# Software Security Module (SSM)

## Introduction

SSM provides following features with software implementation:

* Key storage
* Key management
* Security API through a TCP interface

## Installation

### Prerequisites

* JRE/JDK 1.8

SSM is developed in JDK1.8, so it requires JRE 1.8+ to run.

* GUI

SSM uses java swing to implement GUI. If running in Linux it requires GUI desktop (X) to be installed.

* JDBC

SSM use JDBC driver to connect database. By default, it uses Oracle JDBC driver (ojdbc6.jar is provided), which shall support Oracle9 and above.

Other databases are not tested, but are theoretically supported if provide correct JDBC driver.

### Installation & Configuration

Unpack ssm.tar.gz

Folder structure:

|  |  |
| --- | --- |
| Folder/File | DESC |
| ./ssm | SSM main folder |
| ./ssm.jar | SSM application jar file |
| ./ssm/log4j.xml | Log4j configuration |
| ./ssm/logs | SSM log folder |
| ./ssm/config | SSM configuration folder |
| ./ssm/config/ssm.properties | SSM configuration file |
| ./ssm/config/ssm.sql | SSM initial sql script |
| ./ssm/ssm\_lib | SSM dependency libraries |
| ./ssm/ssm\_lib/commons-logging-1.2.jar | Apache commons logging |
| ./ssm/ssm\_lib/log4j-1.2.7.jar | Log4j |
| ./ssm/ssm\_llib/ojdbc6.jar | Oracle JDBC driver |

Run ./ssm/config/ssm.sql into database. This will create table for SSM.

Update ./ssm/config/ssm.properties if necessary:

|  |  |
| --- | --- |
| Property | Desc |
| ssm.jdbcURL | JDBC connection string. |
| ssm.jdbcDriver | JDBC driver class. |
| ssm.host | Host IP to bind for SSM service.  0.0.0.0 for all interface. |
| ssm.port | Port for SSM service. |
|  |  |

Start SSM by running command in ssm dir:

> cd $HOME/ssm

> java -jar ssm.jar

## Key storage

All keys are stored in database. All keys are double length.

TABLE KEYS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | PK | Description |
| EntityID | varchar2(8) | Not Null | PK | ‘99999999’ for LMK/TMK.INIT/PVK/CVK/MCK/EMK  Customer Defined for other KeyType |
| KeyType | varchar2(8) | Not Null | PK | ‘LMK’  ‘TMK.INIT’  ‘TMK.ATM’  ‘TPK.ATM’  ‘ZPK’  ‘PVK’  ‘CVK’  ‘MCK’  ‘EMK’ |
| KEV | varchar2(32) |  |  | Double length Key Value (Encrypted) |
| KCV | varchar2(6) |  |  | Key Check Value |

\*

LMK: Encrypted by a private key. Generated by console input

TMK.INIT: Only one, encrypted by LMK. Generated by console input

TMK.ATM: One for each ATM, encrypted by LMK. Dynamically generated by interface

TPK.ATM: One for each ATM, encrypted by LMK. Dynamically generated by interface

ZPK: One for each 3rd party entity, encrypted by LMK. Generated by console input

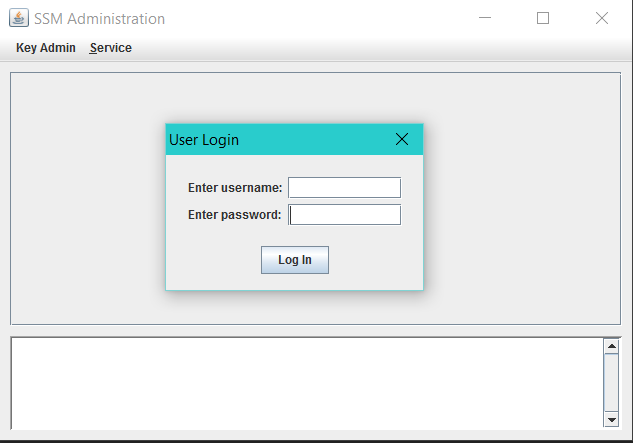
## Key management

SSM provide a GUI console for key management.

### Login

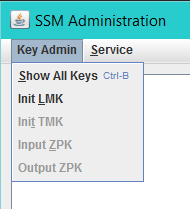
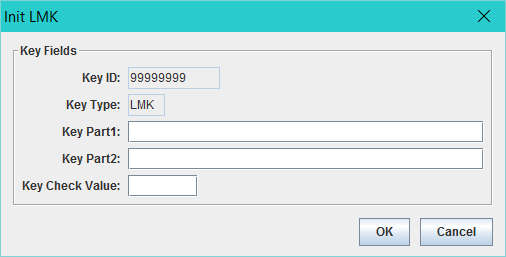
SSM use database authorization for access control.

At startup, operator must provide database user and password to login (connect).



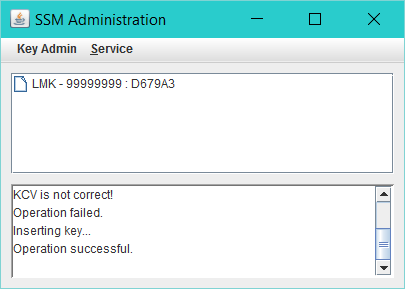
### Initialize LMK

At first logon, there’s no key in database. Before any other operation, LMK must be inited first.

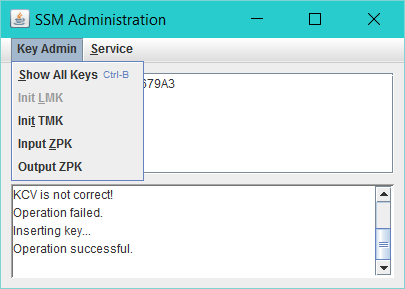
 

LMK’ key id is ‘99999999’ and key type is ‘LMK’. These are fixed value that can not be changed.

After LMK is initialized, it will be shown in the key tree.

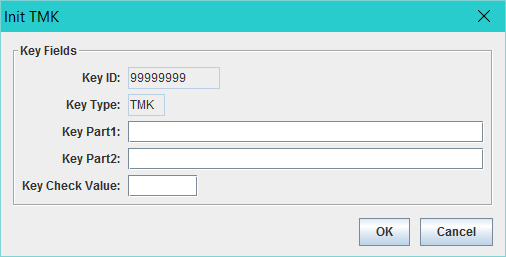


The operation is only allowed once. After operation is done, the menu will be greyed out.

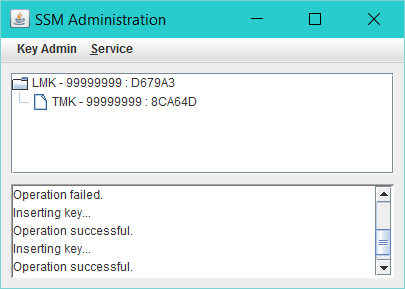


### Initialize TMK.INIT

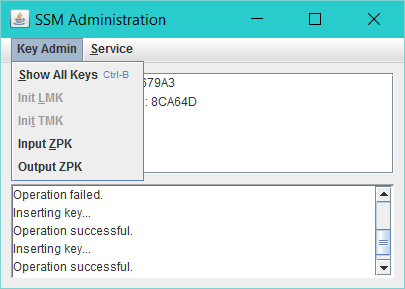
TMK.INIT must be initialized before SSM start service api. Otherwise, any ATM exchange Key will fail.



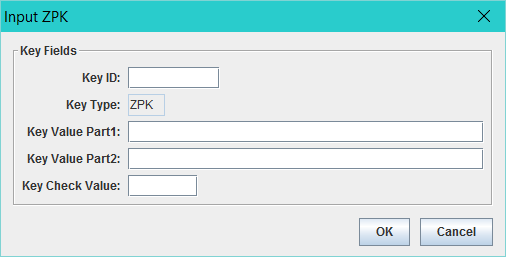
After TMK.INIT is initialized, it will be shown as a child key of LMK.



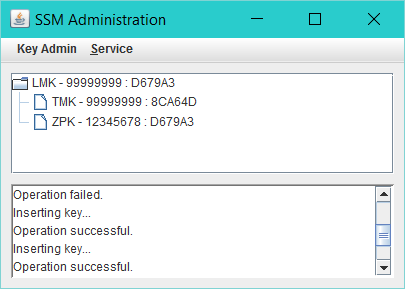
Initialize TMK.INIT is only allowed once. After operation is done, the menu will be greyed out.



### Input ZPK

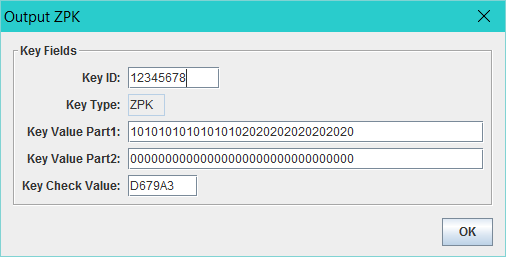


After input ZPK, it’s shown as child key of LMK



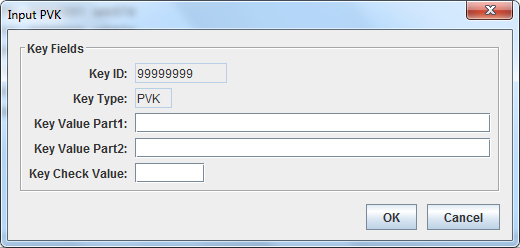
### Output ZPK

Input Key ID and press ‘Enter’, clear key value will be shown as Key Part 1.



### Input PVK/CVK/MCK/EMK

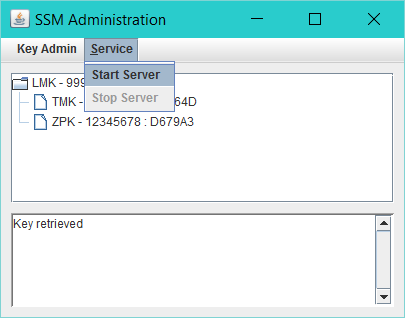
The procedure is same as init TMK



## Interface

SSM act as TCP server, third party connect a permanent link to SSM.

Click SSM menu ‘Start server/Stop server to switch service status.



SSM support these commands:

* IM: Initiate TMK.ATM
* XK: Exchange TMK.ATM / TPK.ATM
* TP: Translate PIN
* CP: Get Clear Pin Block

### ‘IM’ - Initiate TMK.ATM

Generate new TMK.ATM, encrypt it by current TMK.INIT.

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘IM’ |
| ATM ID | 8 A | Unique ID for each ATM |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘im’ |
| Error Code | 2 A | ‘00’: No error  ‘01’: TMK.INIT isn’t initiated yet. |
| Key | 32 H | The new key, encrypted by the TMK.INIT. |
| Key Check Value | 6 H | The key check value. |

### ‘XK’ - Exchange TMK.ATM / TPK.ATM

Generate new TMK.ATM or TPK.ATM, encrypt it by current TMK.ATM.

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘XK’ |
| Mode | 1 A | ‘0’: Exchange TMK.ATM  ‘1’: Exchange TPK.ATM |
| ATM ID | 8 A | Unique ID for each ATM |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘xk’ |
| Error Code | 2 A | ‘00’: No error  ‘01’: TMK.ATM isn’t initiated yet. |
| Key (under LMK) | 32 H | The new key, encrypted by current TMK.ATM. |
| Key Check Value | 6 H | The key check value. |

### ‘TP’ - Translate PIN:

Translate a PIN from TPK to ZPK Encryption.

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘TP’ |
| ATM ID | 8 A | ATM ID for TPK.ATM |
| Entity ID | 8 A | Entity ID for ZPK |
| PIN Block | 16 H | Source PIN Block |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘tp’ |
| Error Code | 2 A | ‘00’: No error  ‘01’: TPK error.  ‘02’: ZPK error. |
| PIN Block | 16 H | Destination PIN block. |

### ‘CP’ – Get Clear PIN:

Get clear PIN from PIN block encrypted by TPK.ATM. PIN Block is in ISO0 format.

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘CP’ |
| ATM ID | 8 A | ATM ID for TPK.ATM |
| PAN | 19 A | PAN (right padding space) |
| PIN Block | 16 H | Source PIN Block |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘cp’ |
| Error Code | 2 A | ‘00’: No error  ‘01’: TPK error  ‘02’: PIN block format error |
| PIN Block | 16 H | Clear PIN block. |
|  |  |  |

### ‘GP’ - Generate PVV

Generate PVV

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘GP’ |
| PVKI | 1 N | PVK Index |
| PAN | 19 A | PAN |
| PIN Block | 16 H | PIN Block, encrypted by ZPK(99999999). |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘gp’ |
| Error Code | 2 A | ‘00’: No error  ‘01’: PVK error.  ‘02’: Format error. |
| PVV | 4 N | PVV |

### ‘VP’ - Verify PVV

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘VP’ |
| PAN | 19 N | PAN |
| PVKI | 1 N | PVK Index |
| PIN Block | 16 H | PIN Block, encrypted by ZPK(99999999). |
| PVV | 4 N | PVV |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘vp’ |
| Error Code | 2 A | ‘00’: Success  ‘01’: PVK error.  ‘02’: Format error  ‘03’: PVV error |

### ‘GC’ - Generate CVV

Generate CVV

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘GP’ |
| PAN | 19 N | PAN |
| Expire Date | 4 N | Expire Date |
| Service Code | 3 N | Service code.  Input ‘999’ for iCVV  Input ‘000’ for CVV2 |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘gp’ |
| Error Code | 2 A | ‘00’: No error  ‘01’: PVK error.  ‘02’: Format error. |
| CVV | 3 N | CVV |

### ‘VC’ - Verify CVV

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘VC’ |
| PAN | 19 A | PAN |
| Expire Date | 4 N | Expire Date |
| Service Code | 3 N | Service code. Input ‘999’ for iCVV |
| CVV | 3 N | CVV |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘vc’ |
| Error Code | 2 A | ‘00’: Success  ‘01’: CVK error.  ‘02’: Format error  ‘03’: CVV error |

### ‘GM’ – Generate MAC

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘GM’ |
| Algorithm | 1 N | 1. ISO 9797 MAC algorithm 3 (= ANSI X9.19 when used with a double-length key) |
| Padding Method | 1 N | 1.ISO 9797 Padding method 1 (i.e. pad with 0x00)  2.ISO 9797 Padding method 2 (i.e. add 0x80 then pad with 0x00) |
| Data Length | 2 N | The length of the following field, in bytes. |
| Data | n N | The message to be MAC’d |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘gm’ |
| Error Code | 2 A | ‘00’: Success  ‘01’: MAC error.  ‘02’: Format error |
| MAC | 8 H | The calculated MAC |

### ‘VM’ – Generate MAC

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘VM’ |
| Algorithm | 1 N | 1. ISO 9797 MAC algorithm 3 (= ANSI X9.19 when used with a double-length key) |
| Padding Method | 1 N | 1.ISO 9797 Padding method 1 (i.e. pad with 0x00)  2.ISO 9797 Padding method 2 (i.e. add 0x80 then pad with 0x00) |
| Data Length | 2 N | The length of the following field, in bytes. |
| Data | n N | The message to be MAC’d |
| MAC | 8 H | The calculated MAC |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘vm’ |
| Error Code | 2 A | ‘00’: Success  ‘01’: MACK error.  ‘02’: Format error  ‘03’: MAC error |

### ‘GQ’ – Generate ARQC

Request:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Command Code | 2 A | ‘GQ’ |
| Data Length | 2 N | The length of the following field, in bytes. |
| Data | n N | The message to be MAC’d |

Response:

|  |  |  |
| --- | --- | --- |
| Field | Length & Type | Details |
| Header | 3 A | ‘SSM’ |
| Response Code | 2 A | ‘gq’ |
| Error Code | 2 A | ‘00’: Success  ‘01’: EVMACK error.  ‘02’: Format error |
| MAC | 8 H | The calculated AC |