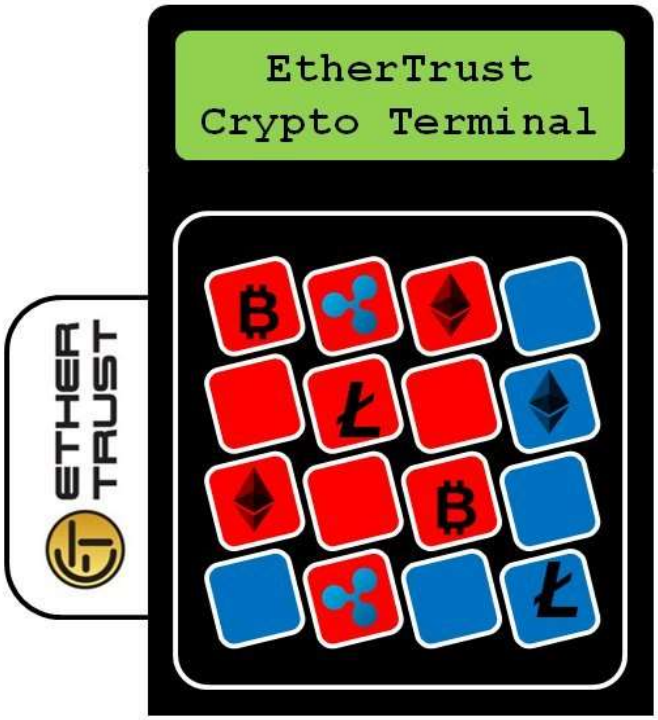
EtherTrust CryptoTerminal
v0.9
Firwmware 0.8d4



1 Content

2 INTRODUCTION 7

2.1 The Crypto Terminal 7

2.2 The Crypto Currency SmartCard (CCSC) 7

2.3 Generating Ethereum transaction from the terminal 8

2.4 Generating transactions with BTOOLS 8

2.5 Overview of the terminal board 9

3 CRYPTOTERMINAL SERIAL COMMANDS 10

3.1 Wrong Command 10

3.2 Empty Command 10

3.3 reboot 10

3.4 echo 10

3.5 disp 10

3.6 edit 11

3.7 enter 11

3.8 debug 11

3.9 nodebug 11

3.10 prompt 11

3.11 noprompt 11

3.12 on 12

3.13 off 12

3.14 status 12

3.15 changeadm 12

3.16 changeuser 12

3.17 changeuser2 13

3.18 adm 13

3.19 user 14

3.20 user2 14

3.21 computekey 14
3.22 genkey 15
3.23 setseed 15
3.24 getseed 16
3.25 getpub 16
3.26 btc 16
3.27 hash160 16
3.28 eth 16
3.29 setkey 17
3.30 setpp 17
3.31 getpriv 18
3.32 setlabel 18
3.33 getlabel 18
3.34 sign 18
3.35 recover 19
3.36 signr 19
3.37 signin 19
3.38 signout 20
3.39 signoutr 20
3.40 signdo 20
3.41 settrans (generation of ether transaction) 21
3.42 signt 21
3.43 gettrans 22
3.44 duplicate 22
3.45 read 23
3.46 write 23
4 CRYPTOTERMINAL MENUS 24
4.1 Boot Message (BM) 24
4.2 Idle Menu (IM) 24

4.3 Boot Menu (BM) 24

4.4 A Menu (AM) 24

4.5 Mode Menu (MM) 24
4.5.1 Mode ON 25
4.5.2 Mode OFF 25
4.5.3 Mode ADM 25
4.5.4 Mode User 25
4.5.5 Mode User2 26

4.6 Key Menu (KM) 26
4.6.1 Key Index 26
4.6.1.1 Sign 26
4.6.1.2 Genkey 27
4.6.1.3 Record Index 27
4.6.2 Key Menu2 (KM2) 27
4.6.2.1 CKey (Compute Key) 27
4.6.2.2 SetTree 28
4.6.2.2.1 SetTree 28
4.6.2.2.2 Read 28
4.6.2.2.3 GenerateTree 29
4.6.2.3 GetL (Get Label) 29
4.6.2.4 SetL (Set Label) 29
4.6.3 Key Menu3 (KM3) 30
4.6.3.1 SetPr (Set Private Key) 30
4.6.3.2 GetPr (Get Private Key) 30
4.6.3.3 GetPub (get public key) 31
4.6.3.3.1 Raw 31
4.6.3.3.2 BTC 31
4.6.3.3.3 Hash160 31
4.6.3.3.4 Hash160 32
4.6.3.4 GetT (Get Tree Seed) 32

4.7 Duplicate Menu (Dup) 32
4.7.1 Remove Card 32
4.7.2 Insert New Card 32
4.7.3 Enter ADM PIN 32

4.8 Transaction Menu (Tr) 33
4.8.1 Ether Menu (EM) 33
4.8.1.1 Create 33
4.8.1.1.1 Address 33
4.8.1.1.2 Ether 33
4.8.1.1.3 Data (text) 33
4.8.1.1.4 Nounce 34
4.8.1.1.5 Gas Price 34
4.8.1.1.6 Gas Limit 34
4.8.1.1.7 Data (Hexadecimal) 34
4.8.1.2 Delete 34
4.8.1.3 Edit 34
4.8.1.3.1 Address 34
4.8.1.3.2 Ether 34
4.8.1.3.3 Data Text 35
4.8.1.3.4 Nounce 35
4.8.1.3.5 Gas Price 35
4.8.1.3.6 Gas Limit 35

4.8.1.3.7 Data Hexadecimal 35
4.8.1.4 Sign 35

5 BTOOLS SCRIPT COMMANDS 36

5.1 Starting a serial terminal 36

5.2 Script Commands 36
5.2.1 start 36
5.2.2 on 36
5.2.3 off 36
5.2.4 adm 36
5.2.5 user 37
5.2.6 user2 37
5.2.7 getpub 37
5.2.8 sign 37
5.2.9 signout 37
5.2.10 setpub 37
5.2.11 signdo 37
5.2.12 signin 38
5.2.13 raw 38

6 BTOOLS CRYPTOTERMINAL SCRIPT EXAMPLES 38

6.1 Ethereum Transaction 38
6.1.1 CryptoTerminal script 38
6.1.2 Transaction Script 38

6.2 Ethereum Signature Request 39
6.2.1 CryptoTerminal Script 39

6.3 Ethereum Transaction Script 39

6.4 Ethereum Signature Read 39
6.4.1 CryptoTerminal Script 40
6.4.2 Transaction Script 40

7 THE CRYPTO CURRENCY SMARTCARD 41

7.1 The Select Command 41

7.2 The Verify UserPin Command 41

7.3 The Verify UserPin2 Command 42

7.4 The Verify AdminPin command 42

7.5 The ChangePin command 43

7.6 The GetStatus command 43

7.7 The Write Command 44

7.8 The Read Command 44

7.9 The Clear KeyPair & InitCurve Command 45

7.10 The InitCurve & InitTree Command 46

7.11 The Generate KeyPair Command 47

7.12 The Dump KeyPair Command 48

7.13 The GetInfo command 50

7.14 The Get KeyParameter Command 50

7.15 The Set KeyParameter Command 51

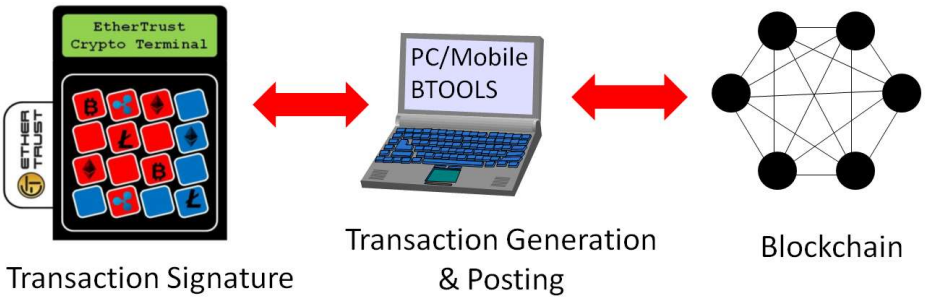
7.16 The SignECDSA command 52

2 Introduction

Blockchain and its distributed ledger architecture will bring spectacular disruption across all industries. However, security, specifically key protection, remains a crucial challenge, as demonstrated by recent hacks.

Leveraging its extensive expertise in cryptography and secure element technologies, EtherTrust brings proven bankcard security to Blockchain transactions. With EtherTrust Crypto Terminal, Blockchain transactions are authorized through an off-line "Chip and PIN" validation.

Keys are processed entirely within the smartcard secure element, and the smart card is never on-line. Furthermore the terminal firmware can be securely flashed in seconds prior the transaction for added protection.



2.1 The Crypto Terminal

The Crypto Terminal is equipped with an USB port supporting the *USB Serial* protocol. It includes a smartcard reader, and delivers the following services:
- Generation and secure storage of cryptographic keys;
- Generation of cryptographic signatures in synchronous mode (controlled by a PC or mobile device) or asynchronous mode (without a connection to a PC or mobile device);
- All PINs are entered directly on the CryptoTerminal, as on a payment-card terminal;
- All signatures require confirmation on the CryptoTerminal;
- Very High Security:
    - The removable secure element (smartcard) is EAL5+ certified;
    - *Bare Metal*, all the Crypto Terminal software, including USB driver can be securely flashed at any time.

2.2 The Cryptocurrency SmartCard (CCSC)

The Cryptocurrency Smartcard (CCSC) can generate, derive according to the BIP32 standard, or import up to 16 elliptic-curve keys for producing ECDSA signatures for cryptocurrencies such as Bitcoin and Ethereum.
The Cryptocurrency Smartcard (CCSC) application, with AID 010203040500, uses three PINs: administrator, user, and user2. The default administrator PIN consists of eight zeros (3030303030303030); the user and user2 PINs each consist of four zeros (30303030).
A 16 KB nonvolatile read/write memory area, protected by a dedicated PIN (User2), is available for storing sensitive information.

2.3 Generating Ethereum transaction from the terminal

The serial command "settrans" creates an Ether transaction using the following seven parameters:
- Key Index, the key to be used for the transaction signature
- Nonce, the nonce of the transaction, an integer positive value, starting from zero, and incremented by one after every transaction.
- GasPrice, the gas price to be used for the transaction in GWEIs
- GasLimit, the maximum instructions to be computed for the transaction, leading to a maximum cost of GasLimit x GasPrice
- Recipient Address: the recipient's Ether address (20 bytes, or 40 hexadecimal digits).
- Amount: the transaction amount in Ether, expressed in decimal notation (for example, 0.0).
- Data, the data associated to the transaction in quotes, text format

The transaction is thereafter written in the non volatile memory of the terminal and is reviewed by the user, before signature.

A signed transaction is retrieved using the "gettrans" serial command.

The transaction, encoded in hexadecimal, can then be copied and pasted into a web interface for broadcasting transactions.



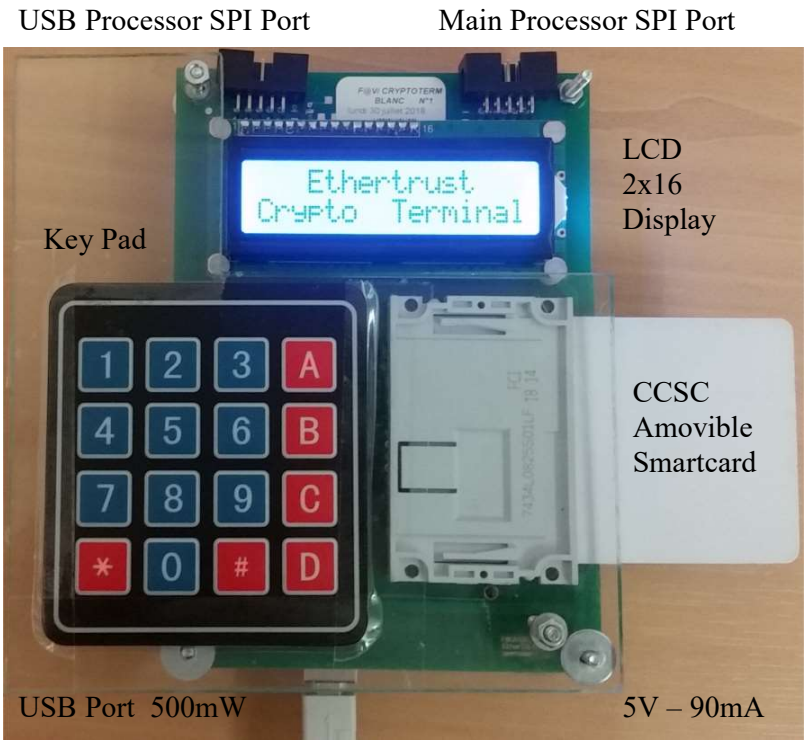
2.4 Generating transactions with BTOOLS

BTOOLS is open software available for Windows, which generates transactions for Bitcoin and Ethereum cryptocurrency platforms. In order to provide a strong security it supports the CryptoTerminal, used for secret key generation and storage, and for the computing of elliptic curve (ECDSA) signature. It is based on the OpenSSL library.

BTOOLS provides the following services:
- Bitcoin address generation (mainnet and testnet);
- Ethereum address generation;

- Bitcoin transaction generation;
- Ethereum transaction generation;
- Simple Bitcoin node client;
- Bitcoin transaction (via the Bitcoin client);
- Ethereum transaction (via WEB APIs);
- Crypto Terminal scripts for key generation and transaction signature (synchronous and asynchronous signature).

2.5 Overview of the terminal board



3 CryptoTerminal Serial Commands

The CryptoTerminal is equipped with an USB port supporting the *USB Serial* protocol.

Serial Configuration: 115200 Bauds, 8 bits, 1 Stop, No Parity
All commands end with the CR (carriage return) and LF (line feed) characters.
An error occurred if the first token of the response is the string ERROR.

3.1 Wrong Command

Action: Command Error

Cmd: WrongCommand CrLf
The first token of error response is **ALWAYS** ERROR
Response: ERROR WrongCommand CrLf

3.2 Empty Command

Action: Empty Command

Cmd: CrLf
Response: ERROR No Command ! CrLf

3.3 reboot

Action: Reboot the CryptoTerminal
Cmd: reboot CrLf
Response: Boot label
!!! The boot response begins by the string "OK ", and is not ended by the CrLf sequence.
!!! Upon reboot the terminal is in the debug mode.

3.4 echo

Action: Terminal Test, force the *nodebug* mode, *prompt* (if any) remain unchanged.

Cmd: echo CrLf
Response: OK CrLf

3.5 disp

Action: Test LCD for debugging purposes; display text at x,y coordinates for 5 seconds

Cmd: disp x-value (0..15) y-value (0...1) text CrLf
Response: OK CrLf
Response: ERROR CrLf

3.6 edit

Action: Test LCD text editor for debugging purposes.

Cmd: edit number-of-characters [Optional text]
Response: [text] CrLf
Response: ERROR CrLf

3.7 enter

Action: Test KEYPAD for debugging purposes; enter data at x,y coordinates (60-second timeout).

Cmd: enter x-value (0..15) y-value (0...1) number-of-characters CrLf
Response: [text] CrLf
Response: ERROR CrLf

3.8 debug

Action: Switch to Debug Mode

Cmd: debug CrLf
Response: OK CrLf

3.9 nodebug

Action: Switch to non-debug (normal) mode.

Cmd: nodebug CrLf
Response: OK CrLf

3.10 prompt

Action: Display the prompt character at the end of the response (>).

Cmd: prompt CrLf
Response: OK CrLf

3.11 noprompt

Action: Remove the prompt character (>) at the end of the response.

Cmd: noprompt CrLf
Response: OK CrLf

3.12 on

Action: Power on the secure element

Cmd: on CrLf
Response: OK CrLf

3.13 off

Action: Power off the secure element

Cmd: off CrLf
Response: OK CrLf

3.14 status

Action: Read the Secure Element (SE) status parameter (10 bytes).
At least one PIN code must have been provided.

Cmd: status CrLf
Response: [SE Status Parameter, 10 bytes] CrLf
Response: ERROR CrLf

3.15 changeadm

Action: Change the administrator PIN
This command powers the SE if needed.
Upon success the terminal in the On Mode

Terminal Display:

```
1234567890123456
Adm Old Pin:
????????

1234567890123456
Adm New Pin:
????????
```

Cmd: changeadm CrLf
Response: OK CrLf
Response: ERROR CrLf

3.16 changeuser

Action: Change the user PIN
This command powers the SE if needed.

Upon success the terminal in the *On* mode

Terminal Display:

```
1234567890123456
User Old Pin:
????

1234567890123456
User New Pin:
????
```

Cmd: changeuser CrLf
Response: OK CrLf
Response: ERROR CrLf

3.17 changeuser2

Action: Change the user2 PIN
This command powers the SE if needed.
Upon success the terminal in the *On* mode.

Terminal Display:

```
1234567890123456
User2 Old Pin:
????

1234567890123456
User2 New Pin:
????
```

Cmd: changeuser2 CrLf
Response: OK CrLf
Response: ERROR CrLf

3.18 adm

Action: Enter the Administrator's PIN on the CryptoTerminal (8 digits), and optionally reset the user's PINs (user1 and user2).
This command powers the SE if needed.

Terminal Display (default):

```
1234567890123456
Adm:????????
D=Del C=No  #=OK
```

Terminal Display (with the reset option):

```
1234567890123456
Adm:????????
Reset Users Pins
```

Cmd: adm [reset] CrLf
Response: OK CrLf
Response: ERROR CrLf

3.19 user

Action: Enter the user PIN on the CryptoTerminal (4 digits).
This command powers the SE if needed.

Terminal Display:

```
1234567890123456
User:????
D=Del C=No  #=OK
```

Cmd: user CrLf
Response: OK CrLf
Response: ERROR CrLf

3.20 user2

Action: Enter the user2 PIN on the CryptoTerminal (4 digits).
This command powers the SE if needed.

Terminal Display:

```
1234567890123456
User2:????
D=Del C=No  #=OK
```

Cmd: user2 CrLf
Response: OK CrLf
Response: ERROR CrLf

3.21 computekey

Action: Compute a key pair (private/public), from a key tree, with hardened keys only according to the BIP32 recommendation.
Administrator mode is required; key generation MUST be confirmed.

Terminal Display:

```
1234567890123456
CKey iii:  yes=#?
.x1.x2...xn  time
```

Cmd: computekey iii x1.x2…xn CrLf
- iii an integer value between 0 and 15);
- xi integer value (between 0 and 2^{31}-1), the hardened key path in the key tree.

Response: OK CrLf
Response: ERROR CrLf

3.22 genkey

Action: Generate a key pair (private/public)
Administrator mode is required; key generation MUST be confirmed.

Terminal Display:

```
1234567890123456
GenK iii: yes=#?
Adm          time
```

Cmd: genkey iii CrLf (iii an integer value between 0 and 15)
Response: OK CrLf
Response: ERROR CrLf

3.23 setseed

Action: Initialize a key tree with a seed or generate a tree with a random seed size
Administrator mode is required; key-tree initialization MUST be confirmed.

Terminal Display:

```
1234567890123456
Seed iii: yes=#?
Adm          time
```

Cmd: setseed iii seed CrLf (iii an integer value between 0 and 15)
Or
Cmd: setseed iii yy CrLf (iii an integer value between 0 and 15, yy integer the size of the random seed))

Response: OK CrLf
Response: ERROR CrLf

Seed: up to 128 hexadecimal digits (64 bytes)

3.24 getseed

Action: Read a tree seed.
Administrator mode is required; the read operation MUST be confirmed.

Terminal Display:

```
1234567890123456
Seed iii: yes=#?
Adm          time
```

Cmd: getseed iii CrLf (iii an integer value between 0 and 15)
Response: [Length (2 bytes), Seed Value (up to 64 bytes) Ascii Encoded] CrLf
Response: ERROR CrLf

3.25 getpub

Action: Read a public key.
Administrator or User PIN must have been provided.

Cmd: getpub iii CrLf (iii an integer value between 0 and 255)
Response: [Length (2 bytes), Public Key Value(65 bytes) Ascii Encoded] CrLf
Response: ERROR CrLf

3.26 btc

Action: Compute a Bitcoin address.
Administrator or User PIN must have been provided.

Cmd: btc iii [xxx] CrLf (iii an integer value between 0 and 255, xxx the network ID an integer value between 0 and 255)
Response: [the Bitcoin address ascii encoded] CrLf
Response: ERROR CrLf

3.27 hash160

Action: Compute a hash160.
Administrator or User PIN must have been provided.

Cmd: hash160 iii CrLf (iii an integer value between 0 and 255)
Response: [the hash160, 40 hexadecimal digits, ascii encoded] CrLf
Response: ERROR CrLf

3.28 eth

Action: Compute an Ethereum address.
Administrator or User PIN must have been provided.

Cmd: eth iii CrLf (iii an integer value between 0 and 255)
Response: [the Ether address, 40 hexadecimal digits, ascii encoded] CrLf
Response: ERROR CrLf

3.29 setkey

Action: Set private and public key.
Administrator PIN must have been provided.

Terminal Display:

```
1234567890123456
SetK iii:yes=#
Adm          time
```

Cmd: setkey iii PrivKey PubKey CrLf (iii an integer value between 0 and 255)
Response: OK CrLf
Response: ERROR CrLf

PrivKey: 64 hexadecimal digits (32 bytes)
PubKey: 130 hexadecimal digits (65 bytes), point encoded with uncompressed format.

Action: Set private and public key.
Administrator PIN must have been provided.

Terminal Display:

```
1234567890123456
SetK iii: yes=#?
Adm          time
```

3.30 setpp

Action: Set private key and compute public key.
Administrator PIN must have been provided.

Terminal Display:

```
1234567890123456
SetK iii: yes=#?
Adm          time
```

Cmd: setpp iii PrivKey CrLf (iii an integer value between 0 and 15)
Response: OK CrLf
Response: ERROR CrLf

PrivKey: 64 hexadecimal digits (32 bytes)

3.31 getpriv

Action: Read a private key.
Administrator PIN must have been provided.
The command must be confirmed.

Terminal Display:

```
1234567890123456
Priv iii: yes=#?
Adm          time
```

Cmd: getpriv iii CrLf (iii an integer value between 0 and 15)
Response: [Length (2bytes) Private Key(32 bytes), Ascii Encoded] CrLf
Response: ERROR CrLf

3.32 setlabel

Action: Set a key label.
Administrator PIN must have been provided.
The command must be confirmed.

Terminal Display:

```
1234567890123456
Label iii:yes=#?
>text
```

Cmd: setlabel KeyIndex (integer 0…15) "text" CrLf
Response:OK
Response: ERROR CrLf

3.33 getlabel

Action: Get a key label.
Administrator or User PIN must have been provided.

Cmd: getlabel KeyIndex (integer 0…15) CrLf
Response: [keylabel] OK
Response: ERROR CrLf

3.34 sign

Action: Generate a signature.
User or Administrator PIN must have been provided.
The command must be confirmed.

Terminal Display:

```
1234567890123456
Sign iii: yes=#?
User        time
```

Cmd: sign iii [hash value, Ascii Encoded] CrLf (iii an integer value between 0 and 15)
Response: [Length(2bytes), Signature (in ASN.1 syntax), Ascii Encoded] CrLf
Response: ERROR CrLf

3.35 recover

Compute and return the last recovery value (0 or 1).

Cmd: recover iii CrLf (iii an integer value between 0 and 15)
Response:[0 or 1] CrLf
Response: ERROR CrLf

3.36 signr

Action: Generate a signature and compute the recover value
User or Administrator PIN must have been provided.
The command must be confirmed.

Terminal Display:

```
1234567890123456
Sign iii: yes=#?
User        time
```

Cmd: signr iii [hash value, Ascii Encoded] CrLf (iii an integer value between 0 and 15)
Response: [Length(2bytes), Signature (in ASN.1 syntax), Ascii Encoded] RecoverValue (0/1) CrLf
Response: ERROR CrLf

3.37 signin

Action: Request a remote signature.
The command must be confirmed.

Terminal Display:

```
1234567890123456
Sign iii: yes=#?
Off         time
```

Cmd: signin iii [hash value, Ascii Encoded] record CrLf

- iii an integer value between 0 and 255
- hash value, the value (up to 32 bytes) to be signed
- record: the record number (default=0, integer 0...20)
Response:OK CrLf
Response: ERROR CrLf

3.38 signout

Action: Read a remote signature
The command must be confirmed.

Terminal Display:

```
1234567890123456
Sign iii: yes=#?
Off          time
```

Cmd: signout iii [HashValue, Ascii Encoded]
- iii an integer value between 0 and 15
- HashValue, the value to be signed
Response: : [Length (2bytes), Signature (in ASN.1 syntax) Hexa Ascii Encoded] [Length (2bytes), Public Key value uncompress format Hexa Ascii Encoded] CrLf
Response: ERROR CrLf

3.39 signoutr

Action: Read a remote signature and compute the recover value
The command must be confirmed.

Terminal Display:

```
1234567890123456
Sign iii: yes=#?
Off          time
```

Cmd: signoutr iii [HashValue, Ascii Encoded]
- iii an integer value between 0 and 15
- HashValue, the value to be signed
Response: : [Length (2bytes), Signature (in ASN.1 syntax) Hexa Ascii Encoded] [Length (2bytes), Public Key value uncompress format Hexa Ascii Encoded] RecoverValur (0/1) CrLf
Response: ERROR CrLf

3.40 signdo

Action: Perform a remote signature
User or Administrator PIN must have been provided
The command must be confirmed.

Terminal Display:

```
1234567890123456
Sign iii: yes=#?
User        time
```

Cmd: signdo iii [hash value, Ascii Encoded] record
- iii an integer value between 0 and 255
- hash value, the value to be signed
- record: the record number (0...20)

Response: : OK CrLf
Response: ERROR CrLf

3.41 settrans (generation of ether transaction)

Action:
The serial command "settrans" creates an Ether transaction using the following seven parameters:
- Key Index, the key to be used for the transaction signature
- Nounce, the nounce of the transaction, an integer positive value, starting from zero, and incremented by one after every transaction.
- GasPrice, the gas price to be used for the transaction in GWEIs
- GasLimit, the maximum instructions to be computed for the transaction, leading to a maximum cost of GasLimit x GasPrice
- Recipient Address: the recipient's Ether address (20 bytes, or 40 hexadecimal digits).
- Amount: the transaction amount in Ether, expressed in decimal notation (for example, 0.0).
- Data, the data associated to the transaction in quotes ("Text"), text format (optional parameter), or hexadecimal format with the $ prefix ($"Hexadecimal")
    - Text examples "Hello World"
    - Hexadecimal example $"313233"

The command must be confirmed.

Terminal Display

```
1234567890123456
Save Transaction
Key: 015  yes=#?
```

Cmd: settrans KeyIndex Nounce GasPrice GasLimit RecipientAddress Amount Data CrLf
Response: :OK CrLf
Response: ERROR CrLf

3.42 signt

Action:
The serial command "signt" signs a previously created Ether transaction

The command must be confirmed.

Terminal Display

```
1234567890123456
Sign 15 : yes=#
User          100

1234567890123456
Sign 15 : yes=#
   Recovering

1234567890123456
Sign 15 : yes=#
      DONE
```

Cmd: signt
Response: :OK CrLf
Response: ERROR CrLf

3.43 gettrans

Action: Read a signed transaction
The command must be confirmed.

Terminal Display

```
1234567890123456
Get Transaction
          yes=#?
```

Cmd: gettrans CrLf
Response: : [Transaction encoded in hexadecimal] CrLf
Response: ERROR CrLf

3.44 duplicate

Action: Duplicate a CCSC smartcard onto another smartcard.
Administrator PIN MUST have PIN provided.
The command must be confirmed.

Cmd: duplicate CrLf
Response: :OK CrLf
Response: ERROR CrLf

Terminal Displays:

```
1234567890123456
```

```
Remove Card
#=Confirm ?

1234567890123456
Insert new card
#=Confirm ?

1234567890123456
Adm:????????
D=Del C=No  #=OK
```

3.45 read

Action: Read data from the smartcard's nonvolatile user memory
User2 PIN or Administrator PIN MUST have PIN provided

Cmd: read address size
address: data address (0...16383, decimal value), 16 KB available.
size: size of data to be read (1..256, decimal value)
Response: : [data (up to 256 bytes), hexadecimal ascii encoded] CrLf
Response: ERROR CrLf

3.46 write

Action: Write data to the smartcard's nonvolatile user memory
User2 PIN or Administrator PIN MUST have PIN provided

Cmd: write adr [data]
adr: data address (0...16383, decimal value), 16 KB available.
data: data to be written (1..255 bytes, hexadecimal ASCII encoded)
Response: :OK CrLf
Response: ERROR CrLf

4 CryptoTerminal Menus

The CryptoTerminal is equipped with a 16 keys keypad, a 2x16 characters LCD display, and a smartcard reader.

4.1 Boot Message (BM)

```
1234567890123456
   EtherTrust
Crypto  Terminal
```

4.2 Idle Menu (IM)

Root Menu

```
1234567890123456
A=Menu     B=Boot
Mode         time
```

Mode: Off/On/Adm/User/User2/User1&2
Time: Local time in seconds, since power on

4.3 Boot Menu (BM)

Path: IM.B

```
1234567890123456
Reboot(type B)=?
Mode         time
```

4.4 A Menu (AM)

Path: IM.A

```
1234567890123456
1=Mode     2=Key?
3=PIN 4=Dup 5=Tr
```

4.5 Mode Menu (MM)

Path: AM.1

```
1234567890123456
1=ON 2=OFF 3=ADM
4=USER 5=USER2 ?
```

4.5.1 Mode ON

Path MM.1

```
1234567890123456
  Starting SE
?              OK
```

4.5.2 Mode OFF

Path MM.2

```
1234567890123456
  Stopping SE
?              OK
```

4.5.3 Mode ADM

Path: MM.3

```
1234567890123456
  Starting SE


1234567890123456
Adm:????????
D=Del C=No  #=OK

1234567890123456
              Adm
?              OK

1234567890123456
              Adm
?           ERROR
```

4.5.4 Mode User

Path: MM.4

```
1234567890123456
  Starting SE


1234567890123456
User:????
D=Del C=No  #=OK
```

```
1234567890123456
            User
?             OK

1234567890123456
            User
?          ERROR
```

4.5.5 Mode User2

Path: MM.5

```
1234567890123456
  Starting SE


1234567890123456
User2:????
D=Del C=No  #=OK

1234567890123456
           User2
?             OK

1234567890123456
           User2
?          ERROR
```

4.6 Key Menu (KM)

Path AM.2

4.6.1 Key Index

```
1234567890123456
KeyIndex: ???
#=OK D=Del  C=No

1234567890123456
1=Sign  2=GenKey
Key:iii 3=Rec A>
```

4.6.1.1 Sign

Path: KM.1

```
1234567890123456
Sign iii: yes=#?
```

```
Record: yy

1234567890123456
    sign iii rec
?             OK

1234567890123456
    sign iii rec
?          ERROR
```

4.6.1.2 Genkey

Path: KM.2

```
1234567890123456
GenK iii: yes=#?
#=OK D=Del  C=No

1234567890123456
      genkey iii
?             OK

1234567890123456
      genkey iii
?          ERROR
```

4.6.1.3 Record Index

Path KM.3

```
1234567890123456
RecordIndex: ??
#=OK D=Del  C=No
```

4.6.2 Key Menu2 (KM2)

Path: KM.A

```
1234567890123456
1=CKey 2=SetTree
3=GetL 4=SetL A>
```

4.6.2.1 CKey (Compute Key)

Path: KM.2.1

```
1234567890123456
```

```
1234.10---------
 ComputeKey: dec

1234567890123456
CKEY iii: yes=#?
.1000.10

1234567890123456
ComputeKey
DONE            ?

1234567890123456
ComputeKey
ERROR           ?
```

4.6.2.2 SetTree

Path: KM2.2

```
1234567890123456
1=SetTree 2=Read
3=Generate Tree
```

4.6.2.2.1 SetTree

Path: KM2.2.1

```
1234567890123456
1234____________
   SetSeed: hexa

1234567890123456
Seed iii: yes=#?
Col:005 n:004 U

1234567890123456
SetSeed
DONE            ?

1234567890123456
SetSeed
ERROR           ?
```

4.6.2.2.2 Read

Path: KM2.2.2

```
1234567890123456
Seed iii: yes=#?
Getseed 3      ?

1234567890123456
GetTreeSeed
ERROR          ?

1234567890123456
0123456789ABCDEF
  TreeSeed: hexa
```

4.6.2.2.3 GenerateTree

Path: Path: KM2.2.2

```
1234567890123456
Length 32 number
#=OK D=Del  C=No

1234567890123456
Seed iii: yes=#?
Col:005 n:004 U

1234567890123456
SetSeed
DONE           ?

1234567890123456
SetSeed
ERROR          ?
```

4.6.2.3 GetL (Get Label)

Path: KM2.3

```
1234567890123456
Hello World !
               ?

1234567890123456
GetLabel
ERROR          ?
```

4.6.2.4 SetL (Set Label)

Path: KM2.4

```
1234567890123456
Hello World !
Setlabel 3

1234567890123456
Label iii:yes=#?
>Hello World!

1234567890123456
SetLabel
DONE           ?

1234567890123456
SetLabel
ERROR          ?
```

4.6.3 Key Menu3 (KM3)

Path: KM2.5

```
1234567890123456
1=SetPr  2=GetPr
3=GetPub  4=GetT
```

4.6.3.1 SetPr (Set Private Key)

Path KM3.1

```
1234567890123456
0123456789ABCDEF
SetPrivKey: hexa

1234567890123456
SetK iii: yes=#?


1234567890123456
SetPrivKey
DONE           ?

1234567890123456
SetPrivKey
ERROR          ?
```

4.6.3.2 GetPr (Get Private Key)

Path: KM3.2

```
1234567890123456
GetPrivKey
ERROR          ?

1234567890123456
Priv iii: yes=#?
getpriv 3

1234567890123456
0123456789ABCDEF
   PrivKey: hexa
```

4.6.3.3 GetPub (get public key)

Path KM3.3

```
1234567890123456
1=Raw       2=BTV
3=H160      4=ETH
```

4.6.3.3.1 Raw

Path KM3.3.1

```
1234567890123456
GetPubKey
ERROR          ?

1234567890123456
0437A4AEF1F8423C
    PubKey: hexa
```

4.6.3.3.2 BTC

Path KM3.3.1

```
1234567890123456
ID:0??   integer
#=OK  D=Del C=No

1234567890123456
1Mf9gx8gF9ocM42b
BTC Address 000
```

4.6.3.3.3 Hash160

Path KM3.3.2

```
1234567890123456
1234567890ABCDEF
 HASH160 Address
```

4.6.3.3.4 Hash160

Path KM3.3.3

```
1234567890123456
1234567890ABCDEF
   ETHER Address
```

4.6.3.4 GetT (Get Tree Seed)

Path KM3.4

```
1234567890123456
Seed iii: yes=#?
Getseed 3      ?

1234567890123456
GetTreeSeed
ERROR          ?

1234567890123456
0123456789ABCDEF
  TreeSeed: hexa
```

4.7 Duplicate Menu (Dup)

Path AM.4

4.7.1 Remove Card

```
1234567890123456
Remove Card
#=Confirm ?
```

4.7.2 Insert New Card

```
1234567890123456
Insert new card
#=Confirm ?
```

4.7.3 Enter ADM PIN

```
1234567890123456
Adm:????????
```

```
D=Del C=No  #=OK
```

4.8 Transaction Menu (Tr)

Path AM.5

```
1234567890123456
KeyIndex: ???
#=OK D=Del C=No?

1234567890123456
Transaction  015
Eth=1          ?
```

4.8.1 Ether Menu (EM)

Path AM.5.1

```
1234567890123456
1=Create   2=Del
3=Edit    4=Sig?
```

4.8.1.1 Create

Path EM.1

```
1234567890123456
1=@ 2=Eth 37=Dat
4=N. 5=Gas 6=Lim

1234567890123456
Save Transaction
          yes=#?
```

4.8.1.1.1 Address

Path EM.1.1

4.8.1.1.2 Ether

Path EM.1.2

4.8.1.1.3 Data (text)

Path EM.1.3

4.8.1.1.4 Nounce

Path EM.1.4

4.8.1.1.5 Gas Price

Path EM.1.5

4.8.1.1.6 Gas Limit

PATH EM.1.6

4.8.1.1.7 Data (Hexadecimal)

Path EM.1.7

4.8.1.2 Delete

Path EM.2

```
1234567890123456
Transaction
Delete    #=yes?
```

4.8.1.3 Edit

Path EM.3

```
1234567890123456
1=@ 2=Eth 37=Dat
4=N. 5=Gas 6=Lim

1234567890123456
Save Transaction
          yes=#?
```

4.8.1.3.1 Address

Path EM.3.1

4.8.1.3.2 Ether

Path EM.3.2

4.8.1.3.3 Data Text

Path EM.3.3

4.8.1.3.4 Nounce

Path EM.3.4

4.8.1.3.5 Gas Price

Path EM.3.5

4.8.1.3.6 Gas Limit

PATH EM.3.6

4.8.1.3.7 Data Hexadecimal

Path EM.3.7

4.8.1.4 Sign

Path EM.4

```
1234567890123456
Sign 015: yes=#?
      DONE
```

5 BTOOLS Script commands

BTOOLS 2.0 is open-source software (https://github.com/purien/btools) available for Windows, Linux, and Raspberry Pi. It generates Bitcoin and Ethereum transactions using the OpenSSL library. For enhanced security, it supports the Cryptocurrency Smartcard (CCSC) for generating and storing private keys and computing ECDSA signatures.

BTOOLS (2.0) is compatible with the CryptoTerminal.

It supports two facilities, a serial terminal for manual interactions and scripts for programmed interactions.

5.1 Starting a serial terminal

Start a serial terminal using the command: btools –terminal.

5.2 Script Commands

A CryptoTerminal script has a filename beginning with the "_" character.

It is a set of command lines ending by the CrLF characters.

5.2.1 start

Action: Start a session with the CryptoTerminal

Cmd: start [optional COM port number]

5.2.2 on

Action: Power the CryptoTerminal secure element

Cmd: on

5.2.3 off

Action: Power off the CryptoTerminal secure element

Cmd: off

5.2.4 adm

Action: Set the CryptoTerminal in Administrator mode

Cmd: adm

5.2.5 user

Action: Set the CryptoTerminal in User mode

Cmd: adm

5.2.6 user2

Action: Set the CryptoTerminal in User2 mode

Cmd: user2

5.2.7 getpub

Action: Get the public key.

Cmd: getpub KeyIndex
- KeyIndex is a decimal value (0...255)

5.2.8 sign

Action: Compute an ECDSA signature.

Cmd: sign KeyIndex
- KeyIndex is a decimal value (0...255)

5.2.9 signout

Action: Export a ValueToSign for offline signature

Cmd: sign KeyIndex RecordIndex
- KeyIndex: a decimal value (0...255)
- RecordIndex: the record index to be used (0..20)

5.2.10 setpub

Action: Set the public key value needed for offline signature
This command is used only for bitcoin family transactions, before signout and signin commands.

Cmd: setpub PublicKeyValue

5.2.11 signdo

Action: Compute the offline signature

Cmd: signdo KeyIndex RecordIndex
- KeyIndex: a decimal value (0...255)
- RecordIndex: the record index to be used (0..20)

5.2.12 signin

Action: Read an offline signature

Cmd: signin KeyIndex RecordIndex
- KeyIndex: a decimal value (0...255)
- RecordIndex: the record index to be used (0..20)

5.2.13 raw

Action: Send a command to the CryptoTerminal

Cmd: raw [CryptoTerminal Command]

Examples:
raw setseed TreeIndex SeedValue
raw compute TreeIndex KeyPath

6 BTOOLS CryptoTerminal Script Examples

6.1 Ethereum Transaction

btools -genethtrans escript.txt ecsript.bin

6.1.1 CryptoTerminal script

```
//////////////////////
// _term.txt     //
//////////////////////

start
wait   100
reboot 100
user
getpub 5
sign     5
off
```

6.1.2 Transaction Script

```
//////////////////////////
//    escript.txt     //
//////////////////////////
```

```
apdu_script _term.txt

nonce     14
gasPrice  30000000000
gasLimit  80000
to          6BAC1B75185D9051AF740AB909F81C71BBB221A6
value      0
data       "Hello World"
```

6.2 Ethereum Signature Request

call btools -genethtrans escript.txt ecsript.bin

6.2.1 CryptoTerminal Script

```
//////////////////////
//   _term.txt   //
//////////////////////

start
wait   100
reboot 100
// Only for bitcoin transaction: insert the setpub command before the signin command
signin 5 7
```

6.3 Ethereum Transaction Script

```
//////////////////////////
//    escript.txt     //
//////////////////////////
apdu_script _term.txt

nonce     13
gasPrice  30000000000
gasLimit  80000
to          6BAC1B75185D9051AF740AB909F81C71BBB221A6
value      0
data       "Hello World"
```

6.4 Ethereum Signature Read

call btools -genethtrans escript.txt ecsript.bin

6.4.1 CryptoTerminal Script

```
//////////////////////
//   _term.txt    //
//////////////////////

start
wait   100
reboot 100
// Note: only for bitcoin transaction, insert the setpub command before the signout command
signout 5 7
```

6.4.2 Transaction Script

```
//////////////////////////
//    escript.txt     //
//////////////////////////

apdu_script _term.txt

nonce     13
gasPrice  30000000000
gasLimit  80000
to          6BAC1B75185D9051AF740AB909F81C71BBB221A6
value      0
data       "hello world"
```

7 The Crypto Currency SmartCard

The Cryptocurrency smartcard (CCSC), of which AID is 010203040500 has three PINs, administrator, user, and user2. The default values are 8 zeros (3030303030303030) for administrator and 4 zeros (30303030) for user and user2.

It is able to generate, to compute according to the BIP32 standard, or to import elliptic curve keys (up to 16), used for the generation of ECDSA signatures used by Bitcoin and Ethereum crypto currencies.

A 16 KB nonvolatile read/write memory area, protected by a dedicated PIN (User2), is available for storing sensitive information.

7.1 The Select Command

This command starts the Crypto Currency smartcard application
Upon success it returns the status word SW1 SW2 = 9000

Command

CLA	INS	P1	P2	P3	AID
00	A4	04	00	06	010203040500

Response

SW1	SW2
90	00

7.2 The Verify UserPin Command

This command verifies the user pin. The UserPin is required for the signature operations.

Upon success it returns the status word SW1 SW2 = 9000
Otherwise it returns SW1=63, SW2=number of remaining tries (3 at the most)

Command

CLA	INS	P1	P2	P3	UserPin
00	20	00	00	04	3030303030

Response

SW1	SW2	Comment
90	00	Success
63	Number of remaining tries	Fail
67	00	Wrong Length
6B	00	Wrong P1P2

7.3 The Verify UserPin2 Command

This command verifies the second user pin. The UserPin2 is required for the memory reading and writing operations.
Upon success it returns the status word SW1 SW2 = 9000
Otherwise it returns SW1=63, SW2=number of remaining tries (3 at the most)

Command

CLA	INS	P1	P2	P3	UserPin2
00	20	00	02	04	3030303030

Response

SW1	SW2	Comment
90	00	Success
63	Number of remaining tries	Fail
67	00	Wrong Length
6B	00	Wrong P1P2

7.4 The Verify AdminPin command

This command verifies the administrator pin. It gives access to all available features of the cryptocurrency application. If P2 is set to FF UserPin and UserPin2 are reset to the default value (four zeros).

Upon success it returns the status word SW1 SW2 = 9000
Otherwise it returns SW1=63, SW2=number of remaining tries (ten at the most)

Command

CLA	INS	P1	P2	P3	AdminPin
00	20	00	01 FF Reset to default	08	30303030303030303030

Response

SW1	SW2	Comment
90	00	Success
63	Number of remaining tries	Fail
67	00	Wrong Length
6B	00	Wrong P1P2

7.5 The ChangePin command

This command sets a PIN (UserPin, UserPin2, AdminPin) to a new value
The P2 value is respectively 00, 02, 01 for UserPin, UserPin2, AdminPin.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=number of remaining tries.

Command

CLA	INS	P1	P2	P3	OldPin	NewPIN
00	24	00	00	10	30303030FFFFFFFF	31313131FFFFFFFF
00	24	00	02	10	30303030FFFFFFFF	30303030FFFFFFFF
00	24	00	01	10	3030303030303030	3131313131313131

Response

SW1	SW2	Comment
90	00	Success
63	Number of remaining tries	Fail
67	00	Wrong Length
6B	00	Wrong P1P2

7.6 The GetStatus command

This command returns the current state of the cryptocurrency application. It required at least the previous checking of one PIN (UserPin, UserPin2, AdminPin).

Command

CLA	INS	P1	P2	P3
00	87	00	00	0A

Response

SW1	SW2	Comment
90	00	Success
63	80	PIN required

This command returns 10 bytes.

byte0: b0= ECDSA, b1=DH, b3=SHA256, b4=HMAC-512, b5= BigNumber (OK= 0x07)
byte1: The maximum number of keys that can be used by the cryptocurrency application (16)
byte2, byte3: The CCSC application version 0x0007 = v0.7)
byte4, byte5: The size of the user memory (for example 4000 for 16 KB)
byte6, byte7: 16 bits (b15...b1b0) indicating the index (bi) of defined keys, for example 0003 for key1 (bit1) and key0 (bit0)
byte8, byte9: 16 bits (b15...b1b0) indicating the index (bi) of defined key trees, for example 0003 for tree1 (bit1) and tree0 (bit0)

7.7 The Write Command

This command writes data in the nonvolatile memory. This service could be used for the secure storage of any information in the area [0000, 0C00[.

It requires the previous checking of PINs UserPin2 or AdminPin.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	Data
00	D0	AdrMSB	AdrLSB	Data Length	Data to be written

The starting address ranging from 0000 to 10FF is encoded by two bytes (P1, P2), P1 being the *most significant byte* (MSB) and P2 the *less significant byte* (LSB).

Address Mapping

Start Address	Length	Comment	PIN required
0000	0C00	Data Area	User2 or Admin
0C00	0400	Key Dump Area	Admin
1000	0100	Key Label - 32 bytes/key	Admin

Response

SW1	SW2	comment
90	00	OK
63	80	PIN required
6D	02	Invalid Address

7.8 The Read Command

This command reads data in the non volatile memory.
It requires the previous checking of PINs UserPin2 or AdminPin.
Upon success it returns the status word SW1,SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3
00	B0	AdrMSB	AdrLSB	Length to be read

The starting address ranging from 0000 to 10FF is encoded by two bytes (P1, P2), P1 being the most significant byte (MSB) and P2 the less significant byte (LSB).

Address Mapping

Start Address	Length	Comment	PIN required
0000	0C00	Data Area	User2 or Admin
0C00	0400	Key Dump Area	Admin
1000	0100	Key Label - 32 bytes/key	Admin

Response

Body	SW1	SW2	comment
Data	90	00	OK
Empty	63	80	PIN required
Empy	6D	01	Invalid Address

7.9 The Clear KeyPair & InitCurve Command

This command MUST be used before any key setting or key generation operation.
This command clears the curve parameters.
The InitCurve command is required to configure the EllipticCurve, excepted when the InitCurve option is used.

It requires the Admin PIN.

The P1=80 option is used to clear either the public or the private, and to initialize the associated curve. Here are some examples

- P1=C0, clear public key and initialize curve SECP256k1.
- P1=A0, clear private key and initialize curve SECP256k1.

The P1=10 option resets and initializes a KeyTree.

Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	PIN required
00	81	00 – Clear Keys 10 – Clear Keys & KeyTree			
00	81	SECP256k1 80 – ClearKeys & InitCurve 40 - Clear Public Key Only 20 - Clear Private Key Only 10 - Reset KeyTree	Key Index [0,15]	00	Admin

Response

SW1	SW2	Comment
90	00	Key Reset Done
63	80	PIN required
69	85	Bad index

7.10 The InitCurve & InitTree Command

This command initializes the elliptic curve parameters and optionally a KeyTree.

The KeyTree is initialized by a seed according to the BIP32 specification. Two mode of seed generation are available:
- The seed value is imported;
- The seed value is randomly generated.

The keys MUST be cleared before this operation.

It requires the Admin PIN.

Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	PIN required
00	89	00 - SECP256k1	Key Index [0,15]	00	Admin
00	89	00 - SECP256k1	Tree Index [0,15]	Seed Length If P3=1, the payload is the size of the seed to be randomly generated	Admin

Response

Data	SW1	SW2	Comment
	90	00	Key Reset Done
TreeStatus 2 bytes b_i= TreeIndex	90	00	KeyTree initialized
	63	80	PIN required
	69	85	Bad index
	64	01	Public Key is defined
	64	02	Private Key is defined
	6A	86	Incorrect P1P2
	69	85	KeyTree initialization error

7.11 The Generate KeyPair Command

This command generates the elliptic curve public and private keys or computes a key according to the BIP32 specification

The keys MUST be cleared before this operation.

It requires the Admin PIN.

If P3 is set to zero a private public key pair is randomly generated.

If P3 is a multiple of 4, a hardened private key is generated whose index a list of n 32 bits word, $IH_1/IH_2/\ldots/IH_n$. The public key is not computed. The MSB bit of IHi 32bits word MUST be set.

Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	PIN required
00	82	00	Key Index [0,15]	00	Admin
00	82	00	Tree Index [0,15]	4 n	Admin

Response

SW1	SW2	Comment
90	00	OK
63	80	PIN required
69	85	Bad index
64	01	Public Key is defined
64	02	Private Key is defined
6D	10	Key Generation Error

7.12 The Dump KeyPair Command

This command dumps the elliptic curve public and private keys.
It returns the size of the data written in the non volatile memory, in the KeyDump area whose address starts at 0C00

It requires the Admin PIN.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	PIN required
00	83	00	Key Index [0,15]	02	Admin
00	83	FF Reset DUMP Area	Not Used	00	Admin

Response

Body	SW1	SW2	Comment
Total Length 2 bytes SECP256k1 0185 SECP256v1 0201	90	00	Key Reset Done
	63	80	PIN required
	69	85	Bad index
	64	01	Public Key is not defined
	64	02	Private Key is not defined
	6A	86	Incorrect P1P2

Dump KeysPair: Memory Mapping

Address = 0C00	Comment
Total Length, 2 bytes	
PubKey A Length, 2 bytes	The length of the A parameter
PubKey A parameter value	The value of the A parameter
PubKey B Length, 2 bytes	The length of the B parameter
PubKey B parameter value	The value of the B parameter
PubKey G Length, 2 bytes	The length of the Generator
PubKey G value	The value of the Generator
PubKey R Length, 2 bytes	The length of the R parameter
PubKey R value	The value of the R parameter (Order of the Generator)
PubKey W Length, 2 bytes	The length of the W parameter
PubKey W value	The value of the W Public Key (EC Point)
PubKey Field Length, 2 bytes	The length of the Field parameter
PubKey Field Value	The value of the prime p of the field Z/pZ
PubKey Size, 2 bytes	The size of the Public Key object
PrivKey A Length, 2 bytes	The length of the A parameter
PrivKey A parameter value	The value of the A parameter
PrivKey B Length, 2 bytes	The length of the B parameter
PrivKey B parameter value	The value of the B parameter
PrivKey G Length, 2 bytes	The length of the Generator
PrivKey G value	The value of the Generator
PrivKey R Length, 2 bytes	The length of the R parameter
PrivKey R value	The value of the R parameter (Order of the Generator)
PrivKey S Length, 2 bytes	The length of the S parameter
PrivKey S value	The value of the S, Private Key (32 bytes)
PrivKey Field Length, 2 bytes	The length of the Field parameter
PrivKey Field Value	The value of the prime p of the field (Z/pZ)
PrivKey Size, 2 bytes	The size of the Private Key object

7.13 The GetInfo command

This command collects a list of information for a given key index, including key label, tree seed and private key.
It requires the Admin PIN.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	PIN required
00	86	00	Key Index [0,15]	00	Admin

Response

Body	SW1	SW2	Comment
Status 2 bytes 0x0000: no tree or key 0x0001: Tree 0x0002: Key 0x0003: Tree & Key Label Length (2 bytes) Label Value TreeSeed Length (2 byte) TreeSeed value Private Key Length (2 bytes) Private key value	90	00	 If status # 0 (tree or key) If tree If key
	63	80	PIN required
	69	85	Bad index

7.14 The Get KeyParameter Command

This command collects the elliptic curve public and private keys parameters.

It requires the User or the Admin PIN.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	PIN required
00	84	00 Parameter A	Key Index [0,15]	00	Admin or User
		01 Parameter B		00	Admin or User
		02 Parameter Field (Z/pZ)		00	Admin or User
		03 Parameter G (generator)		00	Admin or User
		04 Parameter K (cofactor)		00	Admin or User
		05 Parameter R (order of G)		00	Admin or User
		06 Parameter W (Public Key)		43	Admin or User
		07 Parameter S (Private Key		22	Admin
		08 Key Label		20	Admin or User
		09 x value of the public key		00	Admin
		0A TreeSeed		00	Admin

Response

Body	SW1	SW2	Comment
Param0 Length – Param0 value Param1 Length – Param1 value Param2 Length – Param2 value Param3 Length – Param3 value Param4 Length – Param4 value Param5 Length – Param5 value Param6 Length – Param6 value Param7 Length – Param7 value	90	00	Response includes a 2 bytes length field for parameters 0 to 7
Param8 (Key Label) Value	90	00	OK
Param9 Length – Param9 value	90	00	The x value of the public key
	63	80	PIN required
	69	85	Bad index
	64	01	Public Key is not defined
	64	02	Private Key is not defined
	6A	86	Incorrect P1P2
	6D	30	Javacard Exception
ParamA Length – ParamA value	90	00	OK

7.15 The Set KeyParameter Command

This command sets the elliptic curve parameters, including public and private keys parameters

It requires the Admin PIN.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	PIN required
00	88	00 Parameter A	Key Index [0,15]	Length	Admin
		01 Parameter B		Length	Admin
		02 Parameter Field (Z/pZ)		Length	Admin
		03 Parameter G (generator)		Length	Admin
		04 Parameter K (cofactor)		Length	Admin
		05 Parameter R (order of G)		Length	Admin
		06 Parameter W (Public Key) The public key is checked according to the private key, and is reset in case of error		41	Admin
		07 Parameter S (Private Key The private key MUST be initialized before setting the public key		20	Admin
		08 Key Label		20	Admin

Response

SW1	SW2	Comment
90	00	OK
63	80	PIN required
69	85	Bad index
64	01	Public Key is defined
64	02	Private Key is defined
6A	86	Incorrect P1P2
6D	40	Javacard Exception

7.16 The SignECDSA command

This command generates an ECDSA signature.

It requires the AdminPin or UserPin.

Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).

Command

CLA	INS	P1	P2	P3	Data	PIN required
00	80	00 Signature without digest	Key Index [0,15]	Length 20	Data to be signed	Admin or User
00	80	21 Signature with SHA256	Key Index [0,15]	Length	Data to be hashed and signed	Admin or User

Response

Body	SW1	SW2	Comment
Length (2 bytes) ASN.1 ECDSA Signature Encoding	90	00	OK
	63	80	PIN required
	69	85	Bad index
	64	01	Public Key is not defined
	64	02	Private Key is not defined
	6A	86	Incorrect P1P2
	6D	20	Signature Error