not be UID diversified and high attention on secure storage on system level of this key is required.

4.4.2.1 Encryption of PICCData

- Prerequisites:* SDMMetaReadKey set to App.KeyX (0x0 - 0x4)
- Offset name:* PICCENCDataOffset
- Length [bytes]:* $32 * n$; $n=1,2,\dots, n$
- Algorithm:* **PICCENCData** = $E(K_{SDMMetaRead}; PICCDataTag [|| UID] [|| SDMReadCtr] || RandomPadding^{(1)})$

Example SUN mapping:

<https://ntag.nxp.com/424?e=EF963FF7828658A599F3041510671E88&c=94EED9EE65337086>

(1) - Random padding generated by the PICC to make the input 16 bytes long. It is only relevant if SDMReadCtr is not mirrored, as SDMReadCtr adds uniqueness already.

4.4.2.2 Decryption of PICCData

Verification side (e.g. backend, RF reader, NFC Mobile application, etc.) needs to know following parameters:

- Prerequisites:* SDMMetaReadKey used
- Offset name:* PICCENCDataOffset in URL
- Length [bytes]:* PICCENCDataLength
- Algorithm:* **PICCData** = $D(K_{SDMMetaRead}; PICCENCData)$

Table 3. Decryption of PICCData

Step	Command		Data Message
1	Encrypted PICCData	=	EF963FF7828658A599F3041510671E88
2	SDMMetaReadKey = App.Key0	=	00000000000000000000000000000000
3	D(K _{SDMMetaReadKey} , PICCENCData)	=	C704DE5F1EACC0403D0000DA5CF60941
4	PICCDataTag	=	C7
5	UID	=	04DE5F1EACC040
6	SDMReadCtr	=	3D0000
7	Random padding	=	DA5CF60941
8	PICCDataTag [bit]	=	1100 0111
9	PICCDataTag - UID mirroring [bit7]	=	1 (UID mirroring enabled)
10	PICCDataTag - SDMReadCtr mirroring [bit6]	=	1 (SDMReadCtr mirroring enabled)
11	PICCDataTag - UID Length [bit3-0]	=	111b = 7d (7 byte UID)

Example for Python 2.7

```

#!/usr/bin/env python -2
from binascii import hexlify, unhexlify
from Crypto.Cipher import AES

#
# PICCData decryption
# PICCData = AES-128_DECRYPT(KSDMMetaRead; PICCDataTag[||UID]
[||SDMReadCtr]||RandomPadding)

IV = 16 * '\x00'
key = 16 * '\x00' # FileAR.SDMMetaRead Key

# Enc_PICC_Data = '\xEF\x96\x3F
\xF7\x82\x86\x58\xA5\x99\xF3\x04\x15\x10\x67\x1E\x88'
Enc_PICC_Data = 'EF963FF7828658A599F3041510671E88'

myaes = AES.new(key, AES.MODE_CBC, IV=IV)
PICCData = myaes.decrypt(unhexlify(Enc_PICC_Data))

PICCDataTag = hexlify(PICCData[0:1])
UID = hexlify(PICCData[1:8])
SDMReadCtr = hexlify(PICCData[8:11])

print '\nDecrypted PICCData: ' + hexlify(PICCData)
print "PICCDataTag: " + PICCDataTag
print "UID      : " + UID
print "SDMReadCtr : " + SDMReadCtr
    
```

Returns:

Decrypted PICCData: c704de5f1eacc0403d0000da5cf60941

PICCDataTag: c7

Table 4. Decryption of PICCData...continued

Step	Command	Data Message
5	PICCDataTag	= C7
6	UID	= 04958CAA5C5E80
7	SDMReadCtr	= 010000
8	Random padding	= 851ECB67D4
9	PICCDataTag [bit]	= 1100 0111
10	PICCDataTag - UID mirroring [bit7]	= 1 (UID mirroring enabled)
11	PICCDataTag - SDMReadCtr mirroring [bit6]	= 1 (SDMReadCtr mirroring enabled)
12	PICCDataTag - UID Length [bit3-0]	= 111b = 7d (7 byte UID)
	Session Key generation	
13	SV1 = C33C 0001 0080 [UID] [SDMReadCtr] [ZeroPadding]	= C33C0001008004958CAA5C5E80010000
14	$K_{SesSDMFileReadENC} =$ $MAC(K_{SDMFileRead}, SV1)$	= 8097D73344D53F963B09E23E03B62336
15	$IVe = E(K_{SesSDMFileReadENC};$ $SDMReadCtr 0^{16})$	= 7B3F3CFC39D3B7FF5868636E38AF7C3A
16	$D(K_{SesSDMFileReadENC}, IVe,$ $SDMENCFileData)$	= 78787878787878787878787878787878

4.4.4 SDMMAC mirror

4.4.4.1 SDMMAC

Prerequisites: SDMFileReadKey set to 0x0 - 0x4 in NDEF file settings.

Offset name: SDMMACOffset, SDMMACInputOffset

Length [bytes]: 16

Algorithm: **SDMMAC** = MACt($K_{SesSDMFileReadMAC}$; DynamicFileData [SDMMACInputOffset :: SDMMACOffset - 1])⁽¹⁾

Example SUN mapping:

<https://ntag.nxp.com/424?e=EF963FF7828658A599F3041510671E88&c=94EED9EE65337086>

(1) - DynamicFileData is the file data as how it is put on the external interface (mirrored) - replacing any placeholders by the dynamic data.

4.4.4.2 SDMMAC calculation

4.4.4.2.1 CMACInputOffset == CMACOffset

NTAG 424 DNA and NTAG 424 DNA TagTamper features and hints

Prerequisites: SDMFileReadKey
Offset name: SDMMACOffset (SDMMACInputOffset == SDMMACOffset)
Length [bytes]: 16
Algorithm: SDMMAC = MACt(K_{SesSDMFileReadMAC}; zero length input)

Table 5. CMAC calculation when CMACInputOffset == CMACOffset

Step	Command		Data Message
1	KeySDMFileReadMAC	=	00000000000000000000000000000000
2	PICCENCDData	=	EF963FF7828658A599F3041510671E88
3	D(K _{SDMMetaReadKey} , PICCENCDData)	=	C704DE5F1EACC0403D0000DA5CF60941
4	PICCCDataTag	=	C7
5	UID	=	04DE5F1EACC040
6	SDMReadCtr	=	3D0000
7	Random padding	=	DA5CF60941
8	PICCCDataTag [bit]	=	1100 0111
9	PICCCDataTag - UID mirroring [bit7]	=	1 (UID mirroring enabled)
10	PICCCDataTag - SDMReadCtr mirroring [bit6]	=	1 (SDMReadCtr mirroring enabled)
11	PICCCDataTag - UID Length [bit3-0]	=	111b = 7d (7 byte UID)
Session Key generation			
12	SV2 = 3CC3 0001 0080 [UID] [SDMReadCtr] [ZeroPadding]	=	3CC30001008004DE5F1EACC0403D0000
13	K _{SesSDMFileReadMAC} = MAC(K _{SDMFileRead} ; SV2)	=	3FB5F6E3A807A03D5E3570ACE393776F
14	SDMMAC = MACt(K _{SesSDMFileReadMAC} ; zero length input)	=	94EED9EE65337086

Example for Java, using Bouncy Castle library:

```
package MFCMAC;

import org.bouncycastle.crypto.BlockCipher;
import org.bouncycastle.crypto.Mac;
import org.bouncycastle.crypto.engines.AESFastEngine;
import org.bouncycastle.crypto.macs.CMac;
import org.bouncycastle.crypto.params.KeyParameter;

public class MFCMAC {

    public byte[] calculateMFCMAC (byte[] key, byte[]
valueToMAC) {

        try {
```

```

int cmacSize = 16;
BlockCipher cipher = new AESFastEngine();
Mac cmac = new CMac(cipher, cmacSize * 8);
KeyParameter keyParameter = new KeyParameter(key);
cmac.init(keyParameter);
cmac.update(valueToMAC, 0, valueToMAC.length);
byte[] CMAC = new byte[cmacSize];
cmac.doFinal(CMAC, 0);

byte[] MFCMAC = new byte[cmacSize / 2];

int j = 0;
for (int i = 0; i < CMAC.length; i++) {
    if (i % 2 != 0) {
        MFCMAC[j] = CMAC[i];
        j += 1;
    }
}

return MFCMAC;

} catch (Exception ex) {
    ex.printStackTrace();
}

return null;

}
}

```

Modern libraries have "zero length" MAC-ing implemented. In case of manual implementation, below reference pseudo-code may be used. CMAC in detailed steps as per NIST Special Publication 800-38B [2].

Subkey Generation

KsesSDMFileReadMAC = 3FB5F6E3A807A03D5E3570ACE393776F
 CIPHK(0b) -> AES-128(KsesSDMFileReadMAC; 0b128)

1. Let L = CIPHK(0b).

L: 0e2f0c519f60eb99497eed68b3f7c5a5

2. If MSB1(L) = 0, then K1 = L << 1; Else K1 = (L << 1) ⊕ Rb;

L << 1: 1c5e18a33ec1d73292fddad167ef8b4a

Rb: 0000000000000000000000000000000087 (defined in NIST)

K1 = (L << 1): 1c5e18a33ec1d73292fddad167ef8b4a

3. If MSB1(K1) = 0, then K2 = K1 << 1; Else K2 = (K1 << 1) ⊕ Rb.

K1 << 1: 38bc31467d83ae6525fbb5a2cfd1694

K2 = (K1 << 1): 38bc31467d83ae6525fbb5a2cfd1694


```

MAC Generation
-----

Step1
-----
K1, K2 produced

Step2
-----
Mlen = 0, n = 1

Step4
-----
If Mn is a complete block:
M1 = K1 ⊕ M1*
else:
M1 = K2 ⊕ (M1* || 10^j)
M1 = b8bc31467d83ae6525fbb5a2cfd1694

Step6
-----
AES-128 (KsesSDMFileReadMAC; b8bc31467d83ae6525fbb5a2cfd1694) =
e194c7ee12d9f7ee8a65c8331b704386

Step7
-----
CMAC = 94eed9ee65337086
    
```

4.4.4.2.2 CMACInputOffset != CMACOffset

https://www.my424dna.com/?picc_data=FD91EC264309878BE6345CBE53BADF40&enc=CEE9A53E3E463EF1F459635736738962&cmac=EC1E7F6C6C73BF6

- Prerequisites:* SDMFileReadKey
- Offset name:* SDMMACOffset (SDMMACInputOffset != SDMMACOffset)
- Length [bytes]:* 16
- Algorithm:* $SDMMAC = MACt(K_{SesSDMFileReadMAC}; DynamicFileData [SDMMACInputOffset :: SDMMACOffset - 1]^{(1)})$

(1) DynamicFileData is the file data as how it is put on the external interface, i.e. replacing any placeholders by the dynamic data.

Table 6. CMAC calculation when CMACInputOffset != CMACOffset

Step	Command	Data Message
1	SDMMACOffset	6A0000
2	SDMMACInputOffset	440000
3	PICCENCDDataOffset	1F0000
4	SDMFileReadKey	00000000000000000000000000000000

Table 6. CMAC calculation when CMACInputOffset != CMACOffset...continued

Step	Command	Data Message
5	ENCDataOffset	440000
6	ENCDataLength	200000
7	UID	= 04958CAA5C5E80
8	SDMReadCtr	080000
9	SDMENCFileData	= CEE9A53E3E463EF1F459635736738962
10	SDMMAC	= ECC1E7F6C6C73BF6
11	PICCENCData	= FD91EC264309878BE6345CBE53BADF40
12	D(K _{SDM} MetaReadKey, PICCENCData)	= C704958CAA5C5E80080000A243C86DFC
Session Key generation		
13	SV2 = 3CC3 0001 0080 [UID] [SDMReadCtr] [ZeroPadding]	= 3CC30001008004958CAA5C5E80080000
14	K _{SesSDMFileRead} MAC = MAC(K _{SDMFileRead} ; SV2)	= 3ED0920E5E6A0320D823D5987FEAFBB1
15	DynamicFileData [SDMMACInputOffset :: SDMMACOffset - 1] (ASCII)	= CEE9A53E3E463EF1F459635736738962&cmac=
16	DynamicFileData [SDMMACInputOffset :: SDMMACOffset - 1] (hex)	= 4345453941353345334534363345463146343539 36333537333637333839363226636d61633d
17	SDMMACfull = MAC (K _{SesSDMFileRead} MAC; DynamicFileData [SDMMACInputOffset :: SDMMACOffset - 1])	= 81EC45C175E72FF6FAC61BC7AB3BAEF6
18	SDMMAC = MACt	= ECC1E7F6C6C73BF6

5 Standard Secure Messaging (SSM)

Standard Secure messaging is the most up-to-date secure messaging mode, with following properties:

- plain/maced/encrypted channel of communication established between PCD and PICC

Table 7. Communication modes in SSM

Communication Mode	Bit Representation	Explanation
CommMode.Plain	X0	Plain communication: No encryption is used at all.
CommMode.MAC	01	MACed communication: The data is transferred in plain, but a 4 bytes or 8 bytes MAC is added to the message.
CommMode.Full	11	Encrypted communication: Full protection for integrity, authenticity, and confidentiality.

- Confidentiality and integrity are protected by using two session keys (generated on both sides - PCD and PICC)
- Standard Secure messaging is established by successful Cmd.AuthenticateEV2First and Cmd.AuthenticateEV2NonFirst - allows cryptographically binding of all messages within one transaction by using a transaction identifier (TI) and a command counter (CmdCtr)
- For an Authorized changes (settings, data)

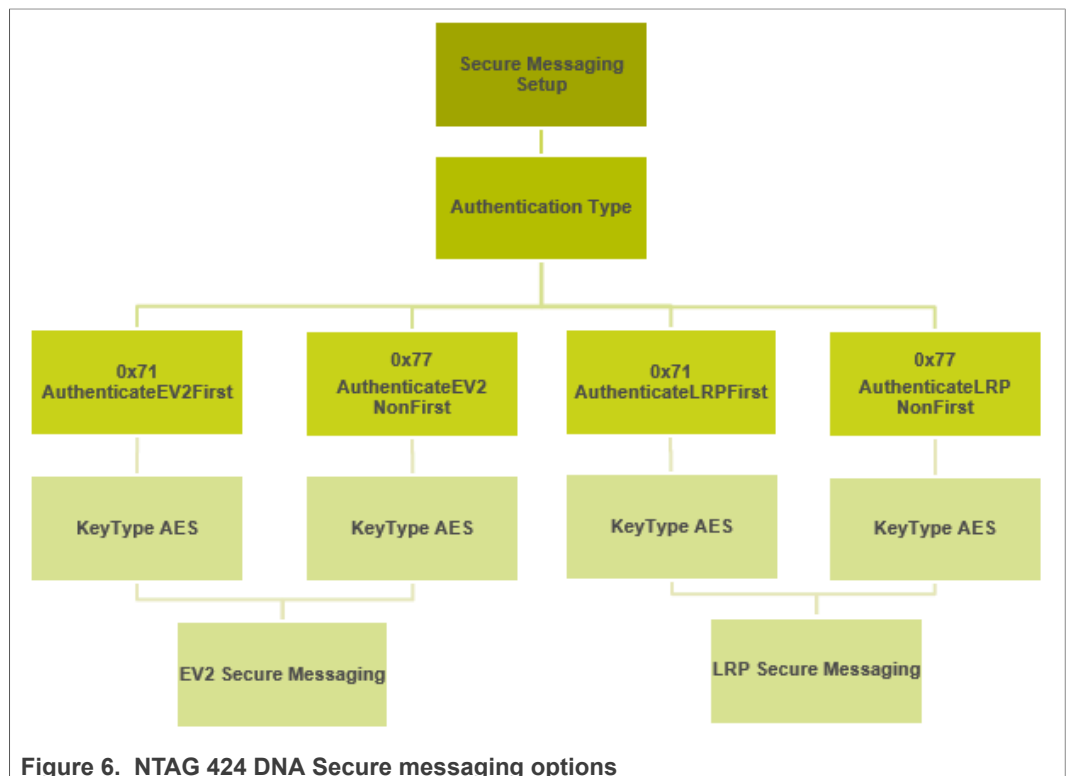


Figure 6. NTAG 424 DNA Secure messaging options

The LRP mode can be permanently enabled using the SetConfiguration command. After this switch, it is not possible to revert to AES mode. More details on LRP can be found in [11] and [12].

5.1 SSM Session Keys generation

As a result of successful authentication, K_{SesSDMFileReadMAC} and K_{SesSDMFileReadENC} keys are generated on PCD and PICC sides, using the same algorithm.

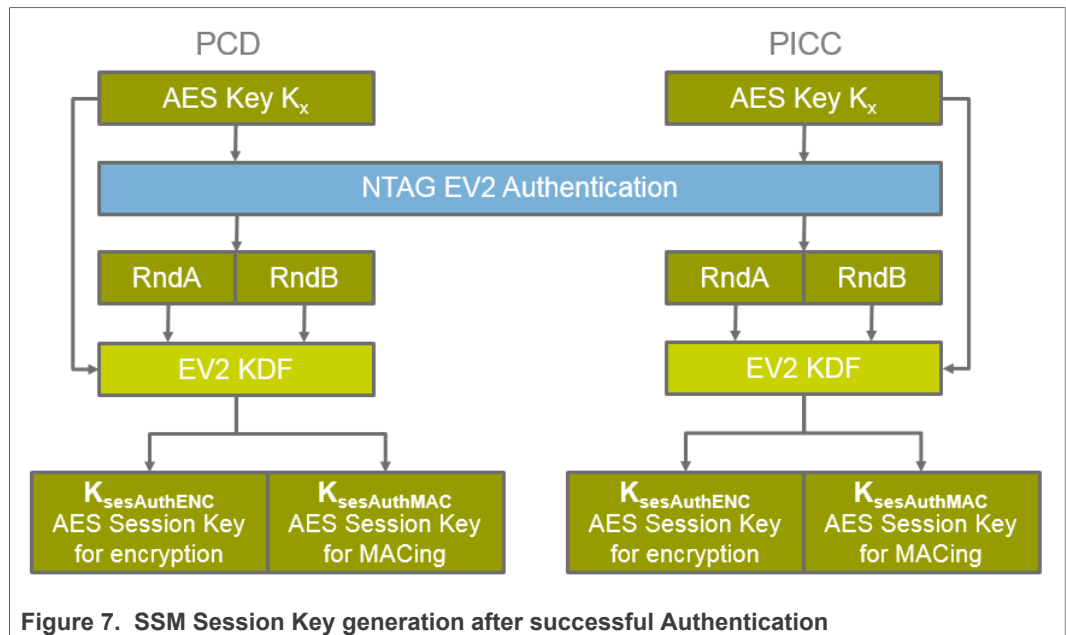


Figure 7. SSM Session Key generation after successful Authentication

5.2 CommMode.Plain

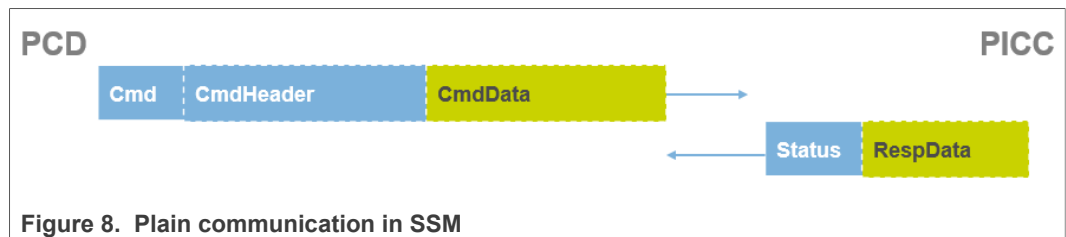


Figure 8. Plain communication in SSM

5.3 CommMode.MAC

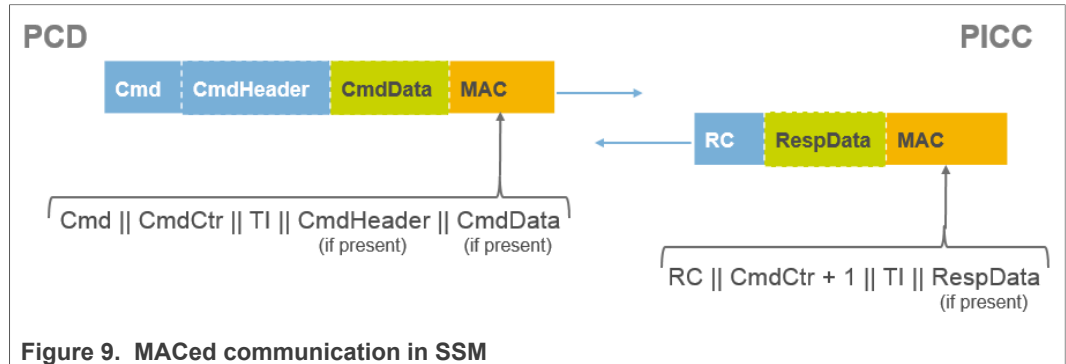


Figure 9. MACed communication in SSM

PCD MAC calculated as:

$$CMAC = MACt (K_{SesAuthMAC}, IV, Cmd \parallel CmdCtr \parallel TI \parallel CmdHeader \text{ (if present)} \parallel EncCmdData \text{ (if present)}) .$$

In following example, session keys $K_{SesAuthENC}$ and $K_{SesAuthMAC}$ were generated during some random AuthenticateEV2First procedure.

Table 8. Example of CommMode.MAC on Cmd.GetFileSettings command

Step	Command	Data Message
1	$K_{SesAuthMAC}$	= 8248134A386E86B7FAF54A52E536CB6
2	Cmd	= F5
3	CmdHeader	= 02
4	CmdData	= n/a
5	TI	= 7A21085E
6	CmdCounter	= 0000
7	Cmd CmdCounter TI CmdHeader n/a	= F500007A21085E02
8	CMAC = MAC ($K_{SesAuthMAC}$, Cmd CmdCounter TI CmdHeader n/a)	= B565AC978FA46D5784C845CD1444102C
9	MACt	= 6597A457C8CD442C
10	Cmd.GetFileSettings C-APDU	> 90F5000009026597A457C8CD442C00
11	R-APDU (ResponseCode ResponseData MACt)	< 000040EEEE000100D1FE001F000044000044000 02000006A00002A474282E7A47986
12	R-APDU's MACt	= 2A474282E7A47986
13	ResponseCode CmdCounter +1 TI ResponseData (without 91)	= 0001007A21085E0040EEEE000100D1FE001F000 04400004400002000006A0000
14	MAC ($K_{SesAuthMAC}$, ResponseCode CmdCounter +1 TI ResponseData)	= DC2A9C473642F3826AE79DA496792086
15	Response CMAC = MACt	2A474282E7A47986

5.4 CommMode.Full

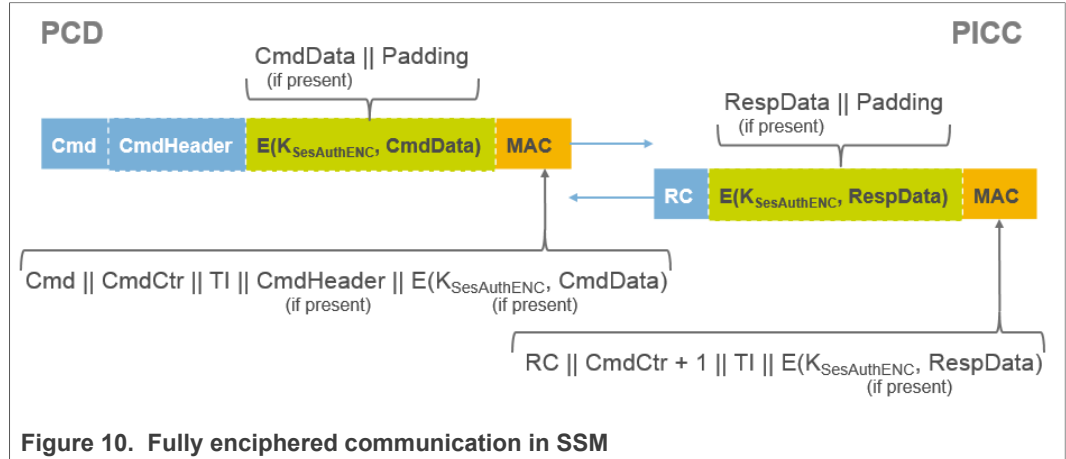


Figure 10. Fully enciphered communication in SSM

$$IV_c \text{ (for command)} = E(K_{SesAuthENC}, IV_c, A55A \parallel TI \parallel CmdCtr \parallel 0^{16})$$

$$IV_r \text{ (for response)} = E(K_{SesAuthENC}, IV_c, 5AA5 \parallel TI \parallel CmdCtr + 1 \parallel 0^{16})$$

In following example, session keys $K_{SesAuthENC}$ and $K_{SesAuthMAC}$ were generated during some random AuthenticateEV2First procedure.

Table 9. Example of CommMode.FULL on Cmd.Write to File No 0x02

Step	Command	Data Message
1	File No	= 02
2	$K_{SesAuthENC}$	= 7305E2CCA5B0377617CDBFEB96C9B358
3	TI	= 856C1841
4	CMAC Session Key $K_{SesAuthMAC}$	= 8B485037C8C2FB400D79BF0AB956F28F
5	IV for Encryption = $E(K_{SesAuthMAC}, 0xA5 \parallel 0x5A \parallel TI \parallel CmdCtr \parallel 0x0000000000000000)$	= $E(K_{SesAuthENC}, A55A \ 856C1841 \ 00000000000000000000) = 81DACD2EB2D257FC556E8952DD665B58$
6	Plain Input Data (NDEF message to write)	= 005ED1015B5500687474703A2F2F7777772E6D6 9666172652E6E65742F70726F64756374732F6E7 46167733F265549443D303432313635344143463 4433830264374723D30303030303126436D61633 D323145323336303832363645334345410000000 00000000000000000000000000000000000000 0000000000000000
7	Length of file	= 800000
8	Chaining	= ISO 14443-4 chaining
9	Encrypted Input Data = $E(K_{SesAuthENC}, \text{Input Data})$	B4716C58E71A09F6D869AB7810C2E94BD02F13 DF2159433D581F50185B11535F3E7A068582B04 B5E4BDE374A788DF7AD8C4C5473F7B30D9496 BD8F3F8ED51D506D3194FDEA51A877C2EB28A 0A8FD2B34E196800A7D2F0AD1CBED98E311E2 F7667DA10DF3CF4CE6A5658B89695EDAD9F5D 9AD1E4C41748D34BC6B15A2B45B050F
10	New IV	= D9AD1E4C41748D34BC6B15A2B45B050F

Table 9. Example of CommMode.FULL on Cmd.Write to File No 0x02 ...continued

Step	Command		Data Message
11	C-APDU (CMAC Response Code)	<	908D00009F02000000800000B4716C58E71A09F6D869AB7810C2E94BD02F13DF2159433D581F50185B11535F3E7A068582B04B5E4BDE374A788DF7AD8C4C5473F7B30D9496BD8F3F8ED51D506D3194FDEA51A877C2EB28A0A8FD2B34E196800A7D2F0AD1CBED98E311E2F7667DA10DF3CF4CE6A5658B89695EDAD9F50000000D9AD1E4C41748D34BC6B15A2B45B050F34765F3E9D2CF701E0C7F781477F7B91B97CBB2A236F876C00
12	CMAC Received from PICC	>	DDDB9EC959B3EFEB
13	CMAC calculated. CMAC = MFCMAC(K _{SesAuthMAC} , Load)	=	A3DDE4DB219E35C96659C2B333EFE2EB
14	CMAC = MFCMAC(K _{SesAuthMAC} , Load) (truncated to 8 Bytes)	=	DDDB9EC959B3EFEB
15	PCD compares CMACs	=	DDDB9EC959B3EFEB == DDDB9EC959B3EFEB

6 Personalization example

Following steps are optional and used as an example only. Final steps shall be defined and configured by customer needs.

Example configuration:

- NDEF content with placeholders: <https://choose.url.com/ntag424?e=00&c=0000000000000000>
- mirroring of PICCdata encrypted (ENCPICCCData)
 - Encryption key ($K_{SDMFileRead}$): 0x02
 - ENCPICCCDataOffset: 0x20
- CMAC computation:
 - CMAC-ing key ($K_{SDMFileRead}$) which is used to generate session key $K_{SesSDMFileReadMAC}$: 0x01
 - SDMMACOffset: 0x43
- empty payload for CMAC input (SDMMACInputOffset == SDMMACOffset)
- GetCounterValue command protected by key 0x01
- No NFCCounter limit
- no mirroring and no encryption of SDMFileData - SDMENCCFileData

Steps:

1. ISO14443-4 PICC Activation [[Section 6.1](#)]
2. Originality signature verification [[Section 6.2](#)]
3. ISO SELECT NDEF application using DF Name [[Section 6.3](#)]
4. Get File Settings [[Section 6.4](#)]
5. GetVersion [[Section 6.5](#)]
6. AuthenticateFirst with ApplicationKey 0x00 [[Section 6.6](#)]
7. Prepare NDEF data [[Section 6.7](#)]
8. Write data to 0xE104 (NDEF File) [[Section 6.8](#)]
9. Change File Settings of 0xE104 [[Section 6.9](#)]
10. AuthenticateEV2First with ApplicationKey 0x03 [[Section 6.10](#)]
11. ISO SELECT Proprietary File 0xE105 [[Section 6.11](#)]
12. Write data to 0xE105 (Proprietary File) [[Section 6.12](#)]
13. ISO SELECT Capability Container file 0xE103 [[Section 6.13](#)]
14. AuthenticateAESNonFirst with ApplicationKey 0x00 [[Section 6.14](#)]
15. Write data to 0xE103 (CC file) - READ-ONLY [[Section 6.15](#)]
16. Change Key - ApplicationKey 0x02 (Using AES-128 Key diversification) [[Section 6.16.1](#)]
17. Change Key - ApplicationKey 0x00 (Master Key) [[Section 6.16.2](#)]

6.1 ISO14443-4 PICC Activation

Reader supporting ISO14443-4 or NFC is required.

Table 10. ISO14443-4 PICC Activation

Step	ISO 14443 Command	NFC Forum Command		Data Message
1	RF field ON		=	true

Table 10. ISO14443-4 PICC Activation...continued

Step	ISO 14443 Command	NFC Forum Command		Data Message
2	REQA	SENS_REQ	>	26
3	ATQA		<	4403
4	Anticollision CL1	SDD_REQ CL1	>	9320
5	CT, UID0, UID1, UID2, BCC		<	8804168913
6	Select cascade level 1	SEL_REQ CL1	>	93708804168913
7	Data		<	04
8	Anticollision CL 2	SDD_REQ CL2	>	9520
9	UID0, UID1, UID2, BCC		<	AA5C5E8028
10	Select cascade level 2	SEL_REQ CL2	>	9570AA5C5E8028
11	SAK		<	20
12	RATS	RATS	>	E080
13	ATS		<	067777610280
14	PPS	ISO-DEP	>	D01100 ⁽¹⁾
15	PPSS		<	D0

(1) Higher data transfer can be set in this step (downlink and uplink). Example shown in

6.2 Originality signature verification

The Symmetric Originality Check is possible only in LRP mode. The asymmetric check can be done prior personalization to assure that it will be done on the NXP delivered IC.

Procedure is described in [\[Section 8.2\]](#).

6.3 ISO SELECT NDEF application using DF Name

Table 11. Select NDEF Application using Cmd.ISOSelect

Step	Command		Data Message
1	ISO7816 AID – DF Application Name	=	D2760000850101
2	CLA	=	00
3	INS	=	A4
4	P1	=	04 (select by DF name)
5	P2	=	0C
6	Lc	=	07
7	Command header	=	00A4040C07
8	Command data (ISO7816 AID – DF Name)	=	D2760000850101
9	Le	=	00
10	Cmd.Select C-APDU	>	00A4040C07D276000085010100

Table 11. Select NDEF Application using Cmd.ISOSelect...continued

Step	Command		Data Message
11	R-APDU	<	9000

6.4 Get File Settings

This step does not reflect default delivered NTAG 424 DNA configuration of NDEF file settings (0000E0EE00010026000CA). Purpose of the example is to show meaning of bytes in response APDU.

Step is optional and may be left out. It is just to identify CommMode: Plain, MACed or FULL and adopt secure messaging of commands in later steps.

Table 12. Get file settings of NDEF File

Step	Command		Data Message
1	Cmd	=	F5
2	Command header	=	02 (file nr. 02)
3	Command data (ISO7816 AID – DF Name)	=	n/a
4	C-APDU = CLA + INS + P1 + P2 + Lc + (CmdHdr + CmdData) + Le	>	90F50000010200
5	R-APDU	<	004300E0000100C1F1212000004300009100

Meaning of R-APDU:

	Length [bytes]	Value	Meaning
FileType	1	00	FileType.StandardData
FileOption	1	40	SDM and Mirror enabled, CommMode.Plain
AccessRights	2	00E0	<ul style="list-style-type: none"> FileAR.ReadWrite = 0 (key nr. 0) FileAR.Change = 0 FileAR.Read = E (free) FileAR.Write = 0
FileSize	3	000100	256d
SDMOptions	1	C1	UID mirror set, SDMReadCounter set, ASCII Encoding mode enabled

SDMAccessRights	2	F121	<ul style="list-style-type: none"> • RFU = F • FileAR.SDMCtrRet = 1 (key nr. 1) • FileAR.SDMMetaRead (PICCENCDData) = 2 • FileAR.SDMFileRead (CMAC) = 1
UIDOffset	3	200000 (n/a by default)	32d
SDMReadCtr Offset	3	430000 (n/a by default)	67d
SDMMACInputOffset	3	n/a	
PICCDataOffset	3	n/a	
SDMMACInputOffset	3	n/a	
SDMENCOffset	3	n/a	
SDMENCLength	3	n/a	
SDMMACOffset	3	n/a	
SDMReadCtrLimit	3	n/a	

6.5 Get Version

Table 13. Get Version

Step	Command		Data Message
1	Cmd	=	60
2	Command header	=	n/a
3	Command data (ISO7816 AID – DF Name)	=	n/a
4	C-APDU = CLA + INS + P1 + P2 + Lc + (CmdHdr + CmdData) + Le	>	9060000000
5	R-APDU	<	0404083000110591AF
6	C-APDU = CLA + INS + P1 + P2 + Lc + (CmdHdr + CmdData) + Le	>	90AF000000
7	R-APDU	<	0404020101110591AF
8	C-APDU = CLA + INS + P1 + P2 + Lc + (CmdHdr + CmdData) + Le	>	90AF000000
9	R-APDU	<	04968CAA5C5E80CD65935D4021189100

NTAG 424 DNA and NTAG 424 DNA TagTamper features and hints

Meaning of R-APDU as per [\[Data sheet\]](#):

	Length [bytes]	Value	Meaning
Vendor ID	1	04	NXP Semiconductors
Type	1	04	NTAG
Sub-Type	1	08	50 pF, Strong back modulation enabled
Major Version	1	30	HW major version number
Minor Version	1	00	HW minor version number
Storage Size	1	11	256 B < storage size < 512 B
Communication Protocol Type	1	05	ISO/IEC 14443-4 support
Software Information			
Vendor ID	1	04	
Type	1	04	
Sub-Type	1	02	
Major Version	1	01	
Minor Version	1	01	
Storage Size	1	11	
Communication Protocol Type	1	05	
Unique Serial Number	7	04968CAA5C5E80	UID if not configured for RandomID
Production Batch Number	4	CD65935D	Production batch number
BatchNo/FabKey	1	40	
Calendar week of Production	1	21	Calendar week of production
Year of production	1	18	Year of production
FabKey ID			

6.6 AuthenticateEV2First with key 0x00

Table 14. Cmd.AuthenticateEV2First using Key No 0x00

Step	Command		Data Message
1	Key No	=	00
2	Key0	=	00000000000000000000000000000000
3	Cmd (INS)	=	71
4	Command header (Key No LenCap)	=	0000
5	Cmd.AuthenticateFirst C-APDU	>	9071000002000000
6	R-APDU (E(K ₀ , RndB) Response Code)	<	A04C124213C186F22399D33AC2A3021591AF
7	E(K ₀ , RndB)	=	A04C124213C186F22399D33AC2A30215
8	Response Code	=	91AF (AF ... additional frame)
9	D(K ₀ , RndB)	=	B9E2FC789B64BF237CCCAA20EC7E6E48
10	PCD generates RndA	=	13C5DB8A5930439FC3DEF9A4C675360F
11	PCD prepares RndB' (rotate left by 1 byte)	=	E2FC789B64BF237CCCAA20EC7E6E48B9
12	RndA RndB'	=	13C5DB8A5930439FC3DEF9A4C675360FE2FC789B64BF237CCCAA20EC7E6E48B9
13	E(K ₀ , RndA RndB')	=	35C3E05A752E0144BAC0DE51C1F22C56B34408A23D8AEA266CAB947EA8E0118D
14	Cmd.AuthenticatePart2 C-APDU (INS = AF)	>	90AF00002035C3E05A752E0144BAC0DE51C1F22C56B34408A23D8AEA266CAB947EA8E0118D00
15	R-APDU E(K _x , TI RndA' PDcap2 PCDcap2)	<	3FA64DB5446D1F34CD6EA311167F5E4985B89690C04A05F17FA7AB2F081206639100
16	E(K _x , TI RndA' PDcap2 PCDcap2)	=	3FA64DB5446D1F34CD6EA311167F5E4985B89690C04A05F17FA7AB2F08120663
17	Response Code		9100
18	D(K ₀ , TI RndA' PDcap2 PCDcap2)	=	9D00C4DFC5DB8A5930439FC3DEF9A4C675360F130000000000000000000000000000
19	TI (4 byte)	=	9D00C4DF
20	RndA' (16 byte)	=	C5DB8A5930439FC3DEF9A4C675360F13
21	PDcap2 (6 byte)	=	000000000000
22	PCDcap2 (6 byte)	=	000000000000
23	RndA (rotate right for 1 byte)	=	13C5DB8A5930439FC3DEF9A4C675360F
24	PCD compares sent RndA (from step 10) and received RndA (from step 20)	=	13C5DB8A5930439FC3DEF9A4C675360F == 13C5DB8A5930439FC3DEF9A4C675360F

6.8.1 Write NDEF File - using Cmd.ISOUpdateBinary, CommMode.PLAIN

Table 17. Write NDEF File - using Cmd.ISOUpdateBinary

Step	Command		Data Message
1	CLA	=	00
2	INS	=	D6
3	P1	=	00
4	P2	=	00
5	Lc	=	53
6	Data	=	0051D1014D550463686F6F73652E75726C2E636 F6D2F6E7461673432343F653D30303030303030 30303030303030303030303030303030303030 303030303026633D3030303030303030303030 30303030
7	Le	=	00
8	C-APDU	>	00D60000530051D1014D550463686F6F73652E7 5726C2E636F6D2F6E7461673432343F653D3030 30303030303030303030303030303030303030 303030303030303030303026633D303030303030 303030303030303000
9	R-APDU	<	9000

6.8.2 Write NDEF File - using Cmd.WriteData, CommMode.FULL

Usually there is no need to write NDEF data over encrypted channel, as NDEF File contains end consumer readable data. This chapter is for demonstrating encrypted data channel exchange - CommMode.FULL.

Table 18. Write NDEF File - using Cmd.WriteData

Step	Command		Data
1	Cmd	=	8D
2	K _{SesAuthMAC} (as generated in Auth. [Table 14])	=	4C6626F5E72EA694202139295C7A7FC7
3	K _{SesAuthENC} (as generated in Auth.)	=	1309C877509E5A215007FF0ED19CA564
4	CmdHeader	=	02 000000 530000
5	CmdCtr	=	0000
6	TI (as generated in Auth.)	=	9D00C4DF
7	CmdData	=	0051D1014D550463686F6F73652E75726C2E636 F6D2F6E7461673432343F653D30303030303030 30303030303030303030303030303030303030 303030303026633D3030303030303030303030 30303030000000000000000000000000000000 00000000000000000000000000000000000000 00000000000000000000800000000000000000 0000000000

NTAG 424 DNA and NTAG 424 DNA TagTamper features and hints

Table 18. Write NDEF File - using Cmd.WriteData...continued

Step	Command	Data
8	IVc = E(K _{SesAuthENC} , A55A TI CmdCtr 0000000000000000)	= E(K _{SesAuthENC} , A55A 9D00C4DF 0000 0000000000000000)
9	IVc	= D2CB7277A17841A06654A48188C1F8F5
10	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	= E(K _{SesAuthENC} , IVc, CmdData 80000000000000000000000000000000)
11	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	= 421C73A27D827658AF481FDFF20A5025B559D0E3AA21E58D347F343CFFC768BFE596C706BC00F2176781D4B0242642A0FF5A42C461AAF894D9A1284B8C76BCFA658ACD40555D362E08DB15CF421B51283F9064BCBE20E96CAE545B407C9D651A3315B27373772E5DA2367D2064AE054AF996C6F1F669170FA88CE8C4E3A4A7BBBEF0FD971FF532C3A802AF745660F2B4
12	Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData)	= 8D00009D00C4DF0200000800000421C73A27D827658AF481FDFF20A5025B559D0E3AA21E58D347F343CFFC768BFE596C706BC00F2176781D4B0242642A0FF5A42C461AAF894D9A1284B8C76BCFA658ACD40555D362E08DB15CF421B51283F9064BCBE20E96CAE545B407C9D651A3315B27373772E5DA2367D2064AE054AF996C6F1F669170FA88CE8C4E3A4A7BBBEF0FD971FF532C3A802AF745660F2B4
13	MAC(K _{SesAuthMAC} , Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData))	= A8D185D964A8E04998965461E7EB3EF3
14	MACT	= D1D9A8499661EBF3
15	Cmd.SetConfiguration C-APDU	> 908D00009F0200000800000421C73A27D827658AF481FDFF20A5025B559D0E3AA21E58D347F343CFFC768BFE596C706BC00F2176781D4B0242642A0FF5A42C461AAF894D9A1284B8C76BCFA658ACD40555D362E08DB15CF421B51283F9064BCBE20E96CAE545B407C9D651A3315B27373772E5DA2367D2064AE054AF996C6F1F669170FA88CE8C4E3A4A7BBBEF0FD971FF532C3A802AF745660F2B4D1D9A8499661EBF300
16	R-APDU (ResponseCode (E(K _{SesAuthENC} , ResponseData) MACT)	< 9100 FC222E5F7A542452
17	R-APDU's MACT	= FC222E5F7A542452
18	Status CmdCounter + 1 TI (E(K _{SesAuthENC} , ResponseData)	= 0001009D00C4DF
19	MAC(K _{SesAuthMAC} , Status CmdCounter + 1 TI (E(K _{SesAuthENC} , ResponseData))	= 96FC5A22A22EC05F377A635407242252
20	MACT	= FC222E5F7A542452
21	Compare R-APDU's MACT (step 17) and calculated MACT from step (step 20)	= true - Integrity of message received from the PICC verified

6.9 Change NDEF File Settings

Table 19. Change NDEF file settings using Cmd.ChangeFileSettings

Step	Command	Data
1	Cmd	= 5F
2	K _{SesAuthMAC}	= 4C6626F5E72EA694202139295C7A7FC7
3	K _{SesAuthENC}	= 1309C877509E5A215007FF0ED19CA564
4	CmdHeader	= 02
5	CmdCtr	= 0100
6	TI	= 9D00C4DF
7	CmdData 40h = FileOption (SDM and Mirroring enabled), CommMode: plain 00E0h = AccessRights (FileAR.ReadWrite: 0x0, FileAR.Change: 0x0, FileAR.Read: 0xE, FileAR.Write: 0x0) C1h = • UID mirror: 1 • SDMReadCtr: 1 • SDMReadCtrLimit: 0 • SDMENCFileData: 0 • ASCII Encoding mode: 1 F121h = SDMAccessRights (RFU: 0xF, FileAR.SDMCtrRet = 0x1, FileAR.SDMMetaRead: 0x2, FileAR.SDMFileRead: 0x1) 200000h = ENCPICCDataOffset 430000h = SDMMACOffset 430000h = SDMMACInputOffset	= 4000E0C1F121200000430000430000
8	IVc = E(K _{SesAuthENC} , A55A TI CmdCtr 0000000000000000)	= E(K _{SesAuthENC} , A55A 9D00C4DF 0100 0000000000000000)
9	IVc	= 3E27082AB2ACC1EF55C57547934E9962
10	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	= E(K _{SesAuthENC} , IVc, 4000E0C1F121200000430000430000 80)
11	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	= 61B6D97903566E84C3AE5274467E89EA
12	Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData)	= 5F01009D00C4DF0261B6D97903566E84C3AE5274467E89EA
13	MAC(K _{SesAuthMAC} , Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData))	= 7BD75F991CB7A2C18DA09EEF047A8D04
14	MACT	= D799B7C1A0EF7A04
15	Lc (Length of 0261B6D97903566E84C3AE5274467E89EAD799B7C1A0EF7A04)	= 19

Table 19. Change NDEF file settings using Cmd.ChangeFileSettings...continued

Step	Command		Data
16	Cmd.ChangeFileSettings C-APDU	>	905F0000190261B6D97903566E84C3AE5274467E89EAD799B7C1A0EF7A0400
17	R-APDU (ResponseCode (E(K _{SesAuthENC} , ResponseData) MACt)	<	9100 57BFF87B1241E93D
18	R-APDU's MACt	=	57BFF87B1241E93D
19	Status CmdCounter + 1 TI (E(K _{SesAuthENC} , ResponseData)	=	0002009D00C4DF
20	MAC(K _{SesAuthMAC} , Status CmdCounter + 1 TI (E(K _{SesAuthENC} , ResponseData))	=	5457D1BFEBF8777B911222411CE9773D
21	MACt	=	57BFF87B1241E93D
22	Compare R-APDU's MACt and calculated MACt from step 14	=	True. Integrity and authenticity of the message received from the PICC - verified

6.10 AuthenticateEV2First with key 0x03

Table 20. Cmd.AuthenticateEV2First using Key No 0x03

Step	Command		Data Message
1	Key No	=	03
2	Key0	=	00000000000000000000000000000000
3	Cmd (INS)	=	71
4	Command header (Key No LenCap)	=	0300
5	Cmd.AuthenticateFirst C-APDU	>	9071000002030000
6	R-APDU (E(K ₃ , RndB) Response Code)	<	B875CEB0E66A6C5CD00898DC371F92D191AF
7	E(K ₃ , RndB)	=	B875CEB0E66A6C5CD00898DC371F92D1
8	Response Code	=	91AF (AF ... additional frame)
9	D(K ₃ , RndB)	=	91517975190DCEA6104948EFA3085C1B
10	PCD generates RndA	=	B98F4C50CF1C2E084FD150E33992B048
11	PCD prepares RndB' (rotate left by 1 byte)	=	517975190DCEA6104948EFA3085C1B91
12	RndA RndB'	=	B98F4C50CF1C2E084FD150E33992B048517975190DCEA6104948EFA3085C1B91
13	E(K ₃ , RndA RndB')	=	FF0306E47DFBC50087C4D8A78E88E62DE1E8BE457AA477C707E2F0874916A8B1
14	Cmd.AuthenticatePart2 C-APDU (INS = AF)	>	90AF000020FF0306E47DFBC50087C4D8A78E88E62DE1E8BE457AA477C707E2F0874916A8B100

Table 20. Cmd.AuthenticateEV2First using Key No 0x03...continued

Step	Command		Data Message
15	R-APDU E(K ₃ , TI RndA' PDcap2 PCDcap2)	<	0CC9A8094A8EEA683ECAAC5C7BF20584206D0 608D477110FC6B3D5D3F65C3A6A9100
16	E(K ₃ , TI RndA' PDcap2 PCDcap2)	=	0CC9A8094A8EEA683ECAAC5C7BF20584206D0 608D477110FC6B3D5D3F65C3A6A
17	Response Code		9100
18	D(K ₀ , TI RndA' PDcap2 PCDcap2)	=	7614281A8F4C50CF1C2E084FD150E33992B048 B9000000000000000000000000
19	TI (4 byte)	=	7614281A
20	RndA' (16 byte)	=	8F4C50CF1C2E084FD150E33992B048B9
21	PDcap2 (6 byte)	=	000000000000
22	PCDcap2 (6 byte)	=	000000000000
23	RndA (rotate right for 1 byte)	=	B98F4C50CF1C2E084FD150E33992B048
24	PCD compares sent RndA (from step 10) and received RndA (from step 20)	=	B98F4C50CF1C2E084FD150E33992B048 == B98F4C50CF1C2E084FD150E33992B048
25	SV 1 = [0xA5][0x5A][0x00][0x01] [0x00][0x80][RndA[15:14] [(RndA[13:8] ⊕ RndB[15:10])] [RndB[9:0] RndA[7:0]	=	A55A00010080B98FDD01B6693705CEA6104948 EFA3085C1B4FD150E33992B048
26	SV 2 = [0x5A][0xA5][0x00][0x01] [0x00][0x80][RndA[15:14] [(RndA[13:8] ⊕ RndB[15:10])] [RndB[9:0] RndA[7:0]	=	5AA500010080B98FDD01B6693705CEA6104948 EFA3085C1B4FD150E33992B048
27	Encryption Session Key (K _{SesAuthENC}) = CMAC (K ₀ , SV1)	=	7A93D6571E4B180FCA6AC90C9A7488D4
28	CMAC Session Key (K _{SesAuthMAC}) = = CMAC(K ₀ , SV2)	=	FC4AF159B62E549B5812394CAB1918CC

6.11 ISO SELECT Proprietary file by EF Name

This step is not needed, if for Writing of the data to the file is done by Cmd.WriteData (and not Cmd.ISOUpdateBinary).

Table 21. Select Proprietary Application using Cmd.ISOSelectFile

Step	Command		Data Message
1	ISO7816 AID – DF Application Name	=	D2760000850101
2	CLA	=	00
3	INS	=	A4
4	P1	=	00 (Select MF, DF or EF, by file identifier)
5	P2	=	0C
6	Lc	=	02
7	Command header	=	00A4000C02

NTAG 424 DNA and NTAG 424 DNA TagTamper features and hints

Table 21. Select Proprietary Application using Cmd.ISOSelectFile...continued

Step	Command		Data Message
8	Command data (ISO7816 – EF Name)	=	E105
9	Le	=	00
10	Cmd.Select C-APDU	>	00A4000C02E10500
11	R-APDU	<	9000

6.12 Write to Proprietary File - using Cmd.WriteData, CommMode.FULL

Table 22. Write Proprietary File (0xE105) - using Cmd.WriteData

Step	Command		Data
1	Cmd	=	8D
2	K _{SesAuthMAC} (as generated in Auth. [Table 20])	=	FC4AF159B62E549B5812394CAB1918CC
3	K _{SesAuthENC}	=	7A93D6571E4B180FCA6AC90C9A7488D4
4	CmdHeader	=	03 000000 000000
5	CmdCtr	=	0000
6	TI (as generated in Auth.)	=	7614281A
7	CmdData	=	0102030405060708090A
8	IVc = E(K _{SesAuthENC} , A55A TI CmdCtr 0000000000000000)	=	E(K _{SesAuthENC} , A55A 7614281A 0000 0000000000000000)
9	IVc	=	4C651A64261A90307B6C293F611C7F7B
10	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	=	E(K _{SesAuthENC} , IVc, 0102030405060708090A 80000000000000)
11	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	=	6B5E6804909962FC4E3FF5522CF0F843
12	Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData)	=	8D00007614281A030000000A00006B5E6804909962FC4E3FF5522CF0F843
13	MAC(K _{SesAuthMAC} , Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData))	=	426CD70CE153ED315E5B139CB97384AA
14	MACT	=	6C0C53315B9C73AA
15	Cmd.SetConfiguration C-APDU	>	908D00001F030000000A00006B5E6804909962FC4E3FF5522CF0F8436C0C53315B9C73AA00
16	R-APDU (ResponseCode E(K _{SesAuthENC} , ResponseData) MACT)	<	9100 C26D236E4A7C046D
17	R-APDU's MACT	=	C26D236E4A7C046D
18	Status CmdCounter + 1 TI E(K _{SesAuthENC} , ResponseData)	=	0001007614281A
19	MAC(K _{SesAuthMAC} , Status CmdCounter + 1 TI E(K _{SesAuthENC} , ResponseData))	=	86C2486D35237F6E974A437C4004C46D

Table 22. Write Proprietary File (0xE105) - using Cmd.WriteData...continued

Step	Command		Data
20	MACt	=	FC222E5F7A542452
21	Compare R-APDU's MACt and calculated MACt from step 14	=	true - Integrity of message received from the PICC verified

6.13 ISO SELECT CC file by EF Name

This step is not needed, if for Writing the data to the file is done by Cmd.WriteData (and not Cmd.ISOUpdateBinary).

Table 23. Select NDEF Application using Cmd.Select

Step	Command		Data Message
1	ISO7816 AID – DF Application Name	=	D2760000850101
2	CLA	=	00
3	INS	=	A4
4	P1	=	04 (select by DF name)
5	P2	=	0C
6	Lc	=	07
7	Command header	=	00A4040C07
8	Command data (ISO7816 AID – DF Name)	=	D2760000850101
9	Le	=	00
10	Cmd.Select C-APDU	>	00A4040C07D276000085010100
11	R-APDU	<	9000

6.14 AuthenticateAESNonFirst with key 0x00

By default, CC file has FileAR.ReadWrite set to 00. Therefore Authentication with Key0 needs to be done.

Table 24. Cmd.AuthenticateEV2NonFirst using Key No 0x00

Step	Command		Data Message
1	Key No	=	00
2	Key0	=	00000000000000000000000000000000
3	Cmd (INS)	=	77
4	Command header (Key No LenCap)	=	0000
5	Cmd.AuthenticateAESNonFirst C-APDU	>	90770000010000
6	R-APDU (E(K ₀ , RndB) Response Code)	<	A6A2B3C572D06C097BB8DB70463E22DC91AF
7	E(K ₀ , RndB)	=	A6A2B3C572D06C097BB8DB70463E22DC
8	Response Code	=	91AF (AF ... additional frame)
9	D(K ₀ , RndB)	=	6924E8D09722659A2E7DEC68E66312B8
10	PCD generates RndA	=	60BE759EDA560250AC57CDDC11743CF6
11	PCD prepares RndB' (rotate left by 1 byte)	=	24E8D09722659A2E7DEC68E66312B869
12	RndA RndB'	=	60BE759EDA560250AC57CDDC11743CF624E8D09722659A2E7DEC68E66312B869
13	E(K ₀ , RndA RndB')	=	BE7D45753F2CAB85F34BC60CE58B940763FE969658A532DF6D95EA2773F6E991
14	<u>Cmd.AuthenticatePart2</u> C-APDU	>	90AF000020BE7D45753F2CAB85F34BC60CE58B940763FE969658A532DF6D95EA2773F6E99100
15	R-APDU E(K ₀ , RndA')	<	B888349C24B315EAB5B589E279C8263E9100
16	E(K ₀ , RndA')	=	B888349C24B315EAB5B589E279C8263E
17	Response Code		9100
18	D(K ₀ , RndA')	=	BE759EDA560250AC57CDDC11743CF660
19	RndA (rotate right for 1 byte)	=	60BE759EDA560250AC57CDDC11743CF6
20	PCD compares sent RndA (from step 10) and received RndA (from step 20)	=	60BE759EDA560250AC57CDDC11743CF6 == 60BE759EDA560250AC57CDDC11743CF6
21	SV 1 = [0xA5][0x5A][0x00][0x01][0x00][0x80][RndA[15:14] [(RndA[13:8] ⊕ RndB[15:10])] [RndB[9:0] RndA[7:0]]	=	A55A0001008060BE1CBA32869572659A2E7DEC68E66312B8AC57CDDC11743CF6
22	SV 2 = [0x5A][0xA5][0x00][0x01][0x00][0x80][RndA[15:14] [(RndA[13:8] ⊕ RndB[15:10])] [RndB[9:0] RndA[7:0]]	=	5AA50001008060BE1CBA32869572659A2E7DEC68E66312B8AC57CDDC11743CF6

Table 24. Cmd.AuthenticateEV2NonFirst using Key No 0x00...continued

Step	Command		Data Message
23	Encryption Session Key (K _{SesAuthENC}) = CMAC (K ₀ , SV1)	=	4CF3CB41A22583A61E89B158D252FC53
24	CMAC Session Key (K _{SesAuthMAC}) = CMAC(K ₀ , SV2)	=	5529860B2FC5FB6154B7F28361D30BF9

6.15 Write to CC - using Cmd.WriteData, CommMode.PLAIN

This step is needed to switch the lifecycle from INITIALIZED to READ-ONLY as per NFC Forum T4T spec.

Table 25. Write CC File (0xE103) - using Cmd.WriteData

Step	Command		Data Message
1	Cmd	=	8D
2	FileNo	=	01
3	Offset	=	0E0000
4	Length	=	120000
5	Data	=	FF0506E10500808283000000000000000000
6	Cmd.WriteData C-APDU	>	908D000019010E0000120000FF0506E10500808283000000000000000000
7	R-APDU	<	9100

6.16 Changing the Key

Only changing of keys nr. 0x2 and 0x0 are shown in the following step. It is highly recommended to configure all the Application Keys during personalization procedure.

6.16.1 KeyNo to be changed does not equal AuthKey

Case 1: Key number to be changed ≠ Key number for currently authenticated session

KeyNo 0x02 will be changed, while currently authenticated with KeyNo 0x00. AES-128 Master Key key diversification is used in the following example. Keys can be diversified using NXP suggested diversification method described in [13].

Table 26. Example for Cmd.ChangeKey in Secure Messaging using Case 1

Step	Command		Data Message
1	K _{SesAuthMAC}	=	5529860B2FC5FB6154B7F28361D30BF9
2	K _{SesAuthENC}	=	4CF3CB41A22583A61E89B158D252FC53
3	Old Key (KeyNo 0x02)	=	00000000000000000000000000000000
4	Master Key (for diversification) (KeyNo 0x02)	=	C8EE97FD8B00185EDC7598D7FEBC818A
	New Key = AES-Diversified New Key by UID	=	F3847D627727ED3BC9C4CC050489B966
5	Version for new key (intended)	=	01
6	Old Key ⊕ New Key	=	F3847D627727ED3BC9C4CC050489B966

Table 26. Example for Cmd.ChangeKey in Secure Messaging using Case 1...continued

Step	Command		Data Message
7	CRC32 (NewKey)	=	789DFADC
8	TI	=	7614281A
9	Initialization vector for encryption	=	A55A7614281A020000000000000000
10	IVe = E(K _{SesAuthENC} , IV)	=	307EDE1814707F30CFE603DD6CA62353
11	KeyData = Old Key ⊕ New Key Version of New Key CRC32 Padding	=	F3847D627727ED3BC9C4CC050489B96601789 DFADC80000000000000000000000
12	E(K _{SesAuthENC} , KeyData)	=	2CF362B7BF4311FF3BE1DAA295E8C68DE0905 0560D19B9E16C2393AE9CD1FAC7
13	Key No	=	02
14	CmdCtr	=	0200
15	Input for CMAC	=	C402007614281A022CF362B7BF4311FF3BE1DA A295E8C68DE09050560D19B9E16C2393AE9CD 1FAC7
16	IV for CMACing	=	00000000000000000000000000000000
17	CMAC	=	EA5D2E0CBFE24C0BCBCD501D21060EE6
	CMACt	=	5D0CE20BCD1D06E6
19	Cmd.ChangeKey C-APDU	>	90C4000029022CF362B7BF4311FF3BE1DAA295 E8C68DE09050560D19B9E16C2393AE9CD1FAC 75D0CE20BCD1D06E600
20	R-APDU	<	203BB55D1089D5879100

6.16.2 KeyNo to be changed equals AuthKey

Case 2: Key number to be changed = Key number for currently authenticated session

KeyNo 0x00 will be changed while being currently authenticated with KeyNo 0x00.

Table 27. Example for Cmd.ChangeKey in Secure Messaging using Case 2

Step	Command		Data Message
1	Old Key (KeyNo 0x00)	=	00000000000000000000000000000000
2	New Key (KeyNo 0x00)	=	5004BF991F408672B1EF00F08F9E8647
3	Version for new key (intended)	=	01
5	K _{SesAuthMAC} Key	=	5529860B2FC5FB6154B7F28361D30BF9
6	K _{SesAuthENC} Key	=	4CF3CB41A22583A61E89B158D252FC53
7	Current IV	=	A55A7614281A03000000000000000000
8	TI	=	7614281A
9	CmdCtr	=	0300
10	Padding	=	80000000000000000000000000000000
11	Plaintext Input (NewKey KeyVer Padding)	=	5004BF991F408672B1EF00F08F9E86470180000 00000000000000000000000000000000
12	IVc	=	01602D579423B2797BE8B478B0B4D27B

Table 27. Example for Cmd.ChangeKey in Secure Messaging using Case 2...continued

Step	Command		Data Message
13	E(K _{SesAuthENC} , IVc, CmdData Padding (if needed))	=	C0EB4DEEFEDDF0B513A03A95A754918 18580503190D4D05053FF75668A01D6FD
14	MAC input = Cmd CmdCtr T1 CmdHeader E(K _{SesAuthEnc} , CmdData)	=	C403007614281A00C0EB4DEEFEDDF0B51 3A03A95A75491818580503190D4D05053FF7566 8A01D6FD
15	CMAC	=	B7A60161F202EC3489BD4BEDEF64BB32
16	CMAcT	=	A6610234BDED6432
17	Cmd.ChangeKey C-APDU	>	90C400002900C0EB4DEEFEDDF0B513A03A95A 75491818580503190D4D05053FF75668A01D6FD A6610234BDED643200
18	R-APDU	<	9100

7 Special functionalities

7.1 Configuration of NDEF application and PICC attributes

Special command: SetConfiguration

Authentication with key AppMasterKey is needed

CommMode.Full needed

It is possible to configure:

- Enable RandomID
- Disable chaining with WriteData
- Enable LRP mode (irreversible)
- Failed authentication counter configuration
- Enable Strong back modulation

7.2 Random ID - RID

This feature is used to retain end consumer privacy, avoid tracking and to meet latest GDPR regulations. In the combination with PICCData encryption, the real NTAG 424's UID cannot be revealed, the PICC responds with random ID (4 bytes) during ISO14443-3 anticollision.

Note:

- If Random ID feature is enabled, the ATQA value is changed to 0x0304 (default is 0x0344).
- Enabling Random ID feature is irreversible process - meaning that it cannot be disabled once it is enabled.

Prerequisites: Active Authentication with the AppMasterKey (AppKey00)

CommMode: FULL

Table 28. Enabling Random ID - RID

Step	Command		Data
1	Cmd	=	5C
2	K _{SesAuthMAC}	=	FE4EDBF46536557E304682F33E63A84F
3	K _{SesAuthENC}	=	7951A705F47F3C29B596454DC1490383
4	CmdHeader - Option (Command option)	=	00
5	CmdCtr	=	0000
6	TI	=	D779B1D0
8	CmdData	=	02 (enable Random ID)
9	IVc = E(K _{SesAuthENC} , A55A TI CmdCtr 0000000000000000)	=	E(K _{SesAuthENC} , A55A D779B1D0 0000 0000000000000000)

Table 28. Enabling Random ID - RID...continued

Step	Command		Data
10	IVc	=	FEFB918047F385563FA8356DE86E5182
11	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	=	E(K _{SesAuthENC} , IVc, 02 80000000000000000000000000000000)
12	E(K _{SesAuthENC} , IVc, CmdData Padding (if necessary))	=	8EA0138A7AF6FC8E99DF2A3A305602C4
13	Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData)	=	5C0000D779B1D0008EA0138A7AF6FC8E99DF2A3A305602C4
14	MAC(K _{SesAuthMAC} , Cmd CmdCounter TI CmdHeader E(K _{SesAuthENC} , CmdData))	=	393A297A133CAA920F28A5C3B9138A4A
15	MAcT	=	3A7A3C9228C3134A
16	Cmd.SetConfiguration C-APDU	>	905C000019008EA0138A7AF6FC8E99DF2A3A305602C43A7A3C9228C3134A00
17	R-APDU (ResponseCode (E(K _{SesAuthENC} , ResponseData) MAcT)	<	910086044208CAD1676A
18	R-APDU's MAcT	=	86044208CAD1676A
19	Status CmdCounter + 1 TI (E(K _{SesAuthENC} , ResponseData)	=	000100D779B1D0
20	MAC(K _{SesAuthMAC} , Status CmdCounter + 1 TI (E(K _{SesAuthENC} , ResponseData))	=	F18690046942890879CAF1D17567336A
21	MAcT	=	86044208CAD1676A
22	Compare R-APDU's MAcT and calculated MAcT from step 14	=	true - Integrity of message received from the PICC verified

Result:

Tap1 (POR, REQA, ISO14443-3 anticollision CL 1): UID = 08F84941

Tap2 (POR, REQA, ISO14443-3 anticollision CL 1): UID = 08B0ADD0

Tap3 (POR, REQA, ISO14443-3 anticollision CL 1): UID = 08BFD57C

7.3 Get UID

With enabled feature of Random ID the only way to retrieve "real" NTAG 424 DNA's UID is to send Cmd.GetCardUID in CommMode.Full.

Prerequisites: Active Authentication with any application key

CommMode: FULL

Table 29. Get NTAG 424 DNA's UID

Step	Command		Data
1	Cmd	=	51
2	SesAuthMACKey	=	379D32130CE61705DD5FD8C36B95D764
3	SesAuthENCKey	=	2B4D963C014DC36F24F69A50A394F875
4	Cmd	=	51
5	CmdCtr	=	0000
6	TI	=	DF055522
7	CmdHeader	=	n/a
8	CmdData	=	n/a
9	MAC(K _{SesAuthMAC} , Cmd CmdCounter TI)	=	CC8E8C2CD015945AFDDD7DA9B19BB9E3
10	MACT	=	8E2C155ADDA99BE3
11	Cmd.SetConfiguration C-APDU	>	90510000088E2C155ADDA99BE300
12	R-APDU (ResponseCode (E(K _{SesAuthENC} , ResponseData) MACT	<	70756055688505B52A5E26E59E329CD6595F672298EA41B79100
13	R-APDU's MACT	=	595F672298EA41B7
14	MAC(K _{SesAuthMAC} , Status Cmd CmdCounter + 1 TI (E(K _{SesAuthENC} , ResponseData))	=	F4593D5FAB671F225798C4EA894195B7
15	Compare R-APDU's MACT and calculated MACT from step 14	=	true - Integrity of message received from the PICC verified
16	ResponseCode (E(K _{SesAuthENC} , ResponseData)	=	70756055688505B52A5E26E59E329CD6
17	CmdCtr	=	0100 (increased by one on the PICC side)
18	IVr for Encryption = E(K _{SesAuthENC} , 5AA5 TI CmdCtr 0000000000000000)	=	7F6BB0B278EA054CBD238C5D9E9E342B
19	D(K _{SesAuthENC} , IV, Response Data Padding)	=	04958CAA5C5E80800000000000000000
20	UID	=	04958CAA5C5E80

7.4 Failed Authentications Counter

This feature improves countermeasures for potential side channel attacks, especially in AES mode. In LRP mode, side channel attack resistance is done by protocol itself, but it can be enabled for LRP mode as well.

Note: Originality keys do not support the failed authentication counter feature. Anyhow Orig.keys (LRP) have SCA resistance by protocol itself.

Every KeyID.AppKeys has its own instance of counter set:

- TotFailCtr (2 bytes)
 - Increases by 1 on each unsuccessful authentication
 - Decreases by value defined with TotFailCtrDecr
 - when TotFailCtrLimit is reached, related key cannot be used for Authentication anymore
- SeqFailCtr (1 byte)
 - Increases by 1 on each consecutive failed authentication
 - If value 50d reached, subsequent authentication attempts are delayed - gradually on all next 50d. Until 255d.
 - successful Authentication resets counter to 0
- SpentTimeCtr (2 bytes)
 - Counts the time "spent" after defined FWT, caused by delayed response of Failed Authentications Counter feature
 - Increased by SpentTimeUnit, which depends on FWT

Default values of counter sets:

- TotFailCtr: 0d
- TotFailCtrLimit: 1000d
- TotFailCtrDecr: 10d
- SeqFailCtr: 0d
- SpentTimeCtr: 0d

When changing an KeyID.AppKey with Cmd.ChangeKey, the related instance set of these three counters is reset to their default values at delivery.

Each failed authentication will increase targeted Key's counter of TotFailCtr by 1. On a successful Authentication, TotFailCtr it is decreased by TotFailCtrDecr value (default 10), to cope with false - positives. By default after 50 consecutive failed authentication attempts NTAG 424 DNA starts to introduce a delay into its response (SW code response of 91AD).

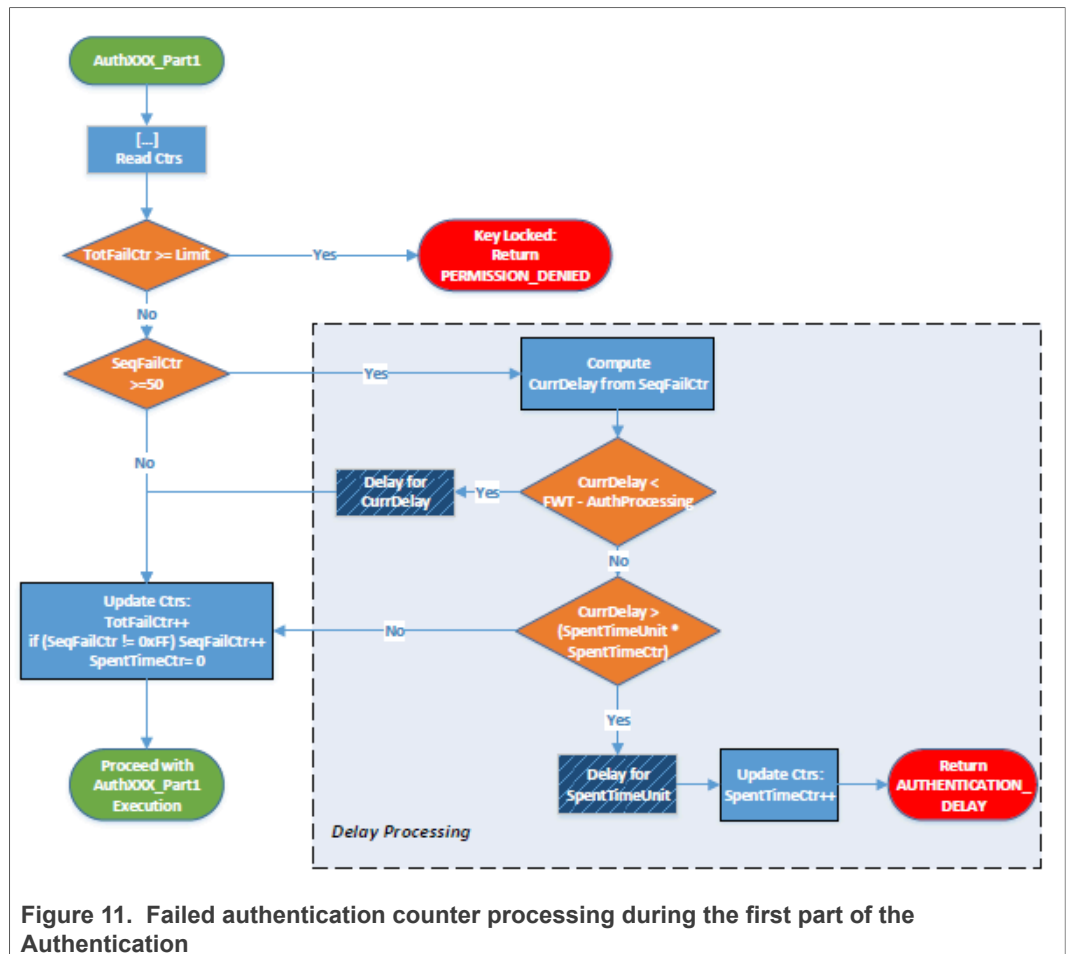
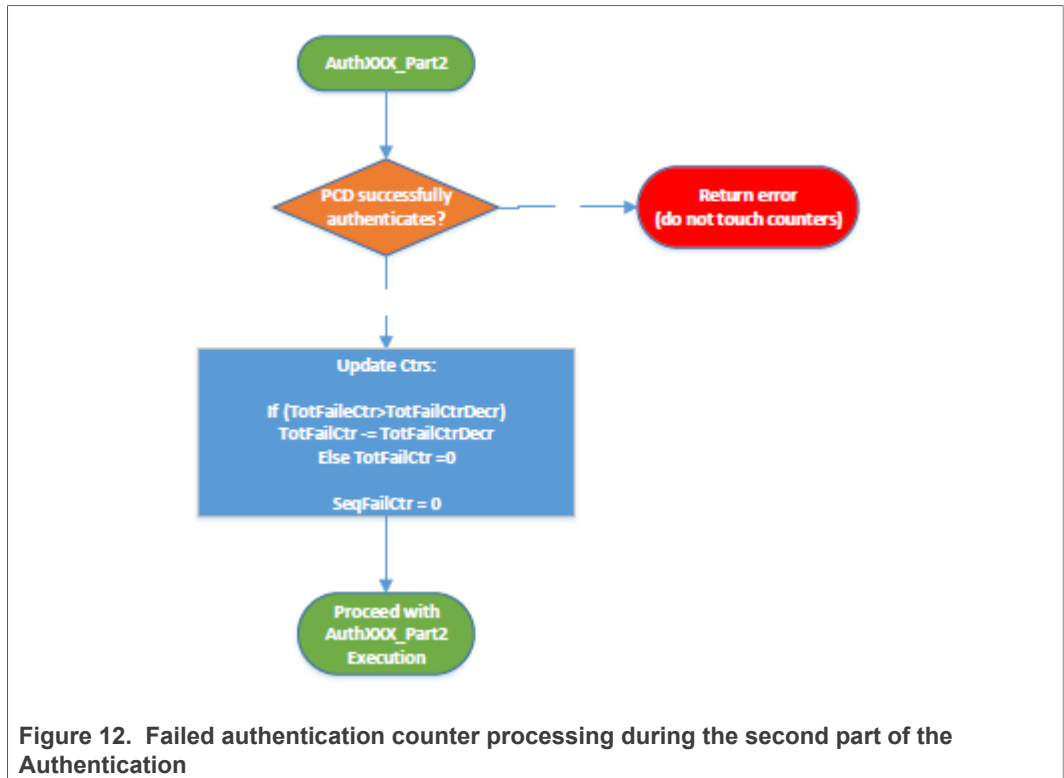


Figure 11. Failed authentication counter processing during the first part of the Authentication



7.5 TagTamper

NTAG 424 DNA TagTamper offers an NFC Forum-compliant solution to reflect, if the sealing of a product is opened. This works without a dedicated app on the NFC reader/writer device. It only requires the capability of reading out NFC Forum Type 4 Tag [5]. NTAG 424 DNA TagTamper has four pads. Two pads are used for antenna connection and the other two used to connect a detection wire. At start-up, the IC checks that the tag tamper wire. If opened, this will be recorded as permanent status in NVM (TTPermStatus). The result can be mirrored in the NDEF message.

Measurement is done automatically during the boot-up of the NTAG 424 DNA. It will be only done during processing of the first ISO/IEC 14443-4 command after complete activation, if the current TTPermStatus is still set to Close. It does not have any influence on any ISO standard time constraints. If PICC detects open tamper loop, TTPermStatus is updated. Measurement on the boot will not be triggered anymore.

In addition, a specific command (Cmd.GetTTStatus) triggers tamper loop measurement and the Tag returns both the permanent (TTPermStatus) and current status (TTCurrStatus) of the tamper loop connection. NTAG 424 DNA is a passive tag powered by an RF field, therefore it cannot trigger measurement by itself. Physical design of a final tag application with counter measures should be used to mitigate fraudulent use - as opening and fixing the tamper loop / seal in between measurements.

7.6 SDMReadCtr Limit

The SDMReadCtrLimit can be enabled by setting a customized value with `Cmd.ChangeFileSettings`. It can be retrieved with `Cmd.GetFileSettings`. This way reading of the NDEF file can be limited after SDMReadCtr reaches SDMReadCtrLimit. When SDMReadCtrLimit is reached, no reading with `Cmd.ReadData` or `Cmd.ISOReadBinary` can be executed. This feature can be a potential risk for DoS attacks.

Main use cases:

- To limit usage/tap number of a single PICC
- To limit conditions for Secure Dynamic Messaging side channel attacks

Feature can be disabled by `Cmd.ChangeFileSettings`, or by setting the SDMReadCtrLimit value to FFFFFFFF.

8 Originality Signature Verification

8.1 Symmetric check

Four (4) secret originality keys (also named as PICC Keys) are present on each individual NTAG 424 DNA, type of AES-128:

- Are written on the IC at the production in the NXP factory
- Keys are created in NXP Fabs HSM and never leave secure environment
- Cannot be changed after the IC leaves the NXP factory
- Originality Check is done by executing a successful LRP Authentication (not AES!) with one of the Originality keys. LRP mode needs to be enabled with command `Cmd.SetConfiguration`.
- These keys are shared only towards NXP's licensees
Sharing procedure of these keys is written in the data sheet [\[1\]](#).

8.2 Asymmetric check

NTAG 424 DNA contains the NXP Originality Signature:

- It is computed according to Elliptic Curve DSA (ECDSA) based on the UID
- Key pair created in NXP Fabs HSM. Private key stored in high secure HSM in NXP premises
- Signature is 56 bytes long and according to SEC standard the **secp224r1** curve is taken

Asymmetric procedure consists of:

- retrieve Originality Signature (56 bytes) from the PICC with `Cmd.Read_Sig` command (NTAG 424 needs to be in ISO14443 - Layer 4 level).
- public key is required by the verifier - available for public below
- ECDSA signature verifying operation needs to be applied - procedure and sample code (C#, Java, C) can be found in Application Note [\[9\]](#)

NTAG public key:

**048A9B380AF2EE1B98DC417FECC263F8449C7625CECE82D9B916C992DA209D68
422B81EC20B65A66B5102A61596AF3379200599316A00A1410**

04 = IETF protocols use the [\[SEC1\]](#) representation of a point on an elliptic curve, which is a sequence of the following fields:

SEC1 point representation	
Field	Description
B0	{02, 03, 04}, where 02 or 03 represent a compressed point (x only), while 04 represents a complete point (x,y)
X	x coordinate of a point
Y	y coordinate of a point, optional (present only for B0=04)

Table 30. Asymmetric Originality Signature verification

Step	Command		Data
1	Cmd	=	3C
2	Cmd header (Address)	>	00
3	C-APDU	>	903C0000010000
4	R-APDU (56 bytes of ECDSA)	=	D1940D17CFEDA4BFF80359AB975F9F6514313 E8F90C1D3CAAF5941AD744A1CDF9A83F883C AFE0FE95D1939B1B7E47113993324473B785D2 19190
5	ECDSA verification		
6	Eliptic curve name	=	secp224r1
7	Input data (UID)	=	04518DFAA96180
8	Public key point coordinate xD (28 bytes)	=	8A9B380AF2EE1B98DC417FECC263F8449C762 5CECE82D9B916C992DA
9	Public key point coordinate yD (28 bytes)	=	209D68422B81EC20B65A66B5102A61596AF337 9200599316A00A1410
10	Signature part 1 r	=	D1940D17CFEDA4BFF80359AB975F9F6514313E 8F90C1D3CAAF5941AD
11	Signature part 2 s	=	744A1CDF9A83F883CAFE0FE95D1939B1B7E47 113993324473B785D21
12	Use ECDSA Verify tools (free online tools)	=	Signature valid

9 System implementation concepts

Most common system used with NTAG 424 DNA is pictured below.

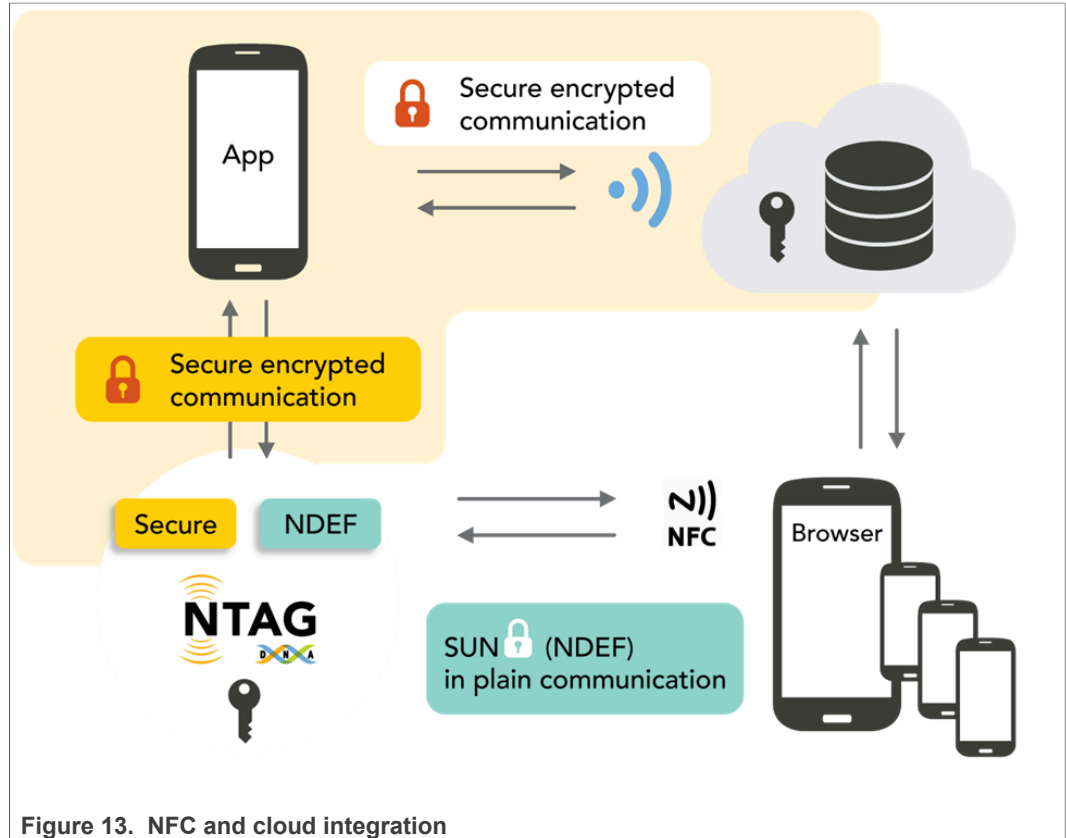


Figure 13. NFC and cloud integration

9.1 Online system

This kind of system is possible with NFC device broadband connectivity (data transfer) and robust backend system - usually cloud based service. By this approach, no keys need to be stored on NFC device, thus no secure element is used on NFC device. It is used only for relaying messages between cloud and the NTAG. It is advisable that all the keys (or master key) are securely stored on backend's HSM.



Figure 14. Main use case system concept

9.2 Offline system

Offline systems usually target closed loop, offline authentication applications. Application provides a proof of authenticity of the Tag or the product to which the Tag is applied to.

For optimal secure solution, host side needs to have:

- AES-128 or LRP crypto algorithm implemented
- secure key storage (e.g. Secure element)

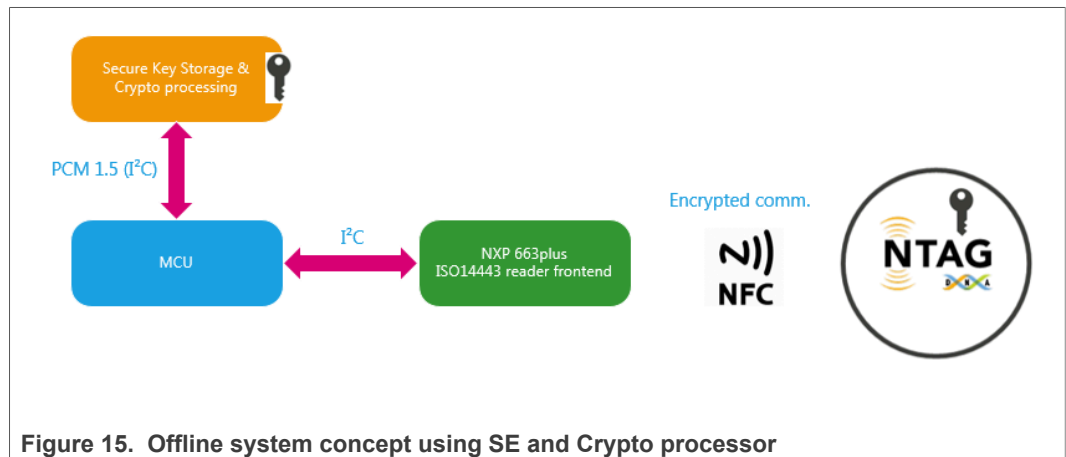


Figure 15. Offline system concept using SE and Crypto processor

10 Supporting tools

10.1 Software

Name	Description	Source
RFIDDiscover 4.5.1.8	PC software tool to evaluate NTAG 424 DNA PICC	NXP DocStore
NXPRdLib	C# API for developing Windows-based applications	NXP DocStore
TapLinX	Java-based SDK for developing Android applications, supporting all NXP RFID products, also NTAG 424 DNA	https://www.mifare.net/
TagInfo	Android and iOS based application to get detailed info of tapped NXP RFID products	Google PlayStore, Apple AppStore
TagWriter	Android application to configure, write NDEF data to NXP RFID products	Google PlayStore

11 References

- [1] **Data sheet** — NTAG 424 Product data sheet, doc.no. 4654**^[1]
- [2] **NIST Special Publication 800-38B** — National Institute of Standards and Technology (NIST) – Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005.
- [3] **ISO/IEC 14443-4:2016** — Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 4: Transmission protocol
- [4] **NFC Data Exchange Format (NDEF)** — Technical Specification — NFC Forum, 24.07.2006, Version 1.0
- [5] **NFC Forum T4T spec.** — Type 4 Tag Technical Specification Version 1.0 2017-08-28 [T4T] NFC Forum™
- [6] **ISO/IEC 7816-4:2005** — ISO JTC 1/SC 27 Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. ISO/IEC 7816-4:2005, January 2005.
- [7] **Application note** — AN11276 NTAG Antenna Design Guide, doc.no. 2421**
- [8] **Application note** — AN11941 Tag Tamper - Tamper Loop design hints, doc.no. 4247**
- [9] **Application note** — AN11350 NTAG Originality Signature Validation, doc.no. 2604**
- [10] **NFC Forum Activity spec.** — Activity Technical Specification Version 1.1 2014-01-23 [ACTIVITY] NFC Forum™
- [11] **Application note** — AN12321 NTAG 424 DNA (TagTamper) features and hints - LRP mode, doc.no. 5244**
- [12] **Application note** — AN12304 Leakage Resilient Primitive (LRP) Specification, doc.no. 5244**
- [13] **Application note** — AN10922 Symmetric key diversifications, doc.no. 1653**

[1] ** ... document version number

12 Legal information

12.1 Definitions

Draft — A draft status on a document indicates that 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 in a draft version of a document and shall have no liability for the consequences of use of such information.

12.2 Disclaimers

Limited warranty and liability — 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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 life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at 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. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out of the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US\$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

12.3 Licenses

Purchase of NXP ICs with NFC technology

Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

12.4 Trademarks

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

NTAG — is a trademark of NXP B.V.

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Tab. 1.	Abbreviations	3	Tab. 18.	Write NDEF File - using Cmd.WriteData	32
Tab. 2.	SDM Session Key Generation	10	Tab. 19.	Change NDEF file settings using Cmd.ChangeFileSettings	34
Tab. 3.	Decryption of PICCData	12	Tab. 20.	Cmd.AuthenticateEV2First using Key No 0x03	35
Tab. 4.	Decryption of PICCData	13	Tab. 21.	Select Proprietary Application using Cmd.ISOSelectFile	36
Tab. 5.	CMAC calculation when CMACInputOffset == CMACOffset	15	Tab. 22.	Write Proprietary File (0xE105) - using Cmd.WriteData	37
Tab. 6.	CMAC calculation when CMACInputOffset ! = CMACOffset	17	Tab. 23.	Select NDEF Application using Cmd.Select	38
Tab. 7.	Communication modes in SSM	19	Tab. 24.	Cmd.AuthenticateEV2NonFirst using Key No 0x00	39
Tab. 8.	Example of CommMode.MAC on Cmd.GetFileSettings command	21	Tab. 25.	Write CC File (0xE103) - using Cmd.WriteData	40
Tab. 9.	Example of CommMode.FULL on Cmd.Write to File No 0x02	22	Tab. 26.	Example for Cmd.ChangeKey in Secure Messaging using Case 1	40
Tab. 10.	ISO14443-4 PICC Activation	24	Tab. 27.	Example for Cmd.ChangeKey in Secure Messaging using Case 2	41
Tab. 11.	Select NDEF Application using Cmd.ISOSelect	25	Tab. 28.	Enabling Random ID - RID	43
Tab. 12.	Get file settings of NDEF File	26	Tab. 29.	Get NTAG 424 DNA's UID	45
Tab. 13.	Get Version	27	Tab. 30.	Asymmetric Originality Signature verification	51
Tab. 14.	Cmd.AuthenticateEV2First using Key No 0x00	29			
Tab. 15.	NDEF Message Creation	30			
Tab. 16.	NDEF Message Creation	31			
Tab. 17.	Write NDEF File - using Cmd.ISOUpdateBinary	32			

Figures

Fig. 1.	UID, NFCCounter (PICCData) and CMAC mirror 8	Fig. 8.	Plain communication in SSM 20
Fig. 2.	E(UID + NFCCounter = PICCData), CMAC mirror 8	Fig. 9.	MACed communication in SSM 21
Fig. 3.	E(PICCData), E(TagTamper status) and CMAC mirror 9	Fig. 10.	Fully enciphered communication in SSM 22
Fig. 4.	E(PICCData), E(Static File Data) and CMAC mirror 9	Fig. 11.	Failed authentication counter processing during the first part of the Authentication 47
Fig. 5.	NFC Reader/Writer technology poll 9	Fig. 12.	Failed authentication counter processing during the second part of the Authentication 48
Fig. 6.	NTAG 424 DNA Secure messaging options 19	Fig. 13.	NFC and cloud integration 52
Fig. 7.	SSM Session Key generation after successful Authentication 20	Fig. 14.	Main use case system concept 52
		Fig. 15.	Offline system concept using SE and Crypto processor 53

Contents

1	Abbreviations	3	6.11	ISO SELECT Proprietary file by EF Name	36
2	Introduction	4	6.12	Write to Proprietary File - using Cmd.WriteData, CommMode.FULL	37
2.1	About this document	4	6.13	ISO SELECT CC file by EF Name	38
2.2	Key benefits using NTAG 424 DNA	4	6.14	AuthenticateAESNonFirst with key 0x00	39
2.3	Target applications	4	6.15	Write to CC - using Cmd.WriteData, CommMode.PLAIN	40
2.4	Standards compliancy	5	6.16	Changing the Key	40
2.4.1	ISO 14443	5	6.16.1	KeyNo to be changed does not equal AuthKey	40
2.4.2	ISO 7816-4	5	6.16.2	KeyNo to be changed equals AuthKey	41
2.4.3	NFC Forum compliancy	5	7	Special functionalities	43
3	Definition of variables used in examples	6	7.1	Configuration of NDEF application and PICC attributes	43
3.1	Byte order	7	7.2	Random ID - RID	43
3.1.1	LSB representation	7	7.3	Get UID	44
3.1.2	MSB representation	7	7.4	Failed Authentications Counter	45
4	Secure Dynamic Messaging (SDM)	8	7.5	TagTamper	48
4.1	Mirroring commons	8	7.6	SDMReadCtr Limit	49
4.2	SUN generation procedure	9	8	Originality Signature Verification	50
4.3	SDM Session Key Generation	10	8.1	Symmetric check	50
4.4	SUN Mirroring	10	8.2	Asymmetric check	50
4.4.1	PICCCData mirror	10	9	System implementation concepts	52
4.4.2	PICCCData Encrypted mirror	11	9.1	Online system	52
4.4.2.1	Encryption of PICCCData	11	9.2	Offline system	53
4.4.2.2	Decryption of PICCCData	11	10	Supporting tools	54
4.4.3	SDMENCFileData mirror	13	10.1	Software	54
4.4.3.1	Encryption of SDMENCFileData	13	11	References	55
4.4.3.2	Decryption of SDMENCFileData	13	12	Legal information	56
4.4.4	SDMMAC mirror	14			
4.4.4.1	SDMMAC	14			
4.4.4.2	SDMMAC calculation	14			
5	Standard Secure Messaging (SSM)	19			
5.1	SSM Session Keys generation	20			
5.2	CommMode.Plain	20			
5.3	CommMode.MAC	21			
5.4	CommMode.Full	22			
6	Personalization example	24			
6.1	ISO14443-4 PICC Activation	24			
6.2	Originality signature verification	25			
6.3	ISO SELECT NDEF application using DF Name	25			
6.4	Get File Settings	26			
6.5	Get Version	27			
6.6	AuthenticateEV2First with key 0x00	29			
6.7	Prepare NDEF message	30			
6.7.1	NDEF	30			
6.7.2	NDEF Length	31			
6.7.3	NDEF Length and header	31			
6.7.4	Prepare NDEF message	31			
6.8	Write NDEF file	31			
6.8.1	Write NDEF File - using Cmd.ISOUpdateBinary, CommMode.PLAIN	32			
6.8.2	Write NDEF File - using Cmd.WriteData, CommMode.FULL	32			
6.9	Change NDEF File Settings	34			
6.10	AuthenticateEV2First with key 0x03	35			

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2020.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

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

Date of release: 17 November 2020

Document identifier: AN12196

Document number: 507218