(DUKPT) Key Filling Method Description

Key store

The device stores ten sets of keys, each of which has the following format:
keyIndex 0: { { trackKsn ,trackIpek } , { emvksn,emvIpek } , { pinKsn,pinIpek } }
keyIndex 1: { { trackKsn ,trackIpek } , { emvksn,emvIpek } , { pinKsn,pinIpek } }
.....
keyIndex 9: { { trackKsn ,trackIpek } , { emvksn,emvIpek } , { pinKsn,pinIpek } }

trackKsn : ksn of encrypted magnetic stripe card data, length 10 bytes
trackIpek : ipek for encrypting magnetic stripe card data, length 16 bytes
emvKsn: ksn for encryption chip card and NFC data, length 10 bytes
emvIpek : ipek for encrypting chip cards and NFC data, length 16 bytes
pinKsn: ksn of the encrypted password, length 10 bytes
pinIpek : ipek for encrypted password, length 16 bytes

Key injection method description:

(1) ciphertext command injection

```
doUpdateIPEKOperation(String keyIndex, String trackksn, String
trackipek, String trackipekCheckvalue, String emvksn
        , String emvipek, String emvipekCheckvalue, String pinksn,
String pinipek, String pinipekCheckvalue)
```

keyIndex : Updates the key of the corresponding corner in the device, ranging from 0 to 9.
trackKsn : ksn of encrypted magnetic stripe card data, length 20, format hex string, plain text
trackIpek : ipek for encrypting magnetic stripe card data, length 32, format hexadecimal string, using masterKey to encrypt plaintext trackIpek
trackCheckValue : checksum value of the incoming ipek, length 16, format hexadecimal string, encrypted with plaintext trackIpek 8 bytes 0
emvKsn: ksn for encryption chip card and NFC data, length 20, format hexadecimal string, plaintext
emvIpek : Encrypt the chip card and NFC data ipek, length 32, format hex string, use the masterKey to encrypt the plaintext emvIpek
emvCheckValue : checksum value of incoming ipek, length 16, format hexadecimal string,

encrypted with plaintext emvIpek 8 bytes 0
pinKsn: ksn of encrypted password, length 20, format hexadecimal string, plaintext
pinIpek : ipek of encrypted password, length 32, format hex string, use masterKey to encrypt plaintext pinIpek
pinCheckValue : checksum value of the incoming ipek, length 16, format hexadecimal string, encrypted with plaintext pinIpek 8 bytes 0

Note: Encryption uses TDES algorithm, double long key, ecb mode encryption
Ipek's update only uses TMK (masterKey), which guarantees security in the device's secure memory.

TMK: 32-bit hex string
Assume:
TMK value is 00000000000000000000000000000000
Update the IPEK with device index 0 to 00000000000000000000000000000000
as follows:
1. TDES encryption uses TMK as the key to encrypt trackIpek in ECB mode and obtains "8CA64DE9C1B123A78CA64DE9C1B123A7"

2. TDES encryption uses the trackIpek plaintext as the key to encrypt 8 bytes in ECB mode to get "8CA64DE9C1B123A7"
The following test uses trackIpek, emvIpek, pinIpek to update to the same value, officially, customers can update to different values according to their needs.

```
pos.doUpdateIPEKOperation(
"00","00000000000000000000","08D7B4FB629D088508D7B4FB629D0885","8CA64DE9C1B123A7",

"00000000000000000000","08D7B4FB629D088508D7B4FB629D0885","8CA64DE9C1B123A7",

"00000000000000000000","08D7B4FB629D088508D7B4FB629D0885","8CA64DE9C1B123A7");
```

(2) Digital envelope injection

Note: The customer's public key, dspread private key is injected during the production process of the device.Currently, Digital Envelope Support Updates TMK, IPEK

Digital envelope description:
1. Sender package information
2. Randomly generate a symmetric encryption key to encrypt information

3. Pos public key encryption random symmetric key generated in the second step
4. Customer private key signature The encrypted information generated in the second step + the encrypted key in the third step
5. Group packet transmission pos. Signature information + encryption information + encryption symmetric key

```
                    int keyIndex = getKeyIndex();
                    String digEnvelopStr = null;
                    try {
                        DukptKeys dukptKeys = new DukptKeys();
                        dukptKeys.setRSA_public_key(clearPubKeyModel);
//                          TMKKey tmkKey = new TMKKey();
//                          tmkKey.setRSA_public_key(clearPubKeyModel);
                        digEnvelopStr =
Envelope.getDigitalEnvelopStrByKey(getAssets().open("rsa_private_key_
pkcs8_test.pem"), dukptKeys, Poskeys.RSA_KEY_LEN.RSA_KEY_1024,
keyIndex);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    pos.udpateWorkKey(digEnvelopStr);
```

i

i DUKPT process analysis:

basic concept:

1. Base Derivation Key (BDK): The root key of the DUKPT key system is generally a double or three times longer T-DES key. Dinghe uses double the length of 16 bytes.

2. KSN (Key Serial Number): A series of 80-bit (20 hexadecimal digits) serial numbers, consisting of 59-bit IKSN (Initial Key Serial Number) and 21-bit EC (Encryption Counter).

3.IPEK: Encrypt emv data, magnetic stripe card data and pin key

Trading simulation:

The decryption end interacts with the POS, and the Initial PEK and Initial KSN are saved in the POS.

Pos processing:

1> Current KSN = IKSN and EC++

2> Current PEK = PEK_Derive(Initial PEK, Current KSN)

3> Encrypted PIN = T-DES (Opr=Encrypt, Current PEK, Clear PIN)

4> Put the Current KSN and Encrypted PIN in the transaction message and send it to the mobile terminal.

Decryption processing (the decryption end is usually the mobile application or the application background):

1> Initial PEK = PEK_Derive (BDK, KSN with EC=0)

2> Current PEK = PEK_Derive(Initial PEK, Current KSN)

3> Clear PIN = T-DES (Opr=Decrypt, Current PEK, Encrypted PIN)

4> Subsequent transaction processing