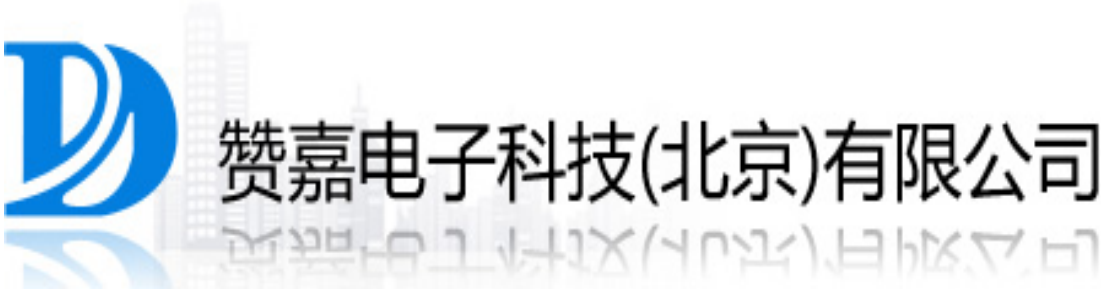

Zanjia Electronic Science & Technology (Beijing) Co., LTD

HSM-ZJ2014

Guidance Documentation

Level 3 Validation

November 2013

赞嘉电子科技(北京)有限公司

Table of Contents

1. Document Overview

This is a guidance document developed for HSM-ZJ2014 from Zanjia Electronic Science & Technology (Beijing) Co., LTD. It describes the detailed information of HSM-ZJ2014 and demonstrates how to install and use it after leaving factory. In this document, the HSM-ZJ2014 is also referred to as “the HSM”.

Note:

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

—Reorient or relocate the receiving antenna.

—Increase the separation between the equipment and receiver.

—Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

—Consult the dealer or an experienced radio/TV technician for help.

2. Product Introduction

2.1. Major Function

HSM-ZJ2014 provides cryptographic services, including encryption, decryption, signature generation and verification, and key management service with various hardware protection mechanisms for its security. The HSM also provides standard interfaces of cryptographic services including PKCS #11. In addition, a modular design makes it convenient to integrate HSM-ZJ2014 with existing information systems.

2.2. Packing List

Table 1 Packing List of HSM-ZJ2014

Component	Quantity
HSM-ZJ2014	1
Directly Connected Ethernet Cable(Grey)	1

Crossover Ethernet Cable(Blue)	1
Fiber Optic Cable	1
Manufacturer's Certificate	1
Packing List	1
Power Line	2
User Guide	1
Crypto Officer Smart Card	8
Product CD	1

Note: this list is only for reference, the packing list in the packing box shall prevail.

2. 3. Parameter Information

Table 2 Parameter Information of HSM-ZJ2014

Parameter	Value
1. Physical	
Specification	4U
Dimensions(Width× Height× Length)	400mm×177mm×490mm
2. Electrical	

Power Supply	220V, 50Hz
Power Dissipation	620W
Network Interface	RJ-45 10/100/1000Mb ×2 Fiber Optic×2
Network Protocol	TCP/IP
MTBF	>50000 hours
3. Environment	
Working Temperature	0℃-40℃
Working Humidity	25%-80%
Storage Temperature	-10℃-55℃

2.4. Approved Mode of Operation

The HSM-ZJ2014provides the following approved mode algorithms:

- AES (ECB and CBC mode, 128/192/256-bit, encryption and decryption)
- ECDSA (NIST p256, signature generation/verification)
- RSA (2048-bit, signature generation/verification)
- SHA-256
- HMAC (HMAC-SHA-256)

- CTR-DRBG based on 256-bit AES

2.5. Service DeliveryChannel

- Ethernet：10M/100M/1000M Self-Adapt, 10000M Fiber Optic
- Based on TCP/IP

2.6. Compatible Standard

- AES (ECB and CBC mode; 256 bit keys):NIST 197, SP 800-38A;
- RSA(2048bit): FIPS 186-4, PKCS#1 v2.1;
- ECDSA(NIST P-256): FIPS 186-4;
- SHA-256: FIPS 180-4, SP-800-107 Rev.1;
- HMAC(SHA-256):FIPS 198-1, SP-800-107 Rev.1;
- DRBG(CTR_DRBGAES256): SP 800-90A;
- Key Generation: SP 800-133.

2.7. Installation Condition

Before Installation，please make sure：

1. The equipment you receive are intact and correspond with Detailed List;
2. Power Switch is OFF;
3. Input voltage keeps within 220V±10%.

2. 8. Schematic Diagram

Front Panel:

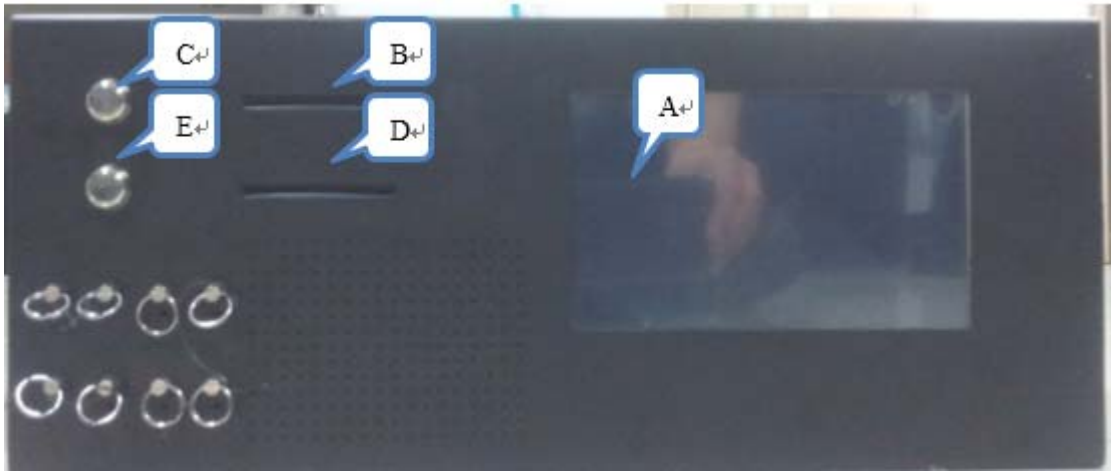

Figure 1 Front View of HSM-ZJ2014

Rear Panel:

Figure 2 Rear View of HSM-ZJ2014

A: Touch Screen

B: Smart Card Reader (ID)

C: Power Switch

D: Smart Card Reader (Key)

E: Reset Switch

N, O: Power Connectors

P: Service Ethernet Port (RJ45)

Q: Management Ethernet Port

R：Service Ethernet Port (Fiber Optic)

2.9. InstallationInstructions

1. Connect the power supply to the HSM.

Note:Before connectingthe power line to the HSM, please make sure power is off first.

2. Connectcable to the HSM. According to your requirement and application scenarios, connect your equipment to the HSM’s RJ45 or Fiber Optic Service Ethernet Port with corresponding cable.

3. Turn on Power Switch, wait until Touch Screen turns to UI below.

Note: If any exception occurs, please contact manufacturer.

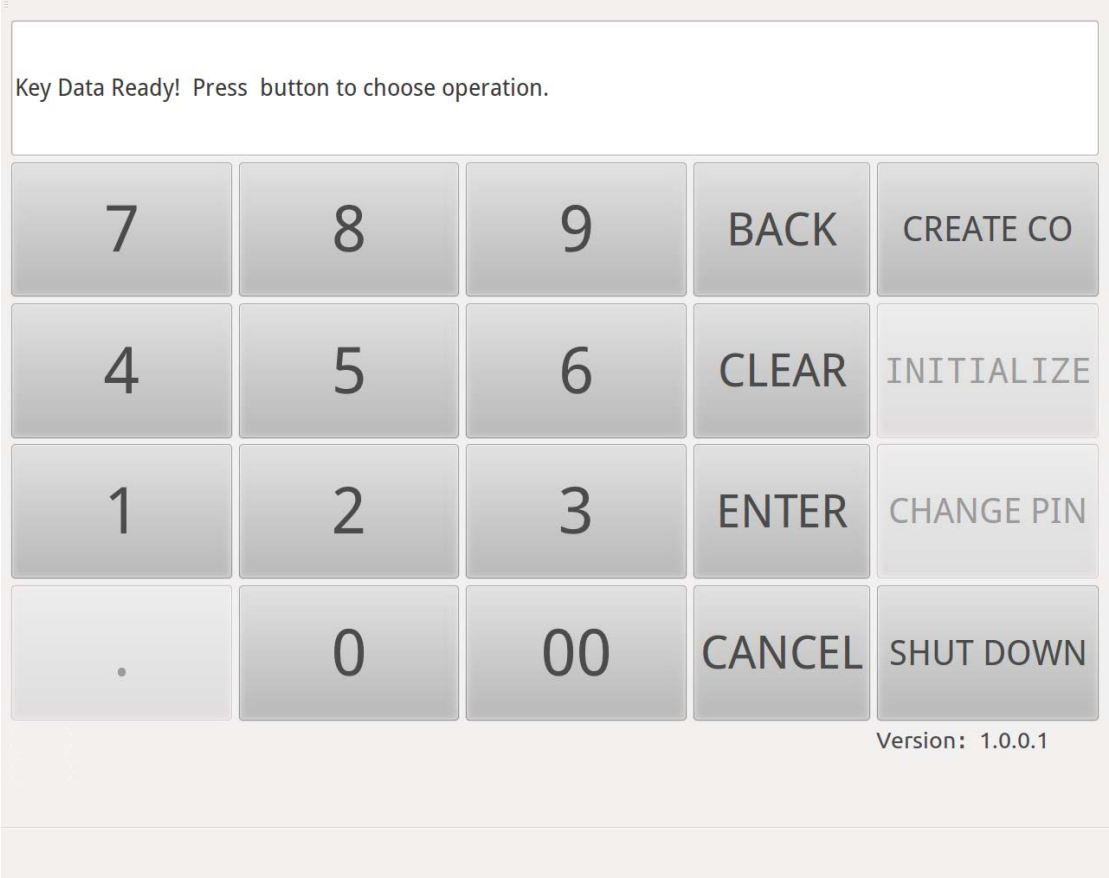

Figure 3 Touch Screen UI of HSM-ZJ2014

3. HSM Operation Guide

Crypto Officer Creation, HSM Initialization, Smart Card PIN Modification and HSM Power Off can be executed using Touch Screen. When firstly used after leaving factory, the HSM must create Crypto Officer and be initialized before using.

Note: the default 8-digit PINs of all smart cards are "12345678".

3. 1. Create Crypto Officer

HSM-ZJ2014 supports 4 types of Crypto Officer, which is shown in the following table.

Table 3 Role Description

Roles	Description
Crypto Officer	**Device Manager:** Executes module initialization, device management and key management functions. Functions are available via the Touch Screen or Remote Management Application. The functions related to accessing key can only executed after Authorizer's authorization.
	Auditor: Executes log audit functions. Functions are available via Remote Management Application. Auditor can view or export the log.
	Authorizer: Executes key authorization functions. Functions are available via Remote Management Application. Authorizer can authorize Device Manager's key management operation.
	Key Manager: Executes master key export and import functions. Functions are available via Touch Screen and Smart Card Reader.

Before coming into use after leaving factory, the HSM needs to generate all Crypto Officers and the corresponding smart cards.

Device Manager, Auditor, Authorizer will be generated at first. 5 Key Managers will not be generated until Master Key is generated, whichis shown in Section 4.2.1.

After Poweringon, HSM-ZJ2014'sTouch Screen will show as below.Press "Create CO" button to create Device Manager, Auditor, and Authorizer.

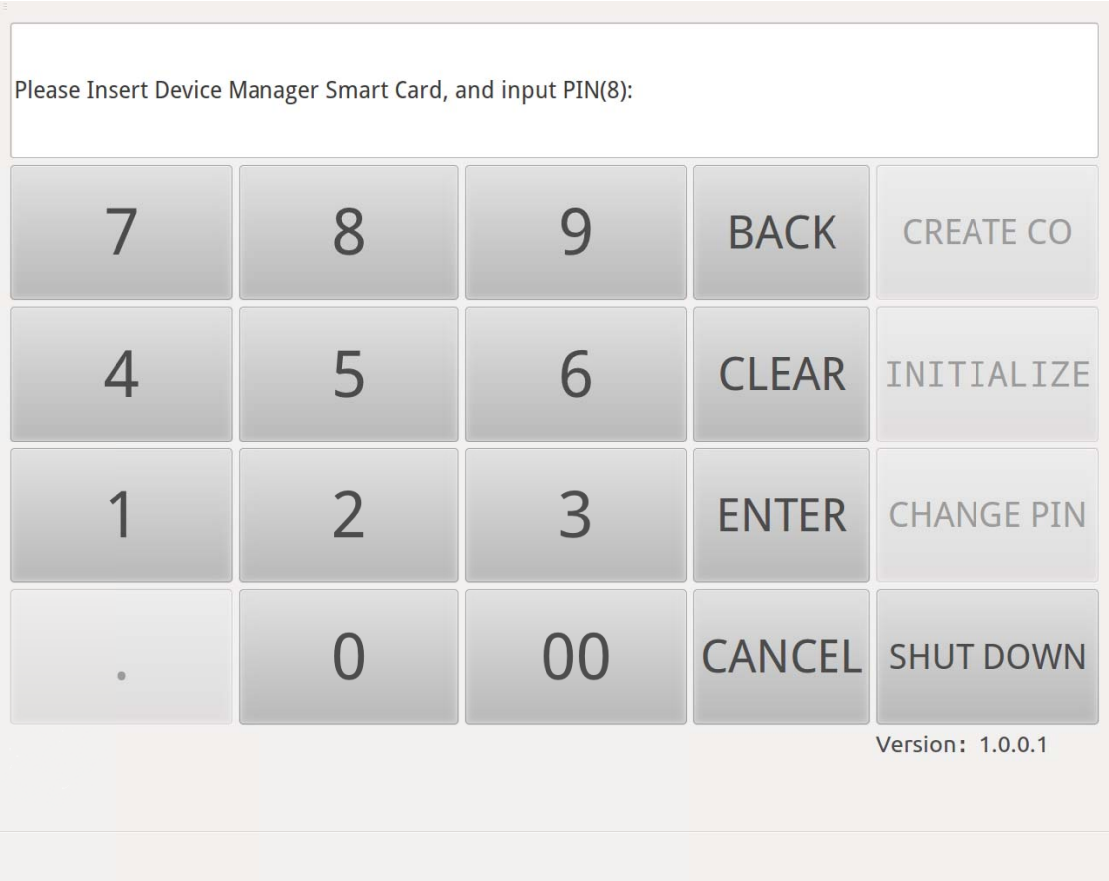

Figure 4 Crypto Officer Creation UI of HSM-ZJ2014

According to the message demonstrated on Touch Screen, insert Device Manager smart card into Smart Card Reader(ID) and input smart card's PIN, then press "ENTER" button.

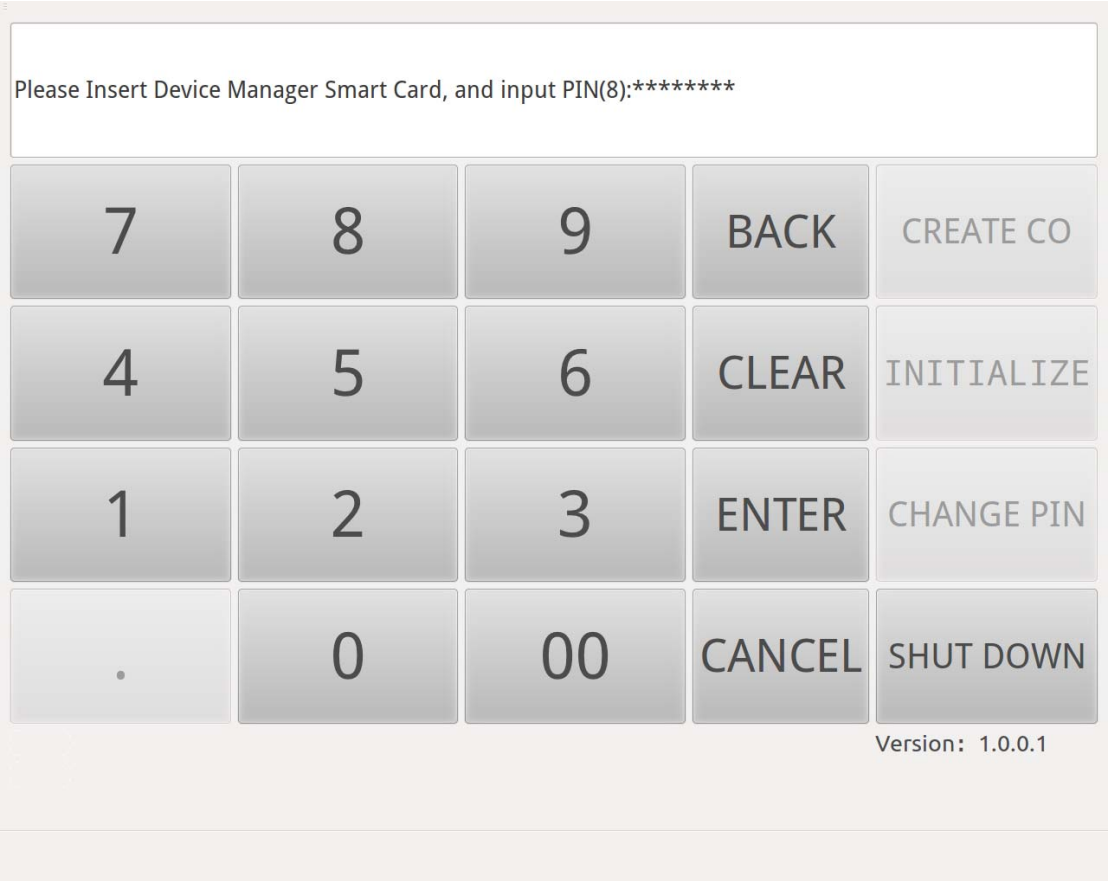

Figure 5 Device Manager Creation UI of HSM-ZJ2014

Then insert Auditor smart card into Smart Card Reader (ID) and input the smart card PIN, then press “ENTER” button.

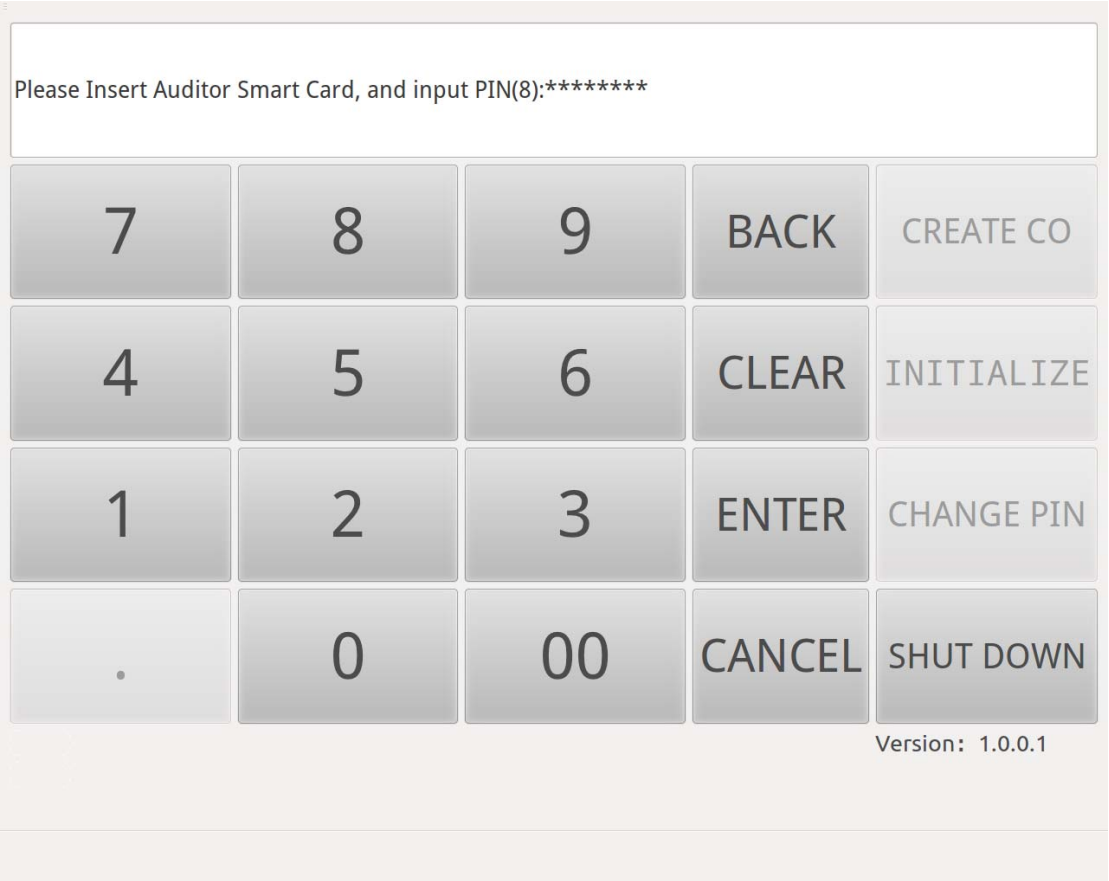

Figure 6 Auditor Creation UI of HSM-ZJ2014

Finally, insert Authorizer smart card into Smart Card Reader (ID) and input the smart card PIN, then press “ENTER” button.

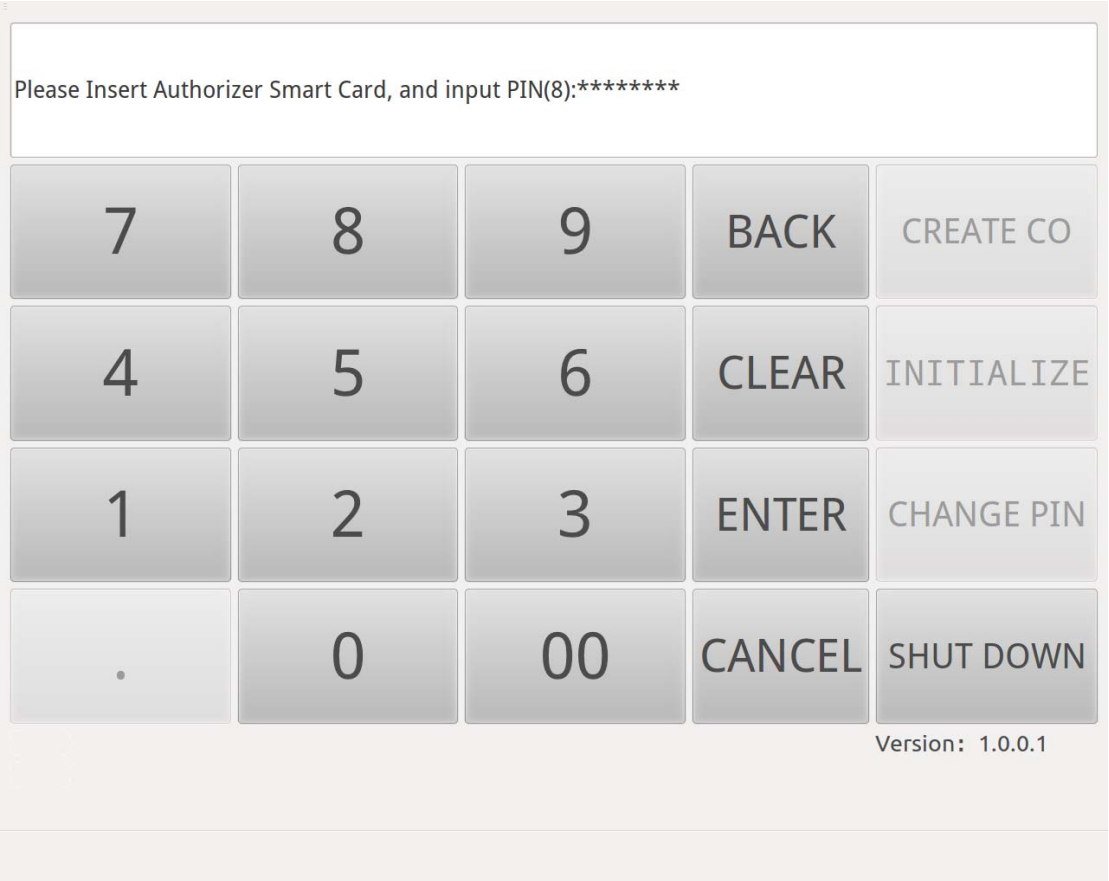

Figure 7 Authorizer Creation UI of HSM-ZJ2014

After Crypto Officers and the corresponding smart cards are generated, Touch Screen will show message “Crypto Officer Smart Card Generated!”

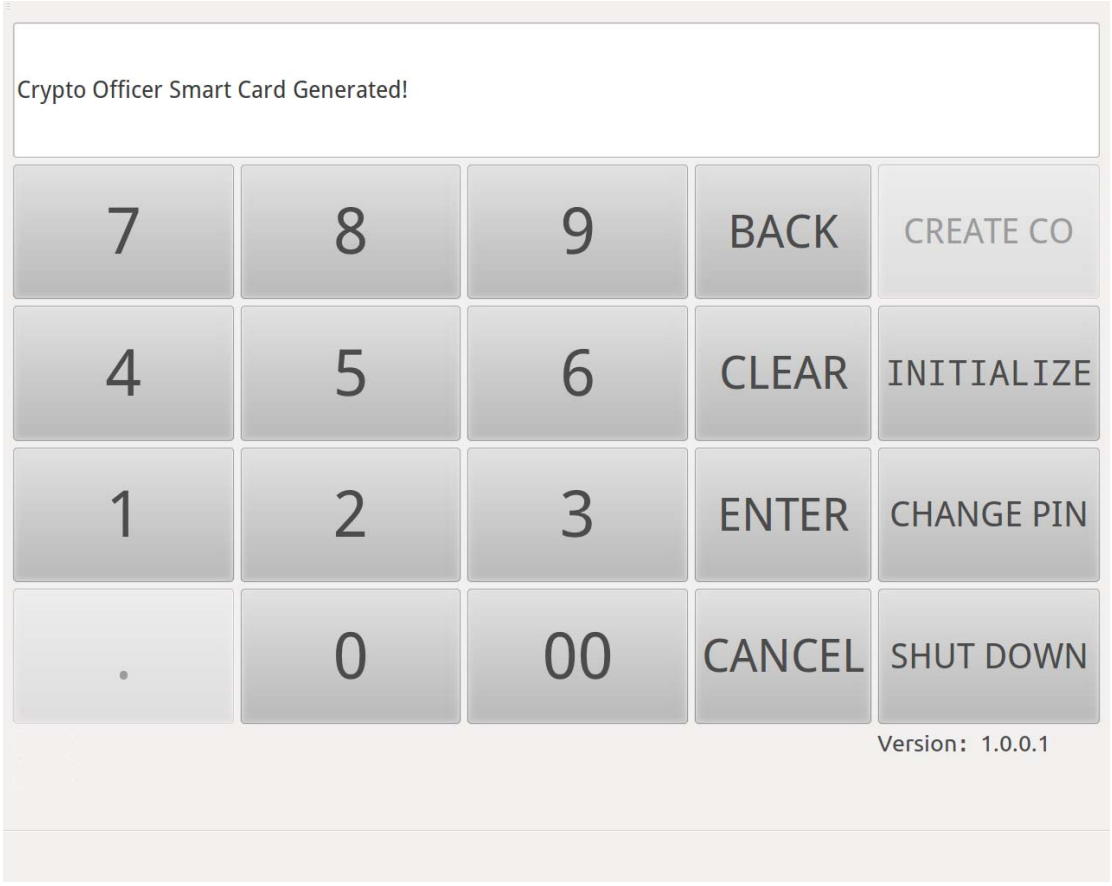

Figure 8 Crypto Officer Generated UI of HSM-ZJ2014

3. 2. Initialize

The HSM needs to be initialized in 2 conditions:

- First used after leaving factory(or restoring factory setting)
- Reboot after Master Key is generated

The two kinds of initialization have a little difference, which will expressin Section 4.2.1 and Section 4.2.2.

3.2.1. Master Key is NotGenerated

When first coming into use after leaving factory, the HSM's master key is

not generated yet. Therefore, Master Key shouldbe generated using the internal RNG and split into 5 shares, andthese 5 shares will be stored into 5 Key Manager smart cards.

Press “INITIALIZE”button, according to the message (Figure 9), firstly insert the Device Manager smart card into Smart Card Reader(ID) and input the corresponding PIN to authenticate Device Manager.

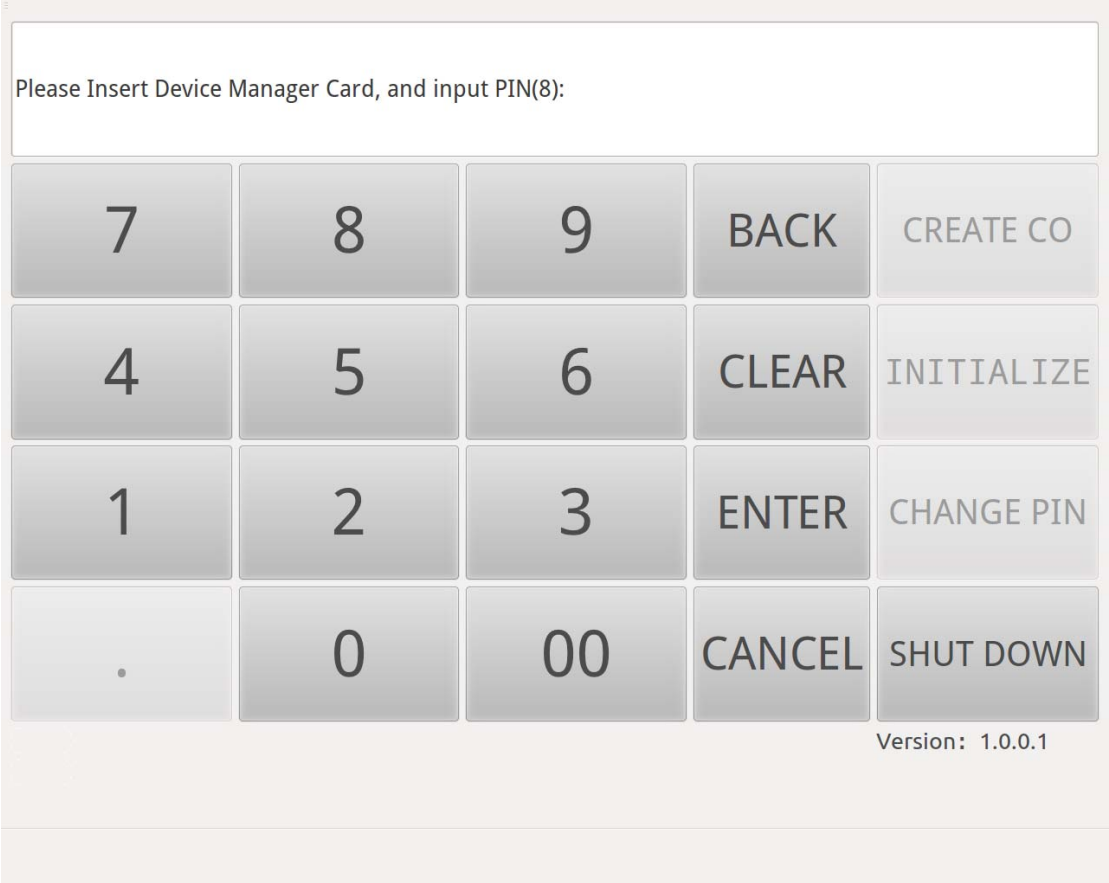

Figure 9Device Manager Authentication in Initialization

After authenticating Device Manager successfully, insert 5 Key Manager smart cards into Smart Card Reader(Key) in order, and input the smart card PIN, then Press “ENTER” button (Figure 10 -Figure 14).

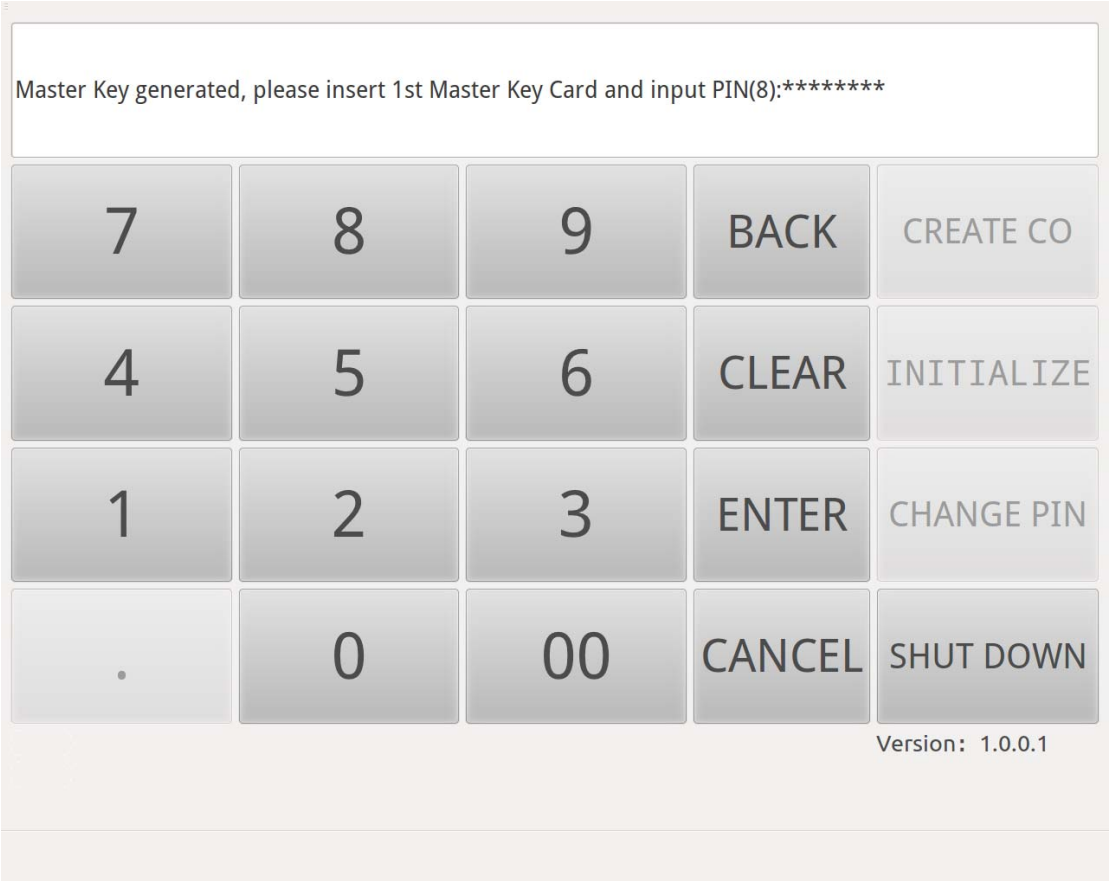

Figure 10Key Manager Authentication in Initialization-1

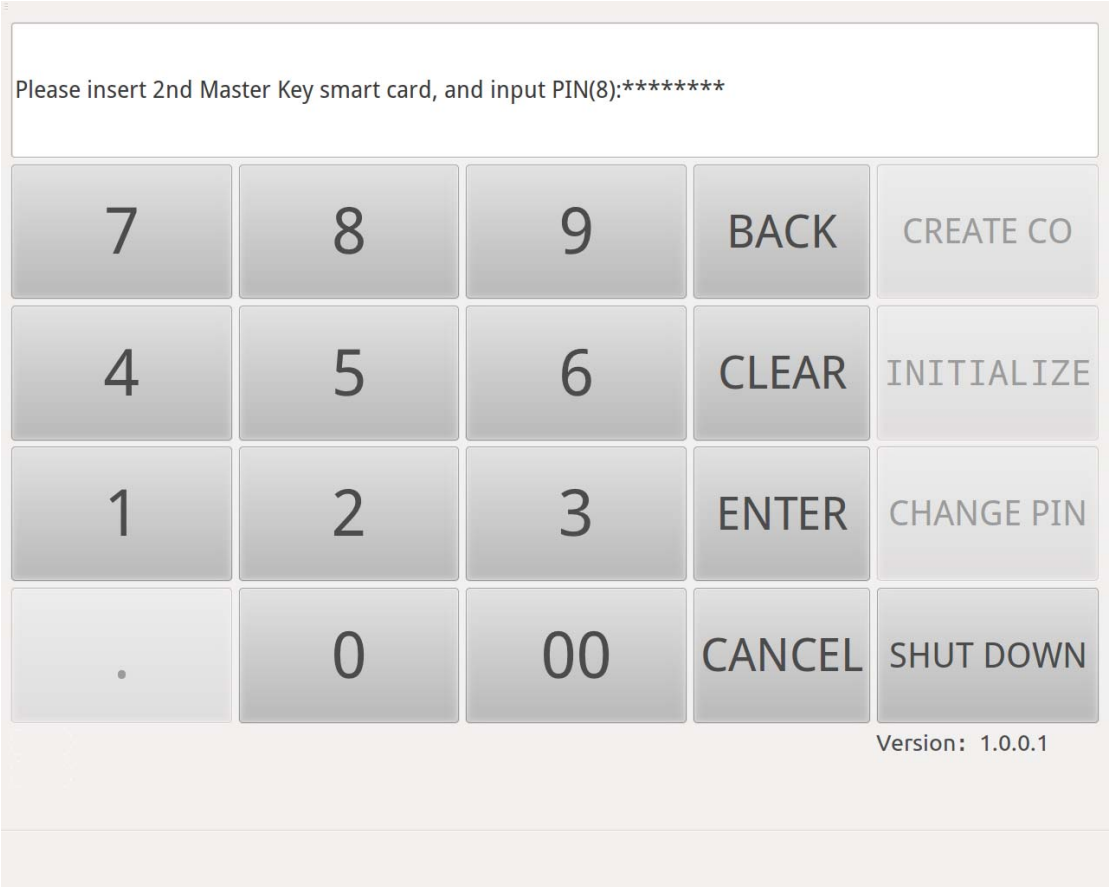

Figure 11Key Manager Authentication in Initialization-2

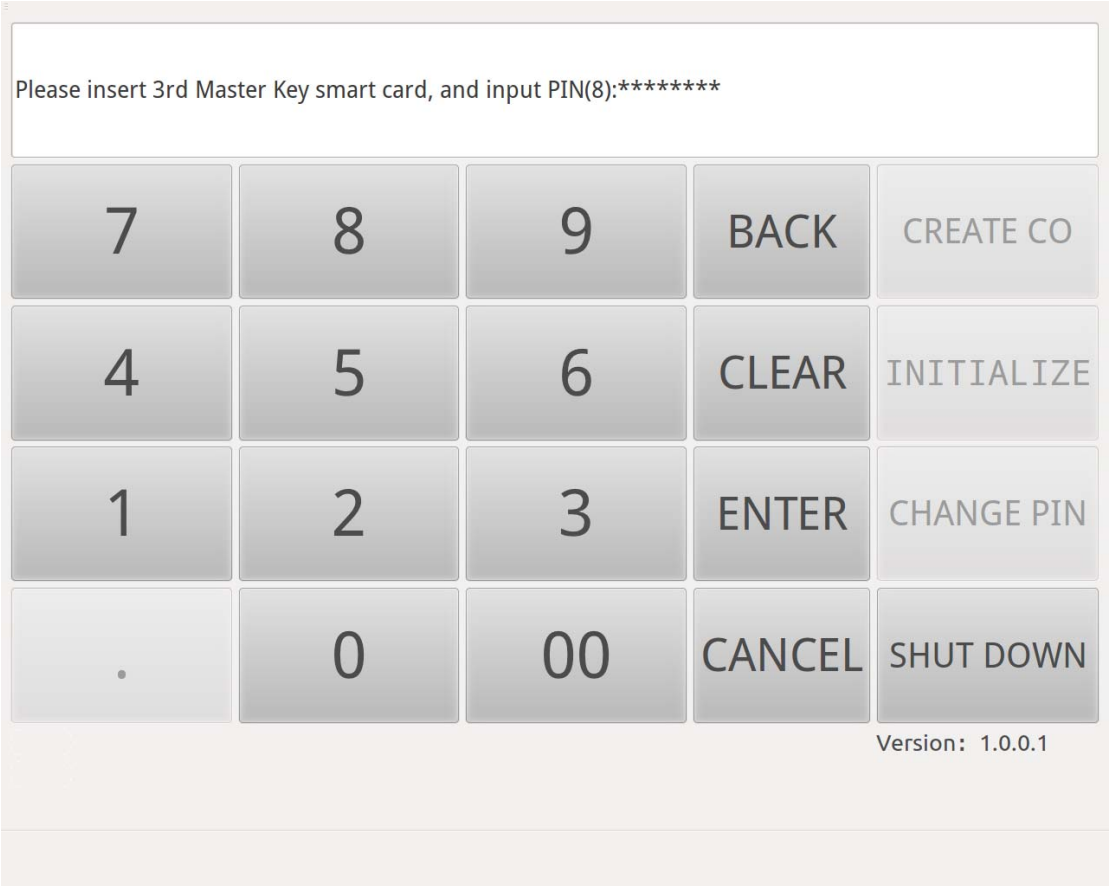

Figure 12Key Manager Authentication in Initialization-3

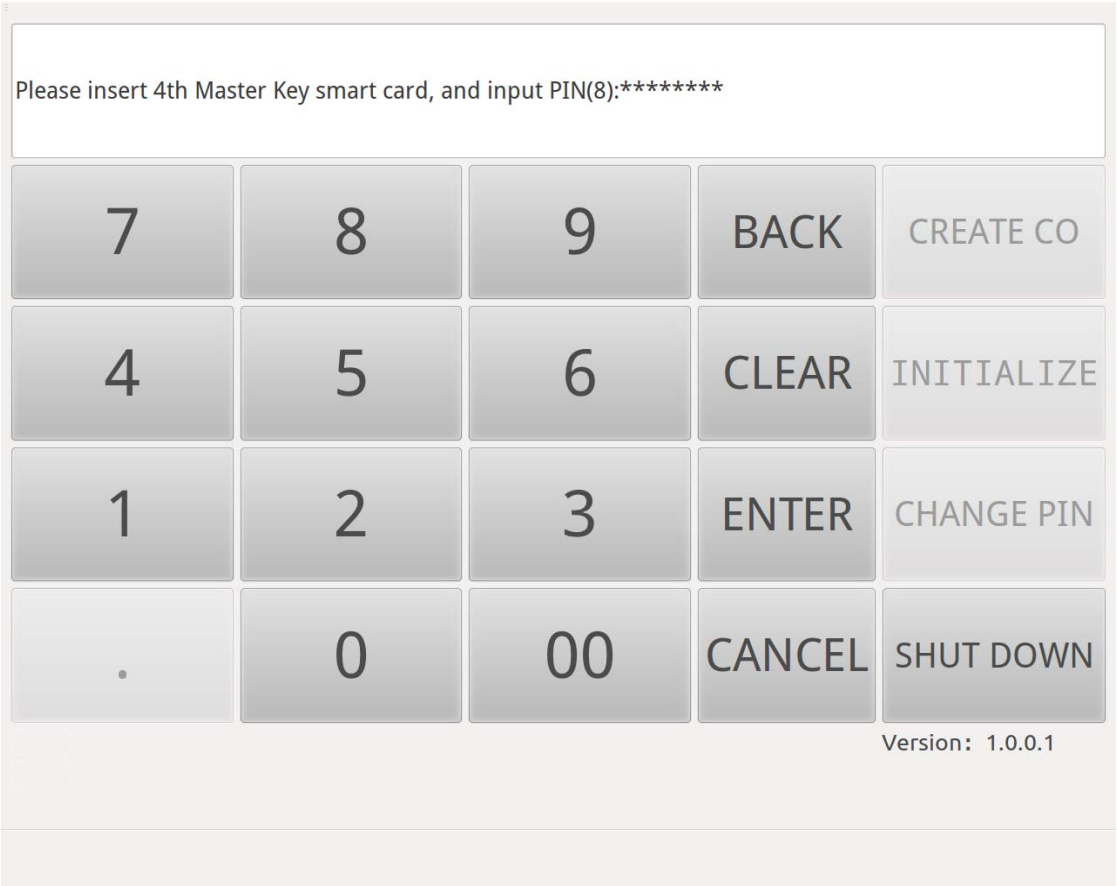

Figure 13Key Manager Authentication in Initialization-4

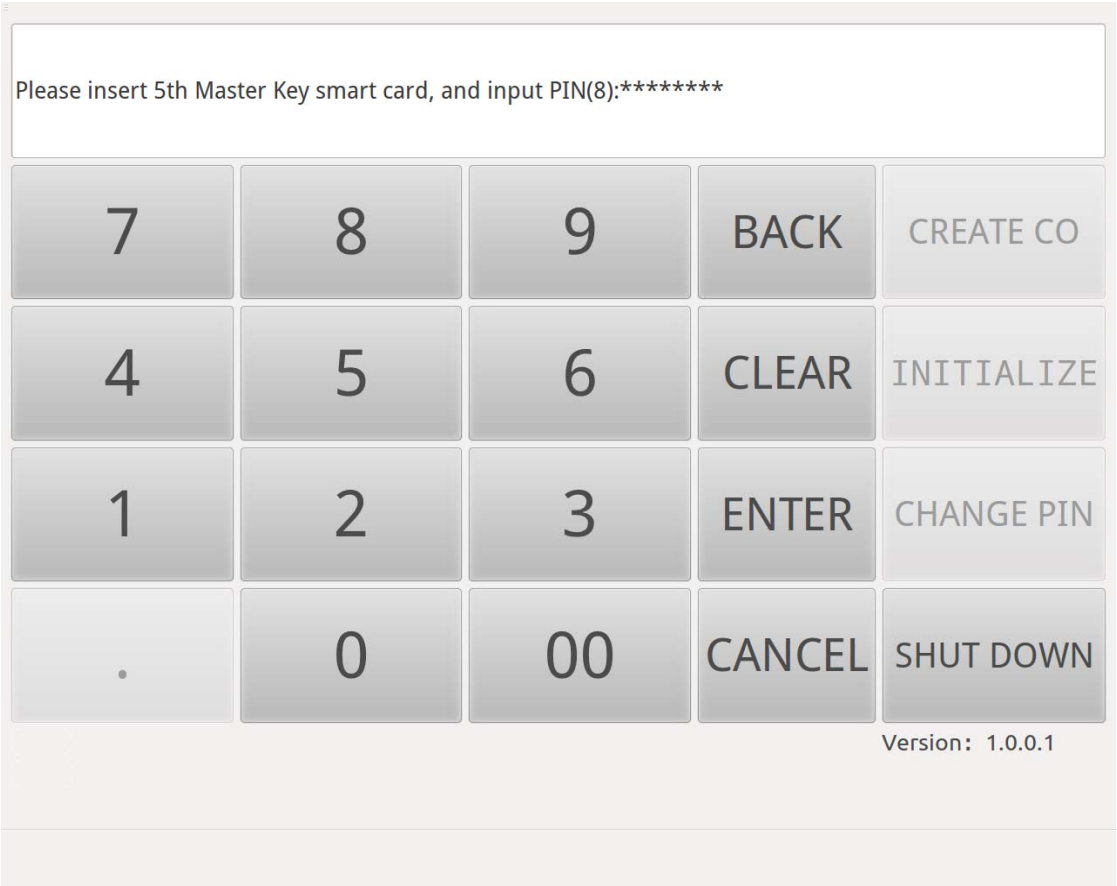

Figure 14Key Manager Authentication in Initialization-5

After 5 shares are stored into 5 Key Manager smart cards, Touch Screen will show message “Master Key Generated! Pressbutton to choose operation. ”

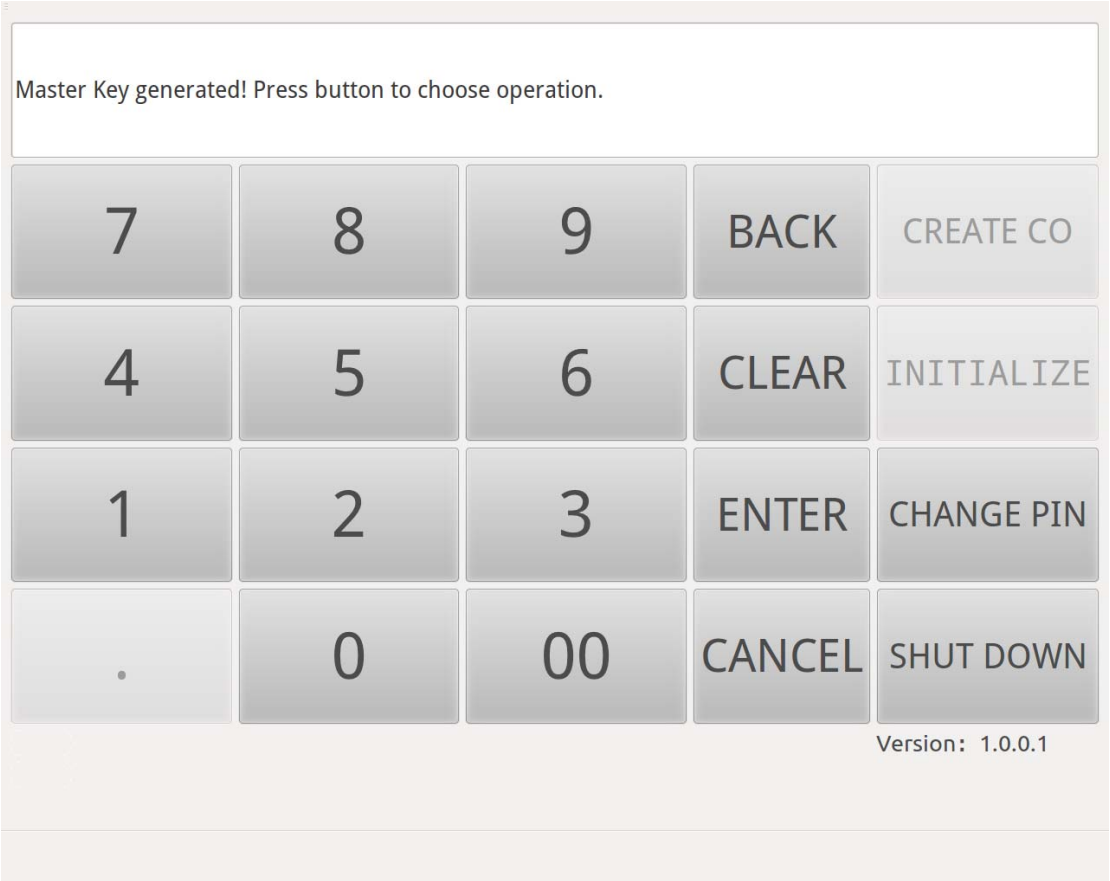

Figure 15Initialization Finished

3.2.2. Master Key is generated

If Master Key has already been stored into 5 Key Manager smart cards, when power on, the HSM needs to combine Master Key using 3 Key Manager smart cards.

Press “INITIALIZE” button, according to the message(Figure 16), first insert the Device Manager smart card into Smart Card Reader (ID) and input the corresponding PIN to authenticate Device Manager.

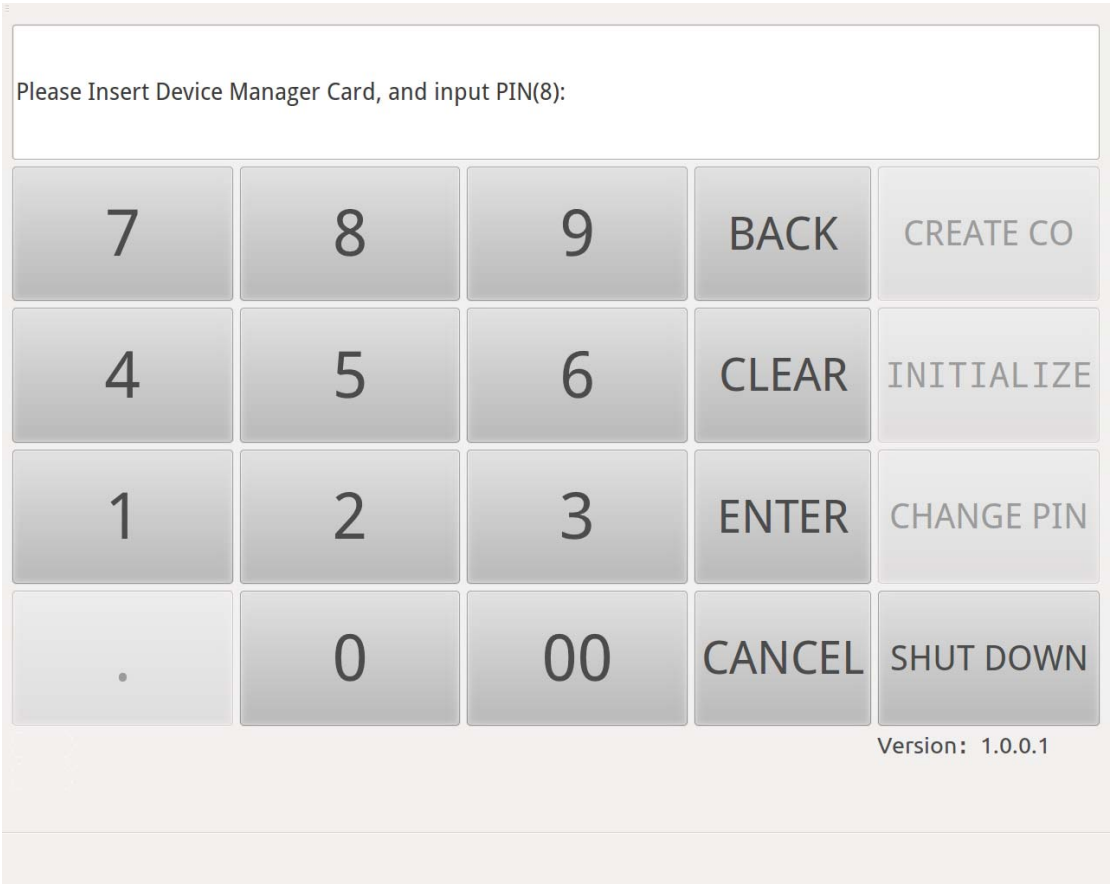

Figure 16 Device Manager Authentication in Initialization

After authenticating Device Manager, insert 3 Key Manager smart cards into Smart Card Reader(Key) in order, and input the smart card PIN(Figure 17 - Figure 19).

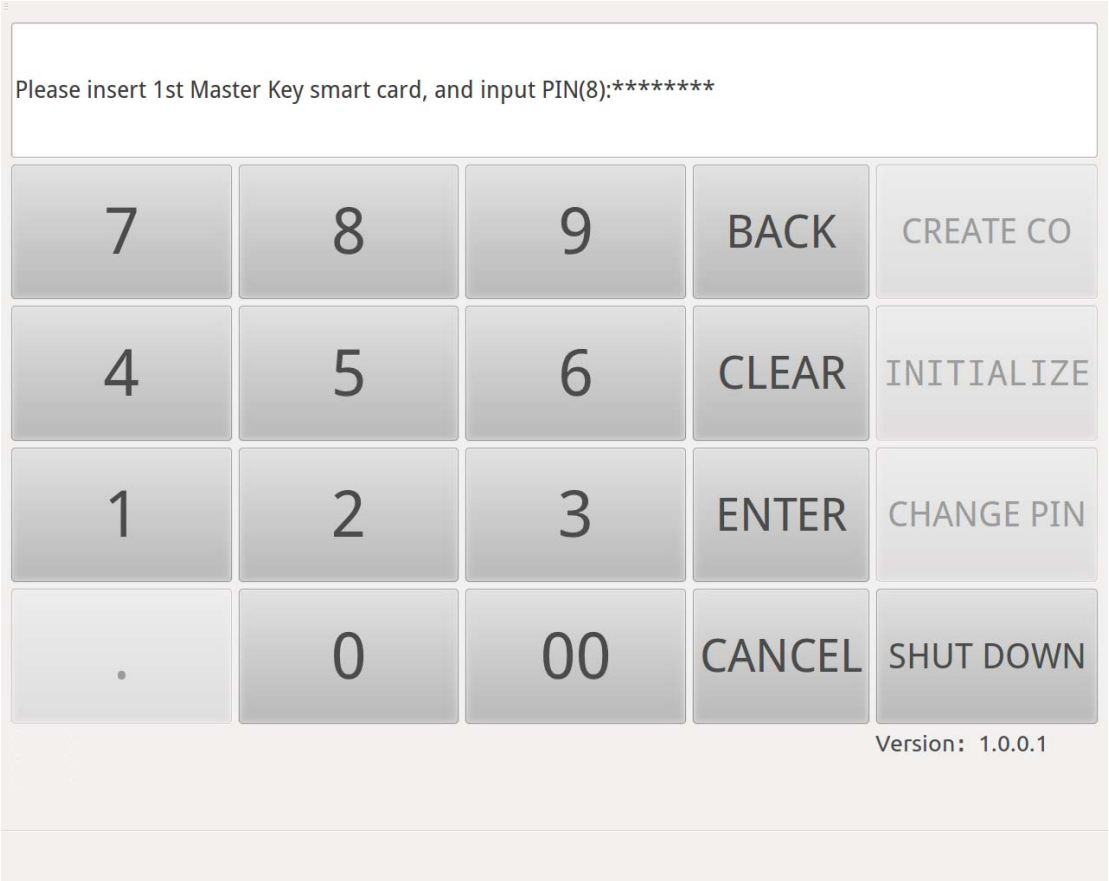

Figure 17 Key Manager Authentication in Initialization-1

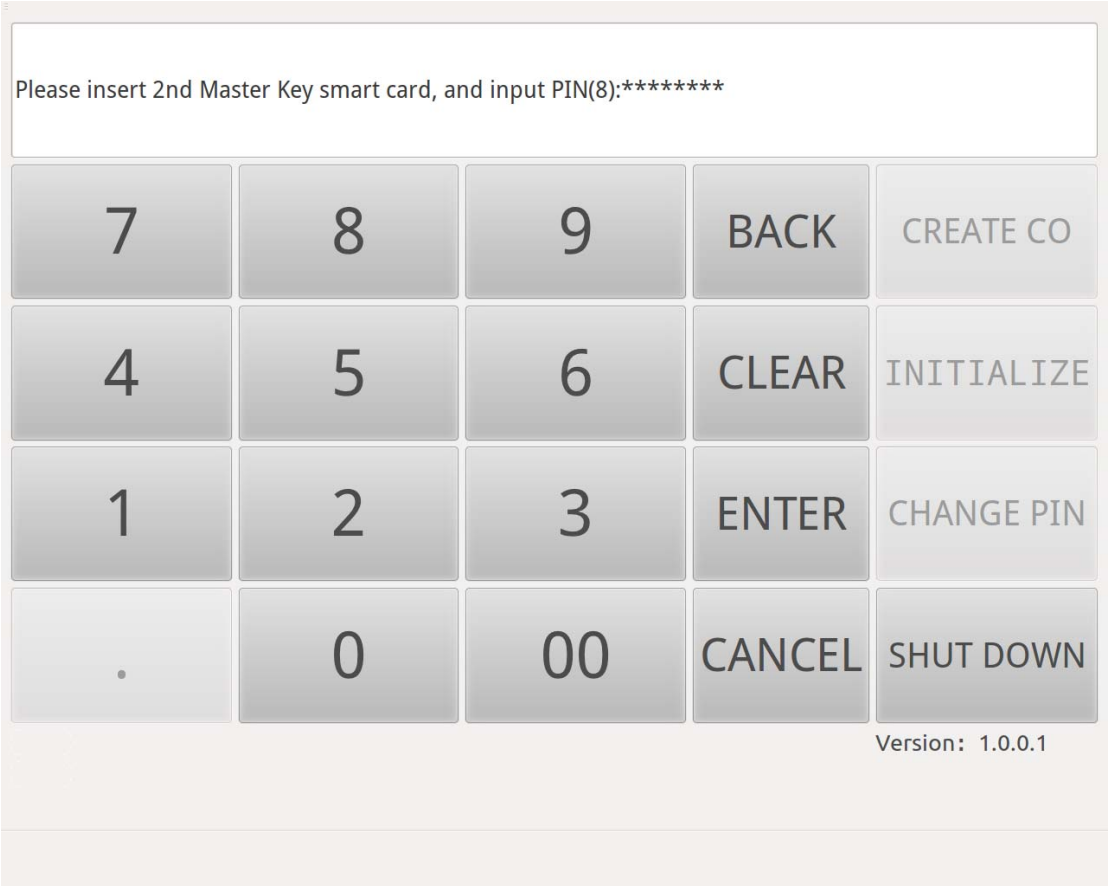

Figure 18 Key Manager Authentication in Initialization-2

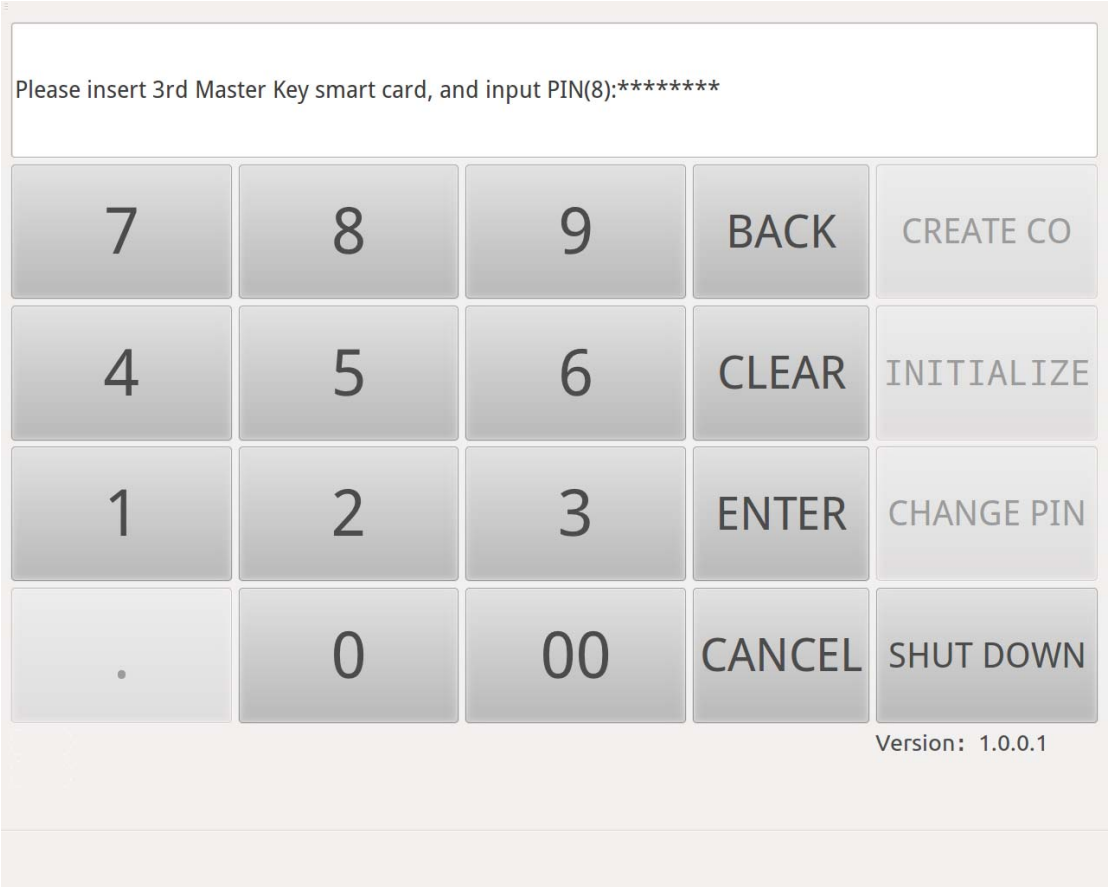

Figure 19 Key Manager Authentication in Initialization-3

After 3 shares are imported into HSM, Master Key will be recovered and Touch Screen will show message“Master Key Imported! Pressbutton to Choose Operation.” (Figure 20)

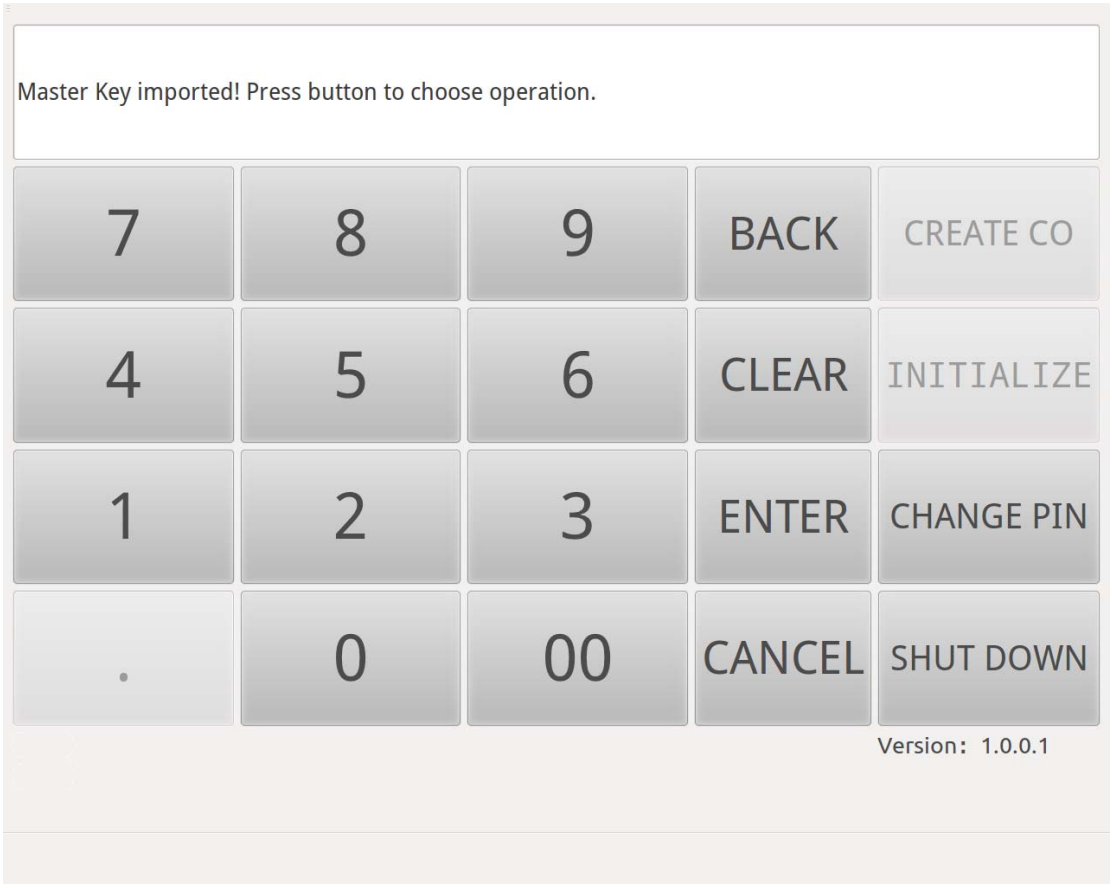

Figure 20 Initialization Finished

3. 3. Modify Smart Card PIN

It is highly recommended that you modify Smart Card PIN immediately Smart Card is created.

If you want to change smart card PIN, please press “CHANGE PIN” button. According to the message (Figure 21), insert the smart card that needs PIN modification into Smart Card Reader(ID) and input the old PIN to authenticate smart card.

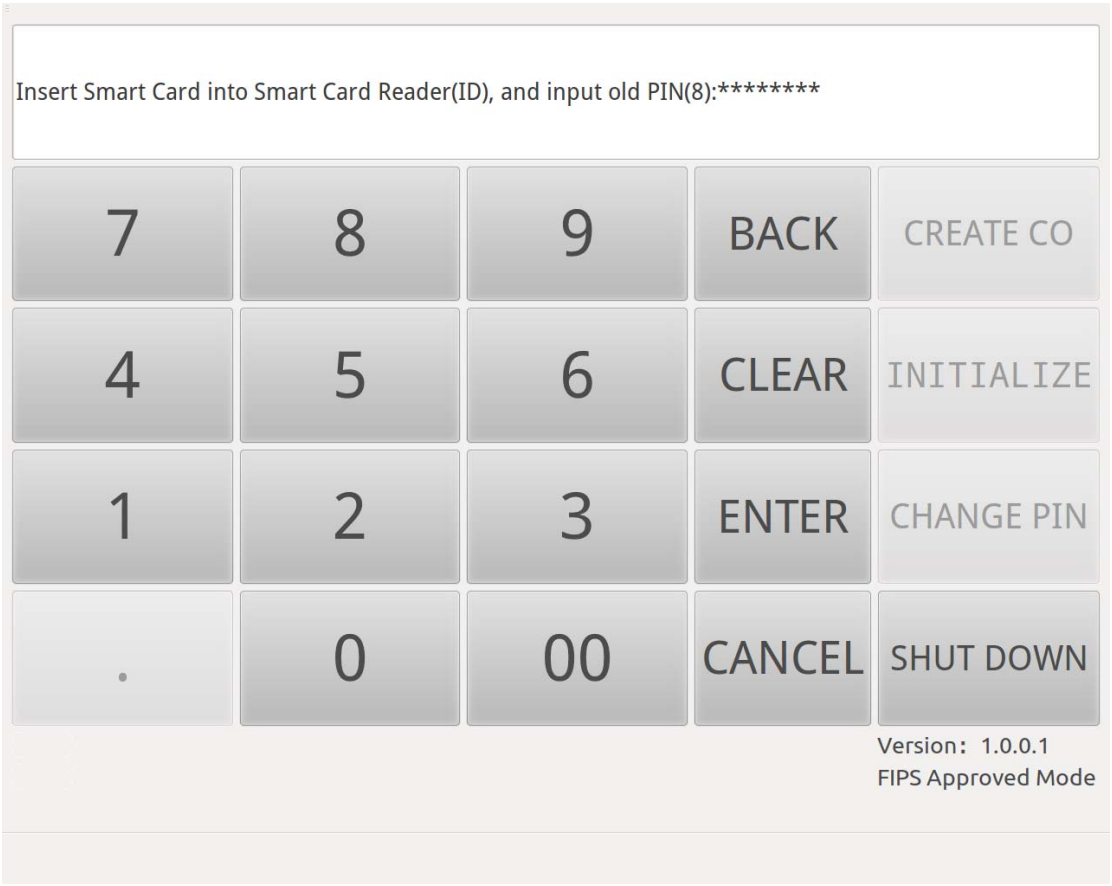

Figure 21 PIN Modification - Input Old PIN

After inputting the old PIN, according to the message, input and confirm the new PIN (Figure 22&Figure 23).

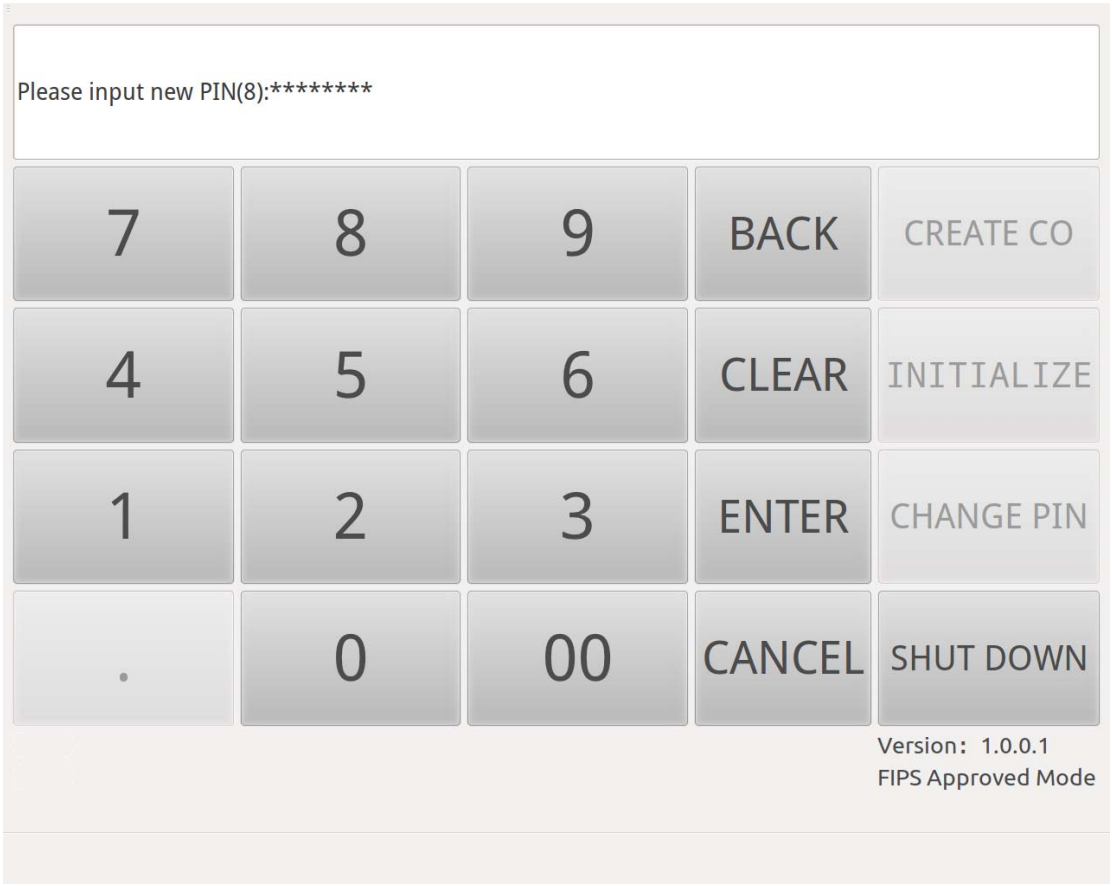

Figure 22 PIN Modification - Input New PIN

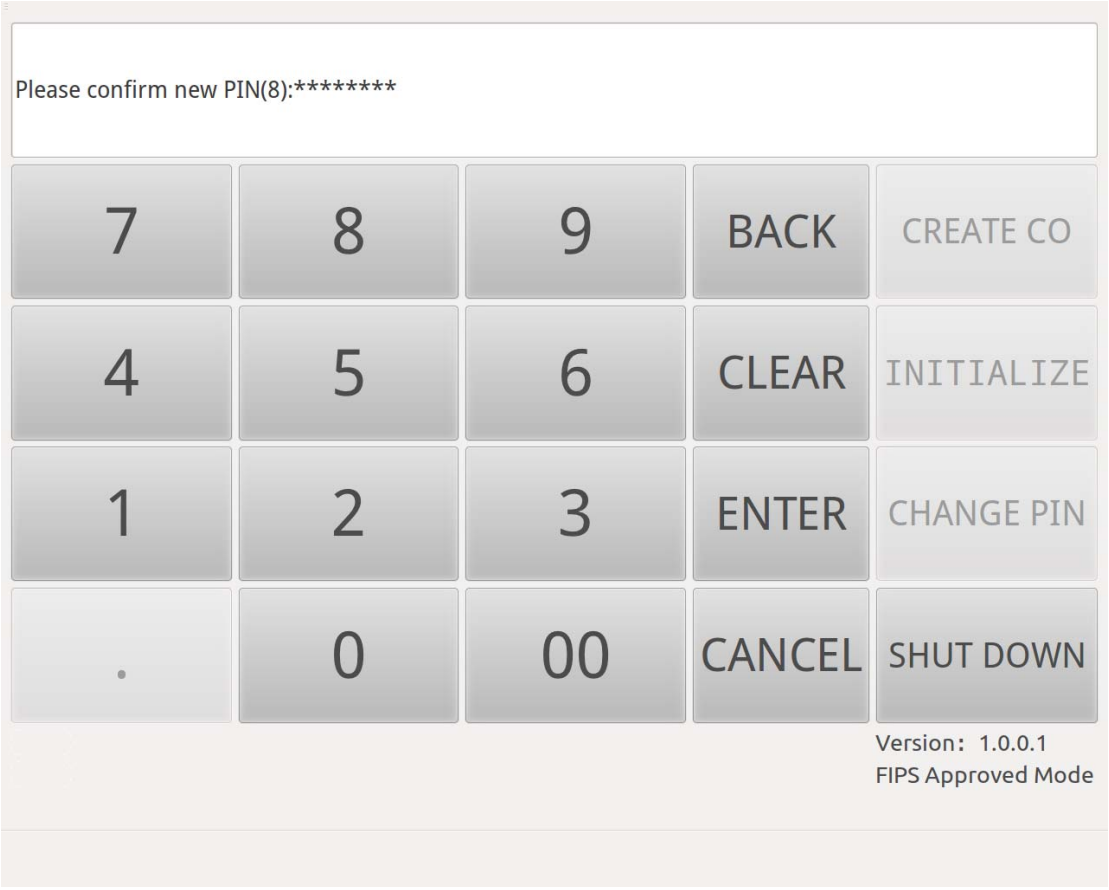

Figure 23 PIN Modification - Confirm New PIN

After the PIN is modified successfully, Touch Screen will show message“PIN Modification Succeeds!”(**Figure 24**)

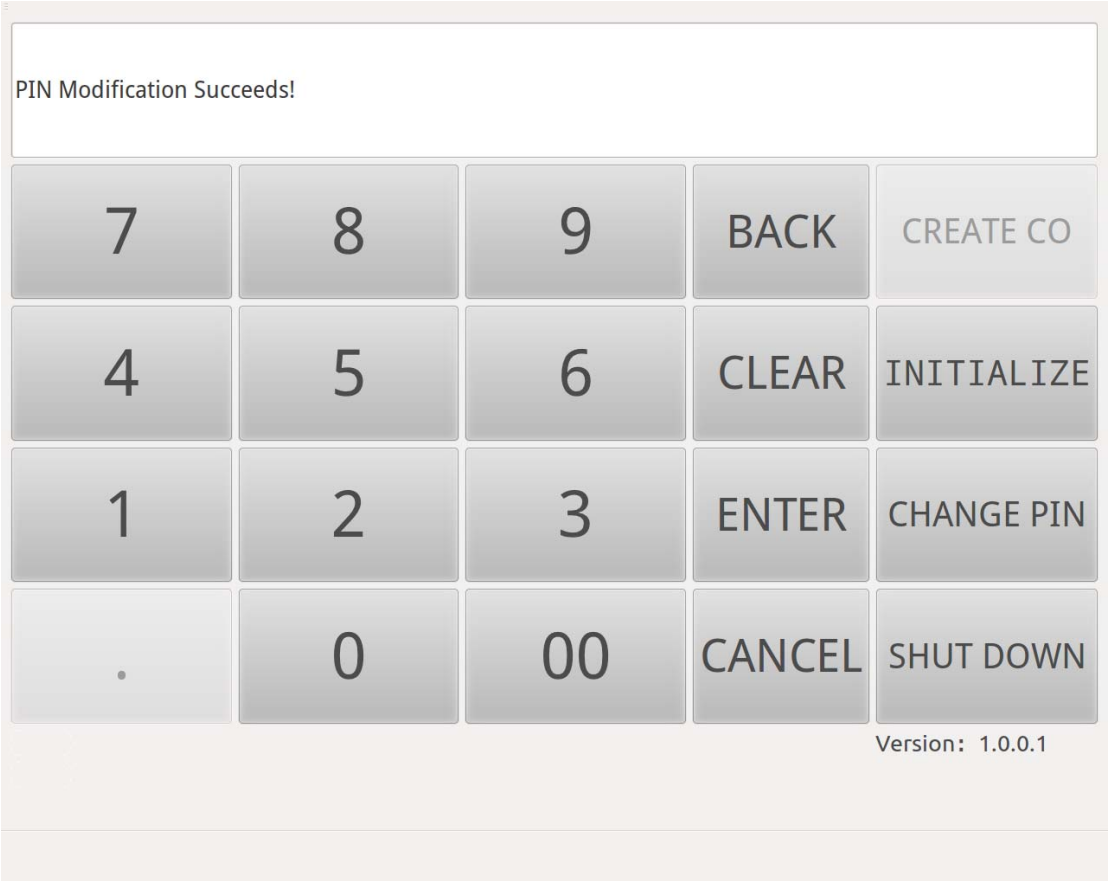

Figure 24 PIN Modification Succeeds

3. 4. Shut Down the HSM

Press “SHUT DOWN” button. According to the message (**Figure 25**), press“ENTER”button to shut down the HSM. If you need not to shut down the HSM right now, press “CANCEL” button to cancel “SHUT DOWN” operation.

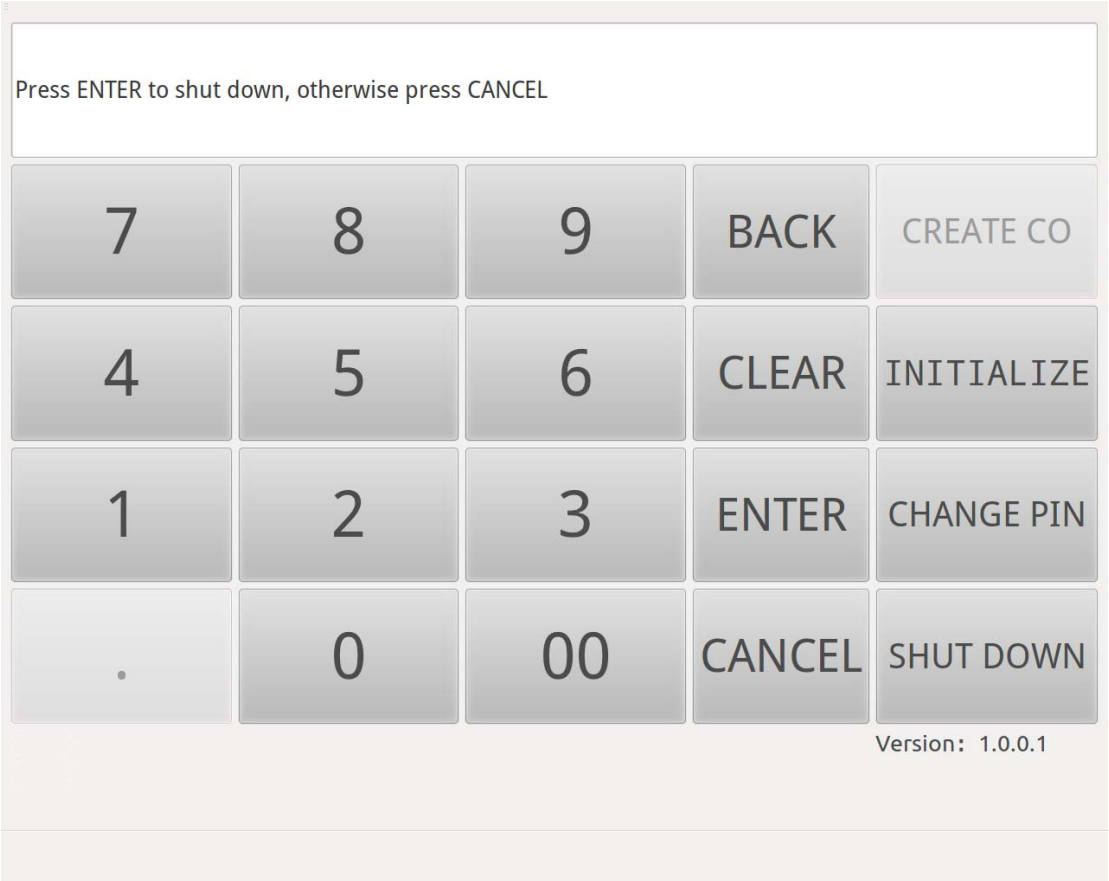

Figure 25 Shut Down the HSM

3.5. Error Message

The following table describes the error message shown on the Touch Screen, gives the probable reason and the solution.

Table 4 Error Message Description

Error Message	Reason	Solution
Fail to Create Device Manager!	**1.** There is no card in Smart Card Reader(ID); **2.** The PIN which you input is not correct.	Make sure you insert card in Smart Card Reader(ID), and input correct PIN.
Fail to Create Auditor!		
Fail to Create		

Authorizer!		
Authentication Failed, please Insert Device Manager Key Card, and Input PIN again (8):	**1.** There is no card in Smart Card Reader(ID); **2.** The card in Smart Card Reader(ID) is not Device Manager Smart card; **3.** The PIN which you input is not correct.	Make sure you insert Device Manager Smart card in Smart Card Reader(ID), and input correct PIN.
Cannot Get Smart Card UID	**1.** There is no card in Smart Card Reader(Key); **2.** The card inserted is invalid.	Make sure you insert HSM-matched card in Smart Card Reader(ID).
The Smart Card has been inserted before!	The Smart Card has been inserted before.	Insert other card that has not been inserted before.
Master Key Recovery Failed!	Master Key combined with 3 Master Key Smart Card does not match with its hash.	**1.** Make sure Smart Card that you insert is Master Key Smart Card. Carry out INITIALIZE again. **2.** Contact manufacturer.
Two PINs is Inconsistent, Please input new PIN again (8):	The two new PINs which you input are inconsistent	You should retype in new PIN twice and make sure the two PINs are consistent.
PIN modification failed!	**1.** There is no card in Smart Card Reader(ID); **2.** The old PIN which you input is not correct. **3.** The card inserted is invalid.	Press “CHANGE PIN” button again to carry on PIN modification and make sure you insert card into the Smart Card Reader(ID) and input correct PIN.
Sensor Triggered, Key Service Stops!	Sensor is triggered.	1. Restore to Factory Setting and reboot.

		2. Contact manufacturer.
Device Key Broken, Key Service Stops!	Device Key is broken.	1. Restore to Factory Setting, and reboot. 2. Contact manufacturer.
Factory Setting Over, Please Reboot!	HSM is in Factory Setting state.	Reboot.
Known Answer Test for Cryptographic Algorithm fails.	The results of Known Answer Test for Cryptographic Algorithm is inconsistent with known results.	1. Reboot. 2. Contact manufacturer.
Known Answer Test for Keyed Hashing Algorithm fails.	The results of Known Answer Test for Keyed Hashing Algorithm is inconsistent with known results.	1. Reboot. 2. Contact manufacturer.
Known Answer Test for Embedded Cryptographic Algorithm is fails.	The results of Known Answer Test for Embedded Cryptographic Algorithm is inconsistent with known results.	1. Reboot. 2. Contact manufacturer.
Protection Card Test fails.	The protection card state is inconsistent with the expected state.	1. Reboot. 2. Contact manufacturer.
RNG Test fails.	The random number bits generated by DRBG of HSM do not pass the random number test.	1. Reboot. 2. Contact manufacturer.
Operating System Test fails.	The operating system state is inconsistent with the expected state	1. Reboot. 2. Contact manufacturer.
Hardware Test fails.	The hardware state is inconsistent with the expected stat.	1. Reboot. 2. Contact manufacturer.
Network	The network configuration is	1. Reboot.

Configuration Test fails.	inconsistent with the expected configuration.	2. Contact manufacturer.
Application Integrity Test fails.	If the hash value of application is inconsistent with the known hash value.	1. Reboot. 2. Contact manufacturer.

4. Remote Management Application

Remote Management Application is an application developed for HSM-ZJ2014 and used for managing the HSM. It needs to run on an individual computer. Before running Remote Management Application, please make sure your PC or laptop meets requirements below:

- OS: Windows Vista/7/8,32/64 bit
- Dynamic link libraries (*.dll) in product CD has been copied to the file folder where Remote Management Application installs.

4. 1. Crypto Officer Logon User Interface

Before using Remote Management Application, please connect your computer where runs Remote Management Application to the HSM's Management Port and configure your network adaptor as following:

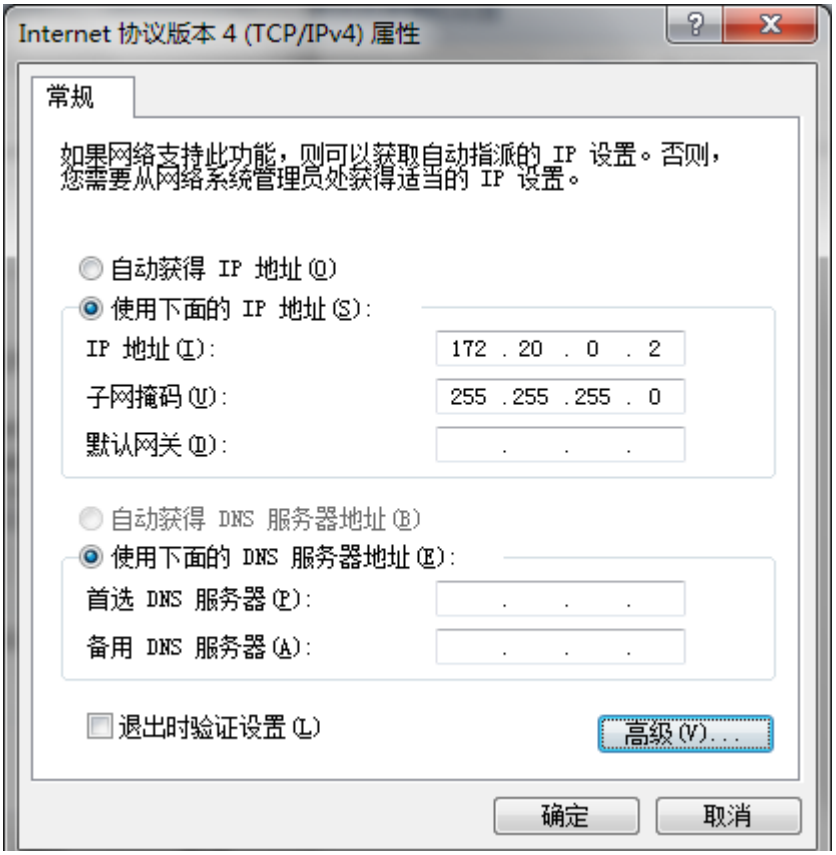

Figure 26 Network Configuration

Launch Remote Management Application, then Crypto Officer Logon User Interface (**Figure 27**) will show up.

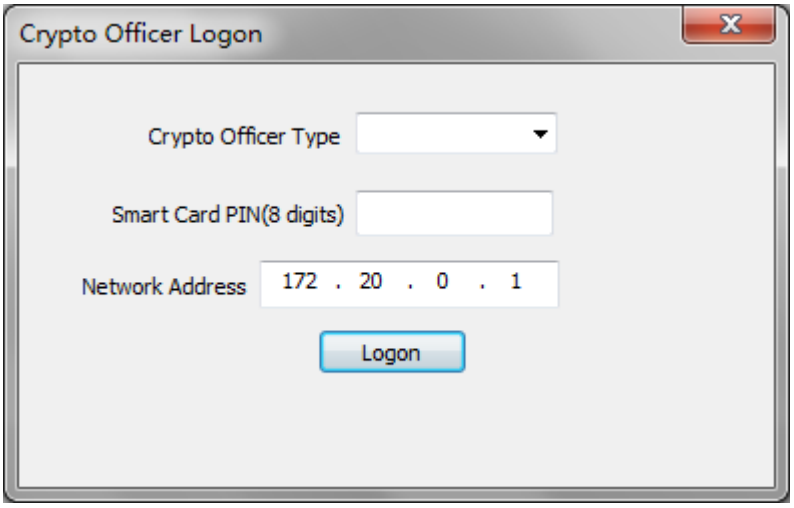

Figure 27 Crypto Officer Logon User Interface

In this UI, you should firstly choose Crypto Officer Type, then insert the corresponding smart card, input its PIN and Management Port IP address, then press “Logon”. After authenticating Crypto Officer’s identity, UI will turn to the corresponding Crypto Officer UI.

4. 2. Device Manager User Interface

In Crypto Officer Logon User Interface (**Figure 28**), choose “Device Manager” as Crypto Officer Type, then insert Device Manager smart card, input its PIN and Management Port IP address (Default is172.20.0.1, generally it needs not to modify), then press “Logon”.

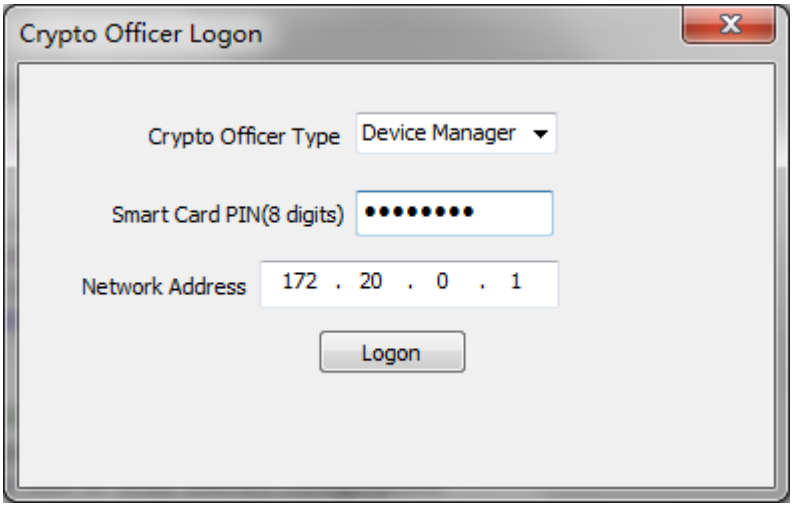

Figure 28 Device Manager Logon Interface

After authenticating Device Manager’s identity, UI will turn to Device Manager UI.

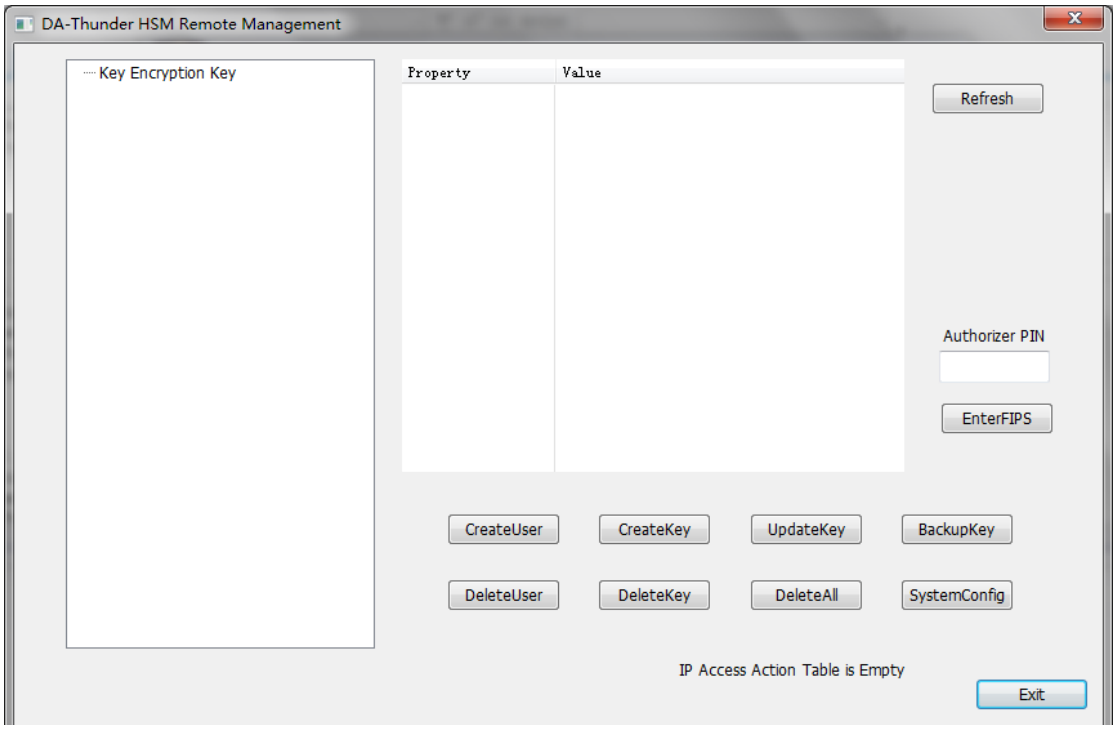

Figure 29 Device Manager UI

4.2.1. User Account and Key Management

4. 2. 1. 1. Create User Account

In Device Manager UI, press “CreateUser” button, the following UI will show up.

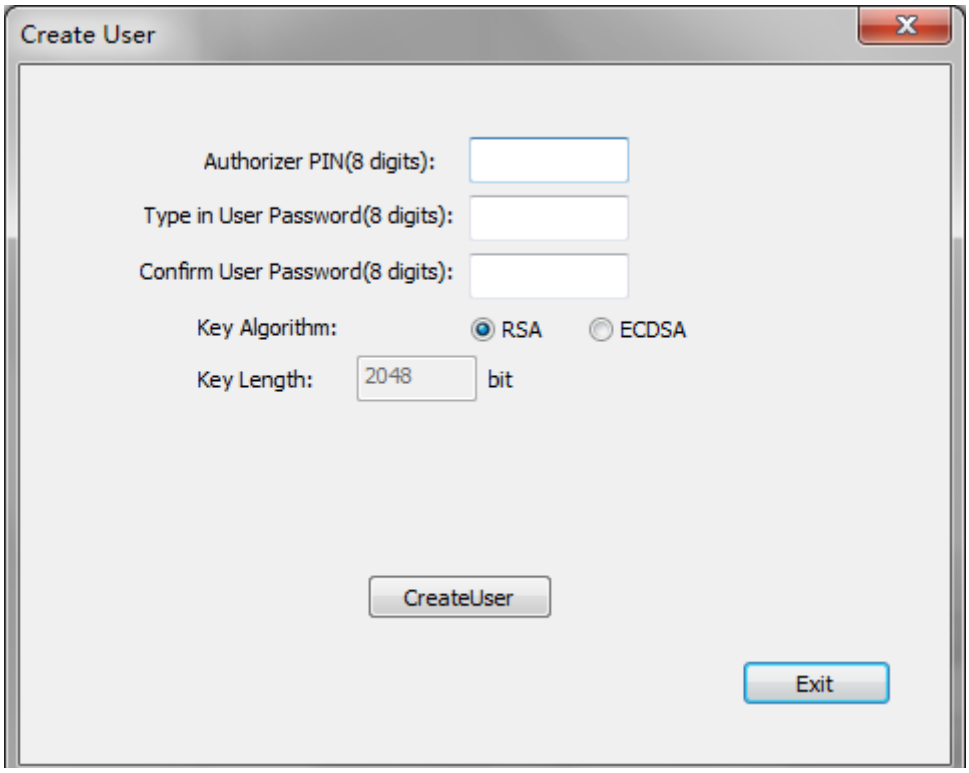

Figure 30 User Account Creation

Insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then input and confirm the user password, then press "CreateUser". If the user is created successfully, the following UI will show up. The default key algorithm of the user key is RSA (2048-bit), ECDSA (NIST p256) can be also chosen as the user key algorithm.

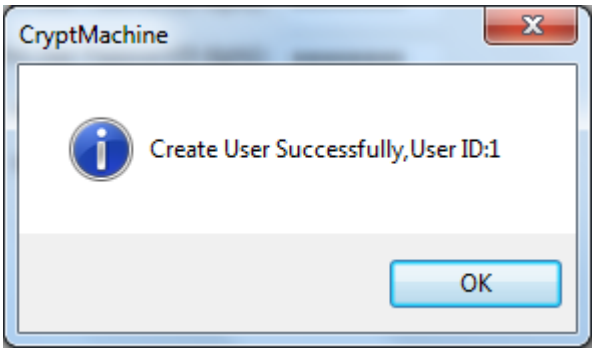

Figure 31 User Account Creation Succeeds

User ID is assigned by the HSM. Each user has 2 default RSA/ECDSA keys, which cannot be deleted.

4. 2. 1. 2. **Delete User Account**

`

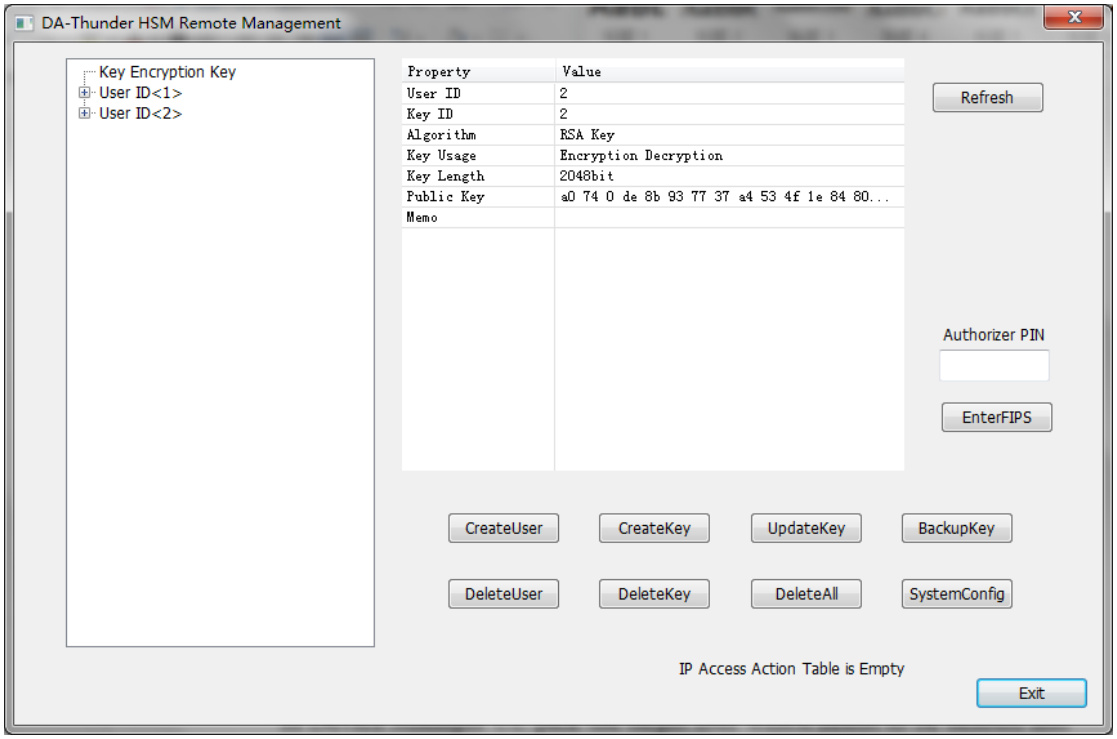

Figure 32 User Account Deletion

In Device Manager UI, pick the target user which needs to be deleted and press “DeleteUser” button, the following dialog will show up.

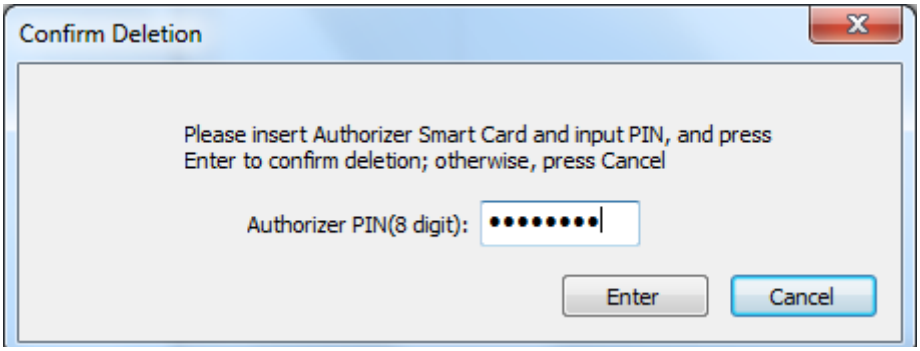

Figure 33 User Account Deletion Confirm

According to the dialog (**Figure 33**), insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then press "ENTER" to confirm user deletion, otherwise press "CANCEL" to cancel. If the user is deleted successfully, the following dialog will show up.

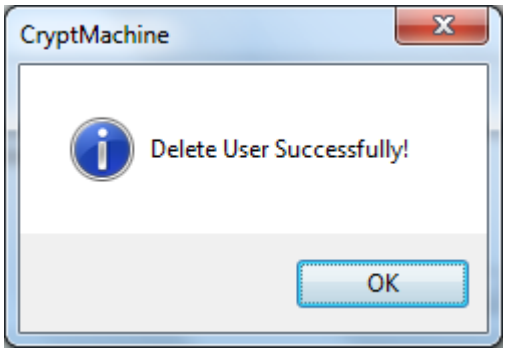

Figure 34 User Deletion Succeeds

4. 2. 1. 3. **Create Key**

In Device Manager UI, press "CreateUser" button, the following UI will show up.

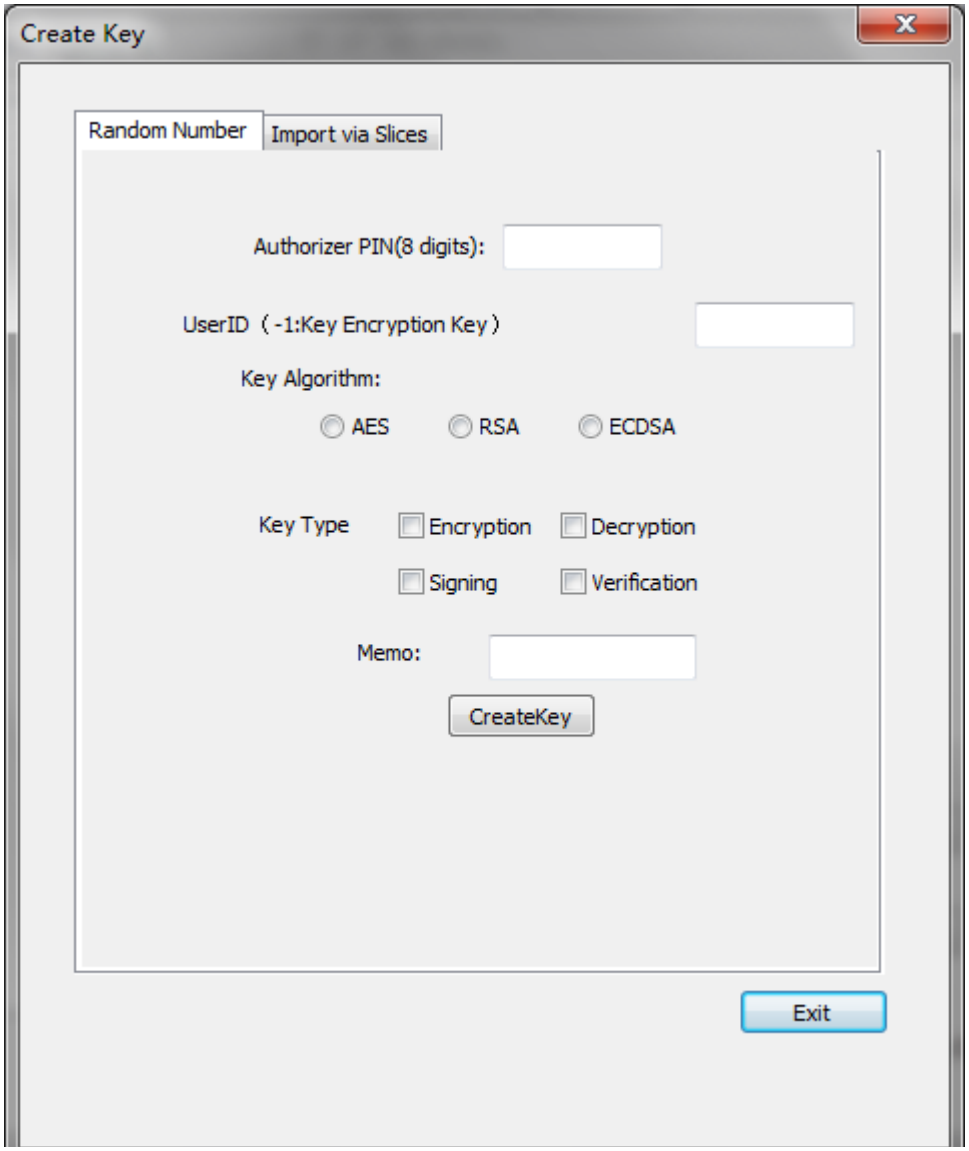

Figure 35 User Key Creation

When creating key, the key value can be generated by internal RNG or imported by 2 shares that Device Manager inputs.

4.2.1.3.1. Generate from Internal RNG

Create Key UI's default page is "Internal RNG", which is shown as below.

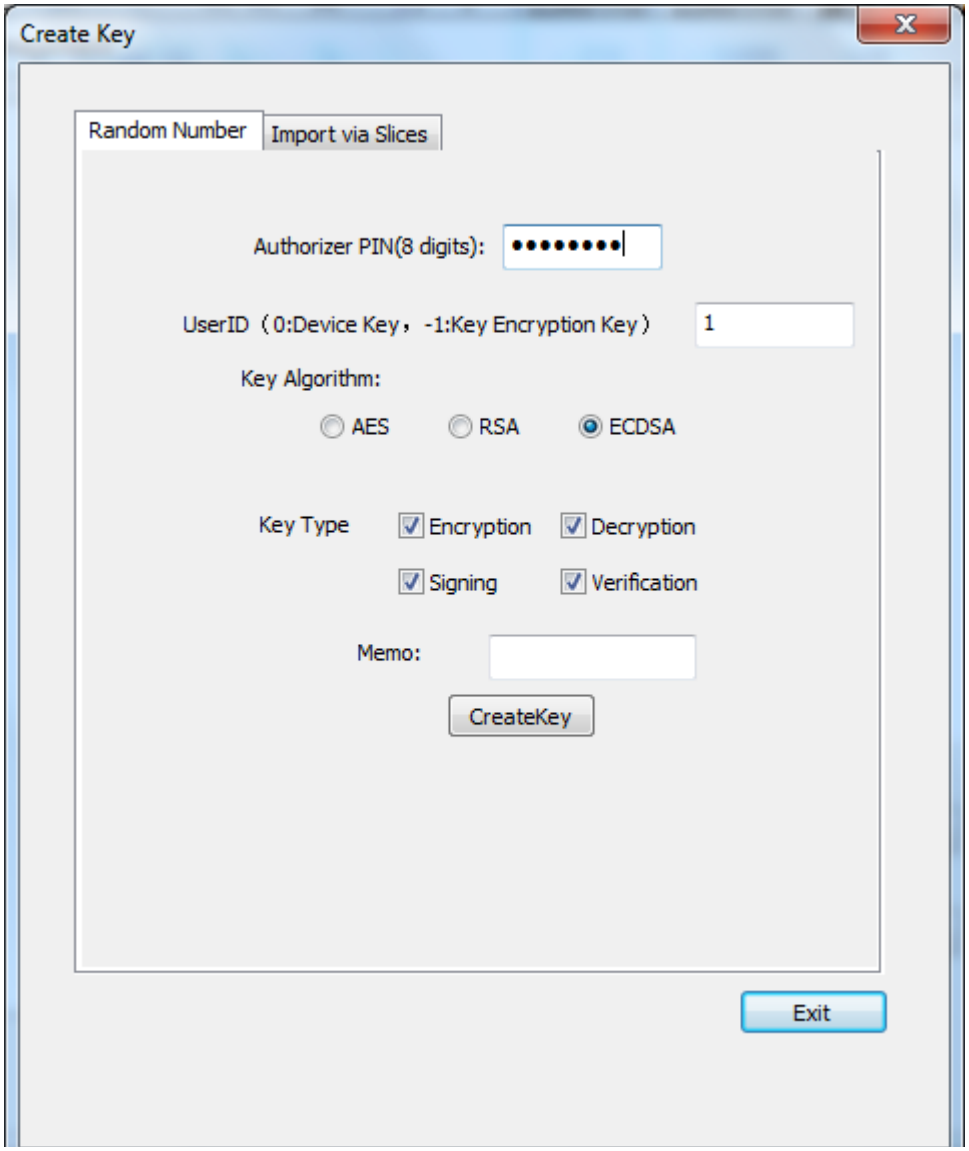

Figure 36 Key Creation via Internal RNG

In this page (**Figure 36**), insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, type in User ID which the key belongs to, choose Key Algorithm, Key Usage and Key Memo (optional), then Press “CreateKey” button. If the key is created successfully, the following dialog will show up.

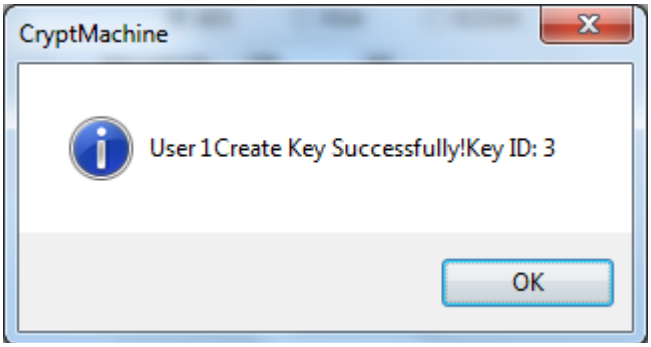

Figure 37 Key Creation Succeeds

Note that Key ID is assigned by HSM.

4.2.1.3.2. Import from Key Shares

In Create Key UI, choose page “Import via Slices”. UI below will show up.

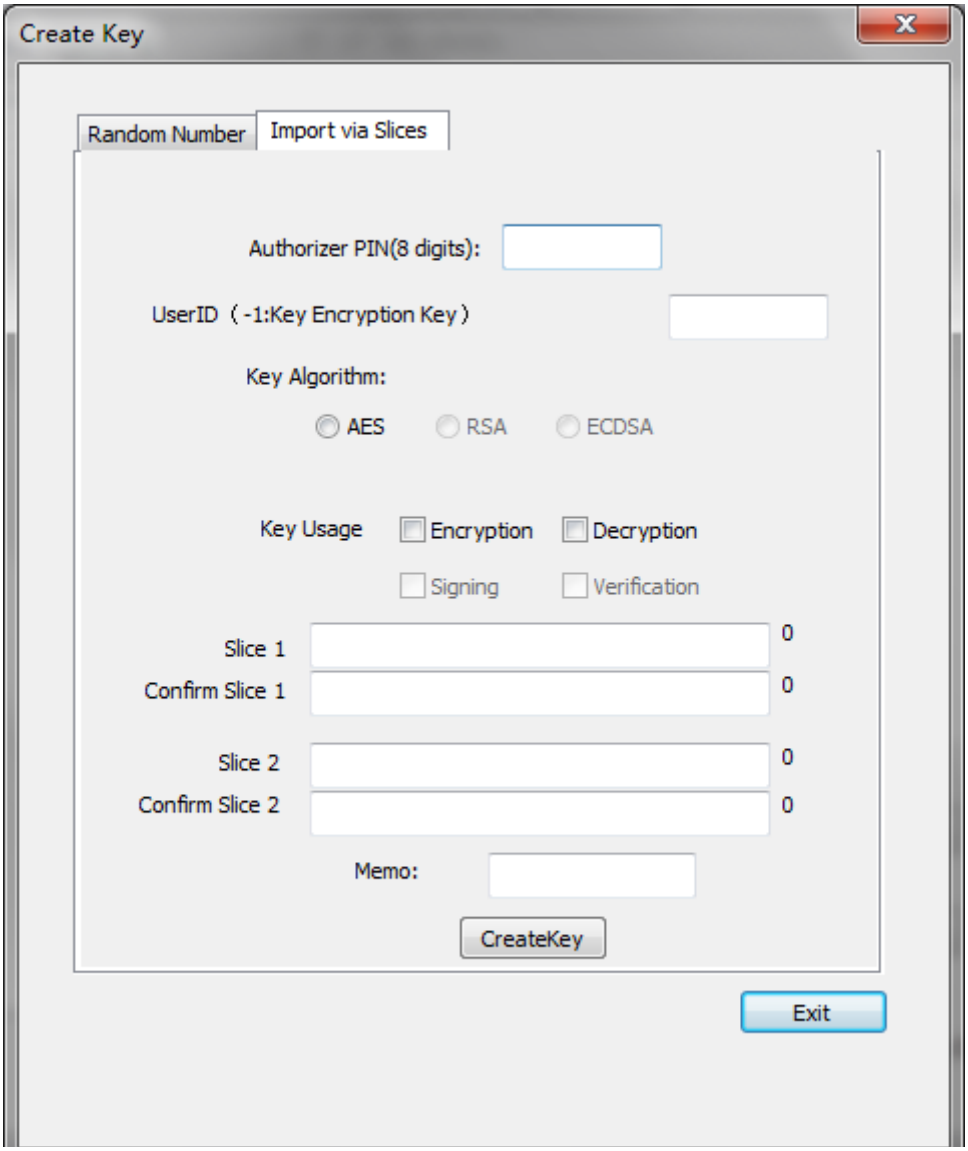

Figure 38Key Creation from Shares

In this page**Figure 38**, insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, type in User ID which the key belongs to, choose Key Algorithm and Key Memo (optional), then Press “CreateKey” button.

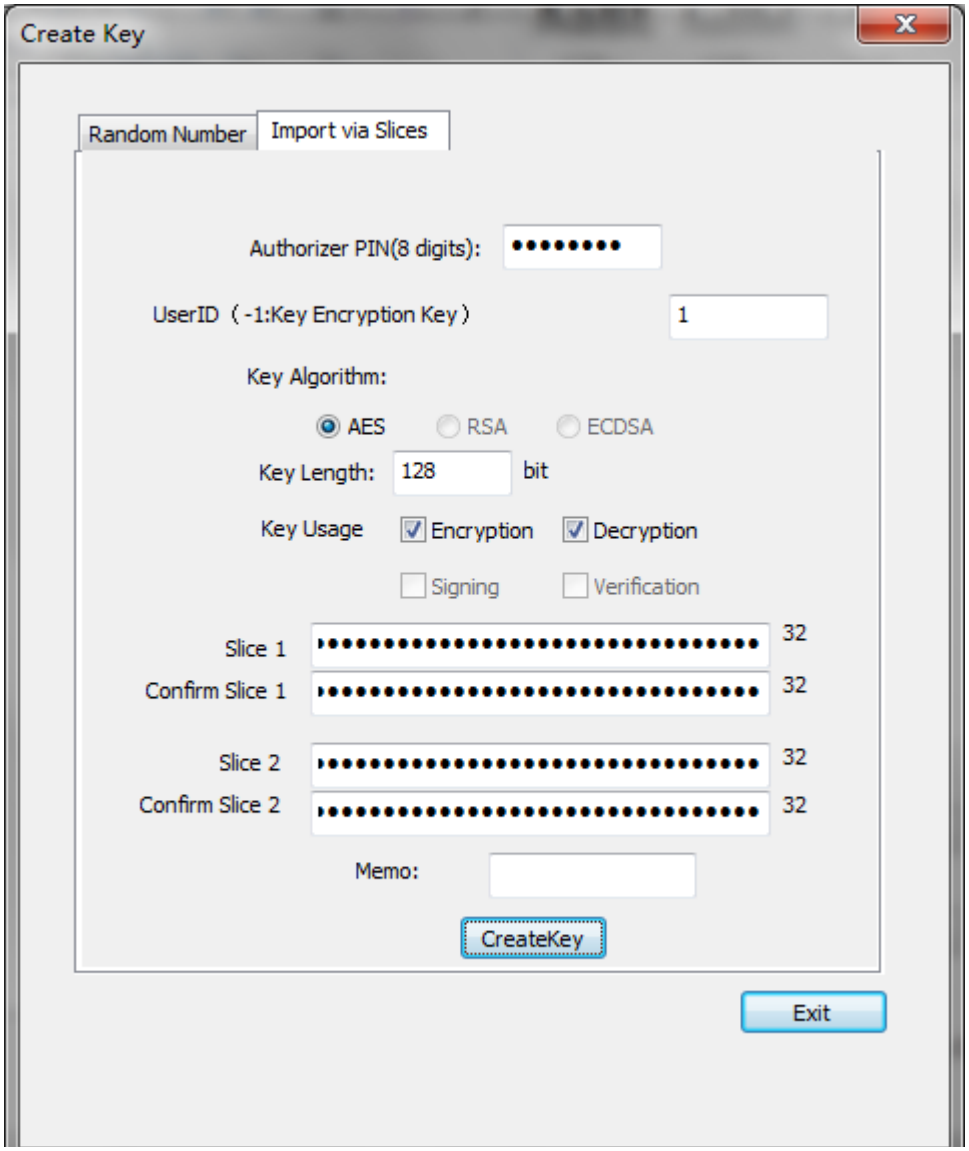

Figure 39 Key Creation from Shares

If the key is created successfully, the following dialog will show up.

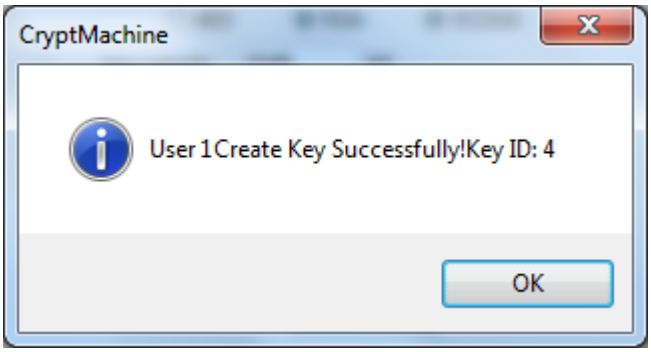

Figure 40 Key Creation Succeeds

Key ID is assigned by HSM.

4. 2. 1. 4. View Key

Click key in the left tree view, the corresponding key information will show as below.

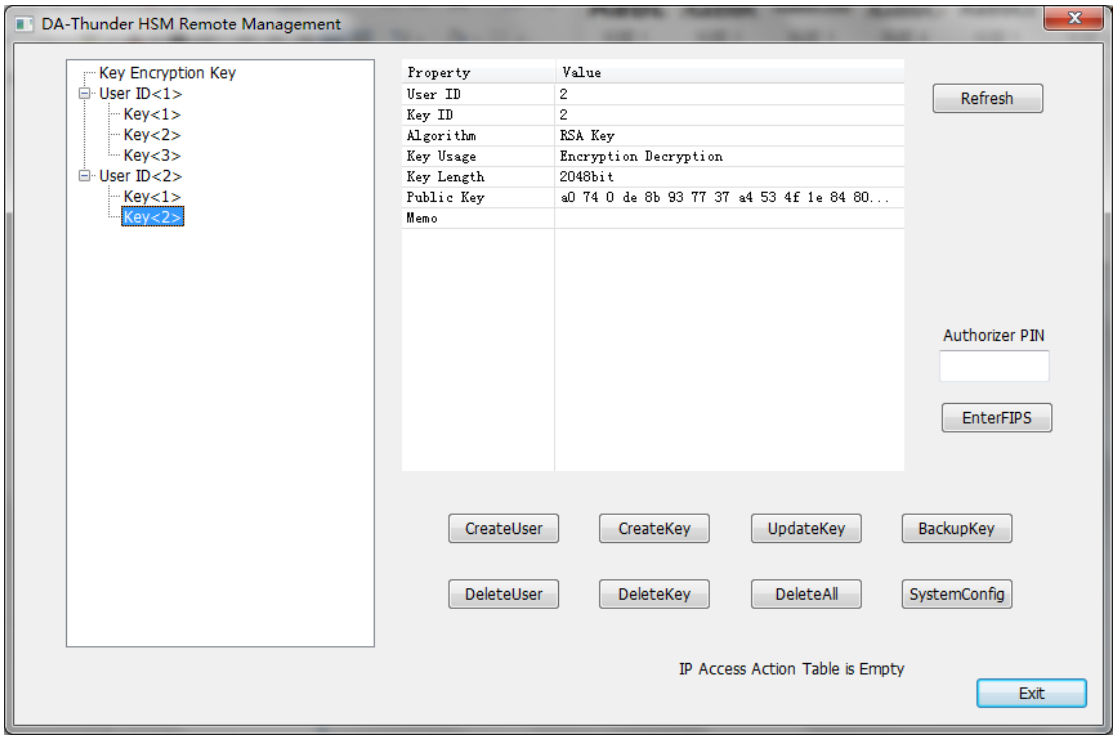

Figure 41View Key Information

4. 2. 1. 5. Update Key

In Device Manager UI (**Figure 29**),pick the target key and press “UpdateKey” button, below UI will show up.

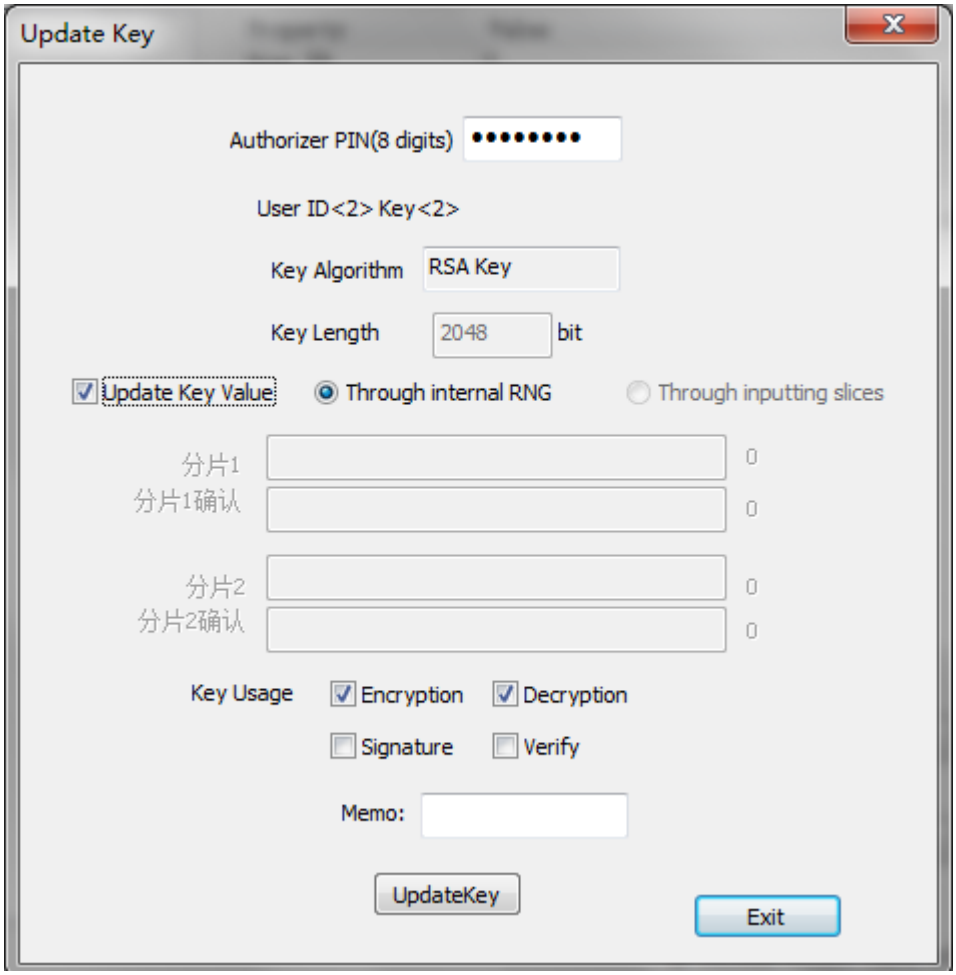

Figure 42 Key Update

Insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN first.

The key information which can be updated is Key Usage and Memo. You can input new information inthe correspondingedit box or tick the corresponding checkbox. Moreover, if you want to update Key Value, you can tick the “Update Key Value” checkbox.

Then press “UpdateKey” button. If the key or key information is updated successfully, the following UI will show up.

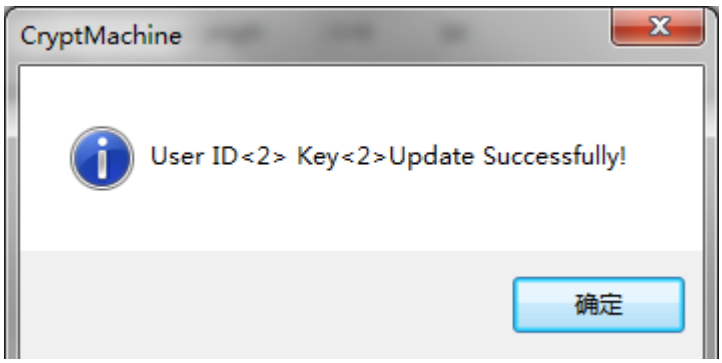

Figure 43 Key Update Succeeds

4. 2. 1. 6. **Delete Key**

In Device Manager UI, pick the target key which needs to be deleted and press “DeleteKey” button, the following dialog will show up.

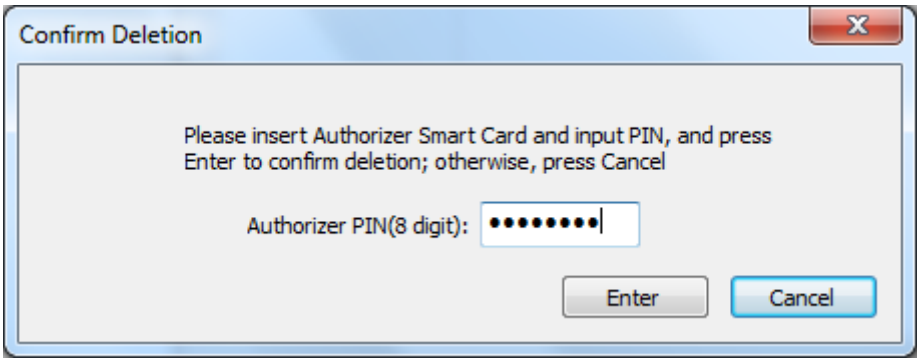

Figure 44 Key Deletion

According to the dialog, insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then press “Enter” to confirm user deletion, otherwise press “Cancel” to cancel. If the key is deleted successfully, the following dialog will show up.

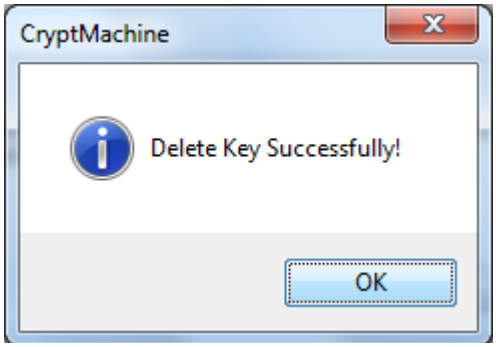

Figure 45 Key Deletion Succeeds

4. 2. 1. 7. Key Backup and Restore Management

Press “BackupKey” button, the following Backup Management UI will show up.

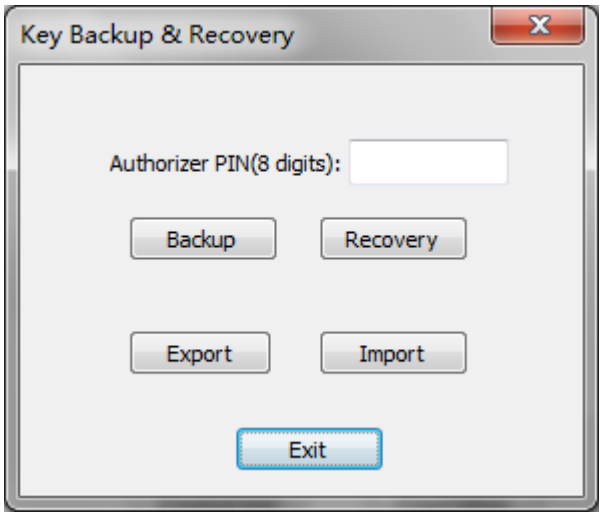

Figure 46 Key Backup and Recovery UI

4.2.1.7.1. Backup Key File

In Backup Management UI (Figure 46), insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then press “Backup” button. The following UI (**Figure 47**) will show up, which

allows to select the file path to store the Key Backup file.

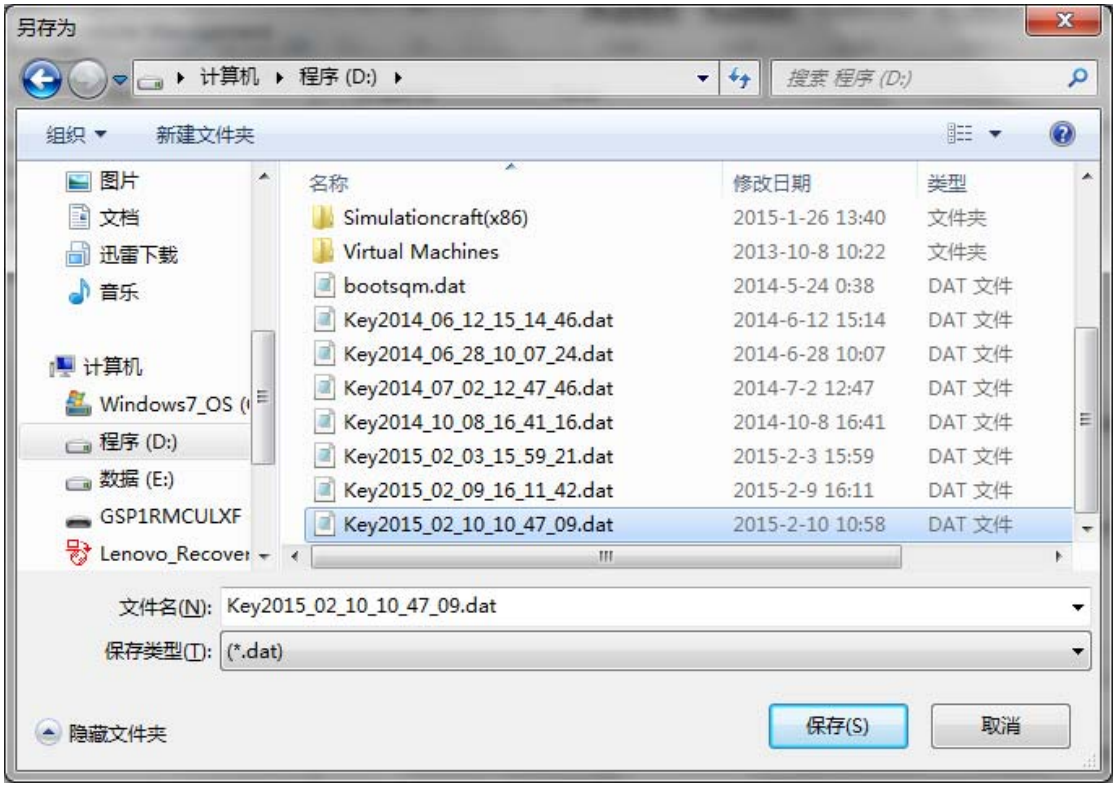

Figure 47 Select File Path to Store Key Backup File

After selecting the file path,the RMA will try to backup Key File. If succeeds, the following dialog will show up, which indicates the Key file has been backed up successfully and showsthe location of the key backup file.

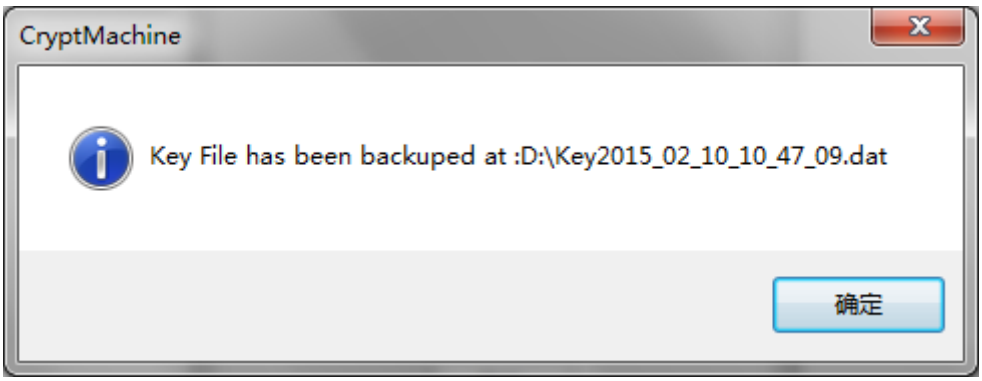

Figure 48 Key FileBackup Succeeds

4.2.1.7.2. Restore Key File

In Backup Management UI (Figure 46), insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then press “Restore” button. If authenticating Authorizer successfully, the following dialog (**Figure 49**) will show up, where you can choose the Key-backup file location.

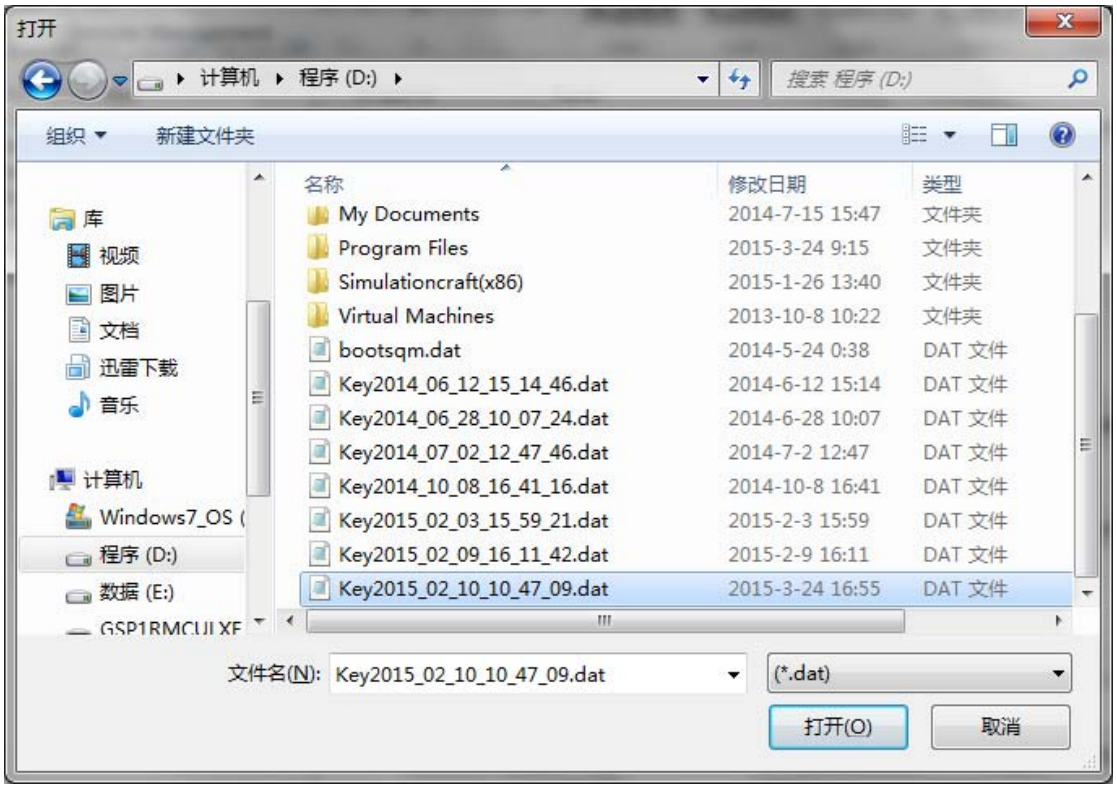

Figure 49 Key File Restoration

After choosing Key-backup file, press “Open”. If restoring Key File successfully, the following dialog will show up and indicate Key File is restored successfully.

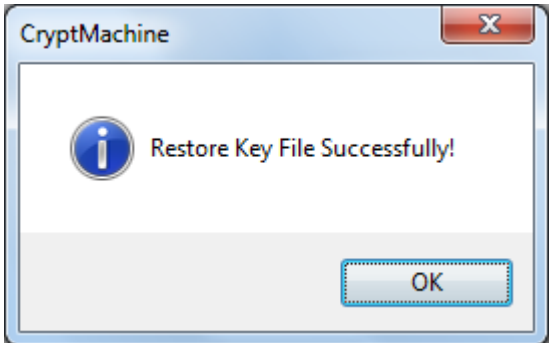

Figure 50 Key Restoration Succeeds

4.2.1.7.3 Export and Import Key File

Key Export and Import function allow backing up Key File from one HSM to the others.

For convenience，we take the source HSM as A, the destination HSM as B.

1. Connect HSM A using Remote Management Application. In Backup Management UI shown in **Figure 46**, insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then press “Export” button.

   If authenticating Authorizer successfully, Key Export Guide UI will show up (**Figure 51**).Before exporting key, 3 Key Managers of HSM A should be authenticated first. According to the message, insert 3 A’s Key Manager smart cards into Smart Card Reader(Key)in order, and input the smart card PIN (**Figure 51 - Figure 53**).

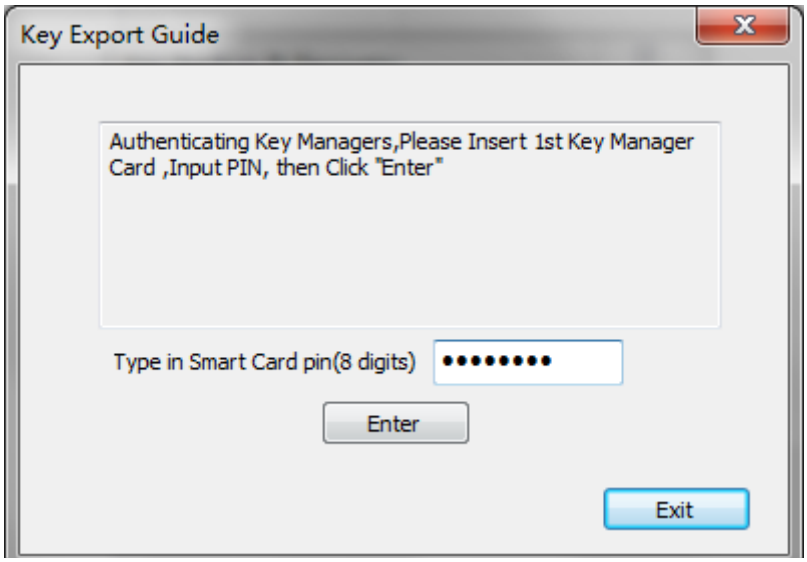

Figure 51 Key Export Guide – 1-1

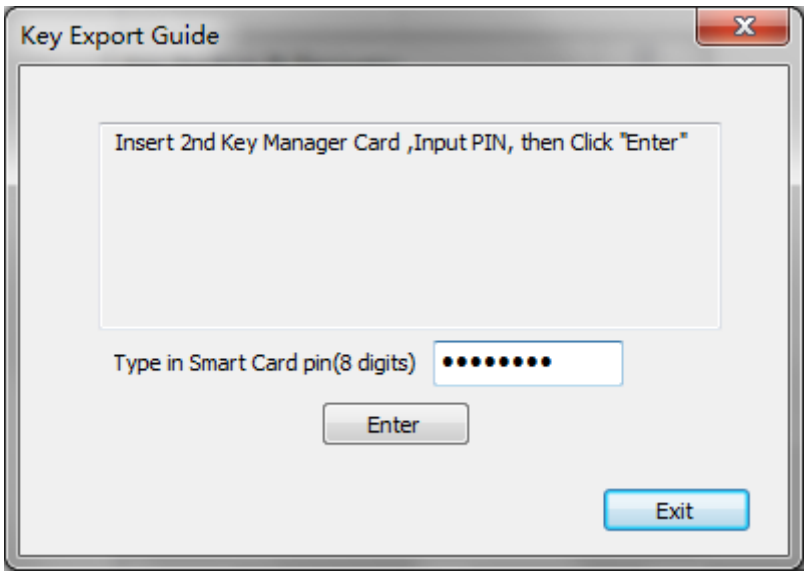

Figure 52 Key Export Guide – 1-2

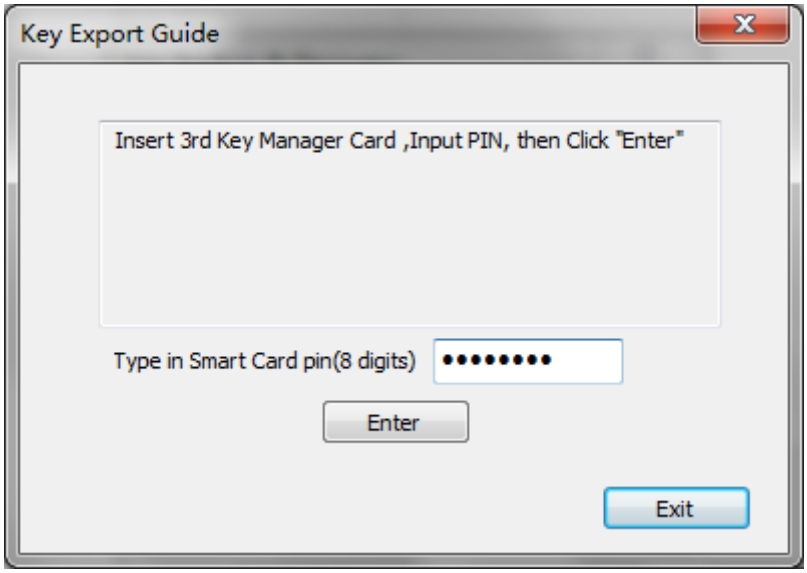

Figure 53 Key Export Guide – 1-3

After authenticating HSM A's 3 Key Manager, you need to import B's Master Key to trans-encrypt A's Key File. According to the message, insert 3 B's Key Manager smart cards into Smart Card Reader (Key) in order and input the smart card PIN (**Figure 54 - Figure 56**).

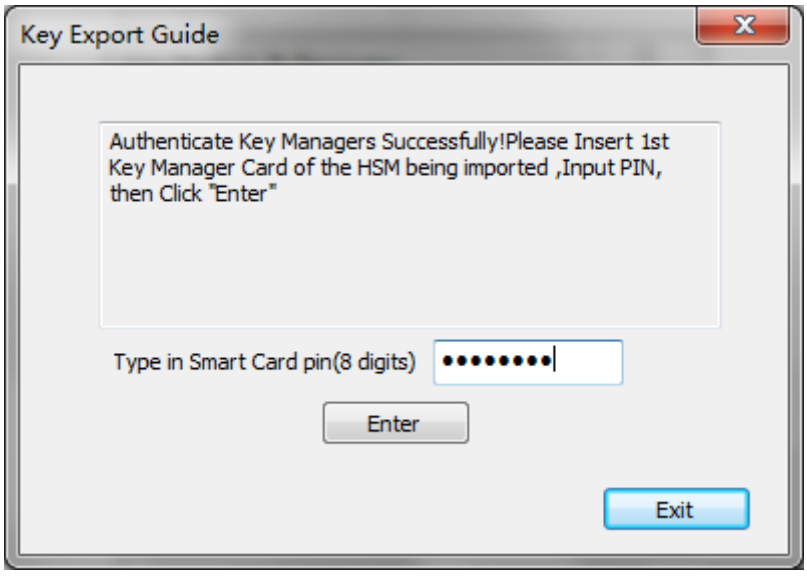

Figure 54 Key Export Guide – 2-1

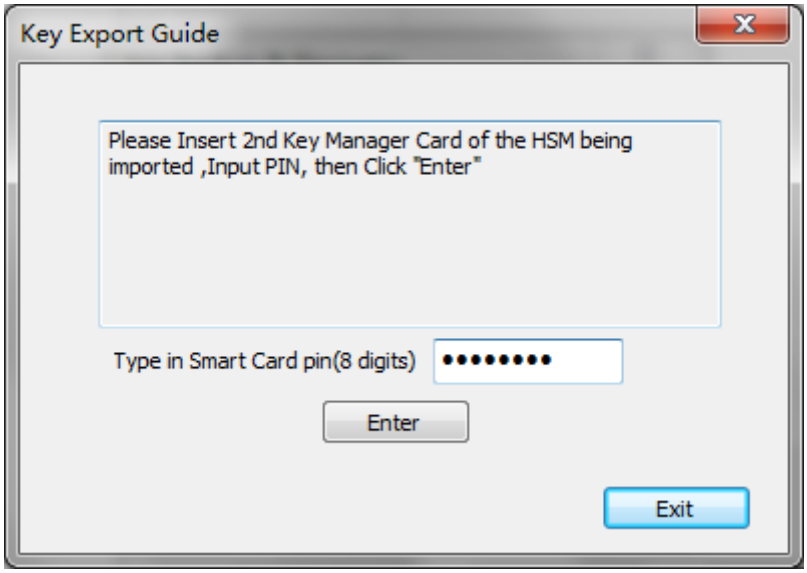

Figure 55 Key Export Guide – 2-2

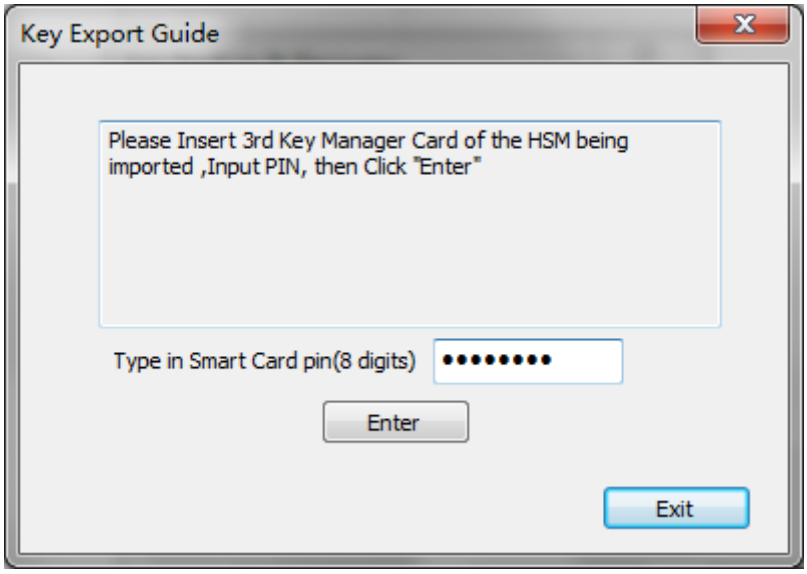

Figure 56 Key Export Guide – 2-3

Then the following UI (**Figure 57**) will show up, which allows to select the file path to store the Key Backup file.

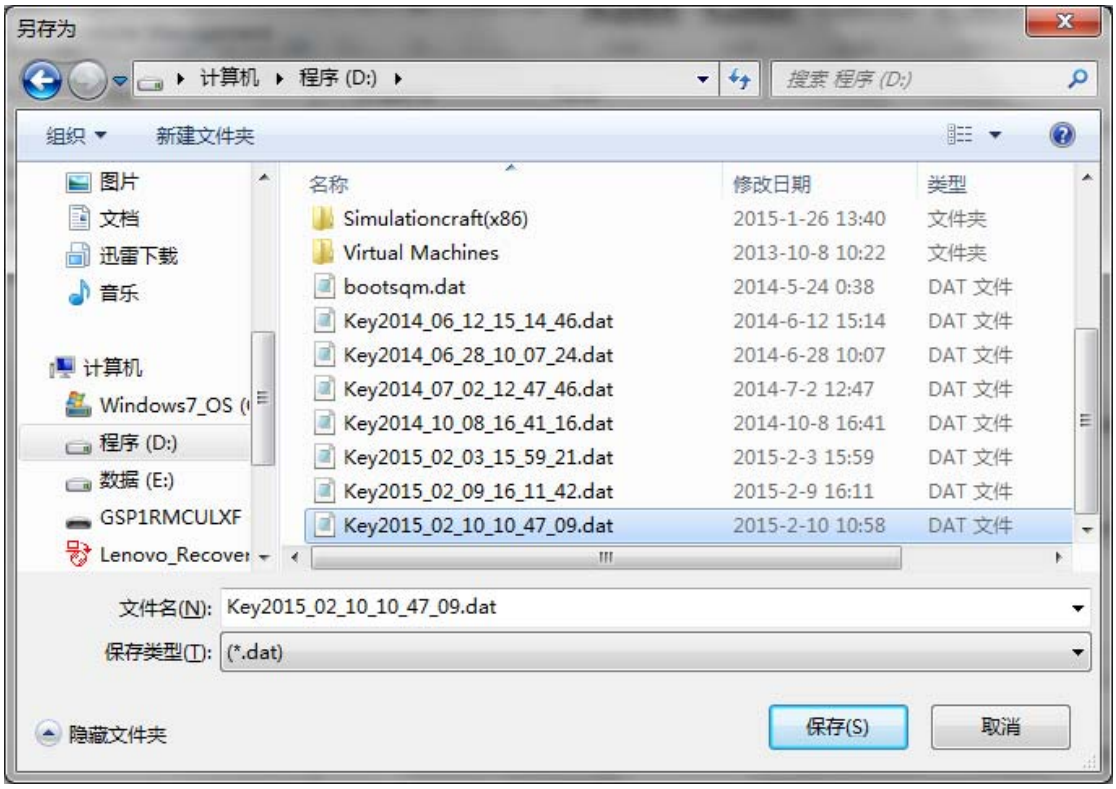

Figure 57 Select File Path to Store Key Backup File

If exporting Key File successfully, the following dialog will show up, which indicates the Key file has been backed up successfully and shows the location of the key backup file.

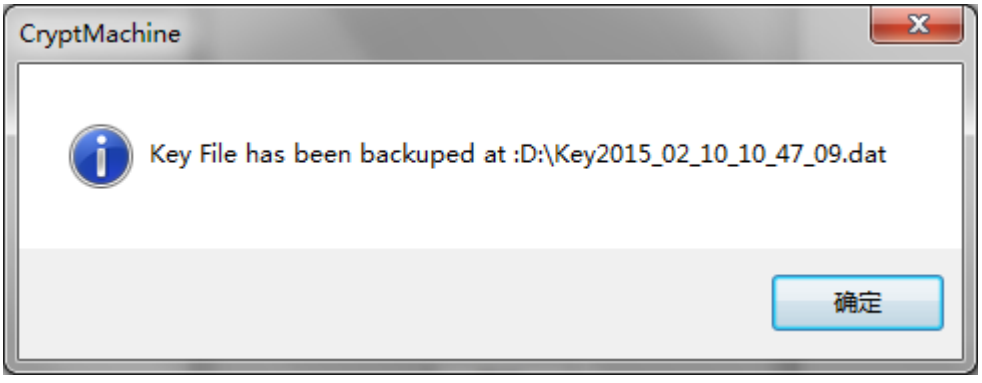

Figure 58 Key File Export Succeeds

2. Launch HSM B, connect it with Remote Management Application and enter Backup Management UI shown in **Figure 46**.Then insert

Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then press “Import” button. If authenticating Authorizer successfully, the following dialog will show up, where you can choose the location of the Key-backup File.

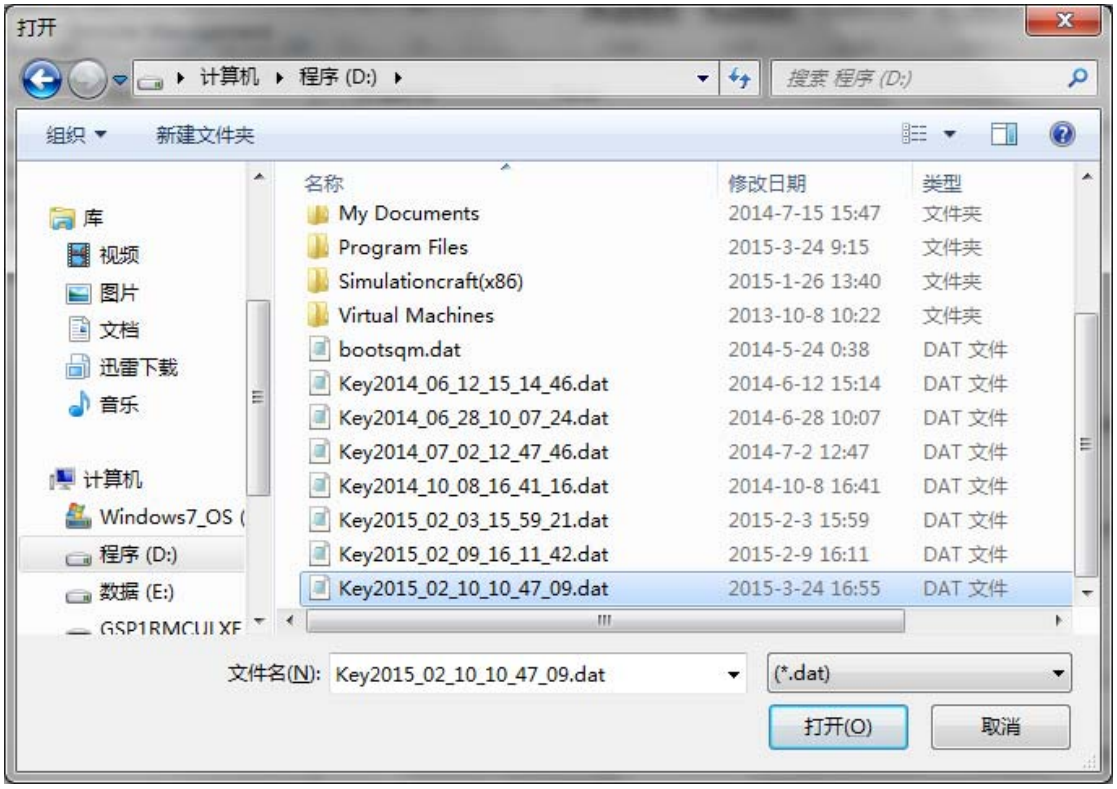

Figure 59 Location of Key Backup File

Choose Key-backup File, then press “Open”, if importing key file successfully, the dialog below will show up.

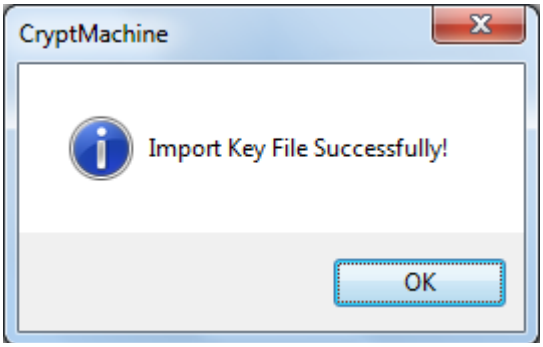

Figure 60 Key File Import Succeeds

In this way, the Key file has been successfully imported from HSM A to B.

4. 2. 1. 8. Zeroize All Users and Keys

In Device Manager UI (**Figure 29**), press “Delete All” button, and the following dialog will show up.

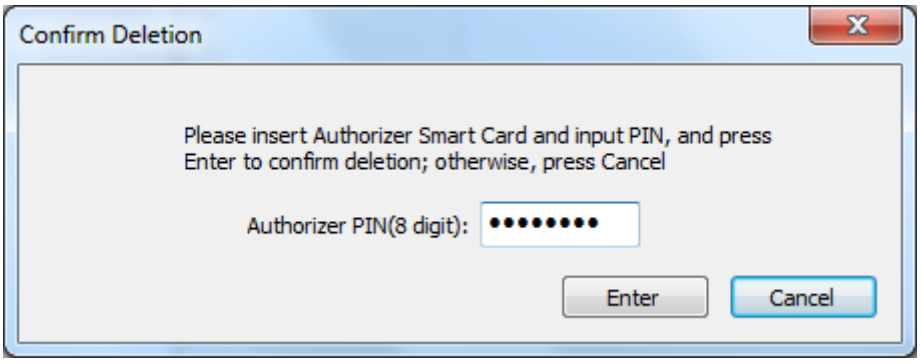

Figure 61 User and Key Zeroization

According to the dialog (**Figure 61**), insert Authorizer smart card into Smart Card Reader (ID) and input the corresponding PIN, then press “ENTER” to confirm user deletion, otherwise press “CANCEL”. If all keys are deleted successfully, the following dialog will show up.

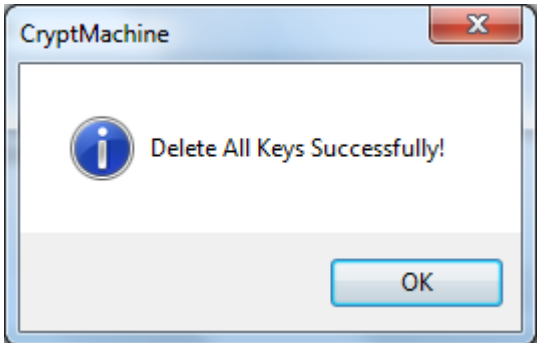

Figure 62 User and Key Zeroization

4.2.2. IP Address Action Table Configuration

IP Address Action Table is used for allowing HSM to provide cryptographic service only for the specific IP address range. Any host whose IP address is not in IP Address Action Table is unable to access HSM for cryptographic service.

In Device Manager UI(**Figure 29**), press “SystemConfig” button, System Configuration UI will show up.

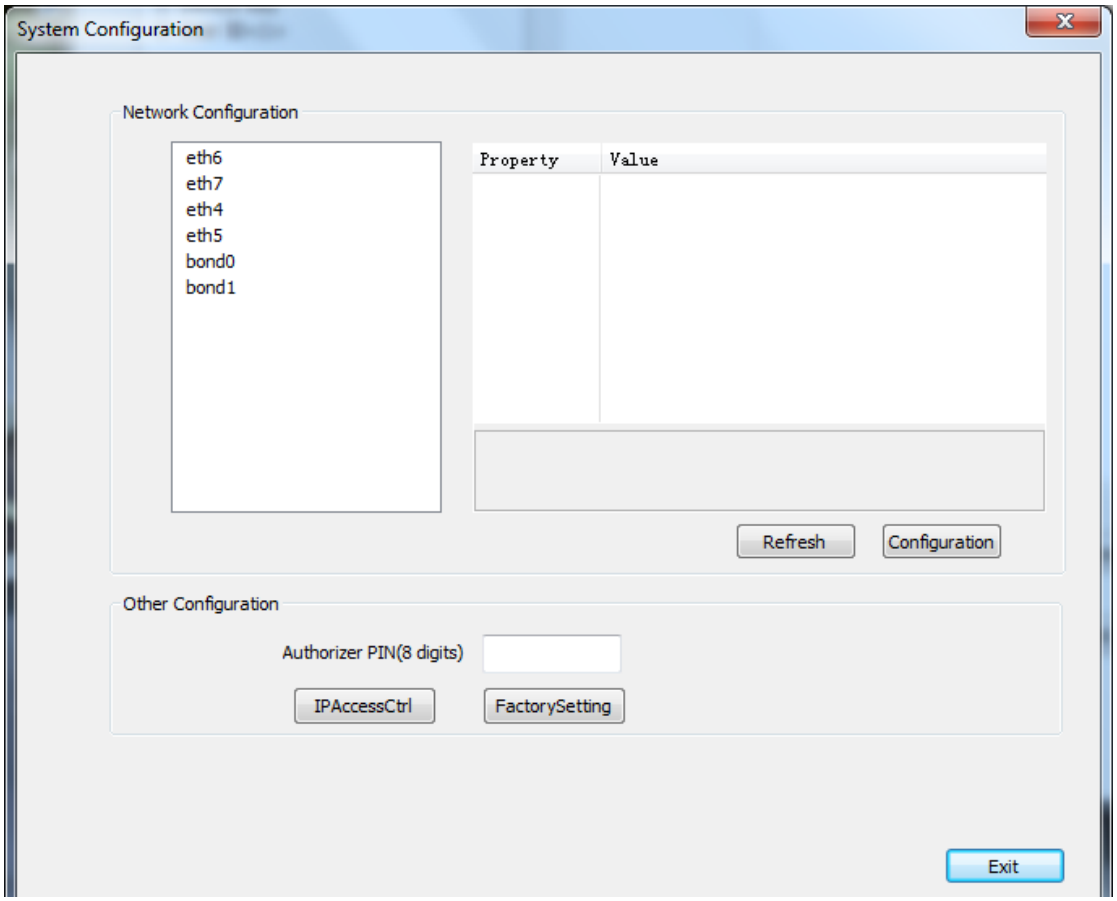

Figure 63 System Configuration

Press “IPAccessCtrl” button, the following UI will show up. You can determine the range of IP address that can access the HSM for cryptographic service.

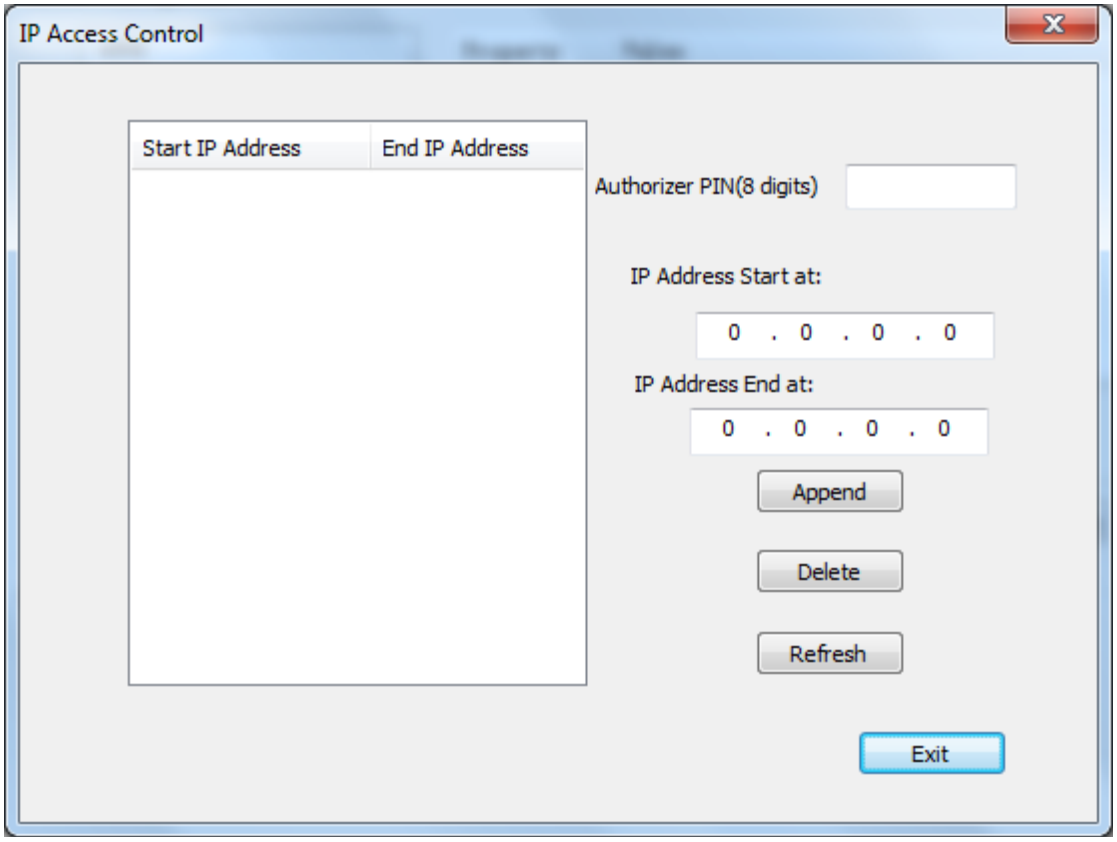

Figure 64 IP Access Control UI

4. 2. 2. 1. Append IP Address Action Table Entry

In IP Access Control UI (**Figure 64**), Input the start IP and end IP of IP Address Action Table entry.

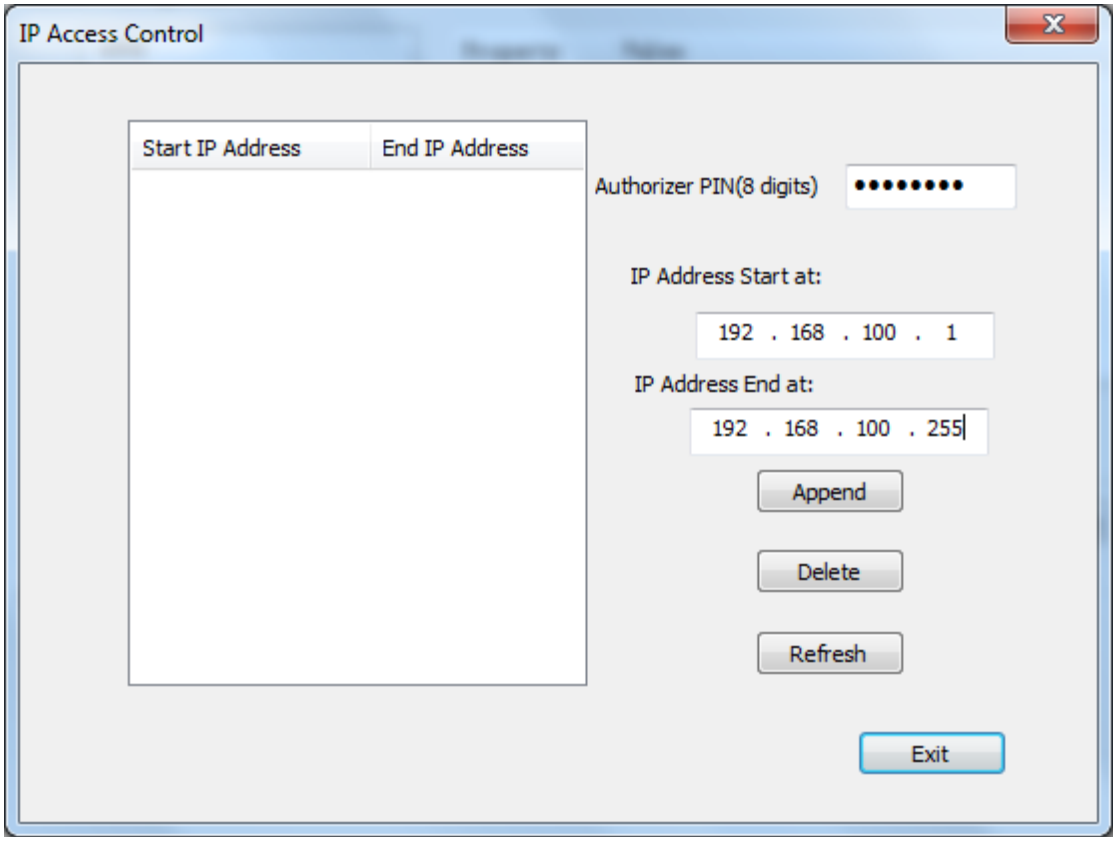

Figure 65Add IP Access Control

Then press “Append”. If appending IP Access Action Table entry successfully, the following dialog will show up, which showsIP Access Action Table Entry is appended successfully.

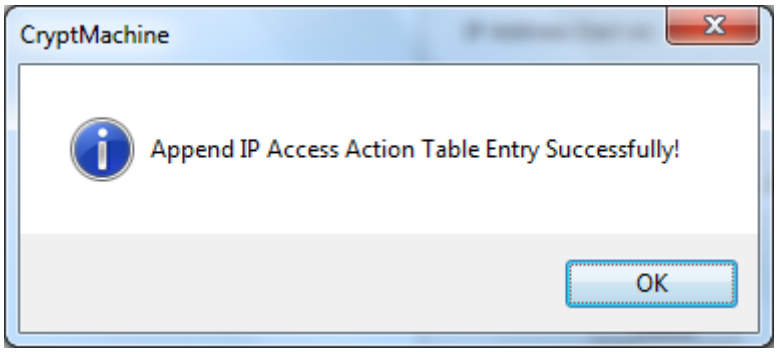

Figure 66Add IP Address Action Table Entry Succeeds

4. 2. 2. 2. Delete IP Address Action Table Entry

Pick the target IP Address Action Table Entry, then press “Delete” button.

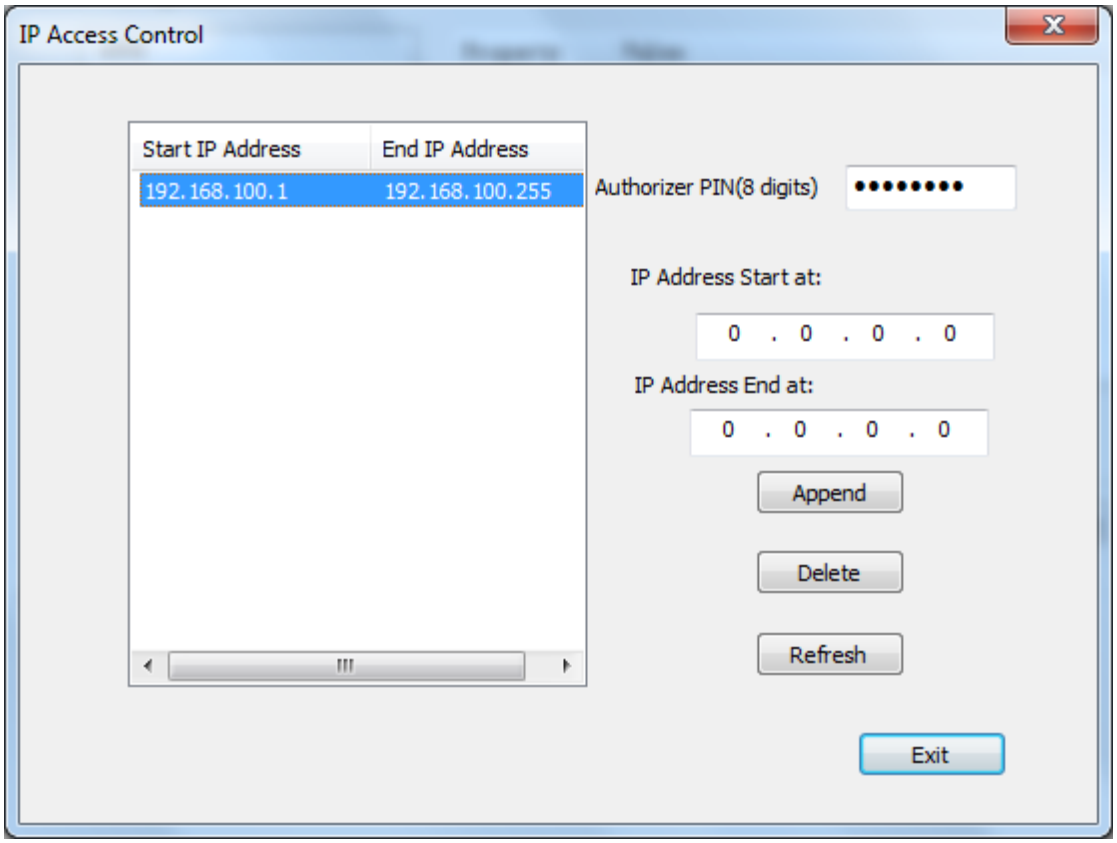

Figure 67 IP Address Deletion

If deleting IP Access Action Table entry successfully, the following dialog will show up, which indicates IP Access Action Table entry is deleted successfully.

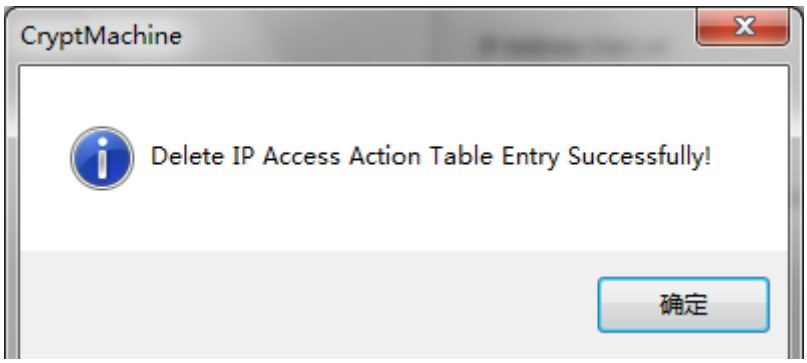

Figure 68 IP Address Deletion Succeeds

4.2.3. Network Configuration

In Device Manager UI(**Figure 69**), press “SystemConfig” button, System Configuration UI will show up.

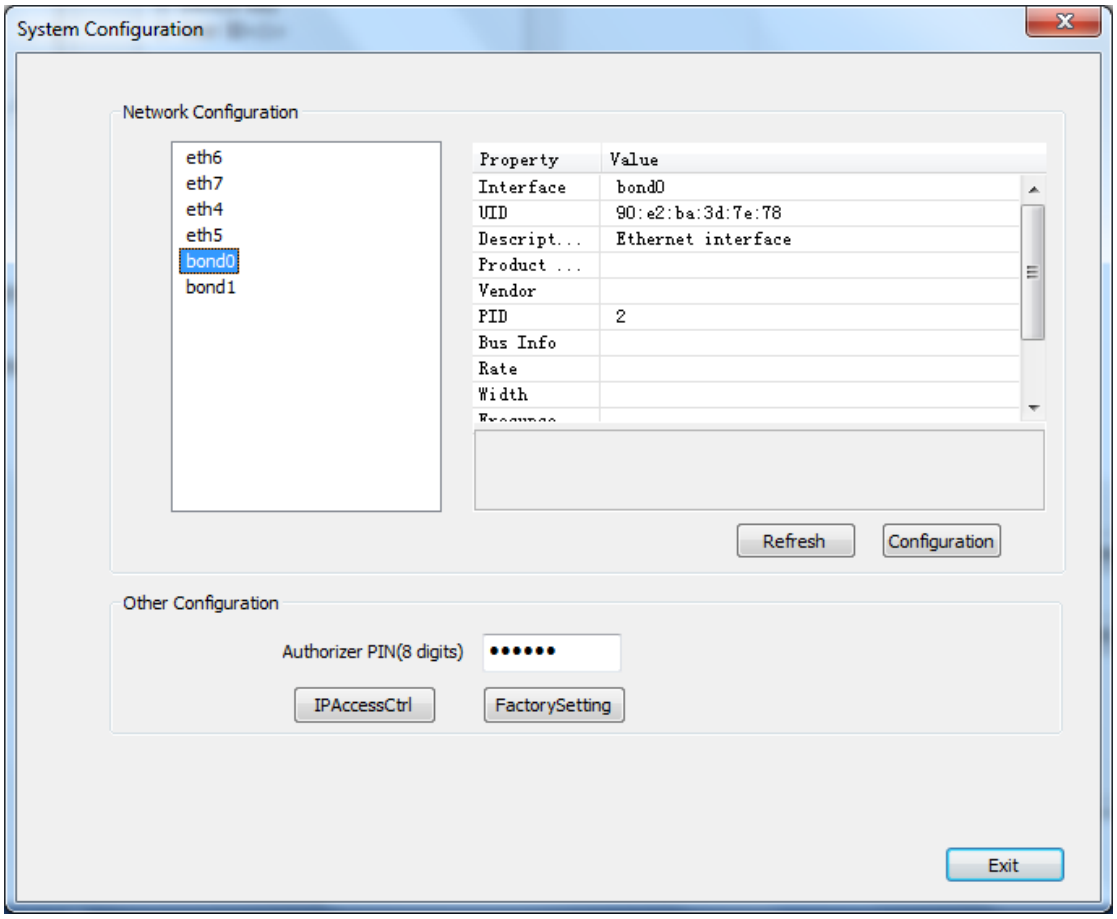

Figure 69 System Configuration UI

Pick the target Network Interface, then press “Configuration” button. The following UI will show up.

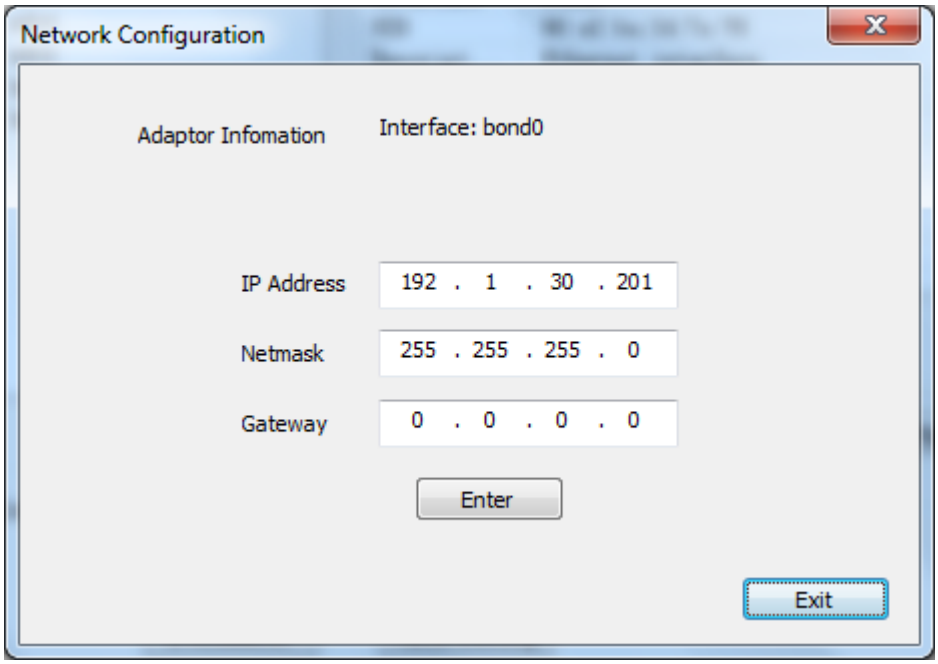

Figure 70 Network Configuration

Input IP address, net mask and default gateway then press “Enter”.

4. 3. Auditor User Interface

In Crypto Officer Logon User Interface, choose “Auditor” as Crypto Officer Type, then insert Auditor smart card, input its PIN and the Management Port IP address, press “Logon”.

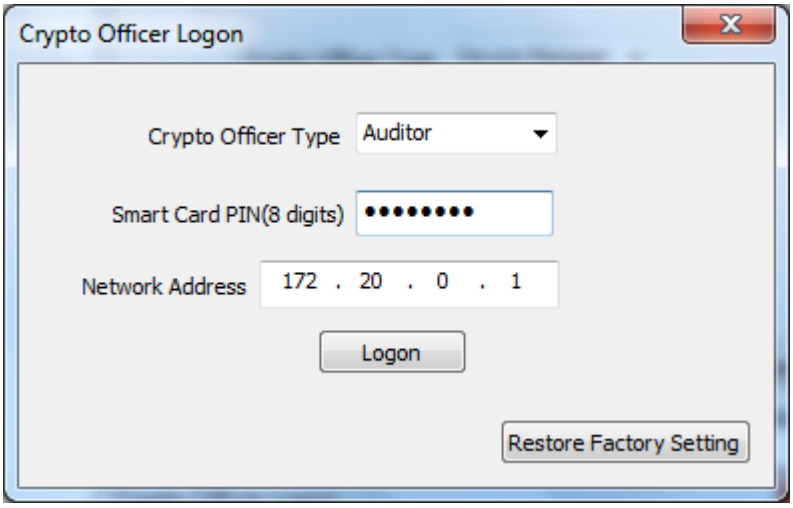

Figure 71 Crypto Officer Logon User Interface

After authenticating Auditor's identity, UI will turn to Auditor UI.

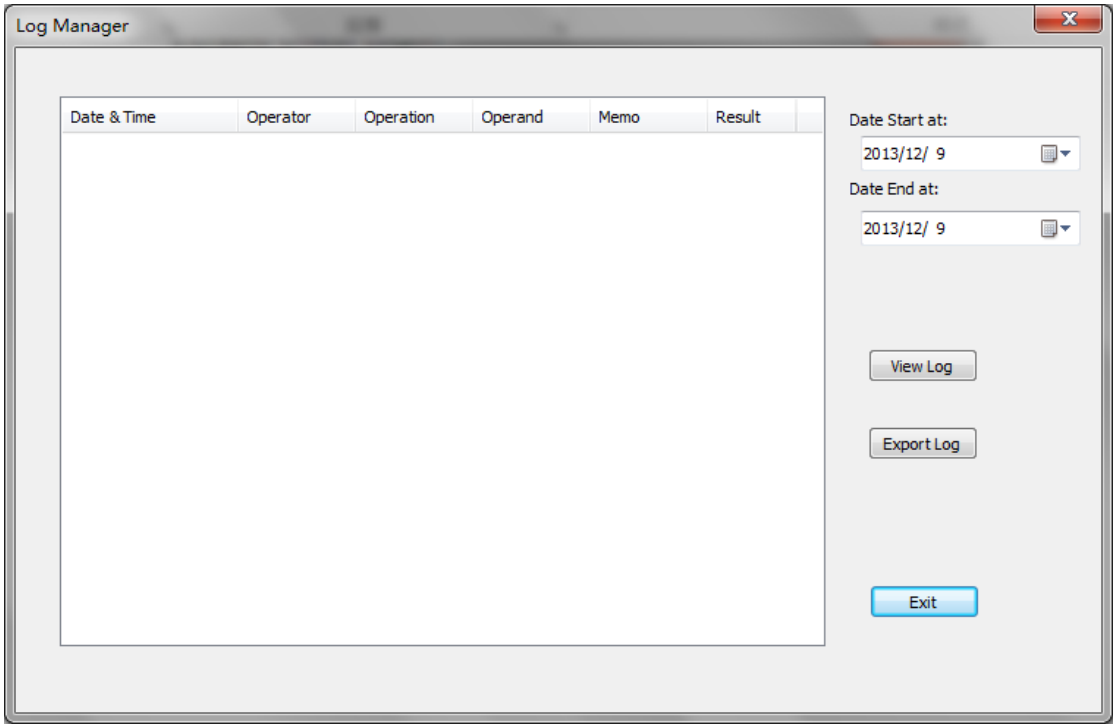

Figure 72 Auditor UI

4.3.1. View Audit Log

Choose start date and end date of the Audit Log that Auditor want to view,

then press “View Log”.

If viewing Audit Log successfully, the Audit Log records will show as below.

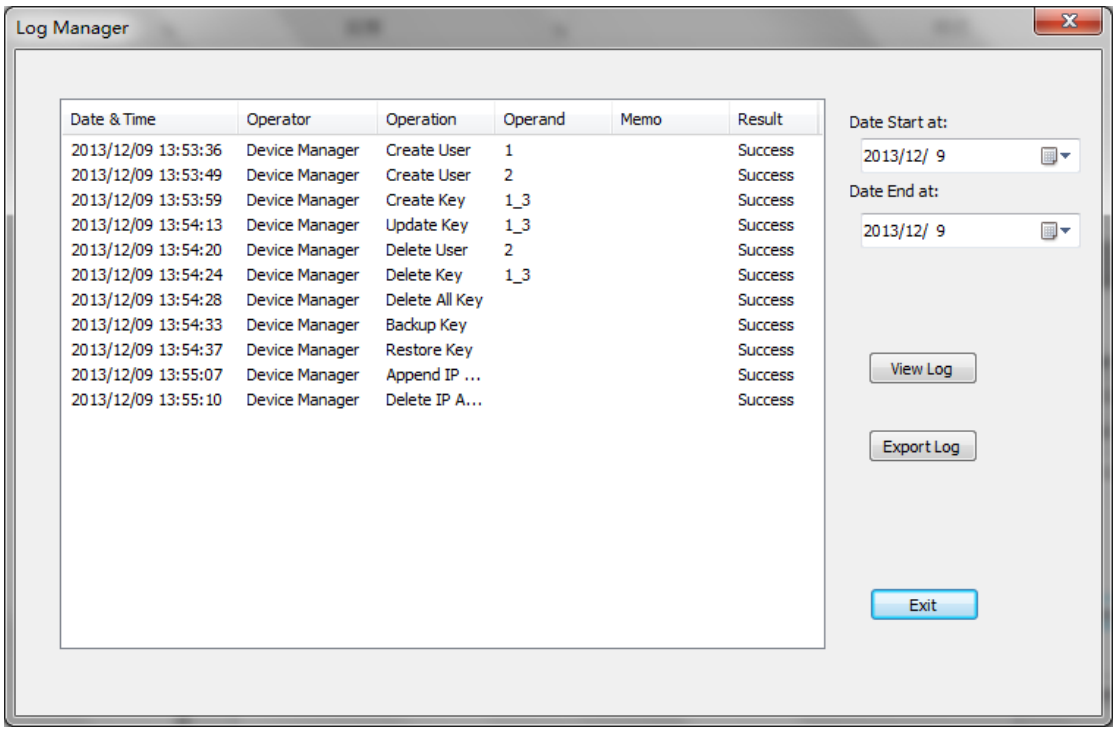

Figure 73 Audit Log

4.3.2. Export Audit Log

Choose start date and end date of the Audit Log that Auditor want to export, then press “Export Log”.

If exporting Audit Log successfully, the following dialog will show up, which indicates that Audit Log is exported successfully.

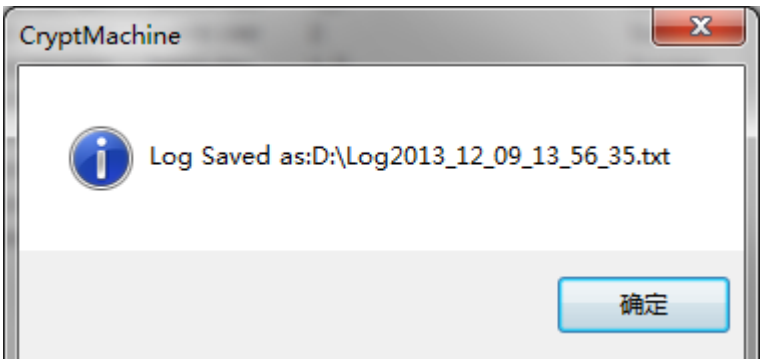

Figure 74 Export Audit Log

The format of Audit Log exported shows as below.

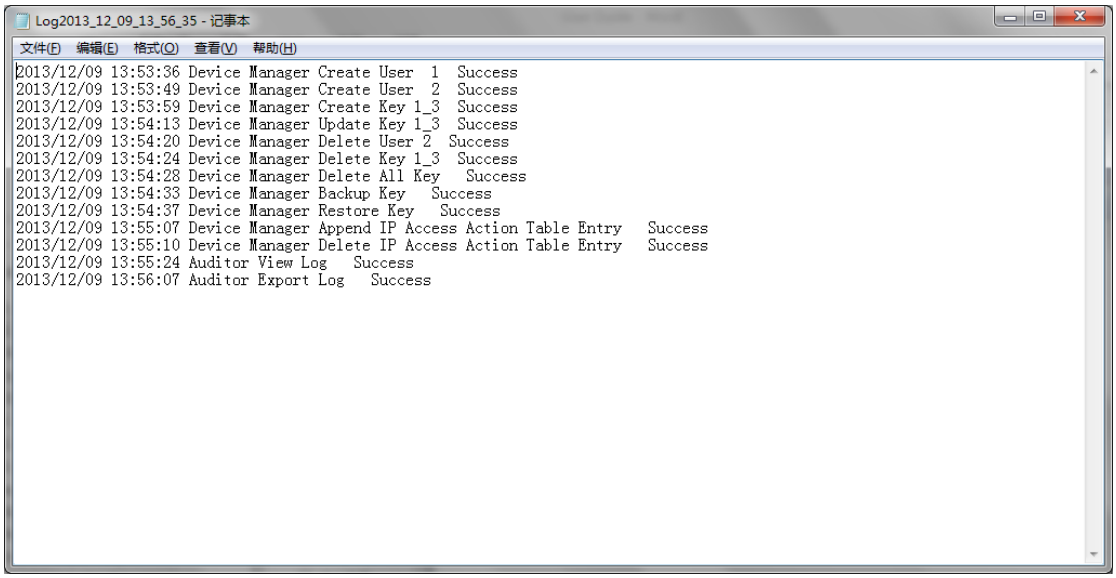

Figure 75 Audit Log

5. Client Guide

When the user account and key are created in HSM-ZJ2014, users can access cryptographic service through software librarywhich is developed for the HSM and compiled according to PKCS#11 standard.

5.1. Supported Operating Systems

We provide 2 versions of the library, which can run respectively on Linux 2.6.0 or later and Windows Vista/7/8.

5.2. Supported PKCS#11 Function

5.2.1 Library Initialization

Function Prototype	CK_RVC_Initialize(CK_VOID_PTR pInitArgs);	
Description	Initial PKCS#11library	
Parameter	pInitArgs	Can only be NULL
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.2 Library Finalization

Function Prototype	CK_RVC_Finalize(CK_VOID_PTR pReserved);	
Description	Clean up application space	
Parameter	pReserved	Reserved, Can only be NULL
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.3 Get Library Information

Function Prototype	CK_RVC_GetInfo(CK_INFO_PTR pInfo);	
Description	Get Library Information	
Parameter	pInfo	Point to Library Information

Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_INFO info;
	CK_RV rv;
	rv = C_Initialize((CK_VOID_PTR) NULL);
	assert(rv == CKR_OK);
	rv = C_GetInfo(&info);
	assert(rv == CKR_OK);
	if (info.version.major == 2) {
	/* Do lots of interesting cryptographic things with the token */
	.
	.
	}
	rv = C_Finalize(NULL_PTR);
	assert(rv == CKR_OK);	

5.2.4 Get Function List

Function Prototype	CK_RVC_GetFunctionList(CK_FUNCTION_LIST_PTR_PTR ppFunctionList);	
Description	Get Library Information	
Parameter	ppFunctionList	Pointer to Pointer to Library List
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_FUNCTION_LIST_PTR pFunctionList;
	CK_C_Initialize pC_Initialize;
	CK_RV rv;
	/* It's OK to call C_GetFunctionList before calling C_Initialize */
	rv = C_GetFunctionList(&pFunctionList);
	assert(rv == CKR_OK);	

	pC_Initialize = pFunctionList->C_Initialize;
	/* Call the C_Initialize function in the library */
	rv = (*pC_Initialize) (NULL_PTR);

5.2.5 Get Slot List

Function Prototype	CK_RVC_GetSlotList(CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount);	
Description	Get slot. If pSlotList is NULL, pulCount returns space size that slot list consumes.	
Parameter	tokenPresent	Boolean value, return TRUE, If slot List only contains present token, otherwise, return FALSE.
	pSlotList	Pointer to Slot List
	pulCount	Pointer to Number of Slots
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_ULONG ulSlotCount, ulSlotWithTokenCount;
	CK_SLOT_ID_PTR pSlotList, pSlotWithTokenList;
	CK_RV rv;
	/* Get list of all slots */
	rv = C_GetSlotList(CK_FALSE, NULL_PTR, &ulSlotCount);
	if (rv == CKR_OK) {
	pSlotList = (CK_SLOT_ID_PTR) malloc(ulSlotCount * sizeof(CK_SLOT_ID));
	rv = C_GetSlotList(CK_FALSE, pSlotList, &ulSlotCount);
	if (rv == CKR_OK) {
	/* Now use that list of all slots */
	..}
	free(pSlotList);
	}	

```
/* Get list of all slots with a token present */
pSlotWithTokenList = (CK_SLOT_ID_PTR) malloc(0);
ulSlotWithTokenCount = 0;
while (1) {
        rv = C_GetSlotList(CK_TRUE, pSlotWithTokenList,
ulSlotWithTokenCount);
if (rv != CKR_BUFFER_TOO_SMALL)
break;
        pSlotWithTokenList = realloc(pSlotWithTokenList,
                                                ulSlotWithTokenList *
sizeof(CK_SLOT_ID));
}

if (rv == CKR_OK) {
/* Now use that list of all slots with a token present */
.
.
}
free(pSlotWithTokenList);
```

5.2.6 Get Slot Information

Function Prototype	CK_RVC_GetSlotInfo(CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo);	
Description	Get Slot Information	
Parameter	slotID	Slot ID
	pInfo	Pointer to Slot Information
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.
Example	CK_ULONG ulCount;	

```
CK_SLOT_ID_PTR pSlotList;
CK_SLOT_INFO slotInfo;
CK_TOKEN_INFO tokenInfo;
CK_RV rv;
rv = C_GetSlotList(CK_FALSE, NULL_PTR, &ulCount);
if ((rv == CKR_OK) && (ulCount > 0)) {
    pSlotList = (CK_SLOT_ID_PTR) malloc(ulCount * sizeof(CK_SLOT_ID));
    rv = C_GetSlotList(CK_FALSE, pSlotList, &ulCount);
    assert(rv == CKR_OK);
/* Get slot information for first slot */
    rv = C_GetSlotInfo(pSlotList[0], &slotInfo);
    assert(rv == CKR_OK);
    ..free(pSlotList);
}
```

5.2.7 Get Mechanism List

Function Prototype	CK_RVC_GetMechanismList(CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount);	
Description	Get Mechanism List	
Parameter	slotID	Slot ID
	pMechanismList	Pointer to Mechanism List
	pulCount	Pointer to Number of the Mechanisms in the List
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.
Example	CK_SLOT_ID slotID; CK_ULONG ulCount; CK_MECHANISM_TYPE_PTR pMechanismList;	

	CK_RV rv;
	.
	.
	rv = C_GetMechanismList(slotID, NULL_PTR, &ulCount);
	if ((rv == CKR_OK) && (ulCount > 0)) {
	pMechanismList = (CK_MECHANISM_TYPE_PTR)
	malloc(ulCount * sizeof(CK_MECHANISM_TYPE));
	rv = C_GetMechanismList(slotID, pMechanismList, &ulCount);
	if (rv == CKR_OK) {
	..}
	free(pMechanismList);
	}

5.2.8 Get Mechanism Information

Function Prototype	CK_RVC_GetMechanismInfo(CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_INFO_PTR pInfo);	
Description	Get Mechanism Information	
Parameter	slotID	Slot ID
	type	Mechanism Type
	pInfo	Pointer toMechanism Information
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_SLOT_ID slotID;
	CK_MECHANISM_INFO info;
	CK_RV rv;
	/* Get information about the CKM_RSA mechanism for this token */
	rv = C_GetMechanismInfo(slotID, CKM_RSA, &info);
	if (rv == CKR_OK) {	

	if (info.flags&CKF_DIGEST) {
	..}
	}

5.2.9 Open Session

Function Prototype	CK_RVC_OpenSession(CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY Notify, CK_SESSION_HANDLE_PTR phSession);	
Description	Open Session	
Parameter	slotID	Slot ID
	type	Session Type
	pApplication	Pointer to Parameters of Callback Function
	Notify	Callback Function
	phSession	Pointer to Session Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.10 Close Session

Function Prototype	CK_RVC_CloseSession(CK_SESSION_HANDLE hSession);	
Description	Close Session	
Parameter	hSession	Session Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.11 Close All Sessions

Function Prototype	CK_RVC_CloseAllSessions(CK_SLOT_ID slotID);

Description	Close All Sessions	
Parameter	slotID	slotID
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_SLOT_ID slotID;
	CK_BYTE application;
	CK_NOTIFY MyNotify;
	CK_SESSION_HANDLE hSession;
	CK_RV rv;
	.
	.
	application = 17;
	MyNotify = &EncryptionSessionCallback;
	rv = C_OpenSession(slotID, CKF_SERIAL_SESSION | CKF_RW_SESSION,
	(CK_VOID_PTR) & application, MyNotify, &hSession);
	if (rv == CKR_OK) {
	..C_CloseSession(hSession);
	}
	rv = C_CloseAllSessions(slotID);	

5.2.12 Login

Function Prototype	CK_RVC_Login(CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen);	
Description	Login	
Parameter	hSession	Session Handle
	userType	User Type, supports only Type USER.
	pPin	Pointer to address that stores PIN, PIN is make up of

		8 or less digits.
	ulPinLen	Length of PIN
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.13 Logout

Function Prototype	CK_RVC_Logout(CK_SESSION_HANDLEhSession);	
Description	Close Session	
Parameter	hSession	Session Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_SESSION_HANDLE hSession;
	CK_UTF8CHAR userPIN[] = { "MyPIN" };
	
	CK_RV rv;
	rv = C_Login(hSession, CKU_USER, userPIN, sizeof(userPIN) - 1);
	if (rv == CKR_OK) {
	..rv == C_Logout(hSession);
	if (rv == CKR_OK) {
	..}
	}	

5.2.14 Create Object

Function Prototype	CK_RVC_CreateObject(CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTRphObject);
Description	Create Object

<table>
<tr><td rowspan="4">Parameter</td><td>hSession</td><td>Session Handle</td></tr>
<tr><td>pTemplate</td><td>Pointer to Template</td></tr>
<tr><td>ulCount</td><td>The number of templates</td></tr>
<tr><td>phObject</td><td>Pointer to Object Handle</td></tr>
<tr><td rowspan="2">Return Value</td><td>CKR_OK</td><td>Success</td></tr>
<tr><td>Non-CKR_OK</td><td>Fail, return error code.</td></tr>
<tr><td>Example</td><td colspan="2">CK_SESSION_HANDLE hSession;
CK_OBJECT_HANDLE hKey;
CK_OBJECT_CLASS
keyClass = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_BBOOLtrue = CK_TRUE;
CK_ATTRIBUTE keyTemplate[] = {
{CKA_CLASS, &keyClass, sizeof(keyClass)}
,
{CKA_KEY_TYPE, &keyType, sizeof(keyType)}
,
{CKA_POINT, point, sizeof(point)}
};

CK_RV rv;
.
.
/* Create an RSA public key object */
rv = C_CreateObject(hSession, &keyTemplate, 3, &hKey);
if (rv == CKR_OK) {
.
.
}</td></tr>
</table>

5.2.15 Destroy Object

Function Prototype	CK_RVC_DestroyObject(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject);	
Description	Destroy Object	
Parameter	hSession	Session Handle
	hObject	Object Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.16 Get Attribute Value

Function Prototype	CK_RVC_GetAttributeValue(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount);	
Description	Get Attribute Value	
Parameter	hSession	Session Handle
	hObject	Object Handle
	pTemplate	Pointer to Template List
	ulCount	Number of Templates
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.
Example	CK_SESSION_HANDLE hSession; CK_OBJECT_HANDLE hObject; CK_BYTE point; CK_ATTRIBUTEtemplate[] = { {CKA_POINT, NULL_PTR, 0}	

```
};


CK_RV rv;
.
.
rv = C_GetAttributeValue(hSession, hObject, &template, 1);
if (rv == CKR_OK) {
        point = (CK_BYTE_PTR) malloc(template[0].ulValueLen);
template[0].pValue = point;
/* template[0].ulValueLen was set by C_GetAttributeValue */


        rv = C_GetAttributeValue(hSession, hObject, &template, 1);
if (rv == CKR_OK) {
        ..}
        free(point);
}
```

5.2.17 Set Attribute Value

Function Prototype	CK_RVC_SetAttributeValue(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount);	
Description	Set	
Parameter	hSession	Session Handle
	hObject	Object Handle
	pTemplate	Pointer to Template List
	ulCount	Number of Templates

Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_SESSION_HANDLE hSession;
	CK_OBJECT_HANDLE hObject;
	CK_UTF8CHAR label[] = { "New label" };
	
	CK_ATTRIBUTEtemplate[] = {
	CKA_LABEL, label, sizeof(label) - 1
	};
	
	CK_RV rv;
	.
	.
	rv = C_SetAttributeValue(hSession, hObject, &template, 1);
	if (rv == CKR_OK) {
	.
	.
	}	

5.2.18 Find Objects Initialization

Function Prototype	CK_RVC_FindObjectsInit(CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount);	
Description	Find Objects Initialization	
Parameter	hSession	Session Handle
	pTemplate	Pointer to Template List
	ulCount	Number of Templates
Return	CKR_OK	Success

Value	Non-CKR_OK	Fail, return error code.

5.2.19 Find Objects

Function Prototype	CK_RVC_FindObjects(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject, CK_ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount);	
Description	Find Object	
Parameter	hSession	Session Handle
	phObject	Pointer to Object Handle
	ulMaxObjectCount	Max number of Objects returned
	pulObjectCount	Pointer to Actual number of Objects returned
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.20 Find Objects Finalization

Function Prototype	CK_RVC_FindObjectsFinal(CK_SESSION_HANDLE hSession);	
Description	Finalize Find Object operation	
Parameter	hSession	Session Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_ATTRIBUTE priKeysTemplate[] = {
	{CKA_CLASS, &priKeys, sizeof(priKeys)},
	};
	rc =FunctionPtr->C_FindObjectsInit(hSession, priKeysTemplate, 1);
	if (rc != CKR_OK) {
	printf("error FindObjectsInit\n");	

	}
	rc = FunctionPtr->C_FindObjects(hSession, &priKeys_h, 1, &ulObjectCount);
	rc = FunctionPtr->C_FindObjectsFinal(hSession);

5.2.21 Encryption Initialization

Function Prototype	CK_RVC_EncryptInit(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey);	
Description	Initialize Encryption	
Parameter	hSession	Session Handle
	pMechanism	Pointer to Mechanism
	hKey	Key Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.22 Encrypt

Function Prototype	CK_RVC_Encrypt(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pEncryptedData, CK_ULONG_PTR pulEncryptedDataLen);	
Description	Encrypt	
Parameter	hSession	Session Handle
	pData	Pointer to Plain Data
	ulDataLen	Length of Plain Data
	pEncryptedData	Pointer to Cipher Data Buffer
	pulEncryptedDataLen	Pointer to Length of Cipher Data Buffer
Return	CKR_OK	Success

Value	Non-CKR_OK	Fail, return error code.

5.2.23 Encrypt Update

Function Prototype	CK_RVC_EncryptUpdate(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen, CK_BYTE_PTR pEncryptedPart, CK_ULONG_PTR pulEncryptedPartLen);	
Description	Encrypt Update	
Parameter	hSession	Session Handle
	pPart	Pointer to AdditionalPlain Data
	ulPartLen	Length of AdditionalPlain Data
	pEncryptedPart	Pointer to Cipher Data Buffer
	pulEncryptedPartLen	Pointer to Length of Cipher Data Buffer
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.24 Encryption Finalization

Function Prototype	CK_RVC_EncryptFinal(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pLastEncryptedPart, CK_ULONG_PTR pulLastEncryptedPartLen);	
Description	Finalize Encryption	
Parameter	hSession	Session Handle
	pLastEncryptedPart	Pointer to Last Cipher Data
	pulLastEncryptedPartLen	Pointer to Length of Last Cipher Data
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example

```
#define PLAINTEXT_BUF_SZ 200
#define CIPHERTEXT_BUF_SZ 256
CK_ULONG firstPieceLen, secondPieceLen;
CK_SESSION_HANDLE hSession;
CK_OBJECT_HANDLE hKey;
CK_BYTE iv[8];
CK_MECHANISM mechanism = {
        CKM_AES_ECB, iv, sizeof(iv)
};

CK_BYTE data[PLAINTEXT_BUF_SZ];
CK_BYTE encryptedData[CIPHERTEXT_BUF_SZ];
CK_ULONG ulEncryptedData1Len;
CK_ULONG ulEncryptedData2Len;
CK_ULONG ulEncryptedData3Len;
CK_RV rv;
.
.
firstPieceLen = 90;
secondPieceLen = PLAINTEXT_BUF_SZ - firstPieceLen;
rv = C_EncryptInit(hSession, &mechanism, hKey);
if (rv == CKR_OK) {
/* Encrypt first piece */
        ulEncryptedData1Len = sizeof(encryptedData);
        rv = C_EncryptUpdate(hSession,
&data[0], firstPieceLen,
&encryptedData[0], &ulEncryptedData1Len);
if (rv != CKR_OK) {
        ..}
```

```
/* Encrypt second piece */
    ulEncryptedData2Len = sizeof(encryptedData) - ulEncryptedData1Len;
    rv = C_EncryptUpdate(hSession,
&data[firstPieceLen], secondPieceLen,
&encryptedData[ulEncryptedData1Len],
&ulEncryptedData2Len);
if (rv != CKR_OK) {
    ..}
/* Get last little encrypted bit */
    ulEncryptedData3Len =
sizeof(encryptedData) - ulEncryptedData1Len - ulEncryptedData2Len;
    rv = C_EncryptFinal(hSession,
&encryptedData[ulEncryptedData1Len +
                                        ulEncryptedData2Len],
&ulEncryptedData3Len);
if (rv != CKR_OK) {
    ..}
}
```

5.2.25 Decryption Initialization

Function Prototype	CK_RVC_DecryptInit(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey);	
Description	Decryption Initialization	
Parameter	hSession	Session Handle
	pMechanism	Pointer to Mechanism
	hKey	Key Handle

Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.26 Decrypt

Function Prototype	CK_RVC_Decrypt(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedData, CK_ULONG ulEncryptedDataLen, CK_BYTE_PTR pData, CK_ULONG_PTR pulDataLen);	
Description	Decrypt	
Parameter	hSession	Session Handle
	pEncryptedData	Pointer to Cipher Data
	ulEncryptedDataLen	Length of Cipher Data
	pData	Pointer to Plain Data Buffer
	pulDataLen	Pointer to Length of Plain Data Buffer
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.27 Decrypt Update

Function Prototype	CK_RVC_DecryptUpdate(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pEncryptedPart, CK_ULONG ulEncryptedPartLen, CK_BYTE_PTR pPart, CK_ULONG_PTR pulPartLen);	
Description	Decrypt Update	
Parameter	hSession	Session Handle
	pEncryptedPart	Pointer to CipherData
	ulEncryptedPartLen	Length of Cipher Data

	pPart	Pointer to AdditionalPlain Data Buffer
	pulPartLen	Pointer to Length of AdditionalPlain Data Buffer
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.28 Decrypt Finalization

Function Prototype	CK_RVC_DecryptFinal(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pLastPart, CK_ULONG_PTR pulLastPartLen);	
Description	Decrypt Finalization	
Parameter	hSession	Session Handle
	pLastPart	Pointer to Last Plain Data
	pulLastPartLen	Pointer to Length of Last Plain Data
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	#define CIPHERTEXT_BUF_SZ 256
	#define PLAINTEXT_BUF_SZ 256
	CK_ULONG firstEncryptedPieceLen, secondEncryptedPieceLen;
	CK_SESSION_HANDLE hSession;
	CK_OBJECT_HANDLE hKey;
	CK_BYTE iv[8];
	CK_MECHANISM mechanism = {
	CKM_AES_ECB, iv, sizeof(iv)
	};
	
	CK_BYTE data[PLAINTEXT_BUF_SZ];
	CK_BYTE encryptedData[CIPHERTEXT_BUF_SZ];
	CK_ULONG ulData1Len, ulData2Len, ulData3Len;	

```
CK_RV rv;
.
.
firstEncryptedPieceLen = 90;
secondEncryptedPieceLen = CIPHERTEXT_BUF_SZ - firstEncryptedPieceLen;
rv = C_DecryptInit(hSession, &mechanism, hKey);
if (rv == CKR_OK) {
/* Decrypt first piece */
      ulData1Len = sizeof(data);
      rv = C_DecryptUpdate(hSession,
&encryptedData[0], firstEncryptedPieceLen,
&data[0], &ulData1Len);
if (rv != CKR_OK) {
      ..}
/* Decrypt second piece */
      ulData2Len = sizeof(data) - ulData1Len;
      rv = C_DecryptUpdate(hSession,
&encryptedData[firstEncryptedPieceLen],
                                    secondEncryptedPieceLen,
&data[ulData1Len], &ulData2Len);
if (rv != CKR_OK) {
      ..}
/* Get last little decrypted bit */
      ulData3Len = sizeof(data) - ulData1Len - ulData2Len;
      rv = C_DecryptFinal(hSession,
&data[ulData1Len + ulData2Len], &ulData3Len);
if (rv != CKR_OK) {
      ..}
}
```

5.2.29 Hash Initialization

Function Prototype	CK_RVC_DigestInit(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism);	
Description	Hash Initialization	
Parameter	hSession	Session Handle
	pMechanism	Pointer to Mechanism
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.30 Hash

Function Prototype	CK_RVC_Digest(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen);	
Description	Hash	
Parameter	hSession	Session Handle
	pData	Pointer to Data to be hashed
	ulDataLen	Length of Data to be hashed
	pDigest	Pointer to Digest Data Buffer
	pulDigestLen	Pointer to Length of Digest Data Buffer
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.31 Hash Update

Function	CK_RVC_DigestUpdate(CK_SESSION_HANDLE hSession,

Prototype	CK_BYTE_PTRpPart, CK_ULONG ulPartLen);	
Description	Hash Update	
Parameter	hSession	Session Handle
	pPart	Pointer to Additional Data Buffer to be hashed
	ulPartLen	Length of Data Buffer to be hashed
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.32 Hash Finalization

Function Prototype	CK_RVC_DigestFinal(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen);	
Description	Hash Finalizaion	
Parameter	hSession	Session Handle
	pDigest	Pointer to Last Digest Data
	pulDigestLen	Pointer to Length of Last Digest Data
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_SESSION_HANDLE hSession;
	CK_MECHANISM mechanism = {
	CKM_SM3, NULL_PTR, 0
	};
	CK_BYTE data[] = {... };
	
	CK_BYTE digest[32];
	CK_ULONG ulDigestLen;	

	CK_RV rv;
	.
	.
	rv = C_DigestInit(hSession, &mechanism);
	if (rv != CKR_OK) {
	.
	.
	}
	rv = C_DigestUpdate(hSession, data, sizeof(data));
	if (rv != CKR_OK) {
	.
	.
	}
	rv = C_DigestKey(hSession, hKey);
	if (rv != CKR_OK) {
	.
	.
	}
	ulDigestLen = sizeof(digest);
	rv = C_DigestFinal(hSession, digest, &ulDigestLen);
	.
	.

5.2.33 Signing Initialization

Function Prototype	CK_RVC_SignInit(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey);

Description	Signing Initialization	
Parameter	hSession	Session Handle
	pMechanism	Pointer to Mechanism
	hKey	Key Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.34 Signing

Function Prototype	CK_RVC_Sign(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen);	
Description	Signing	
Parameter	hSession	Session Handle
	pData	Pointer to Data to be signed
	ulDataLen	Length of Data to be signed
	pSignature	Pointer to Signed Data
	pulSignatureLen	Pointer to Length of Signed Data Buffer
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.35 Signing Update

Function Prototype	CK_RVC_SignUpdate(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen);	
Description	Signing Update	
Parameter	hSession	Session Handle

	pPart	Pointer to Additional Data Buffer to be signed
	ulPartLen	Length of Data Buffer to be signed
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.36 Signing Finalization

Function Prototype	CK_RVC_SignFinal(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen);	
Description	Signing Finalization	
Parameter	hSession	Session Handle
	pSignature	Pointer to Last Signed Data
	pulSignatureLen	Pointer to Length of Last Signed Data
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_SESSION_HANDLE hSession;
	CK_OBJECT_HANDLE hKey;
	CK_MECHANISM mechanism = {
	CKM_RSA, NULL_PTR, 0
	};
	CK_BYTE data[] = {... };
	
	CK_BYTE mac[64];
	CK_ULONG ulMacLen;
	CK_RV rv;
	.
	.
	rv = C_SignInit(hSession, &mechanism, hKey);	

	if (rv == CKR_OK) {
	rv = C_SignUpdate(hSession, data, sizeof(data));
	..ulMacLen = sizeof(mac);
	rv = C_SignFinal(hSession, mac, &ulMacLen);
	.
	.
	}

5.2.37 Verification Initialization

Function Prototype	CK_RVC_VerifyInit(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey);	
Description	Initialize Verification	
Parameter	hSession	Session Handle
	pMechanism	Pointer to Mechanism
	hKey	Key Handle
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.38 Verification

Function Prototype	CK_RVC_Verify(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen);	
Description	Verification	
Parameter	hSession	Session Handle

	pData	Pointer to Source Datato be verified
	ulDataLen	Length of Source Data to be verified
	pSignature	Pointer to Signed Data Buffer to be verified
	ulSignatureLen	Length of Signed Data Buffer to be verified
Return Value	CKR_OK	Success，Verification Passed
	CKR_SIGNATURE_INVALID	Success，Signature Invalid
	Non-CKR_OK	Fail, return error code.

5.2.39 Verification Update

Function Prototype	CK_RVC_VerifyUpdate(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen);	
Description	Verification Update	
Parameter	hSession	Session Handle
	pPart	Pointer to Additional Data Buffer to be verified
	ulPartLen	Length of Data Buffer to be verified
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.40 Verification Finalization

Function Prototype	CK_RVC_VerifyFinal(CK_SESSION_HANDLE hSession, CK_BYTE_PTRpSignature, CK_ULONG ulSignatureLen);	
Description	Verification Finalization	
Parameter	hSession	Session Handle
	pSignature	Pointer to Last Data to be verified

	ulSignatureLen	Length of Last Data to be verified
Return Value	CKR_OK	Success， Verification Passed
	CKR_SIGNATURE_INVALID	Success， Signature Invalid
	Non-CKR_OK	Fail, return error code.

Example	CK_SESSION_HANDLE hSession;
	CK_OBJECT_HANDLE hKey;
	CK_MECHANISM mechanism = {
	CKM_RSA, NULL_PTR, 0
	};
	CK_BYTE data[] = {... };
	
	CK_BYTE mac[64];
	CK_RV rv;
	.
	.
	rv = C_VerifyInit(hSession, &mechanism, hKey);
	if (rv == CKR_OK) {
	rv = C_VerifyUpdate(hSession, data, sizeof(data));
	..rv = C_VerifyFinal(hSession, mac, sizeof(mac));
	.
	.
	}	

5.2.41 Take Random number as Seed

Function Prototype	CK_RVC_SeedRandom(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSeed, CK_ULONG ulSeedLen);
Description	Take Random number as Seed

Parameter	hSession	Session Handle
	pSeed	Pointer to Seed
	ulSeedLen	Length of Seed
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

5.2.42 Generate Random Number

Function Prototype	CK_RVC_GenerateRandom(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pRandomData, CK_ULONG ulRandomLen);	
Description	Generate Random Number	
Parameter	hSession	Session Handle
	pRandomData	Pointer to Random Number
	ulRandomLen	Length of Random Number
Return Value	CKR_OK	Success
	Non-CKR_OK	Fail, return error code.

Example	CK_SESSION_HANDLE hSession;
	CK_BYTE seed[] = {... };
	CK_BYTE randomData[] = {... };
	
	CK_RV rv;
	.
	.
	rv = C_SeedRandom(hSession, seed, sizeof(seed));
	if (rv != CKR_OK) {
	.
	.
	}	

	rv = C_GenerateRandom(hSession, randomData, sizeof(randomData));
	if (rv == CKR_OK) {
	.
	.
	}