

EtherTrust CryptoTerminal V1.0 Firmware 1.01b



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 from Android mobile	8
2.5	Generating transactions with BTOOLS.....	9
2.6	Overview of the ECTv2 crypto terminal board.....	9
3	UPLOADING FIRMWARE	10
3.1	USPASP Security procedure.....	10
3.2	Downloading the CryptoTerminal firmware	10
3.3	Downloading the Bluetooth Low Energy firmware	10
4	CHECKING FIRMWARE INTEGRITY.....	11
5	KEYPAD.....	11
5.1	Root Menu (Idde Menu) Keys	12
6	CRYPTO-TERMINAL SERIAL COMMANDS	13
6.1	Wrong Command.....	13
6.2	Empty Command	13
6.3	reboot	13
6.4	echo.....	13
6.5	disp.....	13
6.6	edit	14
6.7	enter.....	14
6.8	debug.....	14
6.9	nodebug.....	14
6.10	prompt.....	14
6.11	noprompt	14

6.12	on.....	15
6.13	off.....	15
6.14	status.....	15
6.15	changeadm.....	15
6.16	changeuser.....	15
6.17	changeuser2.....	16
6.18	adm.....	16
6.19	user.....	17
6.20	user2.....	17
6.21	computekey.....	17
6.22	genkey.....	18
6.23	setseed.....	18
6.24	getseed.....	19
6.25	getpub.....	19
6.26	btc.....	19
6.27	hash160.....	19
6.28	eth.....	19
6.29	setkey.....	20
6.30	setpp.....	20
6.31	getpriv.....	21
6.32	setlabel.....	21
6.33	getlabel.....	21
6.34	sign.....	21
6.35	recover.....	22
6.36	signr.....	22
6.37	signin.....	22
6.38	signout.....	23
6.39	signoutr.....	23
6.40	signdo.....	23

6.41	settrans (generation of ether transaction).....	24
6.42	signt.....	24
6.43	gettrans.....	25
6.44	settransf (generation and signature of ethereum transaction)	25
6.45	duplicate.....	26
6.46	read.....	27
6.47	write.....	27
6.48	bMAC.....	27
7	CRYPTOTERMINAL MENUS.....	28
7.1	Boot Message (BM).....	28
7.2	Idle Menu (IM).....	28
7.2.1	Serial USB Mode.....	28
7.2.2	Bluetooth Mode.....	28
7.2.3	BLE Loader Mode.....	28
7.2.4	Bluetooth Menu (BTM).....	28
7.2.5	bMAC Menu (BMM).....	28
7.2.6	Boot Menu (BM).....	28
7.2.7	A Menu (AM).....	28
7.3	Bluetooth Setting Menu (BSM).....	28
7.3.1	AT mode.....	28
7.3.2	BT Pin.....	29
7.3.3	BT Name.....	29
7.4	bMAC Run Menu (BRM).....	29
7.4.1	Run bMAC.....	29
7.4.2	Run bMAC+EEPROM.....	29
7.4.3	Run bMAC BLE.....	29
7.5	Mode Menu (MM).....	29
7.5.1	Mode ON.....	29
7.5.2	Mode OFF.....	29
7.5.3	Mode ADM.....	29
7.5.4	Mode User.....	30
7.5.5	Mode User2.....	30
7.6	Key Menu (KM).....	30
7.6.1	Key Index.....	30
7.6.1.1	Sign.....	30
7.6.1.2	Genkey.....	30
7.6.1.3	Record Index.....	31
7.6.2	Key Menu2 (KM2).....	31
7.6.2.1	CKey (Compute Key).....	31
7.6.2.2	SetTree.....	31
7.6.2.2.1	SetTree.....	31
7.6.2.2.2	Read.....	31

7.6.2.2.3	GenerateTree	31
7.6.2.3	GetL (Get Label).....	32
7.6.2.4	SetL (Set Label).....	32
7.6.3	Key Menu3 (KM3).....	32
7.6.3.1	SetPr (Set Private Key)	32
7.6.3.2	GetPr (Get Private Key).....	32
7.6.3.3	GetPub (get public key)	32
7.6.3.3.1	Raw.....	32
7.6.3.3.2	BTC.....	33
7.6.3.3.3	Hash160	33
7.6.3.3.4	Ethereum Address.....	33
7.6.3.4	GetT (Get Tree Seed).....	33
7.7	Duplicate Menu (Dup).....	33
7.7.1	Remove Card	33
7.7.2	Insert New Card.....	33
7.7.3	Enter ADM PIN.....	33
7.8	Transaction Menu (Tr).....	33
7.8.1	Ether Menu (EM).....	33
7.8.1.1	Create.....	34
7.8.1.1.1	Address	34
7.8.1.1.2	Ether.....	34
7.8.1.1.3	Data (text).....	34
7.8.1.1.4	Nounce.....	34
7.8.1.1.5	Gas Price.....	34
7.8.1.1.6	Gas Limit.....	34
7.8.1.1.7	Data (Hexadecimal).....	34
7.8.1.2	Delete.....	34
7.8.1.3	Edit.....	34
7.8.1.3.1	Address	34
7.8.1.3.2	Ether.....	34
7.8.1.3.3	Data Text.....	34
7.8.1.3.4	Nounce.....	34
7.8.1.3.5	Gas Price.....	34
7.8.1.3.6	Gas Limit.....	34
7.8.1.3.7	Data Hexadecimal	34
7.8.1.4	Sign.....	34
8	BTOOLS SCRIPT COMMANDS.....	35
8.1	Starting a serial terminal	35
8.2	CryptoTerminal Script	35
8.2.1	Index Array.....	35
8.2.2	start.....	35
8.2.3	on.....	35
8.2.4	off.....	36
8.2.5	adm.....	36
8.2.6	user.....	36
8.2.7	user2.....	36
8.2.8	getpub.....	36
8.2.9	sign	36
8.2.10	signout	36
8.2.11	setpub.....	37
8.2.12	signdo.....	37

8.2.13	signin.....	37
8.2.14	raw.....	37
9	BTOOLS CRYPTOTERMINAL TRANSACTION SCRIPT EXAMPLES.....	38
9.1	Ethereum Transaction Generation	38
9.1.1	CryptoTerminal script.....	38
9.1.2	EthereumTransaction Script.....	38
9.2	Bitcoin Transaction Generation	38
9.2.1	CryptoTerminal script.....	38
9.2.2	Bitcoin Transaction Script.....	39
10	THE CRYPTO CURRENCY SMARTCARD.....	40
10.1	The Select Command.....	40
10.2	The Verify UserPin Command	40
10.3	The Verify UserPin2 Command.....	41
10.4	The Verify AdminPin command	41
10.5	The ChangePin command.....	42
10.6	The GetStatus command	42
10.7	The Write Command.....	43
10.8	The Read Command	43
10.9	The Clear KeyPair & InitCurve Command	44
10.10	The InitCurve & InitTree Command.....	45
10.11	The Generate KeyPair Command.....	46
10.12	The Dump KeyPair Command	47
10.13	The GetInfo command	49
10.14	The Get KeyParameter Command.....	49
10.15	The Set KeyParameter Command.....	50
10.16	The SignECDSA command.....	51

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, or on-line "Chip and PIN" validation.

Keys are processed entirely within the smartcard secure element, 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, and a *Bluetooth Low Energy* (BLE) module with the RFCOMM profile. It includes a smartcard reader, and delivers the following services:

- Generation and secure storage of cryptographic keys;
 - Generation of cryptographic signature in synchronous mode (driven from PC or mobile), or in asynchronous mode (without connection to PC or mobile);
 - All PINs are entered (like in the case of bank card terminals) on the Crypto Terminal;
 - All signatures are acknowledged by the Crypto Terminal;
 - Very High Security:
- *Removable Secure Element* (smartcard) has EAL5+ certification
 - *Bare Metal*, Crypto Terminal firmware, and the BLE module firmware can be flashed at any time.

2.2 The Crypto Currency SmartCard (CCSC)

The Crypto Currency smartcard (CCSC) 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 for crypto currencies such as Bitcoin and Ethereum.

The Crypto Currency smartcard application (CCSC), of which AID is *010203040500*, has three PINs, administrator, user, and user2. The default values are 8 zeros for administrator and 4 zeros for user and user2.

A Read/Write non volatile memory (16KB), protected by a dedicated PIN (User2), is available for the storage of any sensitive information.

2.3 Generating Ethereum transaction from the terminal

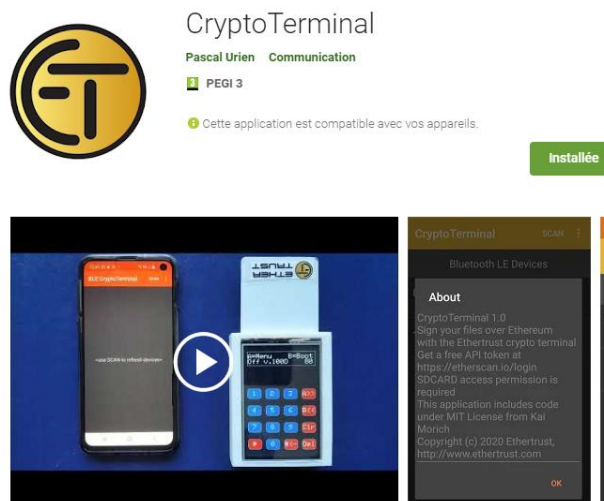
The serial command "settransf" generates an Ethereum transaction according to the seven following parameters:

- 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 Ether address (20 bytes, 40 hexadecimal digits) of the transaction recipient
- Amount, the amount of the transaction (in Ether) in decimal format, such as 0.0.
- Data, the data associated to the transaction, at the most 448 ASCII characters (# prefix) or 224 bytes represented in hexadecimal format (#\$ prefix).

The "settransf" returns the Ethereum transaction in hexadecimal format.

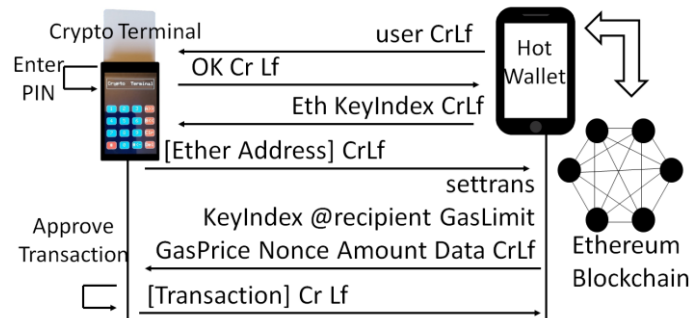
2.4 Generating transactions from Android mobile

A CryptoTerminal application for Android is available on PlayStore. This application "signs" a mobile file thanks to an Ethereum transaction.



https://play.google.com/store/apps/details?id=com.ethertrust.simple_terminal

A serial (RFCOMM) communication is opened between the crypto terminal and an Android mobile phone. The App sends the "user" command that trigs PIN entering by the user; this operation unlocks the smartcard. Afterwards it collects Ethereum addresses bound to cryptographic keys, and finally requests (via the "settransf" command) the generation and signature of a transaction, which must be approved by the user



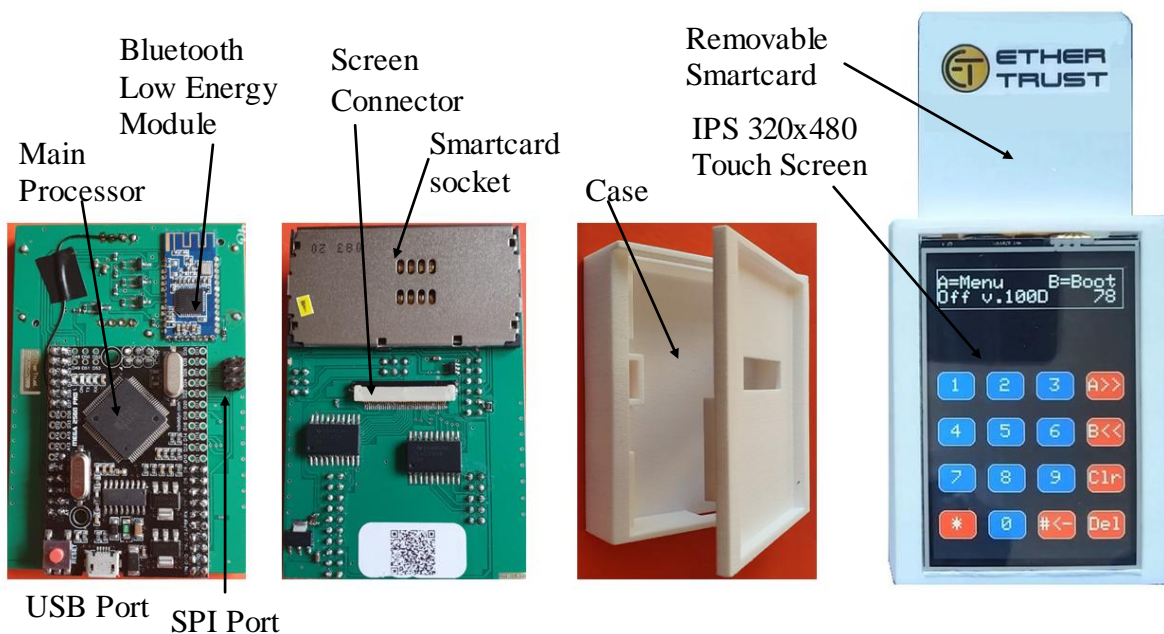
2.5 Generating transactions with BTOOLS

BTOOLS is open software available for Windows or Linux, which generates transactions for Bitcoin and Ethereum crypto currency platforms. In order to provide a strong security it supports the Crypto Terminal, 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:

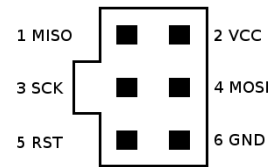
- Bitcoin address generation (mainnet and testnet);
- Ethereum address generation;
- Bitcoin transaction generation;
- Ethereum transaction generation;
- Simple Bitcoin node client;
- Bitcoin transaction (via the Bitcoin client);
- Ethereum transaction (via WEB APIs);
- Crypto Terminal scripts for key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

2.6 Overview of the ECTv2 crypto terminal board



3 Uploading Firmware

The CryptoTerminal firmware can be downloaded with any ISP Programmer. The Crypto Terminal ICSP port has 6 pins.



We recommend using the open USBASP ISP programmer for which Ethertrust has developed strong security features. This programmer is equipped with an open bootloader customized by Ethertrust.

3.1 USBASP Security procedure

10 PINs ICSP



- Make a shortcut between RESET pin and ground.
- The green LED blinks (three times), what indicates that the bootloader is unlocked, and ready to upload a firmware during 5 seconds.
- Download the bootloader static integrity probe firmware. If the test is OK green and red LEDs are blinking alternatively.

- Download the bootloader dynamic integrity probe firmware. After 20 seconds a five digit security code is displayed by blinking LEDs. Check the security code.
- If the two previous tests are OK, download the USBASP firmware.

3.2 Downloading the CryptoTerminal firmware

Connect the 6 PINs ISP cable to the crypto terminal ICSP port. A 6 PINs to 10 PINs adapter is usually needed with ISP programmer

Download the CryptoTerminal.hex file. The processor is ATMEGA2560.

3.3 Downloading the Bluetooth Low Energy firmware

Connect the CryptoTerminal to a PC, via an USB cable.



Start the BLE-Loader software on the PC and select the COM port used by the CryptoTerminal.

When asked by the BLE-Loader, press the key 3 on the CryptoTerminal in order to enter the *Load-BLE-Firmware* mode.

4 Checking Firmware Integrity

The Crypto-Terminal firmware and the BLE-module firmware integrity can be checked at any time.
On the main menu press the '7' key.



1=bMAC 3=bMACble
2=bMAC+EEPROM ?

Press '1' for checking Crypto-Terminal firmware integrity. Key '2' performs the same operation, but the EEPROM is erased. Press '3' for checking BLE-Module firmware

Seed: -----
#=Ok D=Del C=No

Enter a *seed* value, an integer value between 0 and 2147483647 ($2^{31} - 1$).

The displayed result comprises three data:
- the computing time (in 4µs unit for Crypto-Terminal, 64 µs for BLE-Module) first line, left part- the security code (first line, right part), a five digit number (16 bits). The security code is the XOR of MAC and computing time.
- the MAC value (32 bytes) expressed in hexadecimal.

81586363 57820
4FE958A1BF7C8658

Crypto-Terminal Checking
(computing time about 326 seconds)

1165679 34846
EA2C02C71657BD04

BLE-Module Firmware Checking
(computing time about 70s)

We recommend using the seed 0 for BLE-Module checking. When the seed 0 is used, BLE-Module checking runs a legacy sha256 procedure. In that case the MAC returned value should be: EA2C02C71657BD04AE7527007D11FAF6436E76225AEA7A15806C5D55C94F881E
For BLE-Module, the computing time may have some time jitter (+- 256 µs), therefore the security code can take a few values.

In summary the procedure that checks the terminal firmware integrity is the following
- Enter the seed for Crypto-Terminal integrity, and check the security code.
- Enter seed 0 for BLE-Module integrity and check the MAC.

5 Keypad

The ten digit keys ('0' to '9') are used to enter decimal digit.

The '#' key is used for ENTER operation.
The 'A' key is used as FORWARD operation.
The 'B' key is used as BACKWARD operation.
The 'C' key is used as CANCEL operation.
The 'D' key is used as DELETE operation.



The '*' key is used as ESCAPE operation.
For hexadecimal representation the sequence '*A' '*B' '*C' '*D' '*#' '**' are used for entering A,B,C,D,E,F hexadecimal digits.

For telephone keypad , the sequences '*1x' '*2' '*3x' '*4x' are used, x being the character index for the decimal key (ranging from 1 to 9)

For ASCII the '*0xyz' sequence is used, xyz being the ASCII code of the character.

For special character, sequences '*91', '*92' and '*93' are respectively used for ' ' , '!' And '@ ' .

For example a line editor can be start from the main menu, by typing 'A' '2' '0#' 'A' '4', what starts the label setting associated to a cryptographic key.

5.1 Root Menu (Idde Menu) Keys

- Key '1' : Serial USB activation
- Key '2' : Bluetooth Low Energy activation
- Key '3' : BLE-Module Loader activation
- Key '5' : Bluetooth options: PIN and Name
- Key '7' : Firmware Integrity Check
- Key '9' : Elliptic Curve Test
- Key 'A' : Main Menu
- Key 'B' : Boot Menu
- Key 'D' : Product Name



6 Crypto-Terminal Serial Commands

The Crypto-Terminal is equipped with an USB port supporting the *USB Serial* protocol, and a *Bluetooth Low Energy* Module with the *RFCOMM* stack.

- **Serial Configuration: 9600 Bauds, 8 bits, 1 Stop, No Parity**
- **All commands end by the Cr (Carriage Return) Lf (Line Feed) characters.**
- **An error occurred if the first token of the response is the string ERROR.**

6.1 Wrong Command

Action: Command Error

Cmd: WrongCommand CrLf

The first token of error response is **ALWAYS** ERROR

Response: ERROR WrongCommand CrLf

6.2 Empty Command

Action: Empty Command

Cmd: CrLf

Response: ERROR No Command ! CrLf

6.3 reboot

Action: Reboot the Crypto Terminal

Cmd: reboot CrLf

Response: Boot label

Boot label: *OK Ready EtherTrust Terminal v.101bL EEPROM Size: 4096 MaxRec:16 Crypto Checked uECC OK* CrLf

6.4 echo

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

Cmd: echo CrLf

Response: OK CrLf

6.5 disp

Action: test LCD for debugging purposes, display a text at x,y coordinates during 5s

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

Response: OK CrLf

Response: ERROR CrLf

6.6 edit

Action: test LCD text editor for debugging purposes.

Cmd: edit number-of-characters [Optional text]

Response: [text] CrLf

Response: ERROR CrLf

6.7 enter

Action: test KEYPAD for debugging purposes, enter data at x,y coordinates (time out 60s).

Cmd: enter x-value (0..15) y-value (0..1) number-of-characters CrLf

Response: [text] CrLf

Response: ERROR CrLf

6.8 debug

Action: Switch to Debug Mode

Cmd: debug CrLf

Response: OK CrLf

6.9 nodebug

Action: switch to non debug (normal) mode.

Cmd: nodebug CrLf

Response: OK CrLf

6.10 prompt

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

Cmd: prompt CrLf

Response: OK CrLf

6.11 noprompt

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

Cmd: noprompt CrLf

Response: OK CrLf

6.12 on

Action: Power the Secure Element

Cmd: on CrLf

Response: OK CrLf

6.13 off

Action: Unpower the Secure Element

Cmd: off CrLf

Response: OK CrLf

6.14 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

6.15 changeadm

Action: Modify 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

6.16 changeuser

Action: Modify 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

6.17 changeuser2

Action: Modify 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

6.18 adm

Action: Enter the Administrator's PIN on the crypto terminal (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

6.19 user

Action: Enter the user's PIN on the crypto terminal (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

6.20 user2

Action: Enter the user2's PIN on the crypto terminal (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

6.21 computekey

Action: Compute a key pair (private/public), from a key tree, with "hardened keys" only according to the BIP32 BITCOIN recommendation.
Administrator or User mode is required; the 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

6.22 genkey

Action: generate a key pair (private/public)

Administrator mode is required; the 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

6.23 setseed

Action: initiate a key tree with a seed or generate a tree with a random seed size

Administrator mode is required; the 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)

6.24 getseed

Action: read a tree seed.

Administrator mode is required; the reading **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

6.25 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

6.26 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

6.27 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

6.28 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

6.29 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 uncompress format.

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

Terminal Display:

1234567890123456

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

6.30 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)

6.31 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

6.32 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

6.33 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

6.34 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

6.35 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

6.36 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

6.37 signin

Action: Remote Signature Request.

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

6.38 signout

Action: Read 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

6.39 signoutr

Action: Read 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

6.40 signdo

Action: Do 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

6.41 settrans (generation of ether transaction)

Action:

The serial command "settrans" generates an Ether transaction according to the seven following 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 Ether address (20 bytes, 40 hexadecimal digits) of the transaction recipient
- Amount, the amount of the transaction (in Ether) in decimal format, such as 0.0.
- Data, the data associated to the transaction (text or hexadecimal, up to 448 characters)
 - Text example: "Hello World"
 - Text example: #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

6.42 signt

Action:

The serial command "signt" generates the signature of an ether transaction previously created

The command must be confirmed.

Terminal Display

```
1234567890123456
```

```
Sign 15 : yes=#  
eth: 0.0
```

```
1234567890123456
```

```
Sign 15 : yes=#  
Recovering
```

```
1234567890123456
```

```
Sign 15 : yes=#  
DONE
```

Cmd: signt

Response: :OK CrLf

Response: ERROR CrLf

6.43 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

6.44 settransf (generation and signature of ethereum transaction)

Action:

The serial command "settrans" generates an Ethereum transaction and associated signature according to the seven following 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 Ether address (20 bytes, 40 hexadecimal digits) of the transaction recipient

- Amount, the amount of the transaction (in Ether) in decimal format, such as 0.0.
- Data, the data associated to the transaction (text or hexadecimal, up to 448 characters)
 - Text example: "Hello World"
 - Text example: #Hello World
 - Hexadecimal example: \$313233

The command must be confirmed.

Terminal Display

```
1234567890123456
Sign 15 : yes=#
eth:0.0
```

```
1234567890123456
Sign 15 : yes=#
Recovering
```

```
1234567890123456
Sign 15 : yes=#
DONE
```

Cmd: settrans KeyIndex Nounce GasPrice GasLimit RecipientAddress Amount Data CrLf
 Response: : [Transaction encoded in hexadecimal] CrLf
 Response: ERROR CrLf

6.45 duplicate

Action: Duplicate a CCSC smartcard to another SCSC device.
 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
```

6.46 read

Action: Read data from smartcard non volatile 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

6.47 write

Action: Write data in smartcard non volatile 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

6.48 bMAC

Action: Check the Crypto Terminal firmware integrity.

Cmd: bmac seed [eeprom]

seed, an integer value between 0 and $2^{31}-1$

eeprom, an option that erases EEPROM

Response: :ComputingTime MAC CrLf

ComputingTime: the computing time value in 4µs unit

MAC: the 32bytes bMAC value

Response: ERROR CrLf

Terminal Displays:

1234567890123456

Seed: 1234567890

bMAC

7 CryptoTerminal Menus

The CryptoTerminal is equipped with an IPS touch screen whose resolution is 320x480. It emulates a 16 keys keypad, and a 2x16 characters LCD display.

7.1 Boot Message (BM)

1234567890123456

EtherTrust
Crypto Terminal

7.2 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

7.2.1 Serial USB Mode

Path IM.1

1234567890123456

Serial USB

7.2.2 Bluetooth Mode

Path IM.2

1234567890123456

Bluetooth

7.2.3 BLE Loader Mode

Path IM.3

1234567890123456

BT Loader

7.2.4 Bluetooth Menu (BTM)

Path IM.5

1234567890123456

1=AT AT=off
2=btPIN 3=btNAME

7.2.5 BMAC Menu (BMM)

Path IM.7

1234567890123456

1=bMAC 3=bMACble
2=bMAC+EEPROM ?

7.2.6 Boot Menu (BM)

Path: IM.B

1234567890123456

Reboot (type B)=?
Mode time

7.2.7 A Menu (AM)

Path: IM.A

1234567890123456

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

7.3 Bluetooth Setting Menu (BSM)

7.3.1 AT mode

Path: BTM.1

1234567890123456
Trying
OK

7.3.2 BT Pin

Path: BTM.2

1234567890123456
000000
btPIN

7.3.3 BT Name

Path: BTM.3

1234567890123456
ECTv2
btNAME

7.4 bMAC Run Menu (BRM)

7.4.1 Run bMAC

1234567890123456
Seed: 1234567890
#=OK D=Del Time

1234567890123456
Seed: 1234567890
bMAC Computing

1234567890123456
81586363 57820
4FE958A1BF7C8658

7.4.2 Run bMAC+EEPROM

1234567890123456
Seed: 130757
#=OK D=Del Time

1234567890123456
Seed: 130757
bMAC+EEPROM

1234567890123456
81776880 08341
F11E62C62DAD1793

7.4.3 Run bMAC BLE

1234567890123456
Seed: 0
#=OK D=Del Time

1234567890123456
Seed: 1234567890
BLE Computing

1234567890123456
1165679 34846
EA2C02C71657BD04

7.5 Mode Menu (MM)

Path: AM.1

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

7.5.1 Mode ON

Path MM.1

1234567890123456
Starting SE
? OK

7.5.2 Mode OFF

Path MM.2

1234567890123456
Stopping SE
? OK

7.5.3 Mode ADM

Path: MM.3

1234567890123456
Starting SE

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

```
1234567890123456
? Adm
? OK
```

```
1234567890123456
? Adm
? ERROR
```

7.5.4 Mode User

Path: MM.4

```
1234567890123456
Starting SE
```

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

```
1234567890123456
? User
? OK
```

```
1234567890123456
? User
? ERROR
```

7.5.5 Mode User2

Path: MM.5

```
1234567890123456
Starting SE
```

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

```
1234567890123456
? User2
? OK
```

```
1234567890123456
? User2
? ERROR
```

7.6 Key Menu (KM)

Path AM.2

7.6.1 Key Index

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

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

7.6.1.1 Sign

Path: KM.1

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

```
1234567890123456
? sign iii rec
? OK
```

```
1234567890123456
? sign iii rec
? ERROR
```

7.6.1.2 Genkey

Path: KM.2

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

```
1234567890123456
? genkey iii
? OK
```

1234567890123456

```
genkey iii
? ERROR
```

7.6.1.3 Record Index

Path KM.3

1234567890123456

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

7.6.2 Key Menu2 (KM2)

Path: KM.A

1234567890123456

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

7.6.2.1 CKey (Compute Key)

Path: KM.2.1

1234567890123456

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

1234567890123456

```
CKEY iii: yes=#?
.1000.10
```

1234567890123456

```
ComputeKey
DONE ?
```

1234567890123456

```
ComputeKey
ERROR ?
```

7.6.2.2 SetTree

Path: KM2.2

1234567890123456

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

7.6.2.2.1 SetTree

Path: KM2.2.1

1234567890123456

```
1234
SetSeed: hexa
```

1234567890123456

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

1234567890123456

```
SetSeed
DONE ?
```

1234567890123456

```
SetSeed
ERROR ?
```

7.6.2.2.2 Read

Path: KM2.2.2

1234567890123456

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

1234567890123456

```
GetTreeSeed
ERROR ?
```

1234567890123456

```
0123456789ABCDEF
TreeSeed: hexa
```

7.6.2.2.3 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 ?
```

7.6.2.3 GetL (Get Label)

Path: KM2.3

1234567890123456

```
Hello World !
?
```

1234567890123456

```
GetLabel
ERROR ?
```

7.6.2.4 SetL (Set Label)

Path: KM2.4

1234567890123456

```
Hello World !
Setlabel 3
```

1234567890123456

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

1234567890123456

```
SetLabel
DONE ?
```

1234567890123456

```
SetLabel
ERROR ?
```

7.6.3 Key Menu3 (KM3)

Path: KM2.5

1234567890123456

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

7.6.3.1 SetPr (Set Private Key)

Path KM3.1

1234567890123456

```
0123456789ABCDEF
SetPrivKey: hexa
```

1234567890123456

```
SetK iii: yes=#?
```

1234567890123456

```
SetPrivKey
DONE ?
```

1234567890123456

```
SetPrivKey
ERROR ?
```

7.6.3.2 GetPr (Get Private Key)

Path: KM3.2

1234567890123456

```
GetPrivKey
ERROR ?
```

1234567890123456

```
Priv iii: yes=#?
getpriv 3
```

1234567890123456

```
0123456789ABCDEF
PrivKey: hexa
```

7.6.3.3 GetPub (get public key)

Path KM3.3

1234567890123456

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

7.6.3.3.1 Raw

Path KM3.3.1

1234567890123456

```
GetPubKey
ERROR      ?
```

1234567890123456

```
0437A4AEF1F8423C
PubKey: hexa
```

7.6.3.3.2 BTC

Path KM3.3.1

1234567890123456

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

1234567890123456

```
1Mf9gx8gF9ocM42b
BTC Address 000
```

7.6.3.3.3 Hash160

Path KM3.3.2

1234567890123456

```
1234567890ABCDEF
HASH160 Address
```

7.6.3.3.4 Ethereum Address

Path KM3.3.3

1234567890123456

```
1234567890ABCDEF
ETHER Address
```

7.6.3.4 GetT (Get Tree Seed)

Path KM3.4

1234567890123456

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

1234567890123456

```
GetTreeSeed
ERROR      ?
```

1234567890123456

```
0123456789ABCDEF
TreeSeed: hexa
```

7.7 Duplicate Menu (Dup)

Path AM.4

7.7.1 Remove Card

1234567890123456

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

7.7.2 Insert New Card

1234567890123456

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

7.7.3 Enter ADM PIN

1234567890123456

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

7.8 Transaction Menu (Tr)

Path AM.5

1234567890123456

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

1234567890123456

```
Transaction 015
Eth=1      ?
```

7.8.1 Ether Menu (EM)

Path AM.5.1

1234567890123456

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

7.8.1.1 Create

Path EM.1

1234567890123456

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

1234567890123456

```
Save Transaction
yes=#?
```

7.8.1.1.1 Address

Path EM.1.1

7.8.1.1.2 Ether

Path EM.1.2

7.8.1.1.3 Data (text)

Path EM.1.3

7.8.1.1.4 Nounce

Path EM.1.4

7.8.1.1.5 Gas Price

Path EM.1.5

7.8.1.1.6 Gas Limit

Path EM.1.6

7.8.1.1.7 Data (Hexadecimal)

Path EM.1.7

7.8.1.2 Delete

Path EM.2

1234567890123456

```
Transaction
Delete #=?
```

7.8.1.3 Edit

Path EM.3

1234567890123456

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

1234567890123456

```
Save Transaction
yes=#?
```

7.8.1.3.1 Address

Path EM.3.1

7.8.1.3.2 Ether

Path EM.3.2

7.8.1.3.3 Data Text

Path EM.3.3

7.8.1.3.4 Nounce

Path EM.3.4

7.8.1.3.5 Gas Price

Path EM.3.5

7.8.1.3.6 Gas Limit

Path EM.3.6

7.8.1.3.7 Data Hexadecimal

Path EM.3.7

7.8.1.4 Sign

Path EM.4

1234567890123456

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

8 BTOOLS Script commands

BTOOLS (2.0) is open software (<https://github.com/purien/btools>) available for WINDOWS and LINUX and RASPBERRYPI, which generates transactions for Bitcoin and Ethereum crypto currency platforms. It is based on the OPENSLL library. In order to provide a strong security, it supports it the Crypto Currency SmartCard (CCSC) uses to generate and store secret keys, and to compute cryptographic (ECDSA) signature.

BTOOLS (2.0) is compatible with the CryptoTerminal.

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

8.1 Starting a serial terminal

A serial terminal is started by the command line : `btools -terminal delay`.
The typical value for delay is 1000 (ms)

8.2 CryptoTerminal Script

A *CryptoTerminal* script has a name that begins by the "_" character.

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

8.2.1 Index Array

An integer value can be stored in an *Index Array* (IA[256]), whose index is %xy, where xy is an hexadecimal two digit integer. For example %A0 (160 in decimal) means that index = TA[160].

An index affectation is done in the transaction script, for example %A0 4, means that KA[160]. = 4

8.2.2 start

Action: start a session with the CryptoTerminal

Cmd: start [optional COM port number] [optional delay in ms]

8.2.3 on

Action: power the CryptoTerminal secure element

Cmd: on

8.2.4 off

Action: unpower the CryptoTerminal secure element

Cmd: off

8.2.5 adm

Action: set the CryptoTerminal in Administrator mode

Cmd: adm

8.2.6 user

Action: set the CryptoTerminal in User mode

Cmd: adm

8.2.7 user2

Action: set the CryptoTerminal in User2 mode

Cmd: user2

8.2.8 getpub

Action: get the public key.

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

8.2.9 sign

Action: compute an ECDSA signature.

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

8.2.10 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)

8.2.11 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

8.2.12 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)

8.2.13 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)

8.2.14 raw

Action: send a command to the CryptoTerminal

Cmd: raw [CryptoTerminal Command]

Examples:

raw setseed TreeIndex SeedValue

raw compute TreeIndex KeyPath

9 BTOOLS CryptoTerminal Transaction Script Examples

9.1 Ethereum Transaction Generation

```
btools -genethtrans escript.txt escript.bin
```

9.1.1 CryptoTerminal script

```
////////////////////////////////////  
// _term.txt //  
////////////////////////////////////  
start 2 1000  
user  
getpub %A0  
sign %A0  
off
```

9.1.2 EthereumTransaction Script

```
////////////////////////////////////  
// escript.txt //  
////////////////////////////////////  
%A0 5  
apdu_script _term.txt  
nonce 14  
gasPrice 30000000000  
gasLimit 80000  
to 6BAC1B75185D9051AF740AB909F81C71BBB221A6  
value 0  
data "Hello World"  
// dataf file.txt
```

9.2 Bitcoin Transaction Generation

```
btools -gentrans bscript.txt bscript.bin
```

9.2.1 CryptoTerminal script

```
////////////////////////////////////  
// _term.txt //  
////////////////////////////////////  
start 2 1000  
user  
getpub %A0  
sign %A0  
off
```

9.2.2 Bitcoin Transaction Script

```
//////////////////////////////////
// bscript.txt //
//////////////////////////////////

sequence ffffffff
locktime 00000000

%A0 4

nb_input 1

input
transaction      3573f3007a83db8b0eb04a808bf76983dba8270be9c2df44a196
                  97517ab2c062
index 1
apdu_script _term.txt

nb_output 2

output "Hello Ethertrust Crypto Terminal !"
output
fee 0.0010
btc 0.5270
adr mtekQilEPrxo5TngwipB31VJTihSCTjxjk
```

10 The Crypto Currency SmartCard

The Crypto Currency smartcard (CCSC), of which AID is 010203040500 has three PINs, administrator, user, and user2. The default values are 8 zeros (30303030303030) 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 Read/Write non volatile memory (16KB), protected by a dedicated PIN (User2), is available for the storage of any sensitive information.

10.1 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

10.2 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

10.3 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

10.4 The Verify AdminPin command

This command verifies the administrator pin. It gives access to all available features of the crypto currency 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	08	303030303030303030
			FF Reset to default		

Response

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

10.5 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

10.6 The GetStatus command

This command returns the current state of the crypto currency 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= BigInteger (OK= 0x07)
 byte1: The maximum number of keys that can be used by the crypto currency 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)

10.7 The Write Command

This command writes data in the non volatile 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

10.8 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

10.9 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, or for P1=10 (Reset Key Tree) User or 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

10.10 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

10.11 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 or for KeyTree (P2 is a KeyTree index and P3#0) User PIN or 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, IH₁/IH₂/.../IH_n. The public key is not computed. The MSB bit of IH_i 32bits word **MUST** be set.

Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW 1=63, SW 2=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

10.12 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

10.13 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

10.14 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 or User
		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

10.15 The Set KeyParameter Command

This command sets the elliptic curve parameters, including public and private keys parameters. It requires the Admin PIN excepted for Parameter 6 (Public Key) for which User or Admin PIN is required.

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

10.16 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