Summary of CPU card issuance

转载 fayeyiwang Posted on 2016-11-28 15:48:35 Reading: 7.8k Collection 14 Likes 3
Category Column: TSM

Overview: The CPU card contains a microprocessor that functions like a microcomputer. The integrated circuits in the CPU card include the central processing unit (CPU), read-only memory (ROM), random access memory (RAM), electrically erasable programmable read-only memory (EEPROM), etc.

FMCOS consists of four functional modules: transmission management, file management, security system, and command interpretation.

Transmission management: supervise the communication between the card and the terminal to ensure the correct transmission of data.

File management: Different from other cards that store data in chunks. The CPU card stores user data in the form of files in EEPROM, guaranteeing fast and secure access to files.

Security system: It involves the identification and verification of cards, and the permission control mechanism when accessing files.

Command explanation: Check whether the parameters are correct according to the received command, and perform corresponding operations.

The above conceptual things are all found in the document, so I only talked about the general situation, and the following is a detailed description of the specific process of CPU card issuance.

Security Mechanisms:

Each directory of FMCOS has a security status register, with a total of 16 security statuses, represented by the hexadecimal number 0-F. The security status of the current directory is set to 0 after the directory is reset or re-selected, and the value of the security state is changed to the subsequent state specified by the key after the password verification or external authentication command in the current directory is passed.

For all files in FMCOS, you need to set the access permissions when you create them, including read permissions, write permissions, use permissions, and change permissions. It is expressed as a hexadecimal XY number of 1 byte. When the security status of the directory is >=Y and <=X, you have the corresponding permissions for the file. If X=0, you have the corresponding permissions for the file when the security status >=Y. When X=Y, security status=Y will have the corresponding permissions for the file. If you don't want people to ask about a file, you can set the file permission to XY(Y>X)

Issuing:

Equipment Required:

1. Fudan CPU card

2. D8 contactless card reader

3. ESAM card (the issuer of the card, which stores all the keys that need to be loaded into the CPU card)

4. PC

5. VB2005 is used as the development language

Card issuance process:

1. Connect the card reader and open the card reader port.

2. Power on and reset the CPU card and ESAM card, and check whether the CPU card and ESAM card are placed in the correct position of the card reader.

3. The first thing to remember is that you need to select the corresponding directory whether you do any operation on the CPU card or ESAM card.

4. External authentication of CPU card:

There is a master key in the key file in the MF directory of the CPU card, and the attribute settings of the key determine whether you have the permission to initialize the card of the CPU card, erase the MF, and modify the card structure.

The so-called external authentication is to check whether you have the legal permission to perform various operations on the CPU card. First of all, you need to have the master key, usually you don't know the content of the master key (unless it is an uninitialized card, the initial master key is determined by the card manufacturer, the initial key of the card I use is all F), the master key is stored in the ESAM card, you need to get the master key CCk through the decentralized instructions provided by the SAM card (two common DES calculations). Then, through the random number instruction provided by the CPU card, the 8-byte random number sRandom (this random number is not only returned to the terminal, but also stored in a certain position in the CPU card), and the sRandom is encrypted with CCK to obtain the result sRet. Then use the external authentication instruction to send sRet to the CPU card, the CPU card uses the master key under MF to decrypt the sRet, and compares the obtained result with the random number just sent to the terminal, if it is the same, the external authentication is passed, and the security register status of the CPU card is set to the subsequent state of the master key setting, and you also have the permission to operate the corresponding operation value of the subsequent state value. If the authentication fails, the number of errors is reduced by one, and you can use another master key to continue the external authentication, and the number of errors will be cleared to 0. Remember that the number of errors cannot exceed the maximum number of errors allowed by the master key setting (3 times by default for empty cards), otherwise the card will be locked.

Because we don't know whether the CPU card to be used to issue the card is empty or has been initialized, we need to use the initial key provided by the vendor and then use the master key in the ESAM card when doing external authentication.

5. Erase MF:

MF is created by default, and you can only clear all the content under MF, not delete MF.

6. Create the file under MF:

MF's directory structure includes:

- Key file

- Catalog data files

- Basic EF files (including binary files, log files, etc.)

- Catalog file DF

The key file is the first thing you need to create when you create a directory, and the various keys in the key file control your operation permissions on the directory and the content under the directory.

INSTRUCTION: 80E00000073F005001AAFFFF

File ID: 0000

File Type: 3F

File space: 0050 (about file space setting, participating: file space)

Read permission: 01

Write permissions: AA

After the key file is created, you need to write various keys to it, including master keys, maintenance keys, etc.

Then you need to create a directory data file and make it a longer record file to describe what directories are under MF.

Finally, create EF and DF

7. Create a file in the 3F01 directory

Generally, the key file is created and the key is loaded to the key file

Since 3F01 is an ADF, there is no need to create a directory data file

Then create an e-wallet/passbook file, an e-wallet transaction history record file, etc

The e-wallet/passbook file and the e-wallet transaction details file are created and maintained by the PBOC as specified.

8. The method of creating other directories is the same as above

File space

File header: 11 bytes (file type + file identifier + file body space size + permission + verification, etc.)

EEPROM space occupied by each basic file: 11 bytes of file header + file body space

The main space of fixed-length, normal wallets, and revolving files: number of records x length of records (the record length of the key file is data length + 8)

EEPROM space occupied by each DF: 11 bytes of DF header + space for all files under DF and +DF name length

Secure message transmission

Three scenarios:

- Line protection: A 4-byte MAC code is attached to the transmitted data, which is first verified by the receiver after receiving it, and only the correct data is accepted, so as to prevent tampering with the transmitted data.
- Line encryption: DES encrypts the transmitted data, so that the transmitted data is ciphertext, and even if the attacker obtains the data, it is meaningless.
- Line encryption protection: DES encrypts the transmitted data and appends a 4-byte MAC address code

How to use Secure Message Delivery:

When creating a file, change the file type, the highest position is 1 for line protection, the next highest position is 1 for line encryption, and the highest and second highest positions are set to 1 for line encryption protection.

When reading and writing a file or using a key, if you want to use secure message transmission, you must set the last half byte of the CLA to hexadecimal "4"

Line Protection MAC Calculation:

Let's say you create a binary file, use line protection, file identity 0x16, file space 0x20.

The instructions to write data to the binary file are:

04D6960024 + val + mac

The CLA is 04, and the last nibble is 4 because the file uses line protection

INS is D6

P1 is 96, where the upper 3 bits are 100 and the lower 5 bits are file identifiers

P2 is 00, which is the offset of the file to be written

LC is 24

val is the data to be written to the binary file, 0x20 bytes

The MAC is the line protection MAC that we need to compute

The mac is calculated according to the write command to remove the mac part, that is, it is calculated according to "04D6960024 + val".

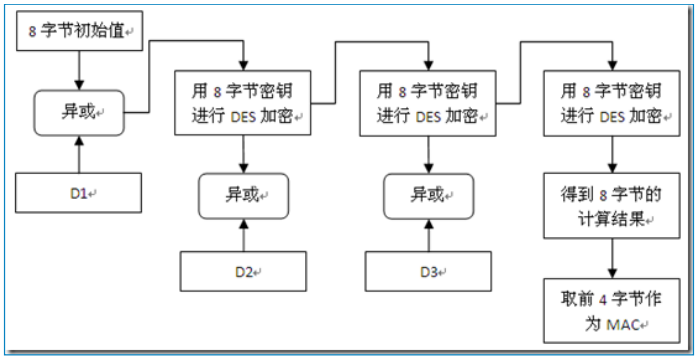
Mac Calculation Steps:

1. The terminal sends the GET CHALLENGE command to the CPU card, obtains a 4-byte random number, and then fills up the "0x00000000", and the obtained 8-byte result is used as the initial value of MAC calculation.

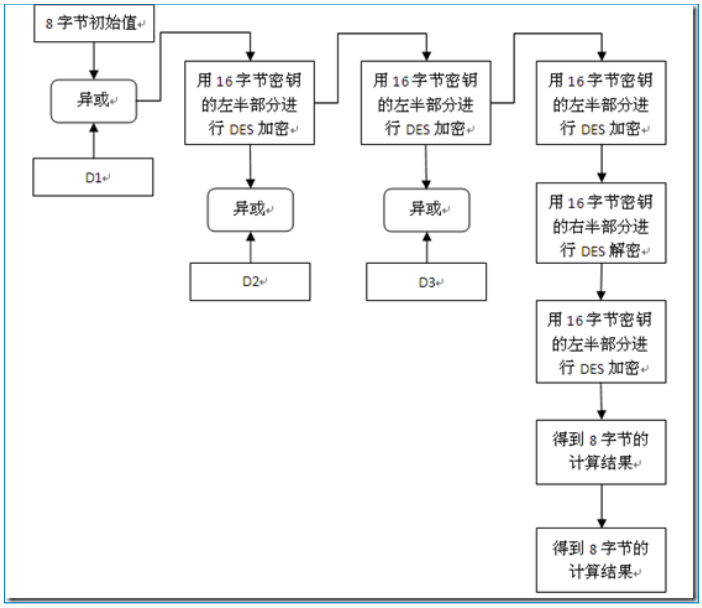
2. Check whether the number of bytes of "04D6960024 + val" is a multiple of 8, if not, make up "0x8000...", if so, add "0x8000000000000000", and then divide the obtained result into D1, D2, D3...

3. Determine the length of the key.

If the key length is 8 bytes, the MAC is calculated as follows:



If the key length is 16 bytes, the MAC is calculated as follows:



Calculate MAC using SAM card:

1. The terminal sends the GET CHALLENGE command to the CPU card, obtains a 4-byte random number, and then fills up the "0x00000000", and the obtained 8-byte result is used as the initial value of MAC calculation.

2. Initialize the general DES

80 1A + P1 (Key Usage) + P2 (Key Version) + Lc + DATA (Dispersion Factor)

3. General DES calculation

80 FA + P1 + P2(00) + Lc + DATA (8-byte random number +) + "04D6960024 + val + 0x8000..."

The value of P1 is the key: 00001001, i.e., 05, represents the MAC address with the initial value

The returned data is a 4-byte MAC

Line Encryption Calculation:

The specific calculation method is the same as the DES/3DES **algorithm**, and the following describes how to use SAM card encryption.

1. Initialize general DES

80 1A + P1 (Key Usage) + P2 (Key Version) + Lc + DATA (Dispersion Factor)

2. General DES calculation

80 FA + P1 + P2(00) + LC + "plaintext"

Here, the value of P1 is the key: 00000000, that is, 00, which represents the encrypted calculation without subsequent speed

The returned data is the encrypted ciphertext

recharge

Recharge offline

1. Select the app directory you want to recharge.

2. Verify the password key.

3. Obtain the recharge key (here the decentralized instruction provided by the encryption machine is used, and the 00B0 key is dispersed with ATS as the dispersion factor to obtain the recharge key).

4. Initialization of the encumbrance transaction.

5. Generate the process key.

6. Compare MAC1 with the process key calculation MAC1 and MAC1 returned by the initialization transaction, and continue to execute if it is consistent.

7. Calculate MAC2 with the process key.

8. Use the memory command to send MAC2 to the CPU card to complete the recharge transaction.

Online top-up

Steps 1, 2, 3 and 4 are the same as offline recharge.

Send the data returned by the initialization of the lap transaction to the backend, and the backend calculates the process key and obtains MAC1. The MAC1 is compared by the background, and if it is consistent, the MAC2 is calculated by the background and returned to the foreground, which uses the circle memory instruction to send MAC2 to the CPU card to complete the recharge transaction.

consume

1. Select the app directory you want to consume.

2. Obtain the consumption key (here the decentralized instruction provided by the encryption machine is used, and the 00B1 key is dispersed with the city code and ATS as the dispersion factor to obtain the consumption key).

3. Initialization of consumer transactions

4. Generate the process key

5. Calculate MAC1 with the process key

6. Send MAC1 to the CPU card with the consumption command for consumption transaction

7. Get TAC and MAC2, verify...

Compound consumption is mainly oriented to the application of the transportation field, which can not only meet the needs of the highway non-stop toll collection system, but also meet the application needs of urban public transportation.

Compared with ordinary consumption, compound consumption has one more update to the composite record file. For example, at the entrance of the expressway or when passengers get on the bus, first do a compound consumption with an amount of 0, so as to correctly record the relevant information of the entrance or boarding station, and then at the exit of the expressway or when the passenger gets off the bus, read the information recorded before, calculate the actual cost, and do a complete compound consumption transaction, so that the segmented toll function in the expressway or public transportation can be realized.

Specific consumption process:

1. Select the app you want to recharge

2. Get the consumption sub-key from the PSAM card or encryption machine

3. Initialize the compound transaction

4. Calculate the process key

5. Use the process key to calculate MAC1, where the transaction type is 09

6. Update the composite application data cache (P1 is the composite consumption flag of the original record of the composite record file in the card)

7. Send MAC1 for consumption

ps: byte7 in the command packet data field of the composite record file is EF (use the line to protect the read, and use the key with the identifier 00 for both reads and writes).