PINPAD Application Development Guide

v1.0

Fujian LANDI Commercial Equipment Co,.Ltd.

Table of Contents

1. Foreword

1.1. Revision History

Date	Version	Description
2017-05-11	1.0	Creation

1.2. Scope

The document provides the PINPAD module application development guide of LANDI high-end POS terminal, including the security mechanism introduction, using guide and security development guide.

This document is suitable for all POS products of EPT-AND, EPT-HQL and APOS/AECR serials.

1.3. Terms and Definitions

Terms	Definition
AES	Advanced Encryption Standard
FK	Fixed Key
DUKPT	Derived Unique Key Per Transaction
MK/SK	Master Key and Session Key
PEK	Pin Encryption Key
MAK	MAC Key
TDK	Track Data Encryption Key
DEK	Data Encryption Key
DUKPT IK	DUKPT Initial Key
DUKPT FK	DUKPT Future Key
N/A	Not Applicable
PED	PIN Entry Device
PIN	Personal Identification Number
RSA	Rivest Shamir Adelman Algorithm
SHA	Secure Hash Algorithm
TDES	Triple Data Encryption Standard
IPP	Internal or Embedded PED of POS terminal
EPP	External PED of POS terminal
KAP	Key Arrangement Pool. It is a logical isolated

	key area for key storing.

1.4. References

[1] EPT-AND Application Development Guide.chm
[2] EPT-HQL Application Development Guide.chm
[3] PED Initial Key Loading User Guide.pdf
[4] PCI PTS Specifications

2. Introduction of PINPAD

2.1. Function Brief

The PINPAD module is mainly responsible for PIN entry/encryption and key management. It provides related APIs to upper-layer applications.

2.1.1. Main Features

The main differences on PINPAD module between the high-end POS and other platforms are as followings：

1. The security mechanism adds multi-key area isolation and authentication.
   In previous platforms, acquirers share 100 sets of terminal master key (TMK). But they will negotiate with each other to choose a TMK index. In this case, applications are difficult to be managed. If there is no related review mechanism and signature mechanism, the security risk of key misuse and mixed use exist. To avoid the risk, high-end POS supports multiple key area mechanism so that the application can be developed more securely and conveniently.
2. Significantly improved the API prototype of PINPAD module based on the fully comprehension of security specification such as PCI/TR-31 to make relevant interfaces design more secure, while more scalable.
3. Provide permission management mechanism for applications, that is, the application manager can allocate different level permissions for different applications.

2.1.2. APIs

The module supports the following APIs:

- EA_pinpad_ucGetAllPinpadsInfo()
- EA_pinpad_ucOpenDevice()
- EA_pinpad_vCloseDevice()
- EA_pinpad_ucSetPinpadCfg()
- EA_pinpad_ucResetPinpad()
- EA_pinpad_ucFormatPinpad()
- EA_pinpad_ucGetPinpadInfo()
- EA_pinpad_ucSetPinpadSerialNum()
- EA_pinpad_ucDispPinpad()
- EA_pinpad_ucLocatePinpadLcd()
- EA_pinpad_ucPinpadBeep()
- EA_pinpad_ucGetRandom()
- EA_pinpad_ucAuthEncKeyKCV()
- EA_pinpad_ucLoadEncKey()
- EA_pinpad_ucGenerateKey()
- EA_pinpad_ucCheckKey()
- EA_pinpad_ucLoadDukptInitialKeyFromKap()
- EA_pinpad_ucDeleteKey()
- EA_pinpad_ucGetKcv()
- EA_pinpad_ucMacInit()
- EA_pinpad_ucMacLoadData()

- EA_pinpad_ucMacGenerate()
- EA_pinpad_ucMacVerify()
- EA_pinpad_ucCalculateDes()
- EA_pinpad_ucEncryptMagTrackData()
- EA_pinpad_ucGetExistentKapIds()
- EA_pinpad_ucCreateKap()
- EA_pinpad_ucDeleteKap()
- EA_pinpad_ucDeleteAllKaps()
- EA_pinpad_ucEraseAllKeysWithinKap()
- EA_pinpad_ucFormatKap()
- EA_pinpad_ucSetKapCfg()
- EA_pinpad_ucGetKapMode()
- EA_pinpad_ucGetKapInfo()
- EA_pinpad_ucGetExistentKeyIdsInKeySystem()
- EA_pinpad_ucStartPinEntry()
- EA_pinpad_ucStartOfflinePin()
- EA_pinpad_ucVerifyOfflinePin()
- EA_pinpad_ucCancelPinEntry()
- EA_pinpad_ucInjectPinEntryFuncKey()
- EA_pinpad_ucGetPinEntryInfo()
- EA_pinpad_ucStartPinEntryAndVerifyWithIcCard()

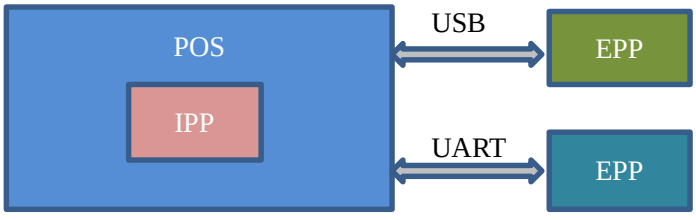
For more details, please refer to PINPAD API of Application Development Guide.

2.2. Main Mechanism

2.2.1. PINPAD Device and Handle

The terminal takes every PINPAD as a virtual device and manages all PINPADs through device handle. Powering on device and gaining its handle must be done before using the device. The handle is equivalent of a permission to use the device.

The terminal supports multiple PINPAD devices, including internal PINPAD and external PINPAD (the device can be connected with USB cable or serial port). Users have to designate the specific accessing device through APIs before using the PINPAD device.



Note: due to the terminal supports multiple PINPAD devices, the specific-used equipment will be automatically set in the boot phase. So users have to connect external PINPAD to POS when powering on the device so that the external PINPAD can be correctly detected and then used.

Related APIs:

The must-have APIs follow below:

EA_pinpad_ucGetAllPinpadsInfo: used to list all available PINPADs detected
EA_pinpad_ucOpenDevice: used to open a PINPAD and obtain its handle
EA_pinpad_vCloseDevice: used to close a PINPAD and release the handle.

The optional APIs follow below:

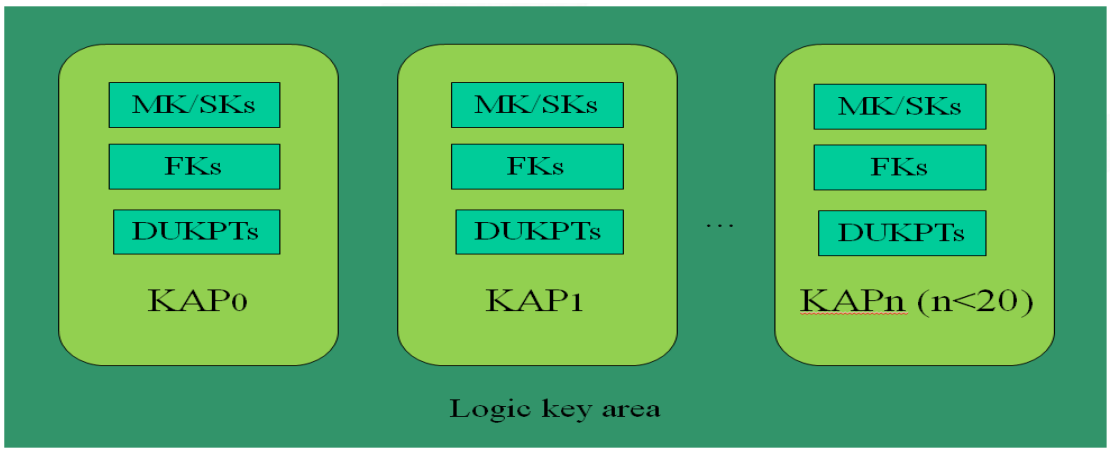
EA_pinpad_ucSetPinpadCfg: configure PINPAD
EA_pinpad_ucResetPinpad: reset PINPAD (logically)
EA_pinpad_ucFormatPinpad: format PINPAD (Restore to the factory state)
EA_pinpad_ucGetPinpadInfo: get PINPAD information
EA_pinpad_ucSetPinpadSerialNum: set the serial number of PINPAD
EA_pinpad_ucGetRandom: read a set of random number from PINPAD

2.2.2. Key Arrangement Pool (KAP)

Key Arrangement Pool is a logically isolated key storage area. Each user can use one or more KAP

Each PINPAD device can support up to 20 KAPs at the same time, that is, each PINPAD device allows up to 20 acquirer applications to be used simultaneously without conflict.

The KAP structure diagram as below:



KAP Identifier

The following data structure is introduced to uniquely identify each KAP:

typedef struct ET_KapId {

```
    ET_RegionId tRegionId;
    ET_KapNum tKapNum;
} ET_KapId;

typedef ushort ET_RegionId;
typedef ushort ET_KapNum;
```

tRegionId：The index of the KAP (Region), used to uniquely identify the app region manager of the certain key user such as bank or third party payment organization. If tRegionId=0, it means that the manager has not been registered in LANDI (the organization does not apply for application root certificate from LANDI or it has one but do not used yet) and the applications will be managed by LANDI uniformly, that is, LANDI acts as a multi-application manager.

If tRegionId=0xFFFF, it refers to all Regions. When signing APP, the application manager can designate the accessible KAP IDs list of the APP. If one of the KAP ID whose tRegionId=0xFFFF, it refers that the APP can access all KAPs of RegionId. Normally, the APP's RegionId is bound to a signature UKEY and cannot be set casually. Only some special APPs can access all KAPs such as middleware APP developed by vendor.

Notes：tRegionId will be allocated by LANDI and will be bound to the signature UKEY that client applied for. Different signature UKEYs will be allocated with different tRegionId to ensure isolation between different regions. By now LANDI divided the world into 8 regions: APAC/CEOP/LAR/NAR/SEPA/SEPAEU/EMEA….

tKapNum：The KAP index inside a special region, used to identify a certain acquirer. Each KAP is securely isolated and independent. Different KAP has different access right. If an organization has multiple independent acquiring businesses and backgrounds, each acquiring business and background should be took as an independent acquirer to ensure the key management between them is isolated.

If tKapNum=0, it refers to a shared KAP under the special RegionId. The KAP is shared as long as the applications are in the same Region. For example, when signing an APP, it will be authorized for KAP ID=0xAAAA0001, which means the APP can access the private KAP of 0xAAAA0001 and the shared KAP of 0xAAAA0000

If tKapNum=0xFFFF, it refers to all KAPs under the special RegionId. When signing an APP, it is authorized by KapNum=0xFFFF, which means the APP can access all KAP within the Region. For instance, when signing an APP, it will be authorized for KAP ID=0xAAAAFFFF, which means the APP can access all KAPs in the range of 0xAAAA0000 to 0xAAAAFFFF.

Notes：

1） For the domestic system, tKapNum is allocated by application management organization. To take smartPOS ecosystem as an example, the operation management platform is responsible to distribute different tKapNum for different acquirer.

2） For the oversea system, tKapNum now is allocated by LANDI. When applying for signature UKEY, every security officer should apply for different tKapNum for different acquirer. LANDI will bind the tKapNum to the specific signature.

For common acquiring application, it is suggested to keep the key in their own private KAP to ensure only

they can access the key. So that KAPs which different APPs can access are independent and isolated from each other.

As for the case that an application needs to use multiple MKs, it is suggested to follow rules below to decide to use one KAP or to use more than one KAP:

1） CASE 1: If an application uses two independent key systems which have their own initial key TMK1 and TMK2. Both TMK1 and TMK2 can separately load their own PIN KEY to encrypt PIN. TMK1 and TMK2 should be isolated in accordance with PCI requirements. And their corresponding background should be taken as two independent acquirers which use their own KAPs.

2） CASE 2：If an application uses two keys, that is, MK1 and MK2, but they all belong to the same key hierarchy and share the same root TMK. MK1 and MK2 are loaded in cipher-text mode so they can be taken as the same acquirer.

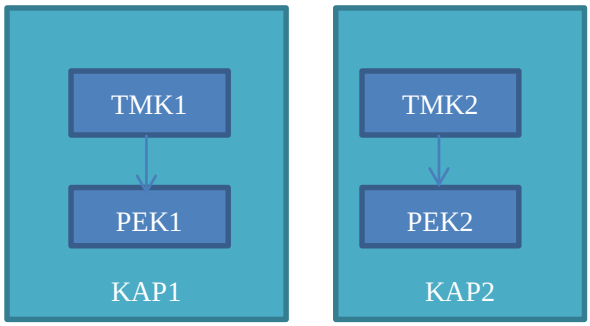
3） CASE 3：If an application uses two independent key hierarchies which have their own initial TMK1 and TMK2 but only TMK1 can be used to load PIN KEY, the other TMK is used in common data function, then TMK2 should not be taken as an independent acquirer. So the two MK can share one KAP or multiple KAP. Generally there are several options:

a) TMK1 and TMK2 are both stored in private KAP.
b) TMK1 is stored in private KAP, but TMK2 is stored in the shared KAP.
c) TMK2 is stored in private KAP, but TMK1 is stored in the shared KAP.
d) TMK1 and TMK2 are both stored in the shared KAP.

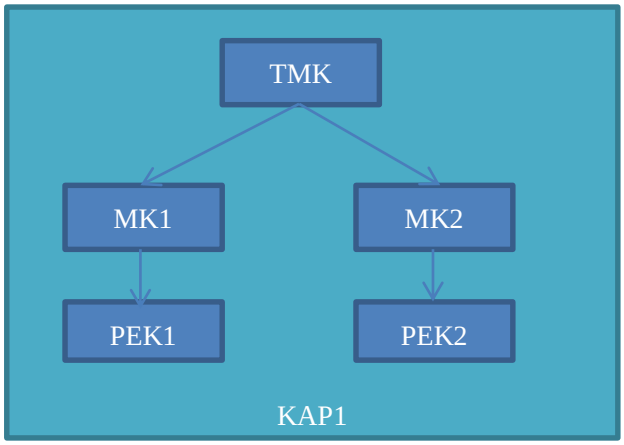
Option a) and b) are recommended, while option c) and d) are not recommended since the key is stored in a shared area.

Different acquirers' keys should be safely isolated to avoid the security risk of key misuse and mixed-use key in accordance with PCI requirements and. The design rules of LANDI POS are to suggest that each acquirer uses an independent KAP to ensure their keys are isolated separately. For the above three cases, the requirements of KAP usage are as followings:

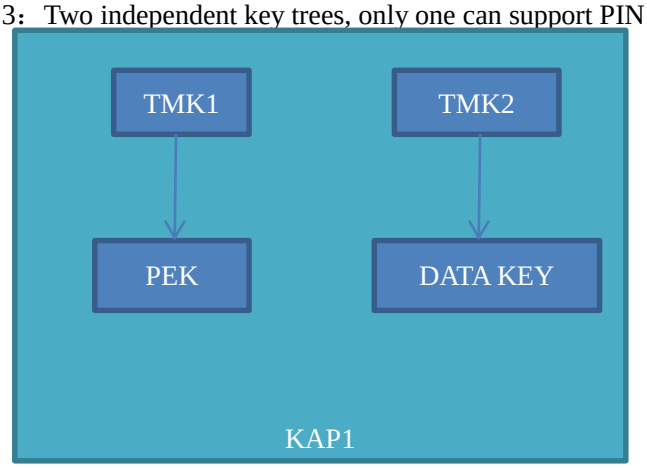
CASE 1：Two independent key tree. Both can be used for acquirer. Then they should use different private KAP



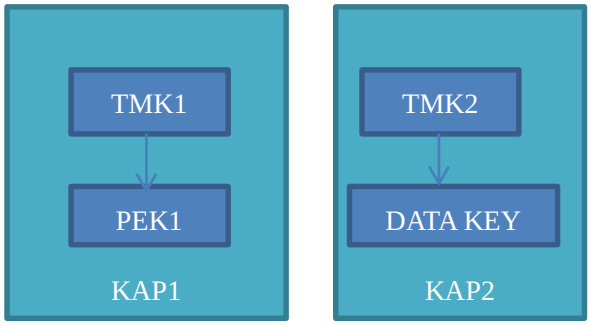
CASE 2：The same key tree, which can use same private KAP



CASE 3：Two independent key trees, only one can support PIN KEY, which can use same KAP



Also to use multiple KAPs:



KAP Management

KAP Using Rules

KAP ID		KAP Property	Access Requirements	Plain-text initial key loading limitation	Multiple Application key isolation
RegionId	KapNum				
=0	0	Global	No requirement, any	None. Any new initial	No isolation

		shared key area	application can access the KAP	plain-text key can be added into key area at any time.	
	[1~15]			Once a key has been loaded, any other new key cannot be loaded into the KAP through non-authorized method unless all keys are cleared or formatted in the whole key area.	No isolation
	[16~0xffff]	Reserved for Future Use			
[1,0xFFFE]	= 0x0000	Shared key area of each user group	Same group can access, that is, as long as the application belongs to the user group (using same application root certificate), then the application can access the KAP.	None. Any new initial plain-text key can be added into key area at any time.	Isolated between groups
	!= 0x0000	Independent and private key area of each user group	Authorized by user group administrator. Only the application with the corresponding permissions can access the key area.	Once a key has been loaded, any other new key cannot be loaded into the KAP through non-authorized method unless all keys are cleared or formatted in the whole key area.	Intra-group isolation
=0xFFFF	N/A	For accessed KAP, it is not allowed to have the value; For accessible KAP list, it refers to all accessible Region			

Notes:

1. LANDI POS provides not only private KAP but also shared KAP to meet specific requirements of certain applications.
2. Each acquirer is requested to load key into their own private KAP and not use shared KAP, either global shared KAP or inter-group shared KAP in accordance with PCI requirements. If a shared KAP is used, other applications can also access or cover it, which may cause key misuse or key mix use between acquirers. And LANDI won't bear relevant responsibilities. Acquirer has an obligation to take their own responsibility of key management to ensure their keys won't be utilized for any illegal purpose. Acquirer application developers who are entrusted by the acquirer should take the responsibility of the APP security. According to the principle: Who signs the app and who is responsible for the app, the signer should be responsible for the risk caused by ill-formed application. So the signer should make sure the application code is reviewed and confirmed and has no risk of security.

KAP Allocation and Management

KAP is consist of the following two parts：

1） tRegionId: represents the group ID. It is allocated by LANDI. LANDI will issue a unique root certificate for each region. Each root certificate has a unique ID which is used to construct the tRegionId. LANDI's allocation system ensures different regions use different root certificate to approach security isolation purpose between regions.

2） tKapNum: the index inside a special region. It is allocated by LANDI. Generally, each region manager needs to apply for signature UKEYs for the acquirer applications signing. During the applying process, the manager should apply for a unique KAP ID for each acquirer. If not applied, the default value of tKapNum(0x0001) is allocated to the UKEY and then any application signed with this UKEY will has same accessible private KAP (= 0x00001), which may cause conflict.

In case one same signer responsible for multiple acquirers applications signing, the signer can apply for binding multiple KAP ID to one UKEY to avoid inconvenience of using multiple UKEYs. But in this case, the signer needs to select a correct KAPID during signing for multiple applications and take responsible for misusing UKEYs.

Note that each KAP ID is related with the acquirer but neither the developer nor the terminal owner/manager.

1) If several applications belong to different acquirers, they are isolated from each other by using different acquirer signature UKEYs.
2) If several applications belong to a same acquirer organization, but belong to different business departments who use different key hierarchies and backend server, they should be regarded as two different independent sub-organizations just as two acquirers and they should be allocated with two independent signature UKEY, KLD and tools.
3) If the several applications belong to one acquirer and have one unified manager, they should be signed with one same signature UKEY.

Related APIs:

EA_pinpad_ucGetExistentKapIds: gain list of the exist KAP
EA_pinpad_ucCreateKap: create KAP
EA_pinpad_ucDeleteKap: delete KAP
EA_pinpad_ucDeleteAllKaps: delete all KAP
EA_pinpad_ucFormatKap: format KAP
EA_pinpad_ucSetKapCfg: configure KAP
EA_pinpad_ucGetKapInfo: gain information of a certain KAP

The KAP-related APIs usually are used by key loading institutions. When loading initial keys, the corresponding KAP will be created. For the subsequent transaction process, there is usually no need to operate

the KAP again. The application manager can allocate different permissions according to the role of applicants. For example, the KAP management developers for key loading application may need an ADMIN permission, but the common application developers don't need this.

2.2.3. Keys

Key Identifier

The data structure, ET_KeyHandle, is used to uniquely identify and describe a key:

```
typedef struct ET_KeyHandle {
    ET_KapId tKapId;
    ET_KeySystem tKeySystem;
    ET_KeyId tKeyId;
} ET_KeyHandle;
```

Key System

Each KAP can store keys for FK, MK / SK, and DUKPT

The data structure, ET_KeySystem, is used to describe a kind of key system:

```
typedef char ET_KeySystem;
#define EM_pinpad_KM_MKSK ( (ET_KeySystem)(0) )
#define EM_pinpad_KM_DUKPT ( (ET_KeySystem)(1) )
#define EM_pinpad_KM_FIXED_KEY ( (ET_KeySystem)(2) )
```

Key ID

```
typedef ushort ET_KeyId;
#define EM_pinpad_MAX_KEY_ID ( (ET_KeyId)(0x00ff) )
```

Note：

1. The key ID is the index number in a key system. The key ID in different key systems may be repeated, and the range of index number is [0,EM_pinpad_MAX_KEY_ID]
2. Please note that the key ID and the key group number which the terminal supports is not the same concept, the two do not have a certain connection. The key group number is the numbers of the storage unit in a PINPAD, it is like the numbers of rooms in a building; but the key ID is just an identifier like as room number. The number is not necessarily increased from 1 to the maximum. In a KAP, an application can arbitrarily designate a key ID as long as it does not exceed the maximum 0xFFFF. Therefore, the key ID may larger than the key group number. For

example, a PINPAD can support at most 250 sets of key, but the key ID can be greater than 256 such as 1000. The key sets are equivalent to the entire storage space of PINPAD, but the key ID is just a unique identifier of key. It is like a building, whose room number can be 3206, but it does not mean that the building has 3206 rooms.

Key Configuration

The data structure, ET_KeyCfg, is used to describe the key configuration：

```
typedef struct ET_KeyCfg {

    ET_KeyUsage tKeyUsage;
    ET_KeyAlgorithm tKeyAlgorithm;
    ET_ModeOfUse tModeOfUse;
    ET_KeyVersionNumber tVerNum;
    ET_KeyExportability tExportability;

} ET_KeyCfg;
```

Key Usage

The definitions of key usage, referring to TR-31 are as follow:

```
typedef char ET_KeyUsage;

#define EM_pinpad_KU_BDK ( (ET_KeyUsage) (0x00) ) //    Corresponding to “B0” defined in TR-31

#define EM_pinpad_KU_DUKPT_INIT_KEY ( (ET_KeyUsage) (0x01) )// "B1". DUKPT init key.

#define EM_pinpad_KU_CVK ( (ET_KeyUsage) (0x02) )// "C0". CVK : Card Verification Key.

#define EM_pinpad_KU_DATA_ENCRYPTION ( (ET_KeyUsage) (0x03) )// "D0". Data encryption Key.

#define EM_pinpad_KU_EMV_IMK_APP_CRYPTOGRAMS ( (ET_KeyUsage) (0x04) )// "E0". IMK :
Issuer Master Key.

#define EM_pinpad_KU_EMV_IMK_SECURE_MSG_FOR_CONFIDENTIALITY ( (ET_KeyUsage)
(0x05) )// "E1". IMK : Issuer Master Key.

#define EM_pinpad_KU_EMV_IMK_SECURE_MSG_FOR_INTEFRITY ( (ET_KeyUsage) (0x06) )//
"E2"
```

#define EM_pinpad_KU_EMV_IMK_DATA_AUTHENTICATION_CODE ((ET_KeyUsage) (0x07))// "E3"

#define EM_pinpad_KU_EMV_IMK_DYNAMIC_NUM ((ET_KeyUsage) (0x08))// "E4"

#define EM_pinpad_KU_EMV_IMK_CARD_PERSONALIZATION ((ET_KeyUsage) (0x09))// "E5"

#define EM_pinpad_KU_EMV_IMK_OTHER ((ET_KeyUsage) (0x0a))// "E6"

#define EM_pinpad_KU_IV ((ET_KeyUsage) (0x0b))// "I0". IV : Initializatio Vector.

#define EM_pinpad_KU_KEY_ENC_OR_WRAP ((ET_KeyUsage) (0x0c))// "K0". including MK.

#define EM_pinpad_KU_KBPK ((ET_KeyUsage) (0x0d))// "K1". KBPK : TR-31 Key Block Protection Key.

#define EM_pinpad_KU_ISO_16609_MAC_ALGORITHM_1 ((ET_KeyUsage) (0x0e))// "M0"

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_1 ((ET_KeyUsage) (0x0f))// "M1"

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_2 ((ET_KeyUsage) (0x10))// "M2"

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_3 ((ET_KeyUsage) (0x11))// "M3". MAC Key.

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_4 ((ET_KeyUsage) (0x12))// "M4"

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_5 ((ET_KeyUsage) (0x13))// "M5"

#define EM_pinpad_KU_PIN_ENCRYPTION ((ET_KeyUsage) (0x14))// "P0". PIN Key, WK.

#define EM_pinpad_KU_KPV ((ET_KeyUsage) (0x15))// "V0". PIN verification, KPV, other algorithm.

#define EM_pinpad_KU_PIN_VERIFICATION_IBM_3624 ((ET_KeyUsage) (0x16))// "V1"

#define EM_pinpad_KU_PIN_VERIFICATION_VISA_PVV ((ET_KeyUsage) (0x17))// "V2"

The following definitions are extended which is not included in TR-31:

#define EM_pinpad_KU_TRACK_DATA_ENCRYPTION ((ET_KeyUsage) (0x20)) magnetic track encryption

#define EM_pinpad_KU_LAKALA_TMK ((ET_KeyUsage) (0x30)) Lakala customized MK

Key Algorithm

```
typedef char ET_KeyAlgorithm;

#define EM_pinpad_KA_AES ( (ET_KeyAlgorithm)('A') )
#define EM_pinpad_KA_DEA ( (ET_KeyAlgorithm)('D') )
#define EM_pinpad_KA_ELLIPTIC_CURVE ( (ET_KeyAlgorithm)('E') )
#define EM_pinpad_KA_HMAC_SHA_1 ( (ET_KeyAlgorithm)('H') )
#define EM_pinpad_KA_RSA ( (ET_KeyAlgorithm)('R') )
#define EM_pinpad_KA_DSA ( (ET_KeyAlgorithm)('S') )
#define EM_pinpad_KA_TDEA ( (ET_KeyAlgorithm)('T') ) // Triple DEA
#define EM_pinpad_KA_SMS4    ( (ET_KeyAlgorithm)('M') ) // SMS4
```

LANDI terminal now only supports DES/TDES,AES, SM4, other algorithms are reserved for extension.
Note: the international version of the LANDI terminal does not support SM4 algorithm, all SM4-related operations will return error.

Key Mode of Use

```
typedef char ET_ModeOfUse

#define EM_pinpad_MOU_ENC_DEC_WRAP_UNWRAP ( (ET_ModeOfUse)('B') ) // Both Encrypt & Decrypt / Wrap &Unwrap. "KUM" : Key Use Mode.

#define EM_pinpad_MOU_GENERATE_AND_VERIFY ( (ET_ModeOfUse)('C') ) // Both Generate & Verify

#define EM_pinpad_MOU_DEC_OR_UNWRAP_ONLY ( (ET_ModeOfUse)('D') ) // Decrypt / Unwrap Only

#define EM_pinpad_MOU_ENC_OR_WRAP_ONLY ( (ET_ModeOfUse)('E') ) // Encrypt / Wrap Only

#define EM_pinpad_MOU_GENERATE_ONLY ( (ET_ModeOfUse)('G') ) // Generate Only

#define EM_pinpad_MOU_NO_RESTRICTIONS ( (ET_ModeOfUse)('N') ) // No special restrictions (other than restrictions implied by the Key Usage)

#define EM_pinpad_MOU_SIGNATURE_ONLY ( (ET_ModeOfUse)('S') ) // Signature Only

#define EM_pinpad_MOU_VERIFY_ONLY ( (ET_ModeOfUse)('V') ) // Verify Only

#define EM_pinpad_MOU_DERIVE_KEYS ( (ET_ModeOfUse)('X') ) // Key used to derive other key(s)
```

Key Management

Key Injection

Plain-text key injection:

LKI solution provided by LANDI is used to inject plain-text initial key safely. For more specific information, please contact to technical support.

Cipher-text key injection:

Provide the following API to load encrypted keys:

EA_pinpad_ucLoadEncKey: Load encrypted keys.

Key Generation

EA_pinpad_ucGenerateKey: to generate a key according to the designated algorithm.

Note: most of the time, keys are generated and injected into terminals by acquirers themselves, only a small part of clients will use PED to generate keys automatically.

Key Check

EA_pinpad_ucCheckKey: to check whether keys are available.

KCV:

EA_pinpad_ucGetKcv: used to calculate the KCV of a set of key, output KCV which is no more than 4 bytes(domestic version) or 3 bytes(international version)

EA_pinpad_ucAuthEncKeyKCV：used to check KCV before loading cipher-text key. (the interface won't output KCV so it is not limited by the byte of KCV

Key Deletion

EA_pinpad_ucDeleteKey: delete a set of key

EA_pinpad_ucEraseAllKeysWithinKap: delete all keys of a certain KAP

Key Using

MAC:

EA_pinpad_ucMacInit: Init MAC parameters

EA_pinpad_ucMacLoadData: Load data to be calculated with MAC algorithm

EA_pinpad_ucMacGenerate: Generate MAC using loaded data

EA_pinpad_ucMacVerify: Compare the generated MAC data with the appointed MAC data

Before using the APIs, a MAK(MAC key) must be loaded first by invoking EA_pinpad_ucLoadEncKey or injection in plaint-text (for Fixed key only).

Data Encryption:

EA_pinpad_ucCalculateDes: Do des/tdes encryption/decryption with appointed key.

Before using the API, a DATA KEY must be loaded first by invoking EA_pinpad_ucLoadEncKey or injection in plaint-text (for Fixed key only).

Account data Encryption:

EA_pinpad_ucEncryptMagTrackData: Encrypt the account data with TDK

Before using the API, a TDK(Track data encryption key) must be loaded first by invoking EA_pinpad_ucLoadEncKey or injection in plaint-text (for Fixed key only).

PIN Encryption:

EA_pinpad_ucStartPinEntry: Start PIN entry and the encrypted PIN BLOCK will returned through the call-back parameter.

Before using the API, a PEK(PIN encryption key) must be loaded first by invoking EA_pinpad_ucLoadEncKey or injection in plaint-text (for Fixed key only).

2.2.4. PIN Collection and Processing

Interactive mode with application:

The terminal provides asynchronous PIN entry and processing API. That is, when entering PIN, the application only needs to set up PIN entry process, and then asynchronously listens the PIN event, including the key pressing event and PIN encryption/verification result event.

PINPAD supports two kinds of PIN input mode through physical keypad or touch screen. Finally which kinds of mode is used is determined by the terminal. Usually:

a) If the terminal supports physical keypad, only entering PIN through physical keypad is allowed.
b) If the terminal only has a touch screen, then entering PIN through touch screen is supported.

As for PIN entry via touch screen, the default UI style is provided by the firmware. There is a specialized system service to do UI interaction of PIN entry process. If the application is not satisfied with the style, he can apply to system service for a customized UI style. For the specific API, please see in the programing guide of PIN Entry UI Interactive Module.

Each PIN keypad event will be informed to the application no matter which kinds of PIN entry process. Applications can deal with it according to the actual situation. But if physical keypad is used, the application is responsible for the realization of the UI, such as the feedback prompt of the PIN entry.

Notes:

a) During PIN entry, application switching in not recommended. One the screen focus is switched to another application, the PIN entry process will be automatically cancelled and exited.
b) The application can use APIs provided by terminal to cancel PIN entry process at any time.
c) When the PINPAD is closed, the driver will quit the PIN entry process automatically.

Offline PIN Processing:

The terminal will automatically send the plain-text PIN to IC card for verification when finishing offline PIN entry. There is no chance for the application to access the plain-text PIN.
The application needs to send related information to PINPAD driver to finish verification. The information are IC card information, public key (if offline cipher-text key is used), IC card PIN verified APDU type and so on.

Related APIs:

EA_pinpad_ucStartPinEntry: to start PIN entry
EA_pinpad_ucCancelPinEntry: to cancel PIN entry
EA_pinpad_ucInjectPinEntryFuncKey: to insert a control button (OK or Cancel) during PIN entry
EA_pinpad_ucGetPinEntryInfo: to gain the results of PIN entry and PIN processing.
EA_pinpad_ucStartPinEntryAndVerifyWithIcCard: offline PIN entry and verification

2.2.5. PINPAD Permission Management

2.2.6. KAP Isolation of Multiple Application

KAP isolation of multiple applications means the key operation in a KAP won't affect keys in other KAPs, they are independent between each other. The mechanism is reflected in the PED's API. Each key-related operation is identified by a KAP ID.

The verification of multiple KAPs means it is needed to check the permission of the visitors when they access a certain KAP. Only apps with legal permission are allowed to access the KAP. As for how to check the permission of apps, it is realized through digital signature technology which will verify the integrity and validity of the target apps, and extract the app permissions authorized (the accessible KAP ID list) during its signing process.

The operating target KAP ID should be indicated in the parameter of the KAP key operation API provided by PED. When calling the API, it will compare the app permission (the accessible KAP ID list) with the target KAP to be operated and determine whether it is allowed.

In order to implement the multi-acquirers application isolation, the application should follow the security development guide below:

a) Developers:
   i. The POS application or the key loader application which is used to load keys should store keys in the private key KAP
   ii. It is recommended to use special key loader to load keys. The POS supports multi-application

isolation mechanism that will compare the KAP IDs which are allowed to access in KLD with that the target POS application want to access. If the result is same, the KAP ID can be accessed.

b) UKEY application and management: an administrator can apply signature UKEY for several acquirers. But before filling application forms, he should precisely define and distinguish the acquirers so that he can apply private KAP for different acquirers.

2.3. API Detailed Description

Please to refer to EPT-AND Application Development Guide.chm

2.4. Development Guide

For the detailed API description of PINPAD module, please see in EPT-AND Application Development Guide.chm. The section mainly focuses on how the most common payment transactions functions are programmed.

2.4.1. Open/ Close device

Usage:

Please refer to PINPAD Device and Handle

Related APIs:

EA_pinpad_ucGetAllPinpadsInfo: use to list all available PINPAD devices identified

EA_pinpad_ucOpenDevice: open a PINPAD device and to gain the handle

EA_pinpad_vCloseDevice: close a PINPAD device and to release the handle

For more API usage, please see EPT-AND Application Development Guide.chm

2.4.2. Plain-text Key Loading

Security Requirements:

According to PCI requirments, the initial plain-text key loading must fulfill the following requirements:

1) Plain-text key should not be accessed by any application, except that the application is part of firmware.
2) Plain-text key can only be stored in secure area or in the buffer temporarily (deleted immediately after used).
3) The interface inputting plain-text key should meet split-knowledge and dual control requirement. For some customers, key components are forbidden to be entered in the terminal manually.
4) Plain-text key loading must be done in secure environment.
5) For devices accepting plain-text key (PED), the operation of downloading the plaintext master key requires authorization. In case of no authorization, all existed previous keys should be erased before loading new keys.

Key Loading Authentication Mechanism:

For the 5 security requirements above, it is requested to verify the compliance of the key source before PED accepts plain-text keys. Or else, all verified keys in PED should be erased.

Hence, PED supports the following secure protection mechanisms of plain-text key loading:

1) Before loading plain-text key, it is requested to erase all previous-loaded keys in the KAP;
2) A password is used to authenticate the operator when loading plain-text key components. The method won't affect existed keys in PED.
3) LKI solution is used to load plain-text keys through a specific Key Loader. PED will authenticate the KLD before loading.
4) Use remote key injection system with mutual authentication.

Please select either of the above methods. LKI is strongly recommended.

Considering the complexity of the requirements, please contact the terminal vendor security team to

discuss the specific initial plain-text key loading solution. LANDI can provide secure and convenient key loading method according to clients' demand, including:

1) **Using LKI device to inject local keys.**
2) **Remote key injection through RKIS system**
3) **Customized according to client's demand.**

2.4.3. Work Key Loading

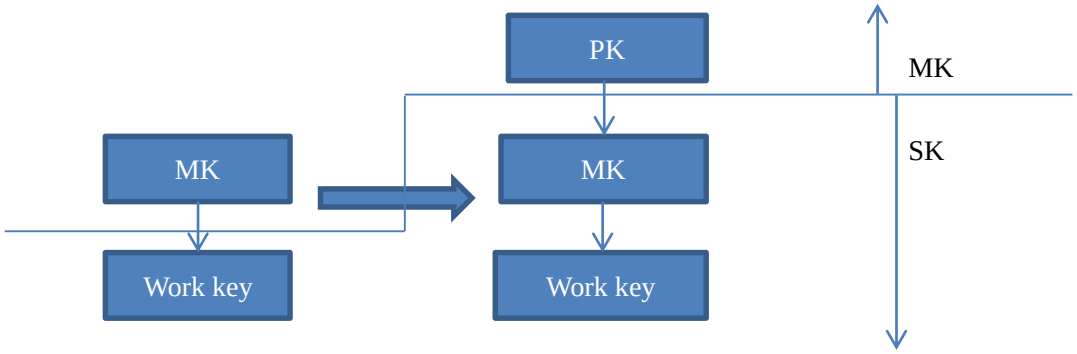
Usage:

Work key is the key used for final data protection or encryption. In MK/SK system it's SK, which may be used to encrypt PIN, calculate MAC, and encrypt magnetic track data etc.

In DUKPT system, work key is automatically generated during transaction and there is no need to load the work key; In FK system, only has one level key structure, then FK is used as the final work key to directly do PIN encryption/MAC calculation/magnetic track data encryption.

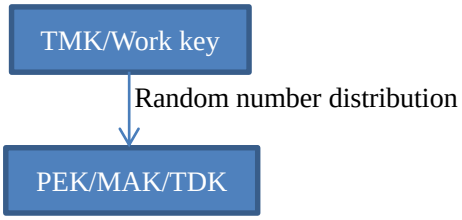
MK/SK system is quite flexible which is can be extended to level two, level three or even more. In theory, only the root key is considered as the MK, and all the lower level keys are SK (session key). "Master key" and "work key" just a common name called. It is mainly based on two-level or three-level key hierarchy in the traditional bank acquiring system. But due to the development of the emerging market, the common name has been difficult to apply to all occasions and is easy to cause ambiguity. For instant:

1) Originally, the traditional bank adopted "master key"+ "work key" 2 key hierarchy. For some reasons, later it changed to a three-level key system, that is, a protection key is added to the top of the original master key, and the master key is loaded after being encrypted. For the new key system, seriously, the MK should be the protection key and all the lower-level keys are SK. But people still habitually call the second level key as the master key.



2) Some clients, such as Lakala, the work key is not directly used to encrypt PIN but another new key dispersed from random number is used. For this situation, the dispersed key is also called "work key", so it is easy to mess up in concept.

SK



Based on reasons above, the APIs of the PINPAD module provided by LANDI only has the two keys: one is the initial key loaded in clear-text way. It is the MK, corresponding to the PK of the three-level key system or to the MK of the two-level key system; the other is the key loaded in cipher-text way, it is the SK, corresponding to the MK in second level of the three-level key system or to work key of two-level key system. This design suits to any kind of variants of the MK/SK system and is not limited by the usage of the traditional POS design for “master key” + ”work key”.

Each key is uniquely identified by its index and usage. No matter which kind of the key, the final use of the key depends on the purpose specified for the key when loading the key. Key usages include:

```
typedef char ET_KeyUsage;

#define EM_pinpad_KU_BDK ( (ET_KeyUsage) (0x00) )  //  BDK of DUKPT

#define EM_pinpad_KU_DUKPT_INIT_KEY ( (ET_KeyUsage) (0x01) )//  IK of DUKPT

#define EM_pinpad_KU_DATA_ENCRYPTION ( (ET_KeyUsage) (0x03) )// data encryption

#define EM_pinpad_KU_KEY_ENC_OR_WRAP ( (ET_KeyUsage) (0x0c) )// key encryption and pack

#define EM_pinpad_KU_KBPK ( (ET_KeyUsage) (0x0d) )// KBPK of TR-31

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_1 ( (ET_KeyUsage) (0x0f) )// MAC key

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_2 ( (ET_KeyUsage) (0x10) )// MAC key

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_3 ( (ET_KeyUsage) (0x11) )// MAC key

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_4 ( (ET_KeyUsage) (0x12) )// MAC key

#define EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_5 ( (ET_KeyUsage) (0x13) )// MAC key

#define EM_pinpad_KU_PIN_ENCRYPTION ( (ET_KeyUsage) (0x14) )// PIN encryption key

#define EM_pinpad_KU_TRACK_DATA_ENCRYPTION      ( (ET_KeyUsage) (0x20) ) // magnetic track data
encryption

#define EM_pinpad_KU_LAKALA_TMK                 ( (ET_KeyUsage) (0x30) ) // Lakala TMK, used to disperse
PEK/MAK/TDK
```

The recommended usages for specific key are as below:

Key System	Key Type	KeyUsage	ModeOfKeyUse
MK/SK	Transmitted protection key of master key	EM_pinpad_KU_KEY_ENC_OR_WRAP	EM_pinpad_MOU_DEC_OR_UNWRAP_ONLY
	master key	EM_pinpad_KU_KEY_ENC_OR_WRAP	EM_pinpad_MOU_DEC_OR_UNWRAP_ONLY
	Work key（PEK）	EM_pinpad_KU_PIN_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
	Work key（MAK）	EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_3	EM_pinpad_MOU_GENERATE_AND_VERIFY
	Work key（TDK）	EM_pinpad_KU_TRACK_DATA_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
	Work key （DEK）	EM_pinpad_KU_DATA_ENCRYPTION	EM_pinpad_MOU_ENC_DEC_WRAP_UNWRAP/ EM_pinpad_MOU_ENC_OR_WRAP_ONLY/ EM_pinpad_MOU_DEC_OR_UNWRAP_ONLY
	Lakala TMK	EM_pinpad_KU_LAKALA_TMK	EM_pinpad_MOU_DERIVE_KEYS
	Lakala PEK	EM_pinpad_KU_PIN_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
	Lakala MAK	EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_3	EM_pinpad_MOU_GENERATE_AND_VERIFY
	Lakala TDK	EM_pinpad_KU_TRACK_DATA_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
DUKPT	DUKPT BDK	EM_pinpad_KU_BDK	EM_pinpad_MOU_DERIVE_KEYS
	DUKPT IK/FK	EM_pinpad_KU_DUKPT_INIT_KEY	EM_pinpad_MOU_DERIVE_KEYS
	PEK derived by FK	EM_pinpad_KU_PIN_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
	MAK derived by FK	EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_3	EM_pinpad_MOU_GENERATE_AND_VERIFY
	TDK derived by FK	EM_pinpad_KU_TRACK_DATA_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
FK	PEK	EM_pinpad_KU_PIN_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
	MAK	EM_pinpad_KU_ISO_9797_1_MAC_ALGORITHM_3	EM_pinpad_MOU_GENERATE_AND_VERIFY
	TDK	EM_pinpad_KU_TRACK_DATA_ENCRYPTION	EM_pinpad_MOU_ENC_OR_WRAP_ONLY
	DEK	EM_pinpad_KU_DATA_ENCRYPTION	EM_pinpad_MOU_ENC_DEC_WRAP_UNWRAP/ EM_pinpad_MOU_ENC_OR_WRAP_ONLY/ EM_pinpad_MOU_DEC_OR_UNWRAP_ONLY

For the key table above, the MK/SK initial key, the BDK of DUKPT and all keys of FK system should be loaded in clear-text way. Others can be loaded in cipher-text way or be generated automatically.

Related API:

EA_pinpad_ucLoadEncKey: Load encrypted keys

For usage of more APIs, please refer to EPT-AND Application Development Guide.chm

Notes:

According to PCI requirements, keys with different usage cannot have same value. Please pay attention to the requirement when programming. Or else, the API may return EM_pinpad_SAME_KEY_VALUE_DETECTED.

1) EA_pinpad_ucLoadEncKey can be used to load cipher-text MK.

2) Key ID and the key Group Number

a) Key ID:

Key ID is a number in the key system which stands for a certain key. Please note that the SK ID cannot be the same under different MKs. For example, there is a SK1 whose KEY ID = 1 under MK1 (Key ID = 0); when adding a new MK2 (KEY ID=3), if its SK2 is KEY ID =1 then the original SK1 will be covered. SK won't be distinguished by MK ID. That is to say, Key ID is just a storage location number of the key. There no any connection between MK ID and SK ID. The application needs to well distribute and manage the key ID, especially when a KAP has multiple key systems.

b) Key group number:

The key storage space of PINPAD is dynamically allocated and is different form product and configuration. In theory, the available key group numbers cannot exceed the total capacity of the PINPAD. But please note that the available group numbers is not equivalent to "the numbers of KAP * the maximum group a KAP supports". For instance, the PINPAD supports at most 250 groups of keys, then each KAP can use at most 250 groups of keys. If the PINPAD can support 20 KAPs, the 20 KAPs share the 250 groups of keys together. It does not mean that each KAP can use a 250-group key independently.

c) The relations between them:

Please note that the key ID and the key group number which the terminal supports are not the same concept, they two do not have a certain connection. The key group number is the numbers of the storage unit in a PINPAD, it is like the numbers of rooms in a building; but the key ID is just an identifier like room number. The number is not necessarily increased from 1 to the maximum. In a KAP, an application can arbitrarily designate a key ID as long as it does not exceed the maximum 0xFFFF. Therefore, the key ID may larger than the key group number. For example, a PINPAD can support 250 sets of key at most, but the key ID can be bigger than 256 such as 1000. The key sets are equivalent to the entire storage space of PINPAD, but the key ID is just a unique identifier of key. It is like a building, whose room number can be 3206, but it does not mean that the building has 3206 rooms.

2.4.4. Online PIN Entry and Encryption

Usage:

The terminal provides asynchronous interfaces for PIN entry and encryption. The application starts PIN entry, and each time when there is a PIN-related event such as key-press event or PIN-entry-finished event happens, PINPAD API will inform application through callback function. Application can realize various human-device interaction functions based on PIN-related event, such as the echo prompt of PIN entry.

All PINs are reported to application with "*" through the whole PIN entry process. It is impossible for the application to touch any clear-text PIN but can only know about the numbers of PIN.

Due to different clients have their own preference, the UI of PIN entry can be designated by application according to clients' demand. The driver provides a default UI for the device with a touch screen. A specific interface can be used to realize customized style if the application wants to change the UI or there is another customized demand.

Related APIs:

EA_pinpad_ucStartPinEntry: Start PIN Entry and encrypt PIN BLOCK

EA_pinpad_ucStartTouchScreenPinEntry: Start PIN Entry and encrypt PIN BLOCK for touch screen PINPAD.

For usage of more APIs, please refer to EPT-AND Application Development Guide.chm

Notes:

1) For the device with a physical keyboard to input PIN, the application should design and realize the UI itself.
2) The rules about the Cancel button, the Clear button and the OK button.
   a) The Cancel button is used to clear all inputted PIN digits. If the number = 0, then return directly; If input on the half way, press the Cancel button for the first time to clear all the inputted digits and the second time to return.

      If clients do not want to support the return function, they can configure the parameter of ET_PinEntryReactionMode to disable the function, that is, press Cancel button to quit PIN entry process.

   b) The Clear button is used to delete the previous inputted PIN digit. Delete a digit per time press till all inputted digits are cleared.

   c) The OK button is used to end PIN entry process. For the situation supporting no PIN entry which means the list of allowed PIN length contains 0x00, pressing the button can return directly. If the numbers of the inputted PIN digits is not expected, meaning it is not within the range of the allowed PIN length list, then pressing the button will be considered as an invalid action.

      For example, if the content of the allowed PIN length list is:"\x00\x06", the PIN entry process can be returned not only when there is no PIN entry, but when a 6-digit PIN has been inputted. The action of pressing OK button will be ignored when the numbers of the inputted PIN digits is neither 0 nor 6.

      When the content of the allowed PIN length list is:"\x04\x06", the PIN entry process can be returned not only when a 4-digit PIN has been inputted, but when a 6-digit PIN has been inputted. This pressing OK button action cannot quit the process when there is no PIN inputted. The action of pressing OK button will be ignored when the numbers of the inputted PIN digits is neither 4 nor 6.

   d) The PIN entry process also supports an automatic function to quit PIN entry process, that is, when there is only one value in the allowed-PIN length list, the process will quit automatically once the length of the inputted PIN digits reaches the value.

If clients do not want to support the return function, they can configure the parameter of ET_PinEntryReactionMode to disable the function, that is, PIN entry process automatically quit.

3) When setting up PIN entry process, an 8-byte card number, that is, a 16-digit card number, must be uploaded**.** The driver of PINPAD will encrypt PIN and card number. The encryption format supports Format0/3/4 etc. specified by ANSI X9.8\ISO9564. At present, most of the function use Format0. The process for PINPAD driver to encrypt card number and PIN is as followed:
   a) Restore the 8-byte zipped card number to a 16-byte PAN, cut out the middle 12-byte data (remove the first 3 bytes and the last 1 byte), fill with 4-byte 0x00 in the front of it to gain a 16-byte data, and finally zip the data to a 8-byte BLOCK1.
   example：
   16-digit card number：1234567890123456
   The zipped 8-byte card number（BCD）: \x12\x34\x56\x78\x90\x12\x34\x56
   The restored 16-byte card number：\x01\x02\x03\x04\x05\x06\x07\x08\x09\x00\x01\x02\x03\x04\x05\x06
   The cut-out PAN：\x04\x05\x06\x07\x08\x09\x00\x01\x02\x03\x04\x05
   The filled-in PAN：\x00\x00\x00\x00 \x04\x05\x06\x07\x08\x09\x00\x01\x02\x03\x04\x05
   After Compressed：\x00\x00\x45\x67\x89\x01\x23\x45
   b) Fill the PIN according to Format0 to gain BLOCK2
   example：
   plain-text PIN：123456
   The filled-in PIN：\x06\x12\x34\x56\xFF\xFF\xFF\xFF
   c) XOR BLOCK1 and BLOCK2 bit by bit to gain the final PIN BLOCK：
   PIN BLOCK = BLOCK1 ^ BLOCK2
   example：
   plain-text PIN：123456
   16-digit card number：1234567890123456
   BLOCK 1: *\x00\x00\x45\x67\x89\x01\x23\x45*
   BLOCK 2: *\x06\x12\x34\x56\xFF\xFF\xFF\xFF*
   PIN BLOCK: *\x06\x12\x71\x31\x76\xfe\xdc\xba*
4) To prevent PIN exhaustive attack, it is recommended for PIN BLOCK to use Format3 or Format4. And it is recommended to use automatic filling format but not the fixed filling format such as Format0 which fills with fixed data like 0xFF.

2.4.5. Offline PIN Entry and Verification

Usage:

PINPAD module provides a specific API to realize offline PIN entry and verification. The API can be used to complete PIN entry and verification safely inside the PINPAD, so the offline PIN won't be revealed to application. When setting up entry process, the inputted PIN will be automatically stored. When the input ends, the PIN will be automatically sent to the IC card for verification by the transmitted information of the API parameters.

To use the API, the required information includes not only the parameters similar to the online PIN entry

interface, but the following information related to offline PIN verification:

1） The verifying card number

2） PIN format (plain-text or cipher-text)

3） The challenge random number.

4） The public key of IC card

5） The APDU message type of the verified PIN

Related APIs：

EA_pinpad_ucStartPinEntryAndVerifyWithIcCard: Offline PIN entry and verification (physical keypad PIN entry)

EA_pinpad_ucStartTouchScreenPinEntryAndVerifyWithIcCard： Offline PIN entry and verification (touch screen PIN entry)

EA_pinpad_ucStartOfflinePin: set up offline PIN entry (PIN will be encrypted and cached into the security area after PIN entry, it is needed to call EA_pinpad_ucVerifyOfflinePin within the specified time to send the cached PIN to IC card for verification)

EA_pinpad_ucVerifyOfflinePin：to verify the offline PIN

For usage of more APIs, please refer to EPT-AND Application Development Guide.chm.

Notes:

1) The application has to use the APIs the complete offline PIN verification without implementing PIN entry function itself. Because the various self-realized offline PIN entry functions may not be able to fulfill PCI etc. security requirements. These self-realized functions are not under the protection of LANDI, so LANDI won't bear any liability if there is any security event happened.
2) For the usage of offline PIN, the application do not need to care about the above details of the implementation process if the related APIs of the latest EMV light core are used in the application architecture. Because the EMV light core module has already completed all processes of the offline PIN.

2.4.6. Magnetic Track Data Encryption

Usage:

PINPAD module provides a specific API to realize the magnetic track data encryption. The application can call the API to encrypt PAN and send to the background of the acquiring institution.

Related API:

EA_pinpad_ucEncryptMagTrackData: Encrypt the account data with TDK

For usage of more APIs, please refer to EPT-AND Application Development Guide.chm.

Notes:

1) According to PCI requirements, when programming, please pay attention to the fact that the magnetic key cannot have the same value with the PIN encryption key. If cannot find the key, please double check whether it is because of the reason which leads to fail to load the key.
2) For the PAN encryption, if the MK/SK or FK system is used, please make sure to at least use a 24-bit TDES key. Or else, it cannot fulfill the requirements of PCI specification etc. If the DUKPT is used, it

is not subject to this limitation because each transaction uses a new key under the DUKPT system.

2.4.7. MAC Generation/Verification

Usage:

PINPAD module provides a specific API to realize the MAC calculation, it supports ISO9797 MAC algorithm 1 which is based on single DES key, that is ANSI X9.9 and ISO9797 MAC algorithm 3 which is based on TDES key, that is ANSI X9.19

PINPAD module supports the segmentation calculation MAC for the batch data (but do not support filling mode2). Users can initialize the algorithm through EA_pinpad_ucMacInit, call EA_pinpad_ucMacLoadData to load data for several groups and times, and finally call EA_pinpad_ucMacGenerate to output the final MAC or call EA_pinpad_ucMacVerify to verify the MAC.

Relates APIs:

EA_pinpad_ucMacInit: Init MAC parameters

EA_pinpad_ucMacLoadData: Load data to be calculated with MAC algorithm

EA_pinpad_ucMacGenerate: Generate MAC using loaded data

EA_pinpad_ucMacVerify: Compare the generated MAC data with the appointed MAC data

For usage of more APIs, please refer to EPT-AND Application Development Guide.chm

Notes:

1) According to PCI requirements, when programming, please pay attention to the fact that the MAC key cannot have the same value with the PIN encryption key. If cannot find the key, please double check whether it is because of the reason which leads to fail to load the key.
2) When loading MAC key, it is allowed to set the usage of MAC key to be “only generate MAC”, or “only for verification”, or both of these two. When using related APIs, they will check whether the corresponding MAC key usage matches. For example, when using EA_pinpad_ucMacGenerate, the usage of the MAC key cannot be “only for verification”, same like when using EA_pinpad_ucMacVerify, the usage of MAC key cannot be “only generate MAC”.

2.4.8. Other Data Encryption and Decryption

Usage:

PINPAD module provides a specific API to realize the DES encryption and decryption of the common data. So the API can be used to encrypt or decrypt message.

A set of DES/TDES key is needed to load before using the API. The usage is similar to the DES/TDES usage of the algorithm library and it can support encryption, decryption, TECB, TCBC etc.

Related APIs:

EA_pinpad_ucCalculateDes: Do des/tdes encryption/decryption with appointed key.

For usage of more APIs, please refer to EPT-AND Application Development Guide.chm

Notes:

1) According to PCI requirements, when programming, please pay attention to the fact that the

encryption/decryption key of the common data cannot have the same value with the PIN encryption key. If cannot find the key, please double check whether it is because of the reason which leads to fail to load the key.

2.5. Security Development Guide.

Notes:

This document provides important security guides for developers on how to use the POS terminal correctly. In order to protect asserts of merchants, acquirers and cardholders, the developers have responsibility to follow the guidance while programming and must read these guides carefully.

If the developers ignore these guides and use other means to achieve a payment transaction, they should take all security related responsibilities while security is compromised. The application manager and auditor must review and inspect the application carefully before signing the application. The POS manufacturer won't accept any security responsibility for the loss to the merchants, the acquirer or the cardholder since it is caused by the application. LANDI only provides security commitments and assurances to organizations who compliance with the following security development requirements. Any other means to achieve payment transaction without following security requirements belong to unauthorized activities, which are not protected by LANDI

1) Please be sure to use the correct API to complete relevant encryption functions. If not, it may result in the situation that the application doesn't meet related requirements of security specification of PCI.
   a) Online PIN entry and encryption:
      Developers have to use EA_pinpad_ucStartPinEntry as the function to enable online PIN entry and encryption. Any other method such as soft encryption is prohibited to implement PIN entry and encryption.
   b) Offline PIN entry and verification
      Developers have to use EA_pinpad_ucStartPinEntryAndVerifyWithIcCard or EA_pinpad_ucStartOfflinePin + EA_pinpad_ucVerifyOfflinePin as the function to enable offline PIN entry and verification. Any other method is prohibited, such as entering PIN on touch screen through a standard android input method or through a self-programming PIN-entry interface, which may not meet the security requirements.
   c) Account data encryption
      - Developers must encrypt the account data of cardholders. If the acquirers' background does not support the cipher-text account data, the acquirer must fully assess the risks and ensure the cardholder's legitimate rights and interests. LANDI won't take related risks. As the commissioned developer of the acquiring organization, application developers are responsible for maintaining the interests of the acquirer and ensuring the security of the cardholder's data. The application developers are obliged to inform the acquirer of the risk in case where the acquirer does not support the encrypted account data of cardholder.
      - Developers have to use EA_pinpad_ucEncryptMagTrackData to encrypt cardholder's account data. Other unauthorized methods are prohibited for data encryption, such as soft encryption or other methods.

2) The application on terminals should try to store user's sensitive data to a minimum
Restrict the data storage as well as the reserved time to keep it at a level of exactly meeting the requirements of business, law, industry regulations and administrative provisions. In fact, terminal applications shall not reserve any user's sensitive data, including but not limited to magnetic track information of bank cards, card number, PIN, CVN2 (even encrypted) etc.

3) Do not store sensitive data
If the whole transaction system has no specific requirements, the applications on terminals shall not store any sensitive data after the identity verification is completed even if the data is encrypted, including track information of bank cards, card number, PIN, CVN2 etc.

4) The application can only contain the needed functions. There is no intentionally or unintentionally hide backdoor. Backdoor includes but no limited to the followings:
    a) The software, without prior consent of applicant and the user of terminal product, automatically connects to the internet or communicates with the network port without confirmation from Applicant or user, to transfer any information of the terminal product.
    b) Open the listening port without permission from Applicant and the user of terminal product.
    c) Use ICMP protocol as a backdoor to perform illegal communication to transfer any information on POS terminal without permission from Applicant and the user of terminal product.
    d) Use libpcap alike tool to construct private network data message to perform illegal communication without permission of from Applicant and the user of terminal product.
    e) Implant backdoors into webpage or threads to transfer any information of POS terminal without permission from Applicant and the user of terminal product.
    f) Store any input event or frame buffer in another space without permission from Applicant and the user of terminal product.
    g) Listen to all kinds of input /output devices in the system without permission from Applicant and the user of terminal product.
    h) Modify (including promote and reduce) the access rights to any software in terminal products without permission from Applicant and the user of terminal product.
    i) Hide the thread with the highest access right without permission from Applicant and the user of terminal product.
    j) Simulate PIN input interface and operation but not call the secure PIN input function provided by LANDI terminal SDK without permission from Applicant and the user of terminal product. Such simulation may mislead the users to input their PINs and that result in PIN leakage.

5) Once the terminal enters the input-PIN interface, the previous transaction content has been determined and the application cannot provide method to change the transaction content (including the amount of money, goods content, etc.)

6) The application rewrites some C library function (if applicable):
When compiling, the memset and the memcmp may be optimized by the compiler, leaving a security risk. So it is recommended:
    a) when compiling the underlying code, the following option must be added:

LOCAL_CFLAGS += **-fno-dse**

b) The following functions are recommended to be used instead of "memcmp" and "memset" separately:

```
uchar EI_ucMemcmp(void *pvInput1, void *pvInput2, uint uiLen)
{
    uchar *   pucInput1 = (uchar*)pvInput1;
    uchar *   pucInput2 = (uchar*)pvInput2;
    uchar   ucRet;
    volatile uint   i;

    ucRet = EM_SUCCESS;
    for(i=0;i<uiLen;i++)
    {
       if(pucInput1[i] != pucInput2[i])
       {
          ucRet = EM_ERROR;
       }
    }
    return ucRet;
}

void EA_vMemset(void *s, uchar ucC, uint uiLen)
{
    uchar *   pucInput1 = (uchar*)s;
    volatile uint   i;
    for(i=0;i<uiLen;i++)
    {
       pucInput1[i] = ucC;
    }
    pucInput1 = NULL;
    ucC = 0;
}
```

7) Clear sensitive data buffer

Sensitive data includes plain-text magnetic-track information, account data, PINBLOCK, keys, passwords etc. After completing sensitive operations, the cached data of the sensitive variable and the cipher-text sensitive data must be cleared to be 0. The sensitive variables include the global variable and the local variable in the function.

8) The use of account data

Account data can be accessed by an authenticated application, which must meet the following requirements:

a) The account data won't be displayed or output in plain text.

b) The display or output of PAN must be truncated, up to show the first six characters and the last four

characters, the remaining characters are replaced by the character "*".

c) The user data must be encrypted when using.

d) Cached data must be cleared to be 0 when the transaction is done.

e) When the transaction timeout occurs, the cached data must be cleared up and the recommended timeout period is less than 2 minutes.

Note: the cached data includes plain-text or cipher-text card data.

		Data Element	Storage Permitted	Render Stored Account Data Unreadable per Requirement 3.4
Account Data	Cardholder Data	Primary Account Number (PAN)	Yes	Yes
		Cardholder Name	Yes	No
		Service Code	Yes	No
		Expiration Date	Yes	No
	Sensitive Authentication Data[1]	Full Magnetic Stripe Data[2]	No	Cannot store per Requirement 3.2
		CAV2/CVC2/CVV2/CID	No	Cannot store per Requirement 3.2
		PIN/PIN Block	No	Cannot store per Requirement 3.2

9) The PIN entry interface should be separated from the non-PIN entry interface.

The entry of PIN and non-PIN cannot be done in the same interface to avoid the situation that the user enters the PIN to the non-PIN input box, leading to leakage of sensitive information.

10) Key Injection

For specific key injection process, please refer to PED Initial Key Loading User Guide

PED Initial Key Loading User Guid