CPU card explained

转载 logaa Posted on 2012-05-16 11:12:09 Reading: 4.1k Collection 21 Likes 2
Article Tags: encrypt algorithm terminal storage decrypt authentication

Part 1 CPU Basics

1. Why use CPU cards?

IC cards can be divided into contact IC cards, contactless IC cards and composite cards in terms of interface mode. In terms of device technology, it can be divided into unencrypted memory cards, encrypted memory cards and CPU cards. The non-encrypted card has no security, and the data in the card can be arbitrarily rewritten, and the encrypted memory card adds a logical encryption circuit on the basis of the ordinary memory card to become an encrypted memory card. Because the logic encryption memory card uses password control logic to control the access and rewrite of EEPROM, it needs to verify the password before it can be written, so it is safe for the chip itself, but it is not safe in the application. It has the following unsafe factors:

1. The password is transmitted in plaintext on the line and is easy to be intercepted;

2. For system vendors, passwords and encryption algorithms are transparent.

3. The logical encryption card cannot authenticate whether the application is legitimate. For example, if someone fakes an ATM and you have no way of knowing its legitimacy, when you insert a credit card and enter your PIN, the password of the credit card is intercepted. Another example is INTENET online shopping, if the card is encrypted logically, the shopper is also unable to determine the legitimacy of the online store.

It is precisely because of the insecurity factors in the use of logical encryption cards that the development of CPU cards has been promoted. The CPU card can achieve the legitimacy authentication of the person, the card, and the system.

2. Three types of authentication for CPU cards

CPU cards have three authentication methods:

Cardholder legitimacy authentication - PIN verification

Card legitimacy certification – internal certification

System legitimacy certification – external certification

Cardholder Legitimacy Certification:

The process of verification by entering a personal order by the cardholder.

System legitimacy certification (external certification) process:

system card,

Send a random number X

Encrypt random numbers [with specified algorithm and key].

[Use the specified algorithm and key] to decrypt Y and get the result Z

Compare X, Z, if the same means that the system is legitimate;

Card legitimacy verification (internal authentication) process:

System card

Send a random number X

Encrypt random numbers with the specified algorithm and key].

[Use the specified algorithm and key] to decrypt Y and get the result Z

Compare X, Z, if the same means that the card is legitimate;

In the above authentication process, the key does not appear in plaintext on the line, it is encrypted by random numbers every time it is sent, and because there is a random number to participate, it is ensured that the content of each transmission is different. If intercepted, it doesn't make any sense. This is not only a password-to-password authentication, but also a method authentication method, just like the cipher telegram used in the early days in the army, the sender encrypts the message into ciphertext according to a certain method and sends it, and then the receiver receives it and decrypts the ciphertext according to a certain method.

Through this authentication method, there is no attack point on the line, and the card can also verify the legitimacy of the application.

However, because the key and algorithm used by the system side for authentication are in the application, it still cannot remove the aggressiveness of the system vendor.

Here, we introduce the concept of SAM card.

SAM card is a CPU card with special performance , which is used to store keys and encryption algorithms, and can complete mutual authentication, password verification, encryption and decryption operations in transactions, and is generally used as an identity mark.

Thanks to the advent of SAM cards, we have a more complete system solution.

When issuing a card, we store the master key into the SAM card, and then the master key in the SAM card encrypts the characteristic byte of the user card (such as the application serial number) to generate a sub-key, and injects the sub-key into the user card. Due to the uniqueness of the application serial number, the sub-key within each user card is different.

Once the key is injected into the card, it does not appear outside the card. When in use, the master key of the SAM card generates a sub-key and stores it in the RAM area to encrypt and decrypt data.

The above certification process takes the following form:

System legitimacy certification (external certification) process:

SAM Card System card

Send a random number X

The SAM card generates a sub-key to encrypt the random number

Decrypt Y to get result Z
Compare X, Z, if the same means that the system is legitimate;

Card legitimacy verification (internal authentication) process:

SAM Card System card

Send a random number X

Encrypt random numbers with the specified algorithm and key].

SAM card decrypts Y, and results Z

Compare X, Z, if the same means that the card is legitimate;

In this way, the key in the application is transferred to the SAM card, and the authentication becomes the card-card authentication, and the system vendor no longer has responsibility.

3. Line protection
When the card transmits data to the outside world, if it is transmitted in clear text, the data can be easily loaded and analyzed. At the same time, it is also possible to tamper with the transmitted data, and to solve this problem, the CPU card provides a line protection function.
There are two types of line protection, one is to encrypt the transmitted data in DES and transmit it in ciphertext form to prevent interception and analysis. The second is to attach a MAC address (security message authentication code) to the transmitted data, which the receiver will first verify after receiving it, and receive it only after the verification is correct, so as to ensure the authenticity and integrity of the data.

Fourth, the hardware structure diagram

EEPROM is used to store user data; The ROM is used to store the COS operating system, and the RAM area is used to store the intermediate variables of the COS runtime.
COS (chip operation system) is a chip operating system, similar to DOS and WINDWOS, and a CPU card without COS cannot be used like a PC without DOS and WINDOWS .
COS is solidified into the ROM area by the chip supplier when the chip leaves the factory, and this process is called masking.
COS is the core part of the CPU card, and it implements the security of the CPU card together with the hardware.

Part 2 Introduction to SmartCOS
SmartCOS is a chip operating system independently developed by Minghua Company, which passed the People's Bank of China certification in June 1999.
COS is divided into four parts:
1. The file system of SMARTCOS
The CPU card is managed in file mode, and SmartCOS supports the following file systems:
1. Files can be divided into MF files, DF files, and EF files
MF: Master file, which is the root of the entire file system and is unique, equivalent to the root directory.
DF: A dedicated file, equivalent to a subdirectory, which can be used to store all the files of an application. A DF can be an application, or multiple DFs can be used for the same application.
EF: A basic file that stores a variety of application data and management information.

2. EF is divided into two types in terms of storage content:
Security basic file: used to store the key, each directory can only create a security basic file, the key file can not be selected by file selection, the key content can not be read, but can be used and modified when the conditions are met.
Working basic file: used to store the actual data of the application, the number and size are limited by space. Reads and writes when conditions are met.

3. Basic file structure
The structure of a basic document can be divided into the following four types:

Binary file:
Data is read and written in bytes, and the length of each read and write cannot exceed 110 bytes. Can be used to store out-of-order data.

Linear Fixed-Length Record Files:
Each record is of fixed length, and records can be accessed by record number, and the record range does not exceed 254;
The length of each record does not exceed 110 bytes, and the key file is a linear fixed-length record file, and the length of each record is fixed at 25 bytes. It can be used to store regular, fixed-length data.
Linear Variable Length Record File:
The length of each record can vary, but the maximum length cannot exceed 110

bytes, which can be accessed by the record number.

Structure of the cyclic fixed-length record file:
It is equivalent to a ring record queue, which is stored according to the first-in, first-out principle, the latest record number is 1, the last written record number is 2, and so on, the earliest record is automatically overwritten after the record is full.

4. File structure diagram
EF and DF can be created under MF;
DF can no longer be created under DF, only EF can be created;

KEY file: used to control the creation, reading, and writing of files in MF
Data files
Key file:
It is used to control the creation and access of files under DF
Data files (e.g. wallet files, etc.)

5. Calculation of file space
MF header file length is 10 bytes + file name length (5-16 bytes)
The length of the DF header file is 10 bytes + the length of the file name
EF file footprint:
Define the space of the record and loop record files = file header space (10 bytes) + number of records * record length
The space of the variable-length record structure file = the header space (10 bytes) + the space requested when it is created
Space occupied by the key file = Header space (10 bytes) + Number of keys × 25 bytes
Wallet file space = file header (10 bytes) + file body (17 bytes)
Space of passbook file = File header (10 bytes) + File body (20 bytes)
The file system is established, so how to ensure the security of the file, the following describes the security system.

Second, the security system of SMARTCOS
1. The state machine is the safety state:
Refers to a security level that the card is currently in, with (0---F) 16 security states. Automatically set to 0 after reset, and the security status of the current application will be automatically cleared to 0 after being successfully selected or reset. A change in security status must be achieved through key authentication.
**Only PIN reconciliation and external authentication in the current directory can change the security status.

2. Security attributes are access rights
Access permissions are specified when the file is created.
It is a compartmental concept, for example, if the read permission of a file is XY, it means that the state machine (security state) M of the current application must meet the requirements of X=〈M=〈Y. If the read permission is set to 2F, the current state machine (security state) reaches 2 or higher, and the file can be read.

3. The relationship between the key and the security state
Each key defines a subsequent state when it is established, that is, the security state that can be achieved after passing the key authentication, and among various keys, only PIN authentication and external authentication can change the security status of the current directory.

4. The relationship between security status and access rights
(Security Status)
Usage permissions-------- key usage ---------- subsequent status---------- file read and write retrieval limits

5. The security status is only valid in the current directory, once you select another application, the state machine will automatically jump to the minimum permission 0.
Security is independent between directories.

Fourth, reset response

Symbolic Byte Content Content Explained
TS3B Forward Convention
T06C TB1 and TC1 exist, and the historical character is 12
The TB100 requires no additional programming voltage
The TC102 requires 2 additional protection periods
T1-TC XX historical characters
Specific meanings of SMARTCOS historical characters:
Symbolic Byte Content Content Explained
The version number of the T1XX SMARTCOS
T2XX card status bytes
T386 Minghua IC card manufacturing organization identification number
T438
T5-TC XX card unique serial number
The card status bytes are described as follows:

B7B6 B5 B4 B3 B2 B1 B0 Status
011XXX XXX 001 XXX XXX XXX XXX The card is initialized and successfully, the card is not initialized, and the card is locked during the initialization process
0 00 0 0 0 X The card is not personalized
0 01 0 X X X X The personalization of the card is not finished
0 11 0 X X X X The card is personalized
0 00 1 x x x x x The card personalization was not successful and the card was locked
0 11 1 X X X X X The card is personalized successfully and the card is locked
。

There are concepts about initialization and personalization in the card state byte, and we look at these two concepts from the life cycle of the card, which generally includes the following parts:

Chip chip producer

Mask COS chip manufacturer

Encapsulated into a card card manufacturer

Initialization of the card (COS enabled) by the card producer
Transmission password protection
Personalization of the card card issuer

Use of the card The consumer of the card

Recycling of cards by card issuers

The initialization process is to activate COS and define the COS version, after which COS can be used.
The personalization process refers to the creation of files, the writing of user data, etc.
The card manufacturer is protected by a transmission password when it is initialized and delivered to the card issuer.

The IC card must support either T=0 or T=1 protocols, but not both, while the terminal must support both T=0 and T=1 protocols.
T=0 communication protocol is an asynchronous half-duplex character transmission protocol;
T=1 communication protocol is an asynchronous half-duplex block transmission protocol;
In the ISO7816-3 standard, these two protocols are specified;
The protocol used by the IC card is specified in TD1, and if TD1 is not in the reset response message, it means that the protocol with T=0 is used for communication. After the reset answer, the IC card and the terminal communicate with the protocol specified by the IC card.

5. Instruction analysis
Here, we take an e-wallet application as an example to explain the instructions of SmartCOS.
1. File structure:

The following file structure is defined in the pedestrian specification:
1) The basic data file of the public application of the e-passbook ED/e-wallet EP application
File structure:
File Identifier (SFI) '21' (decimal)
File Type: Transparent
File size 30
File Access Control Read = Free Rewrite = Security Information Required
Byte data object length
1-8 Issuer ID 8
9Application Type Identification 1
10 App Version 1
11-20 Apply serial number 10
21-24 App Enabled Date 4
25-28 Apply effective date 4
29-30 Issuer-defined FCI data 2

2) Cardholder basic data file of e-passbook ED/e-wallet EP application

File Identifier (SFI) '22' (decimal)
File Type: Transparent
File size 39
File Access Control Read = Free Rewrite = Security Information Required
Byte data object length

1 Card Type Identification 1
2 Employee Identification of the Bank 1
3-22 Cardholder Name 20
23-38 Cardholder ID number 16
39 Cardholder ID Type 1

3) Electronic passbook ED transaction details file

File Identifier (SFI) '24' (decimal)
File Type: Loop
File Access Control Read = PIN Protection
Rewrite = Not allowed
Record size 23
Byte data object length
1-2ED or EP online or offline transaction serial number 2
3-5 Overdraft Limit 3
6-9 Transaction amount 4
10Transaction Type Identifier 1
11-16 Terminal No. 6
17-20 Trading Date (Terminal) 4
21-23 Trading hours (terminal) 3

2. The safety design is as follows:
1) External authentication can be carried out after checking the password;
2) The No. 01 external authentication key is used to control the storage of the e-wallet;
3) The No. 02 external authentication key is used to control the modification of the basic file and the modification of the key;
4) You can make a purchase after checking the password.

The following keys are installed in the KEY file:
KEY TYPE IDENTIFIES THE KID STATUS OF THE PERMISSION TO USE THE KEY AND DESCRIBES THE FUNCTION OF THE KEY
0B01 0F 1 PIN for PIN verification
0801 11 2 External authentication key for e-wallet storage
0802 1F F external authentication key, which is used to modify basic files and keys
0101 22 No captive key, used to generate captive MAC
0002 01 No wallet consumption key, which is used to generate wallet consumption MAC
0201 03 No TAC key, which is used to generate TAC for deposit, consumption, cash withdrawal, and modification of overdraft limit
0501 33 No App Maintenance Key for MAC to generate App Lock, App Unlock, Card Lock, Card Lock, and Read, Update Binary, Log commands

3. Instruction sequence:
1) During the card issuance process, create a file and install a key on the card
[Create MF] APDU Command: 80 (CLA) e0 (INS) 00 (P1) 00 (P2) 18 (Lc) ff ff ff (8 byte transfer code) ff (Establish security attributes of files under MF) 01 (short file identifier for directory files) 31 50 4159 2e 53 59 53 2e 44 44 46 30 31 (name of the file created)

[Create DF] APDU Command: 80 (CLA) E0 (INS) 01 (P1) 00 (P2) 0D (LC Info Length) 2F01 (File Identifier) ff (Establish File Permissions) 00 (COS Reserved) A0 00 00 00 03 86 98 07 01 (ADF Name)

Create File in DF APDU Command: 80 (CLA) E0 (INS) 02 (P1) 00 (P2) 07 (File Information Length) 6F 02 (Key File Identifier) 05 (File Type) FF (Add Permission for New Key) 00 (COS Reserve) 09 (Number of Records) 19 (Record Length):

Install PIN [Write Key] APDU command: 80 (CLA) E8 (INS) 00 (P1) 00 (P2) 0A (key information length) 01 (key identifier) 01 (key version number) 00 (algorithm identification) 0B (key type) 0F (usage permission) 01 (subsequent status) 2F (modify permission) 33 (error counter) 12 34 (personal password)

Install an external authentication key (DEAK) [Write Key] APDU command: 80 (CLA), E8 (INS), 00 (P1), 00 (P2), 18 (key information length), 01 (key identifier), 01 (key version number), 00 (algorithm identifier), 08 (key type), 11 (usage rights), 02 (subsequent status), FF (modify permissions), 33 (error counters), XX XX XXXX (Key Contents)

Install an external authentication key (DEAK) [Write Key] APDU command: 80 (CLA), E8 (INS), 00 (P1), 00 (P2), 18 (key information length), 02 (key identifier), 01 (key version number), 00 (algorithm identifier), 08 (key type), 1F (usage permission), 0F (subsequent status), FF (modify permission), 33 (error counter), xx xx XXXX (Key Contents)

。 Install the consumption key DPK of the e-wallet EP. [Write Key] APDU command: 80 (CLA) E8 (INS) 00 (P1) 00 (P2) 18 (key information length) 02 (key identifier) 01 (key version number) 00 (algorithm identification) 00 (key type) 01 (use permission) 00 (subsequent status) FF (modify permission) 00 (error counter) XX XX XXXX (KEY CONTENT)

Install the Wallet's Trap Key DLK [Write Key] APDU Command: 80 (CLA) E8 (INS) 00 (P1) 00 (P2) 18 (Key Information Length) 01 (Key Identifier) 01 (Key Version Number) 00 (Algorithm Identifier) 01 (Key Type) 22 (Usage Permissions) 00 (Subsequent Status) FF (Modify Permissions) 00 (Error Counters) XX XX XXXX (Key Contents)

。 DTK [Write Key] APDU command: 80 (CLA), E8 (INS) 00 (P1), 00 (P2) 18 (key information length) 01 (key identifier) 01 (key version number) 00 (algorithm identifier) 07 (key type) 0F (usage permission) 00 (subsequent status) FF (modify permission) 00 (error counter) XX XX XXXX XX XX (Key Contents)

Install the application maintenance key DAMK[Write Key] APDU command: 80 (CLA), E8 (INS), 00 (P1), 00 (P2), 18 (key information length), 01 (key identifier), 01 (key version number), 00 (algorithm identifier), 05 (key type), 0F (usage permission), 00 (subsequent status), FF (modify permission), 00 (error counter), XX XX XX XXXX (Key Contents)

Create File for Public Application APDU Command: 80 (CLA) E0 (INS) 02 (P1) 00 (P2) 07 (LC Message Length) 00 15 (File Identifier) 00 (Binary File Type) 0F (Read Permission) FF (Update Permission) 00 1e (File Length)

Create File [Create File] APDU command: 80 (CLA) E0 (INS) 02 (P1) 00 (P2) 07 (file information length) 00 16 (file identifier) 00 (binary file type) 0F (read permission) FF (update permission) 00 27 (file length)

Create File APDU Command: 80 (CLA) E0 (INS) 02 (P1) 00 (P2) 07 (File Information Length) 00 18 (File Identifier) 03 (Circular Record File Type) 1F (Read Permission) 10 (Update Permission) 0a 17 (File Length)

Write the public application base data file [Update Binary] APDU command: 00 (CLA) D6 (INS) 95 (P1) 00 (P2) 1E (message length) A0 00 00 00 03 00 0001 (issuer ID) 03 (application type ID) 01 (application version) 00 00 19 98 08 15 00 00 00 01 (application serial number) 2000 10 01 (application activation date) 20 02 12 31 (application expiration date) 55 66 (issuer-defined FCI data)

Write card holder basic data file [Update Binary] APDU command: 00 (CLA) D6 (INS) 96 (P1) 00 (P2) 27 (file information length) 00 (card type identification) 00 (Bank employee identification) 53 41 4d 50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (cardholder name) 31 31 30 31 30 38 37 30 30 33 31 3731 38 39 00 (Cardholder ID Number) 00 (Cardholder ID Type)

Create Wallet EP File [Create File] APDU Command: 80 (CLA) E0 (INS) 02 (P1) 00 (P2) 07 (Lc) 0001 (File Identifier) 06 (File Type) 00 (Permission 1) 00 (Permission 2) 00 (LEN1) 00 (LEN2)

Create End (DF) APDU Command: 80 (CLA) E0 (INS) 01 (P1) 01 (P2) 02 (File Identifier Length) 2F 01 (File Identifier)

MF [Create End] APDU Command: 80 (CLA) E0 (INS) 00 (P1) 01 (P2) 02 (File Identifier Length) 3F 00 (File Identifier)

2) Transaction process
Here, we take the captive process and consumption process as an example:
A. Circle deposit - deposit the amount into the card
Before captivity, the personal population order and the external certification No. 01 must be proofread;

Select Application[Select File]APDU Command: 00 (CLA) A4 (INS) 00 (P1) 00 (P2) 02 (Length) 2F01 (File Identifier)

Verify PIN [Verify] APDU command: 00(CLA) 20(INS) 00(P1) 00(P2) 02(length) 1234(PIN)

External Certification: [
Take the random number [Get Challenge] APDU command: 00(CLA) 84(INS) 00(P1) 00(P2) 08

3DES encryption of random numbers with No. 01 external authentication key;

External Authentication APDU command: 00(CLA) 82(INS) 00(P1) 01
(P2) 08 (length) XX XX XX (ENCRYPTED RANDOM NUMBER)
]

Captivity:[
Initialize Trap [Initalize For Load] APDU command: 80 (CLA) 50 (INS) 00 (P1) 02 (P2) 0B (length) 01 (key identifier) 00 00 10 00 (transaction amount) 00 00 00 00 00 01 (final). Endpoint number)

If the initialization of the captive is successful, the content of the data field is answered:
Description length (bytes)
Old balance of e-passbook or e-wallet4
e-Passbook or e-Wallet online transaction serial number 2
Key version number 1
Algorithm ID 1
Pseudo-random number ICC 4
MAC1 4

3DES encryption is carried out with the content of the circled response (4-byte pseudo-random number ICC + 2-byte electronic passbook or e-wallet online transaction number +80 00) to generate the process key;

[Credit For Load] APDU Command: 80 (CLA) 52 (INS) 00 (P1) 00 (P2) 0B (LC)
YYYY MM DD (Trading Date)HH MM SS (Trading Hours)XX XX XX XX (MAC2)

MAC2 CALCULATIONS:
Initial value: 00 00 00 00 00 00 00 00
Key: The process key generated above
Data generated for MAC2: 00 00 10 00 (4-byte transaction amount) 02 (transaction type ID) 00 00 00 0001 (6-byte terminal number) YY YY MM DD (transaction date) HH MM SS (transaction time)

If the deposit transaction is successful, the online transaction number of the e-wallet file is added by 1, the transaction amount is added to the balance of the e-wallet, and a record is added to the transaction detail file.

B. Consumption process:
Verify the PIN before making a purchase:
Verify PIN [Verify] APDU command: 00(CLA) 20(INS) 00(P1) 00(P2) 02(length) 1234(PIN)

[Initialize For Purchase] APDU command: 80(CLA) 50(INS) 01(P1) 02(P2)0B (length) 01 (consumption key identifier) 00 00 00 01 (consumption amount) 00 00 00 00 00 01 (terminal number)

If the initialization consumption is successful, the data field content is answered:
Description length (bytes)
Old balance of e-passbook or e-wallet4
e-Passbook or e-Wallet online transaction serial number 2
Overdraft Limit 3
Key version number 1
Algorithm ID 1
Pseudo-random number ICC 4

Debit For Purchase APDU Command: 80 (CLA) 54 (INS) 01 (P1) 00 (P2) 0F (LC)
YYYY YY YY (Terminal Transaction Number) YY YY MM DD (Terminal Transaction Date) HH MM SS (Trading Hours) XX XX XXXX (MAC1)

Use the consumption/withdrawal key pair (4-byte pseudo-random number ICC + 2-byte electronic passbook or e-wallet online transaction serial number + the last 2 bytes of the terminal transaction serial number) to encrypt the process key;

MAC1 Calculation:
Initialization: 00 00 00 00 00 00 00 00 00 00 (8 bytes)
Key: The process key
The data generated by the MAC code: 4 bytes of transaction amount + 1 byte of transaction type identification + 6 bytes of terminal number + 4 bytes of terminal transaction date + 3 bytes of terminal transaction time

If the transaction is successful, add 1 to the online transaction number of the e-wallet file, minus the transaction amount from the e-wallet, and add a record to the transaction details file.

6. Analysis of other instructions
The following directives have many doubts in the process of use, which are explained as follows:
1. The ciphertext installation process of the key:
In version 3.2, the ciphertext installation of keys will be different, and the process is as follows:
1) Create a key file:
When you create a key file, the following table describes the file information:
LC about file information
07 File Identifier (2 bytes) File Type (1 byte) Permission 1 (1 byte) Permission 2 (1 byte) Len1 (1 byte) Len2 (1 byte)
Permission 1 indicates the permission to add a new key, and permission 2 indicates how to install the key. Permission 2 is set to 80h, which means that the installation is in ciphertext.
2) Use the application master key of the previous layer (SmartCOS 3.2 specifies: the external authentication key with the key identifier 01 is the application master key) to encrypt the key information (the key information is: plaintext key information length + key information + 80 00 0000 00 00 00), and the encryption mode of the key information is standard Triple DES or Single DES, if the key is 16 bytes, it is encrypted with TripleDES, if it is 8 bytes, It is encrypted with Single DES.
3) Generate MAC code, the initial value is: 4 bytes of random number + 00 00 00 00, the data generated MAC code is: 5 command headers + encrypted key information.
4) The ciphertext installation of other keys is encrypted with the application master key.
5) When installing the master key of the application under MF in ciphertext, use the transmission key of the card to install it.

2. Use security messages to write binary files
1) When creating a binary file, the fourth digit of the file type is 1, which indicates that a security packet is used.
2) LC is the number of bytes written + 4
3) Calculation of Security Messages (MAC):
The initial value is: 4 bytes of random number + '00 00 00 00'
The key is: the application maintenance key

Code value

CLA04
INSD6
P1Xx
P2Xx
LC writes data length +4
DATA Write data + 4 bytes MAC

8. Unlock PIN Unblock personal password
The command packet is encoded as follows:
Code value
CLA84
INS24
P100
P201 - Unlock PIN
Lc0C
DATA encrypts the PIN data element + Packet Authentication Code (MAC) data element, using the PIN unlock key.
Thereinto:
Encrypted PIN Token:
The PIN unlock key is encrypted and generated by PIN plaintext length + PIN + 80 00 (a multiple of 8).
MAC Generation:
Initial value: 4-byte random number + 00 00 00 00
Password: PIN unlock key
MAC address code generated data: CLA+INS+P1+P2+LC+encrypted PIN (8 bytes)

6. Use of CPUDEMO

7. Instructions for using the RD reader library
(1)int cpu_reset(int icdev,unsigned char *data_buffer);
Description: CPU card reset operation
Parameter: icdev: The device identifier returned by initialization
data_buffer: The string returned by the reset response
Returns: = 0 correct
< 0 error (see error code)

(2)int cpu_protocol(int icdev,int len, unsigned char *send_cmd,
unsigned char *receive_data)
Note: A command string is sent to the reader based on the transmission protocol of the CPU card T=1
Parameter: icdev: The device identifier returned by initialization
send_cmd: Command string for card operation (T=1 format)
len: The length of the command string
receive_data: The string of data returned by the CPU card
Returns: = 0 correct
< 0 error (see error code)

(3)int cpu_comres(int icdev,int len, unsigned char *send_cmd,
unsigned char *receive_data)

Notes:
1) Regardless of whether it is a T=0 or T=1 card, the RD reader transmits commands in T=1 format.
The send_cmd of the command strings sent includes:
NAD+PCB+LEN+COMMAND+BCC
For T=0 cards, NAD and PCB can be filled with '00';
The return value receive_data structure is the same as Send_cmd.

2) cpu_comres is different from cpu_protocol ().
a. When the upper-layer library receives the state byte SW1SW=61XX, it Cpu_protocol () function for further processing, and issues a response byte command (GetRespond) to the reader, and returns the data and the state byte together; The cpu_comres() function does nothing and returns the state byte SW1SW2 directly.
b. When the upper-layer function library receives the state byte SW1SW=6CXX, it Cpu_protocol the () function for further processing, assigns XX to le and resends the last instruction, and returns the data and the state byte together; The cpu_comres() function does nothing and returns the state byte SW1SW2 directly.