```
This is the protocol of an authentification using key 1 (read & write access) that was
a CHANGED DES key (key value 'd10023456789abcd'). After the 'setKeyVersion to 0' the resulting key is 'd00022446688aacc'.

method: authenticateD40
key length: 8 data: d10023456789abcd keyNo: 1
method: authenticateD40
step 01 get encrypted rndB from card
method: authenticateD40
- send auth apdu   apdu     length: 7 data: 900a0000010100
method: authenticateD40
- receive response response length: 10 data: eb0533b4bc89afcf91af
method: authenticateD40
step 02 get the encrypted rndB from response data
method: getData
responseAPDU length: 10 data: eb0533b4bc89afcf91af
method: getData
responseData length: 8 data: eb0533b4bc89afcf
method: authenticateD40
- encryptedRndB length: 8 data: eb0533b4bc89afcf
method: authenticateD40
step 03 setKeyVersion to 00 for DES keys
method: authenticateD40
- DES key provided      length: 8 data: d10023456789abcd
method: setKeyVersion
a length: 8 data: d10023456789abcd offset: 0 length: 8 version: 0
method: authenticateD40
- DES key w/keyVersion 0 length: 8 data: d00022446688aacc
method: authenticateD40
step 04 get a TDES key from the DES key
method: getTDesKeyFromDesKey
key length: 8 data: d00022446688aacc
method: getTDesKeyFromDesKey
TDES key length: 24 data:
```

```
d0002244668 8aaccd0002244668 8aaccd0002244668 8aacc
method: authenticateD40
- DES key   length: 8 data: d00022446688aacc
method: authenticateD40
- TDES key length: 24 data: d00022446688aaccd00022446688aaccd00022446688aacc
method: authenticateD40
step 06 decrypt the encRndB using TripeDES.decrypt with key key length: 8 data: d00022446688aacc iv0 length: 8 data: 0000000000000000
method: authenticateD40
- encrypted rndB length: 8 data: eb0533b4bc89afcf
method: authenticateD40
- decrypted rndB length: 8 data: c4a3468fb0d87e74
method: authenticateD40
step 06 rotate the decrypted rndB by 1 position/byte to the left
method: authenticateD40
- rndB              length: 8 data: c4a3468fb0d87e74
method: rotateLeft
data length: 8 data: c4a3468fb0d87e74
method: authenticateD40
- rndB left rotated length: 8 data: a3468fb0d87e74c4
method: authenticateD40
step 07 generate a random rndA
method: getRandomData
key length: 8 data: 0000000000000000
method: getRandomData
length: 8
method: authenticateD40
- rndA length: 8 data: 45cc39928713e1c0
method: authenticateD40
step 08 concatenate rndA || rndB left rotated
method: authenticateD40
- rndA || rndB left rotated length: 16 data: 45cc39928713e1c0a3468fb0d87e74c4
method: authenticateD40
step 09 copy encryptedRndB to iv1 from position 0 to 8
```

```
method: authenticateD40
iv1 length: 8 data: eb0533b4bc89afcf
method: authenticateD40
step 09 copy encryptedRndB to iv1 from position
method: authenticateD40
step 10 encrypt rndA || rndB left rotated
method: authenticateD40
        Note: we are encrypting the data by DEcrypting the plaintext due to PICC characteristics
method: authenticateD40
using mode case SEND_MODE = XOR w/ previous ciphered block --> decrypt
method: authenticateD40
step 10 encryption magic starting ********************
method: tripleDesSendModeDecryption
*** start of the manual decryption ***
method: tripleDesSendModeDecryption
the ciphertext is 16 bytes long so we need to run 2 rounds to decrypt (length / 8)
method: tripleDesSendModeDecryption
******** manual decryption text start ********
method: tripleDesSendModeDecryption
SEND mode means: XORing the ciphertext with previous ciphered block, than DEcrypt
method: tripleDesSendModeDecryption
tdesKey length: 24 data: d00022446688aaccd00022446688aaccd00022446688aacc
method: tripleDesSendModeDecryption
1 starting with an empty 'cipheredBlock' of 8 bytes length = DES block length
method: tripleDesSendModeDecryption
cipheredBlock    length: 8 data: 0000000000000000
method: tripleDesSendModeDecryption
2 split the ciphertext into blocks of 8 bytes
method: tripleDesSendModeDecryption
ciphertext      length: 16 data: 45cc39928713e1c0a3468fb0d87e74c4
method: tripleDesSendModeDecryption
ciphertextBlock1 length: 8 data: 45cc39928713e1c0
```

```
method: tripleDesSendModeDecryption
ciphertextBlock2 length: 8 data: a3468fb0d87e74c4
method: tripleDesSendModeDecryption
3 XORing ct1 with cipheredBlock
method: tripleDesSendModeDecryption
ct1 Xored        length: 8 data: 45cc39928713e1c0
method: tripleDesSendModeDecryption
4 decrypt ct1Xored using TripleDES.decrypt
method: tripleDesSendModeDecryption
ct1Xored decrypt length: 8 data: 88e199b02da83367
method: tripleDesSendModeDecryption
5 copy ct1XoredDecrypted to cipheredBlock
method: tripleDesSendModeDecryption
cipheredBlock    length: 8 data: 88e199b02da83367
method: tripleDesSendModeDecryption
6 XORing ct2 with cipheredBlock
method: tripleDesSendModeDecryption
ct2Xored         length: 8 data: 2ba71600f5d647a3
method: tripleDesSendModeDecryption
7 decrypt ct2Xored using TripleDES.decrypt
method: tripleDesSendModeDecryption
ct2 Xored decrypt length: 8 data: 557208d962ae4b4f
method: tripleDesSendModeDecryption
8 Note: for more data this would be extended but we are ready now
method: tripleDesSendModeDecryption
9 concatenate ct1XoredDecrypted and ct2XoredDecrypted to plaintext
method: tripleDesSendModeDecryption
plaintext length: 16 data: 88e199b02da83367557208d962ae4b4f
method: tripleDesSendModeDecryption
******** manual decryption text end **********
method: authenticateD40
XOR w/ previous ciphered block --> decrypt
method: authenticateD40
data before XORing data length: 16 data: 45cc39928713e1c0a3468fb0d87e74c4 cipheredBlock length: 8 data: 0000000000000000
method: authenticateD40
running a 2 round loop to XOR rndArndBLeftRotated
```

```
with the previous cipheredBlock and DEcrypt the block using TripleDES
method: authenticateD40
The outer loop is running for i=0 to <16 in steps of 8
method: authenticateD40
outer loop i: 0
method: authenticateD40
The inner loop is running for j=0 to <8 in steps of 1
method: authenticateD40
TripleDES.decrypt cipheredBlock length: 8 data: 88e199b02da83367
method: authenticateD40
 copying cipheredBlock to ciphertext from i = 0 length 8
method: decrypt
 ciphertext length: 16 data: 88e199b02da833670000000000000000
method: authenticateD40
outer loop i: 8
method: authenticateD40
The inner loop is running for j=0 to <8 in steps of 1
method: authenticateD40
TripleDES.decrypt cipheredBlock length: 8 data: 557208d962ae4b4f
method: authenticateD40
 copying cipheredBlock to ciphertext from i = 8 length 8
method: decrypt
 ciphertext length: 16 data: 88e199b02da83367557208d962ae4b4f
method: authenticateD40
step 10 encryption magic ending    ********************
method: authenticateD40
manual decryption: SUCCESS
method: authenticateD40
- encrypted rndA || rndB left rotated length: 16 data: 88e199b02da83367557208d962ae4b4f
```

```
method: authenticateD40
step 11 send the encrypted data to the PICC using the 0xAF command (more data)
method: authenticateD40
- send auth apdu   apdu     length: 22 data: 90af00001088e199b02da83367557208d962ae4b4f00
method: authenticateD40
- receive response response length: 10 data: 6ccc27d21352c5ee9100
method: authenticateD40
step 12 the response data is the encrypted rndA from the PICC
method: authenticateD40
        Note: the received (encrypted) rndA is left rotated
method: getData
responseAPDU length: 10 data: 6ccc27d21352c5ee9100
method: getData
responseData length: 8 data: 6ccc27d21352c5ee
method: authenticateD40
- encrypted rndA left rotated length: 8 data: 6ccc27d21352c5ee
method: authenticateD40
encryptedRndA length: 8 data: 6ccc27d21352c5ee
method: authenticateD40
The iv is set to 8 * 0x00
method: authenticateD40
iv0 length: 8 data: 0000000000000000
method: authenticateD40
step 13 decrypt the encrypted rndA left rotated using TripeDES.decrypt with key key length: 8 data : d00022446688aacc iv0 length: 8 data: 0000000000000000
method: authenticateD40
- encrypted left rotated rndA length: 8 data: eb0533b4bc89afcf
method: authenticateD40
- decrypted left rotated rndA length: 8 data: cc39928713e1c045
method: authenticateD40
step 14 rotate decrypted left rotated rndA to
```

```
RIGHT
method: authenticateD40
- decrypted rndA length: 8 data: 45cc39928713e1c0
method: authenticateD40
step 15 compare self generated rndA with rndA received from PICC
method: authenticateD40
- rndA generated length: 8 data: 45cc39928713e1c0
method: authenticateD40
- rndA received  length: 8 data: 45cc39928713e1c0
method: authenticateD40
- rndA generated and received are equals: true
method: authenticateD40
step 16 generate the DES Session key from rndA and rndB
method: authenticateD40
- rndA           length: 8 data: 45cc39928713e1c0
method: authenticateD40
- rndB           length: 8 data: c4a3468fb0d87e74
method: getSessionKey
rndA length: 8 data: 45cc39928713e1c0 rndB length: 8 data: c4a3468fb0d87e74
method: authenticateD40
- This are the first 4 bytes of rndA and rndB, the DES Session key is
method: authenticateD40
- rndA first 4 bytes || rndB first 4 bytes
method: authenticateD40
- rndA first 4 Bytes    45CC3992
method: authenticateD40
- rndB first 4 Bytes            C4A3468F
method: authenticateD40
- SessionKey is 8 Bytes 45CC3992C4A3468F (length: 8)
method: authenticateD40
**** auth result ****
method: authenticateD40
*** AUTHENTICATED ***
method: authenticateD40
********************

```