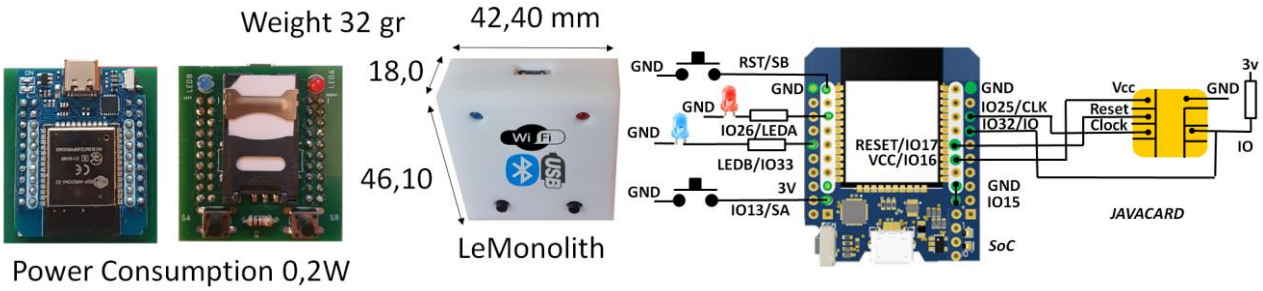
LeMonolith
A New Paradigm For Secure Element

LeMonolith Dev Kit (LEM) v0.4

1 Foreword

In a world increasingly digital, organizations and individuals need to protect their digital assets. Protection is done through encryption which is based on encryption keys. If a hacker is able to access your encryption keys, then you lose that protection. As a result securing encryption keys is a paramount objective. Organizations have deployed HSM (Hardware Security Modules), which are security hardened, intrusion and tamper resistant, to store and provision cryptographic keys for encryption, decryption and authentication. HSMs are expensive to purchase and manage. LeMonolith offers better security, at a fraction of the price, in a form factor that permits individual usage, and significantly reduces operational costs. In cube the size of a matchbox, it leverages the ubiquitous secure element (the chip in banking cards), highly vetted open-source security middleware, along with Bluetooth and WiFi for easy integration onto your home/office environment.

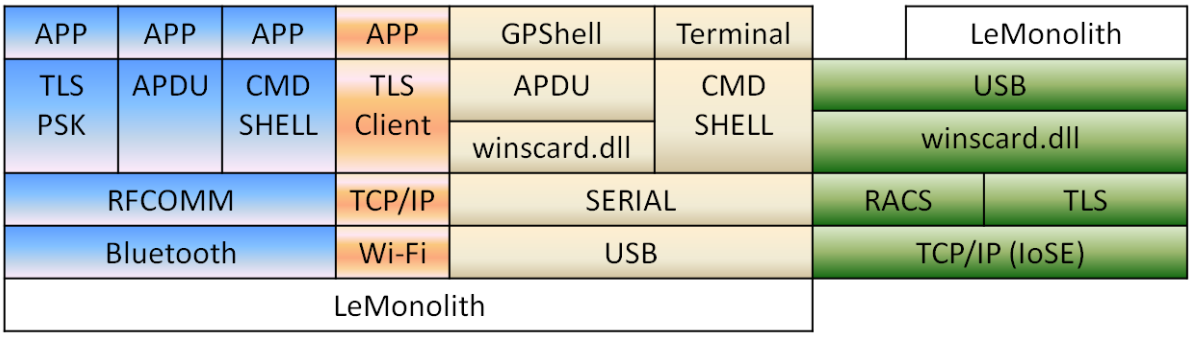
2 About LeMonolith



LeMonolith has two main components: an ESP32 D1 Mini board and a javacard. Its goal is to demonstrate security nano-servers, based on secure elements. An introduction to online TLS-SE secure element is available on youtube see:
see https://www.youtube.com/watch?v=0cLtrcMNjQ4

LeMonolith is an open device based on the ESP32-WROOM-32 module, which is made with two parts: an ESP32 System on Chip (SoC) comprising a RF section that implements Wi-Fi 2.4 GHz and Bluetooth 4.2, and a 4 MB serial FLASH. The ESP32 is clocked at 240 MHz, it is a dual-core system with two Harvard architecture Xtensa LX6 CPUs. Internal memories comprise 448 KB ROM and 520 KB SRAM. It includes CP2102 or CH9102 USB to UART Bridges.

Software development environment uses ARDUINO Integrated Development Environment (IDE), and Oracle Java Card Development Kit (JDK). Four JAVACARD applications (.cap files) are available: *TLS-SE*.cap (TLS for secure element), *CC*.cap (Crypto Currency), *TLS-IM*.cap (extended TLS Identity Module), and TLS-IM0.cap (minimal TLS Identity Module).



LeMonolith development kit (LEM DevKit) is a set of software tools that perform the following operations:
- Software downloading for ESP32 SoC and javacard.
- Smartcard use through the USB interface (i.e. smartcard reader)
- Smartcard use through the TLS-SE (TLS for Secure Element) Wi-Fi interface
- Smartcard as TLS identity module (TLS-IM) for Wi-Fi TLS server; a smartcard reader is available over Wi-Fi, secured by TLS-PSK (TLS pre-shared-key)
- Smartcard use from Bluetooth interface, including mobile application for Android and TLS-SE services over Bluetooth.

3 Initialization of the LEM DevKit

To install CP210x drivers for windows, see
https://www.silabs.com/documents/public/software/CP210x_Windows_Drivers.zip
To install CH9102 drivers for windows,
see https://www.wch-ic.com/downloads/CH343SER_ZIP.html

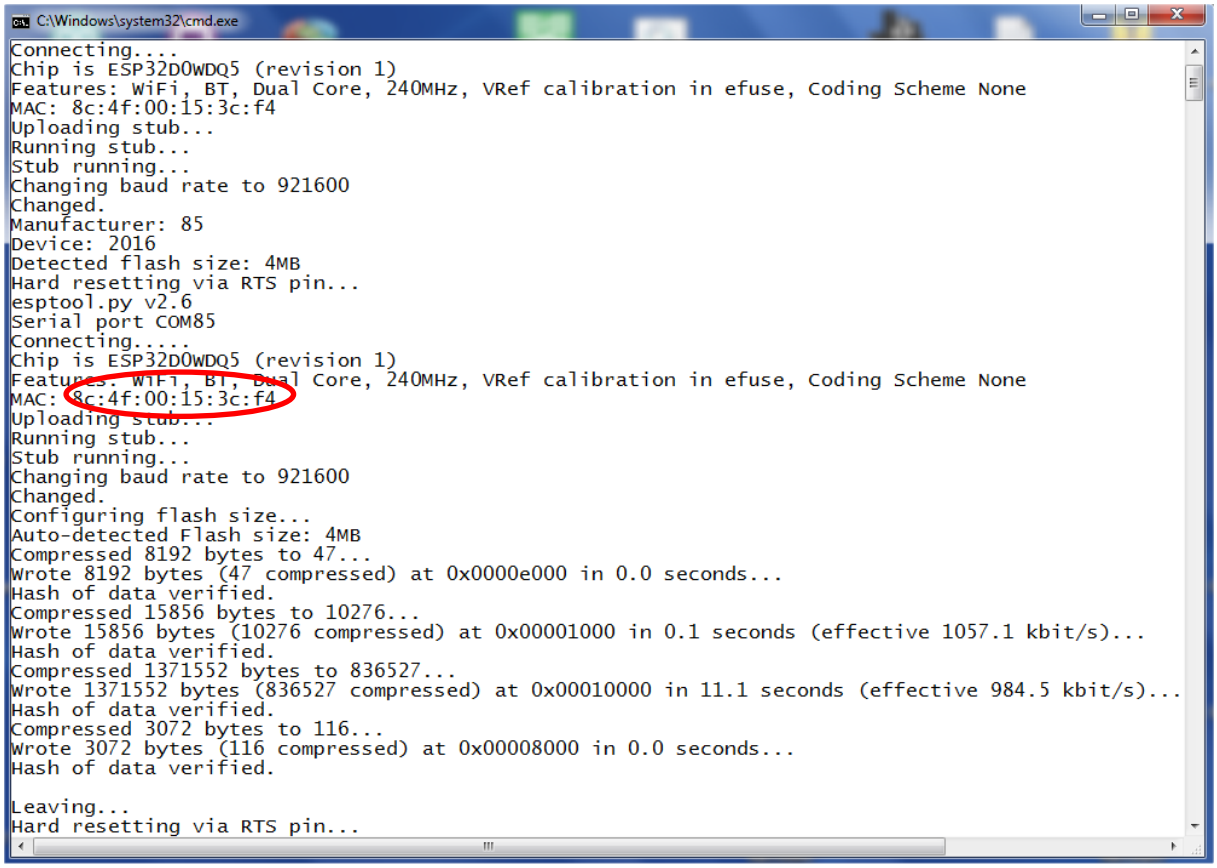
- Connect LeMonolith to USB port
- Go to /

In the file MAKE.bat enter the IP address is you already know it:
set MYIP=192.168.1.35
To use IoSE server, comment this line:
REM set MYIP=192.168.1.35
Execute MAKE.bat…the USB serial port in which is plugged LeMonolith is detected.

4 Loading software

4.1 ESP32 software loader

Goto /ESP32loader, execute loader41.bat for version 4.1 (loader2.bat for version 2, loader.bat for version 1)



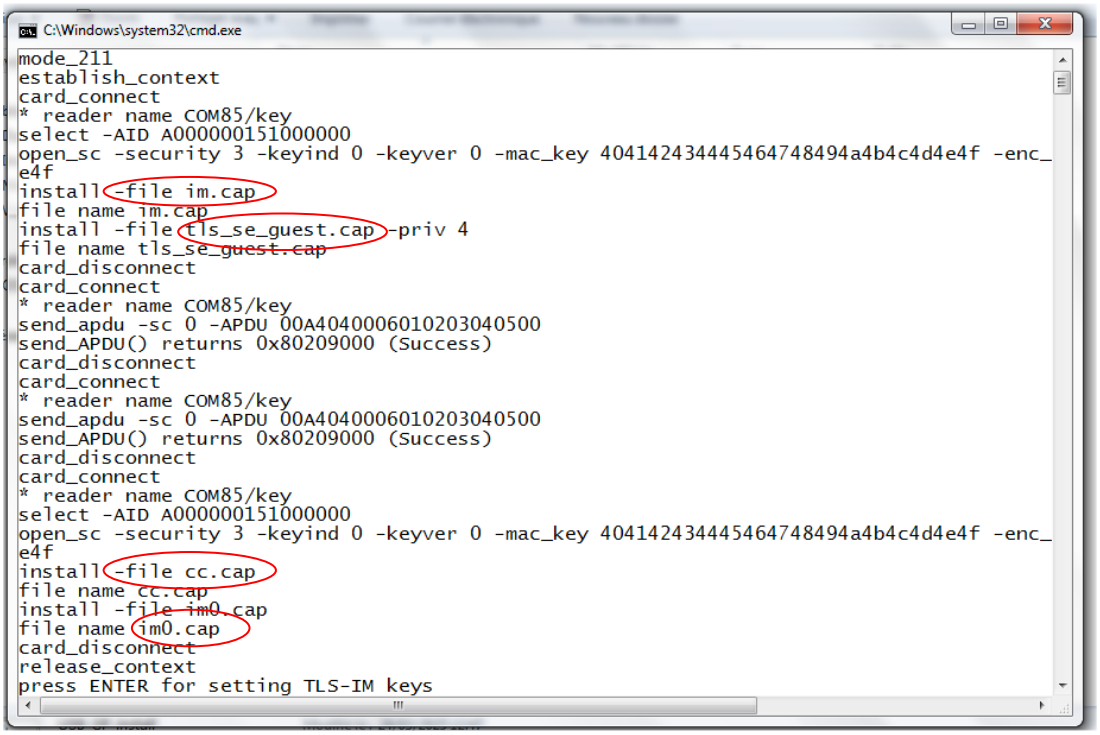
In this example the Wi-Fi MAC address is 8C:4F:00:15:3C:F4. The Bluetooth address is obtained by adding 2 to the Wi-Fi address (8C:4F:00:15:3C:F6).
Upon software downloading completion, LeMonolith is in the USB MODE.

4.2 Secure Element Software Loader

Go to /

4.2.1 Load applications in SE

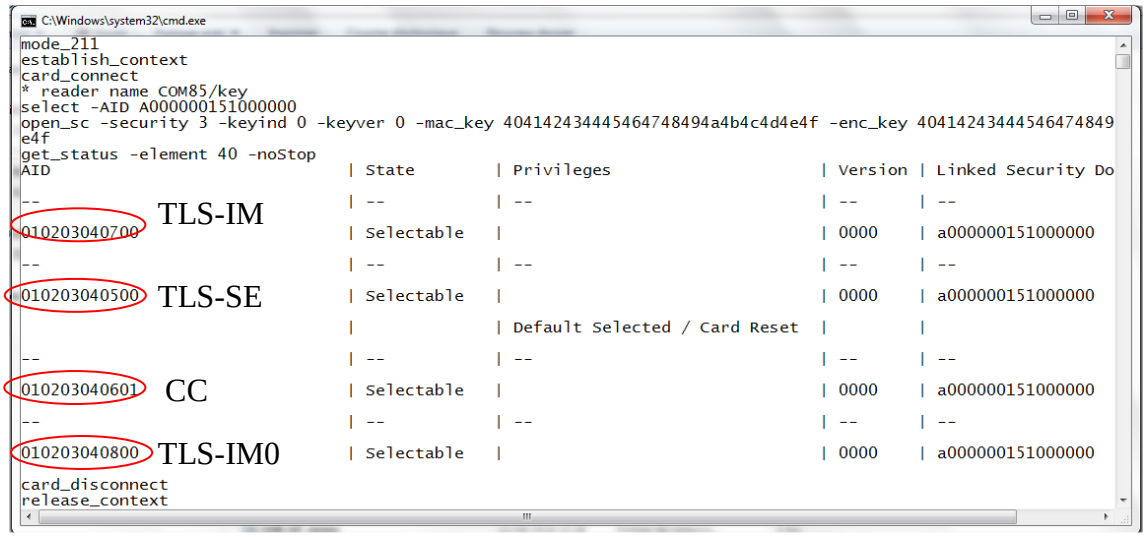
- Execute USB_GP_delete.bat, which removes all applications.
- Execute USB_GP_install.bat, which installs all applications.



4.2.2 List Applications stored in Secure Element

Execute USB_GP_list.bat, which lists all applications

Applications stored in the secure element are identified by a set of bytes called AID (Application IDentifier): for TLS-SE AID=010203040500, for CC AID=010203040601, for TLS-IM AID=01020304050700, for TLS-IM0 AID=01020304050800



5 Working mode selection

LeMonolith has three main working modes:

- USB, which works like smartcard reader
- Wi-Fi, which realizes a TCP/IP personal Hardware Secure Module (pHSM)
- Bluetooth, for applications with smartphone

There are two ways to select the working mode:
- By using the two push buttons
- By using a serial terminal

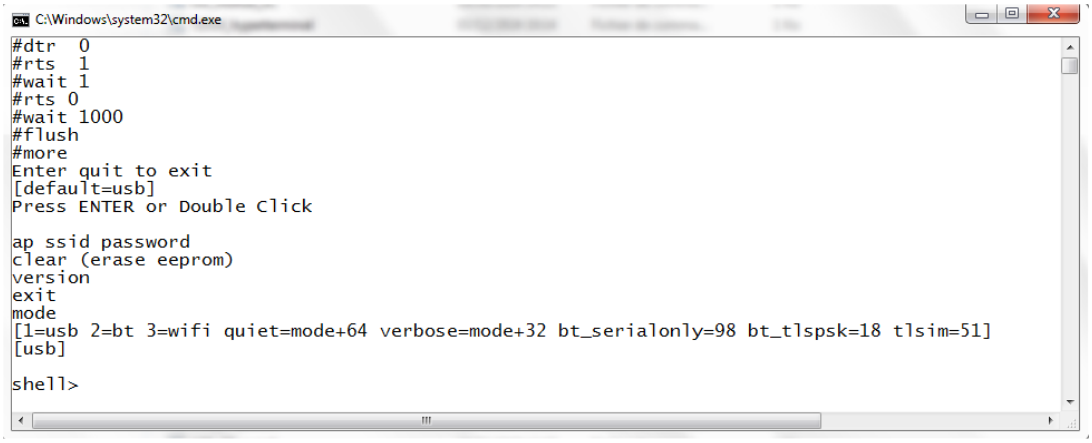
5.1 Using push buttons

- Hold ACK button
- Press shortly RESET button
- Release ACK button, when the blue LED is blinking
- Double press ACK button, the current working mode is displayed
- Double press ACK button to select another working mode
- Press RESET button to restart LeMonolith

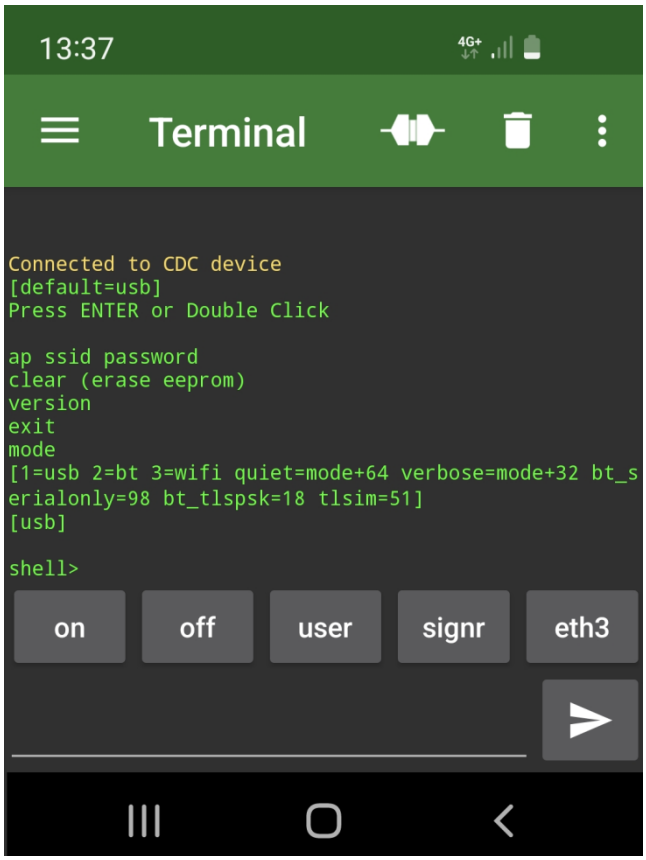
Mode	Blue LED	Red LED
USB	On	On
Bluetooth	On	Off
Wi-Fi	Off	On
Bluetooth USB	Blinking	Blinking
Bluetooth TLS-PSK	Blinking	Off
Wi-Fi + TLS-IM	Off	Blinking

5.2 Using Serial Terminal

The serial baudrate is 115200, with 8 bits, and no parity.
LeMonolith SDK provides the old version of windows HYPERTERMINAL and a dedicated terminal.
Execute TERM_terminal.bat (for dedicated terminal), OR TERM_hyperterminal.bat (for hyperterminal).



LeMonolith can also be powered by OTG under ANDROID, and works with the "Serial USB Terminal" application using baudrate=115200, 8 bits, no parity, end of line CR LF, and local echo.
See https://play.google.com/store/apps/details?id=de.kai_morich.serial_usb_terminal



- Hold ACK button
- Press shortly RESET button
- Release ACK button when blue LED is blinking

The following lines are displayed:

```
[default=usb]
Press ENTER or Double Click
```

Press ENTER to enter the configuration menu, the following lines are displayed:

```
ap ssid password
clear (erase eeprom)
version
exit
mode
[1=usb 2=bt 3=wifi quiet=mode+64 verbose=mode+32 bt_serialonly=98
bt_tlspsk=18 t tlsim=51]
[usb]

shell>
```

To fix the ssid and password for Wi-Fi, enter the following command
ap YourSSID YourPASSWORD, and press ENTER

```
>ap YourSSID YourPASSWORD
New ssid/passwd has been written in EEPROM

>shell
```

To select a mode type mode number and press ENTER

```
>mode 98
new mode 98 has been written in EEPROM

>shell
```

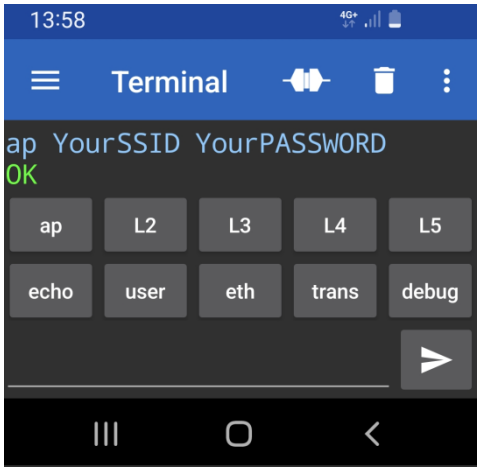
Mode	Comment
1	USB mode, works with TLS-SE shell (no debug)
2	BLUETOOTH mode, works with CC-SE (Crypto Currency) shell (debug)
3	Wi-Fi mode, works with TLS-SE (debug)
98	USB Bluetooth mode, works with CC-SE shell
18	Bluetooth with TLS-PSK (experimental)
33	USB debug mode
65	USB nodebug mode
34	Bluetooth debug mode
66	Bluetooth nodebug mode
35	Wi-Fi debug mode
67	Wi-Fi nodebug mode
51	Wi-Fi + TLS-IM debug mode
83	Wi-Fi + TLS-IM nodebug mode

5.3 Using Bluetooth mode for setting Wi-Fi parameters

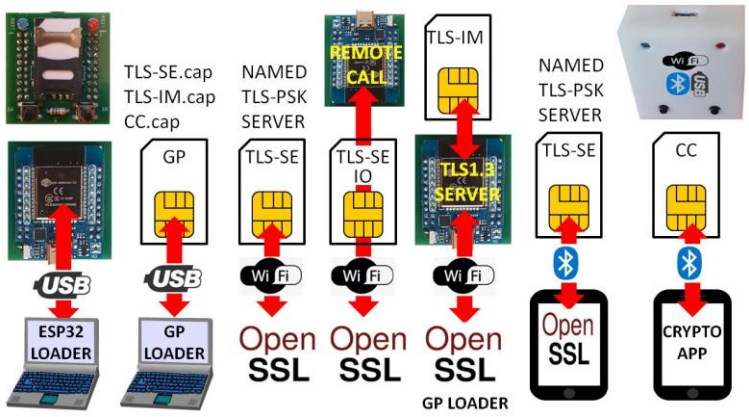
LeMonolith is compatible with ANDROID bluetooth application such as "Serial Bluetooth Terminal" (9600 bauds, end of line CR LF, and local echo)
https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal

In the Bluetooth mode a command (*ap*) is available in order to fix SSID and PASSWORD parameters.

- Select the Bluetooth mode
- Associate to *LeMonolith* device (RFCOMM profile, 9600 bauds, no parity)
- Type the command: ap YourSSID YourPASSWORD followed by ENTER



6 Overview



6.1 USB Mode

After software downloading LeMonolith is working in USB mode. Thanks to USB serial interface (115200 bauds, no parity, one bit stop) a *command shell* gives access to the secure element. It provides three main commands

- *on* powers on the secure element
- *A [Ascii hexadecimal encoded APDU]* sends ISO7816 command to secure element and returns response
- *off* powers off the secure element

Advanced commands (see section 7) may be used to modify some functional parameters such as frequency (F), baud rate (pts), transmission protocol (t0, t1),...

There are two ways for interfacing the serial element:

- a serial terminal associated to a script,
- a SHIM (*winscard.dll*, with a configuration file *cardconfig.txt*) that realizes a logical bridge between PC/SC API and serial USB. Open software, like GPSHELL, are compatible with LeMonolith if this SHIM is located in their repertory.

#dtr 0	F
#rts 1	ifs
#wait 1	on
#rts 0	hist
#wait 1000	A 00A4040006010203040500
#flush	A 0020000108303030303030303030
#timeout 10000	A 0081000000
null	A 0089000000
null	A 0082000000
pts	test
ta	off

Example of script used by serial terminal (in config/USB00.bat)

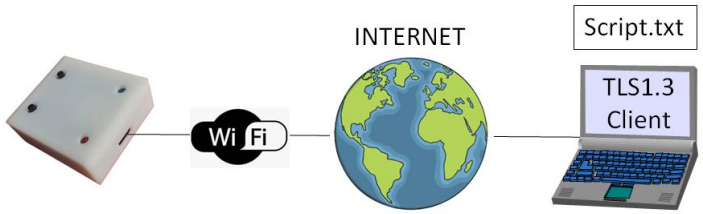
6.2 USB Bluetooth mode

This mode is similar to the USB-Mode, but it uses a different command shell (see section 8) dedicated to the CC (CryptoCurrency) javacard application.
The script below creates (*genkey 2*) an Ethereum account, using key at index 2, and generates (*settransf*) an Ethereum transaction in the file trans.txt *(#write*).

#dtr 0	adm 00000000
#rts 1	genkey 2
#wait 1	eth 2
#rts 0	eip155 11155111 key nonce GasPrice GasLimit
#wait 1500	#timeout 10000
#flush	settransf 2 45 10 100000
#timeout 2000	86F9E3E33BA7E42AB1128DA9291F675FA82546FF 0.0 #hello
#file trans.txt	#write
null	#timeout 2000 recipient address amount data
null	off

Script used by serial terminal for Ethereum transaction (./config/USB_TRANS.bat)

6.3 Wi-Fi Mode



In this mode LeMonolith is an internet server, providing a command SHELL over a TLS1.3 server; so it supports a worldwide access. On the client side a TLS client is required, for example OPENSSL or WolfSSL. Commands implemented by the TLS-SE javacard application, are listed in section 9.
Commands like c02 (clear keys at index 2), g02 (generate keys at index 2), p02 (get public key at index 2), or s02[data] (sign data with private key at index 2) are executed by the secure element, over an end to end TLS session.
A connection to LeMonolith with OPENSSL is realized thanks to the following line

openssl s_client -tls1_3 -connect ip:444 -servername key1.com -groups P-256 -cipher DHE -ciphersuites TLS_AES_128_CCM_SHA256 -debug -tlsextdebug -msg -no_ticket -psk [PSK in hexadecimal]

A set of commands can be sent by using a dedicated TLS1.3 client (*client.exe*), for example:

client.exe -H #p02 -H #?02 –S key1.com -s –p 444 -h [*LeMonolith-IP*]
-l TLS13-AES128-CCM-SHA256 -H psk[*PSK in hexadecimal*]

A software tool (*parser2.exe*) can extract response recorded in a file, for example with:

- client.exe -H #p02 -H #?02 -S key1.com -s -p 444 -h [*LeMonolith-IP*] -l TLS13-AES128-CCM-SHA256 -H psk[*PSK in hexadecimal*] 1>log.txt

The command line: *parse2.exe log.txt*, creates rx_i.txt and tx_i.txt files, which contain i^{th} request (-H #request) and associated i^{th} response.

6.3.1 Using multiple identities

By default the TLS-PSK identity is *Client_identity*, which is associated to a root account in the secure element.

The command *?A501* enables a guest account. The main command available for this account is the reading of a memory record (Ixx, see section 10).

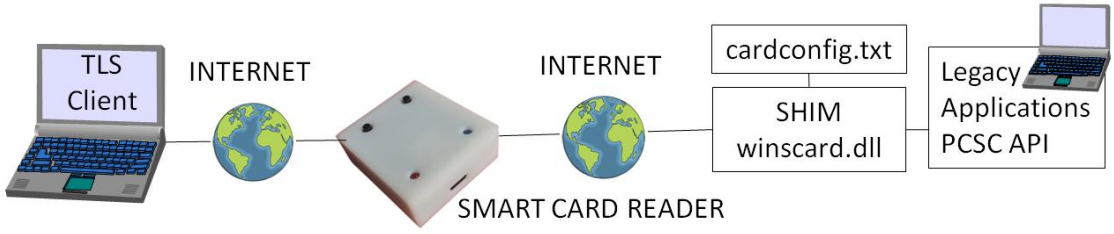
A connection to LeMonolith to an account *psk_identity* with OPENSSL is realized thanks to the following line :

openssl s_client -tls1_3 -connect ip:444 -servername key1.com -groups P-256 -cipher DHE -ciphersuites TLS_AES_128_CCM_SHA256 -no_ticket -psk [PSK in hexadecimal
-psk_identity [psk-identity]

A set of commands can be sent to an account psk-identity by using a dedicated TLS1.3 client (*client.exe*), for example:

client.exe -H #p02 -H #?02 –S key1.com -s –p 444 -h [*LeMonolith-IP*]
-l TLS13-AES128-CCM-SHA256 -H psk[*PSK in hexadecimal*] -H identity[*psk-identity*]

6.4 Wi-Fi TLS-IM Mode



In this mode LeMonolith provides a TLS1.3 server running in the ESP32 processor and secured by a TLS Identity Module (TLS-IM), i.e. the dedicated TLS-IM javacard application. Two security mechanisms are supported: X509 certificate that only realizes an echo procedure, and pre-shared-key (PSK) that gives access to a command shell (see section 9). When TLS1.3 PSK is used (with the TLS_AES_128_GCM_SHA256 cipher suite) *LeMonolith* is a remote smartcard reader.

LeMonolith software routes TLS packets to TLS-SE javacard application for cipher suite TLS_AES_128_CCM_SHA256, and to TLS-IM command shell for cipher suite TLS_AES_128_GCM_SHA256.

A connection to LeMonolith with OPENSSL is realized thanks to the following line:

openssl s_client -tls1_3 -connect ip:444 –groups P-256 –cipher DHE –ciphersuites TLS_AES_128_GCM_SHA256 –debug –tlsextdebug –msg -no_ticket
–psk [PSK in hexadecimal]

A set of commands can be sent by using a dedicated TLS1.3 client (*client.exe*), for example:

client.exe -H console -s –p 444 –h [*LeMonolith-IP*] -l TLS13-AES128-GCM-SHA256
–H psk[*PSK in hexadecimal*]

The *TLS-IM command shell* gives access to the secure element, with three main commands

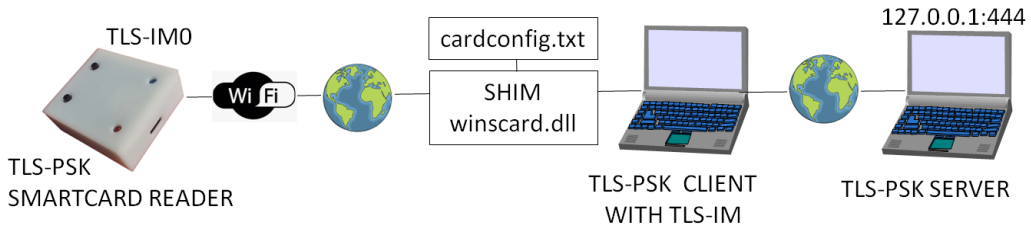
- *on* powers on the secure element
- *A [Ascii hexadecimal encoded APDU]* sends ISO7816 request secure element and returns response
- *off* powers off the secure element

There are two ways for interfacing the on-line smartcard reader:
- a TLS1.3 client,
- a SHIM (*winscard.dll*, with a configuration file *cardconfig.txt*) that realizes a logical bridge between PC/SC API and serial USB. Open software like GPSHELL (https://github.com/kaoh/globalplatform) are compatible with LeMonolith if this SHIM is located in their repertory.

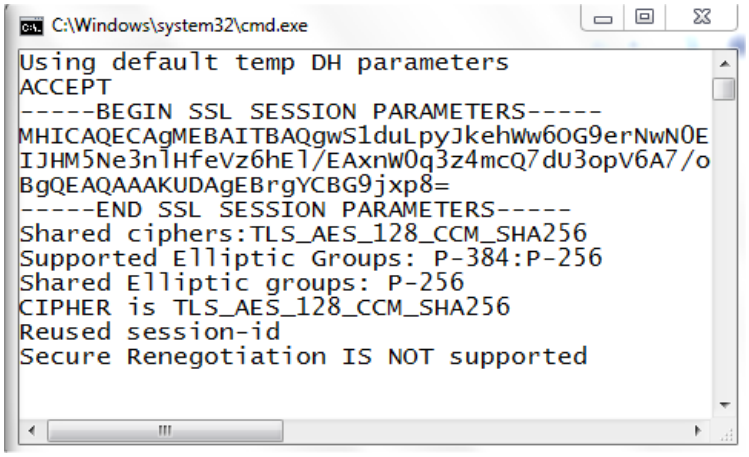
```
seid 192.168.1.35:444/key
psk 0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20
tx 1
```

Example of cardconfig.txt file used by wincard.dll SHIM.

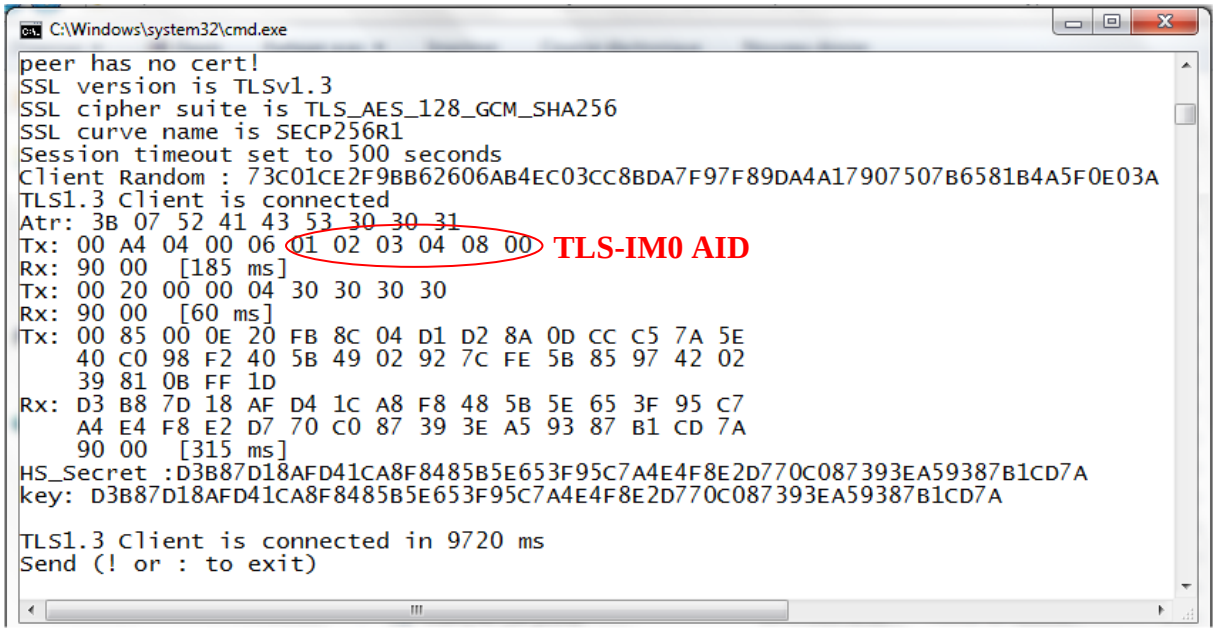
6.4.1 An example of remote smartcard reader use



A TLS-PSK server is started on 127.0.0.1:444
(TLSIM_SERVER_LOCAL_PSK_AESCCM.bat).

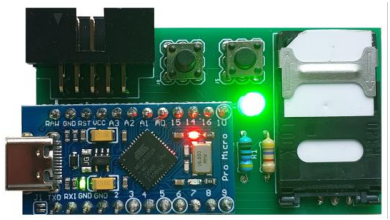


A TLS-PSK client works with a TLS-IM0 smartcard over PC/SC API. This smartcard is remotely used thanks to the SHIM winscard.dll
(TLSIM_SERVER_LEM_CONNECT_REMOTE.bat).



6.5 TLS Identity Module

TLS-IM tokens can be used on client side, in order to secure the TLS session. A TLS-IM token includes a smartcard with the TLS-IM javacard application. There are several form factors such as smartcard or multi-factor authentication (MFA) module.



Example of TLS-IM MFA Token

Under *USB mode* LeMonolith is a TLS-IM token, it can be used through USB serial interface or with a *winscard.dll* SHIM for PC/SC.

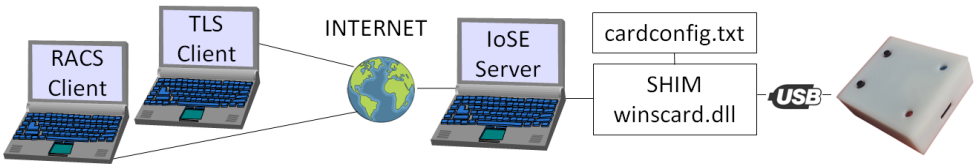
Here is an illustration of a line command with client.exe using USB serial (com58) that performs a connection to a TLS1.3 server *127.0.0.1:444*, with server name *key1.com*:

- client.exe -H $script.txt -H com58 -H hw1 -H aid010203040700 -H pin0000 -H console -S key1.com -s -p 444 -h 127.0.0.1 -l TLS13-AES128-CCM-SHA256
- The file script.txt is used to start LeMonolith (for example #dtr 0, #rts 1, #wait 1, #rts 0, #wait 1000, #flush).

Here is a line command with client.exe using PC/SC and *winscard.dll* SHIM (see section 5.1) that performs a connection to a TLS1.3 server 127.0.0.1:444 with server name key1.com:

- client.exe -H im -H aid010203040700 -H pin0000 -H console -S key1.com -s -p 444 -h 127.0.0.1 -l TLS13-AES128-CCM-SHA256

6.6 IoSE server for Wi-Fi mode



Internet of Secure Element (IoSE) server manages two TCP daemons: one for *Remote APDU Call Secure* (RACS) server (port 7777), and another for TLS1.3 front server port (8888). For windows the *winscard.dll* SHIM makes a bridge between PC/SC and USB serial. Within the IoSE server the secure element identifier (SEID) is *999*, and the secure element name server (SEN) is *COMX001*.

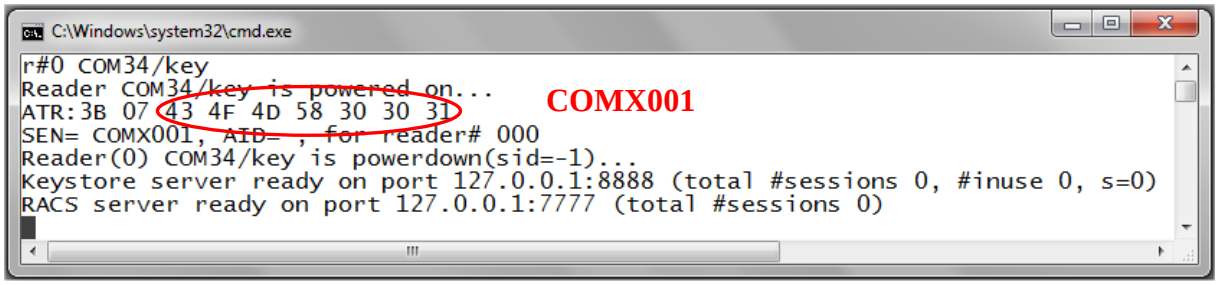
RACS uses TLS1.2 with X509 certificates for both server and client.
The LEM dev kit provides tools that generate certificates for certification authority CA in repertory *makecert/MakeID/DemoC/makeca.bat*, for server (*CommonName*=server) and for client (*CommonName*=client) in repertory *makercert/MakeID/makeClientServer.bat*.

With openssl a connection to RACS server, is realized by the following command line:

- Openssl s_client -tls1_2 -connect 127.0.0.1:7777 -cipher AES128-GCM-SHA256 -cert client.pem –key clientkey.pem -CAfile root.pem -verify 1 -pass pass:pascal

With openssl a connection to LeMonolith embedded TLS1.3 PSK server, is realized by the following command line:

- openssl s_client -tls1_3 -connect 127.0.0.1:8888 -servername COMX001 -groups P-256 -cipher DHE -ciphersuites TLS_AES_128_CCM_SHA256 -no_ticket -psk [PSK_Hexadecimal_Value]

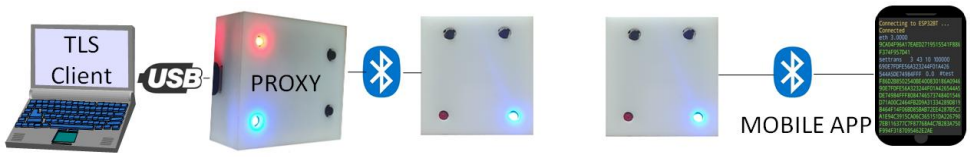


Secure elements hosted by RACS server are described in the winscard.dll SHIM: the server ip (127.0.0.1), tcp port (7777), and SEID (999) are provided by the file cardconfig.txt. Default CA is *root.pem*, default certificate is *client.pem*, and default keyfile is *clientkey.pem*. Thanks to this facility, software like GPHELL is transparently used for remote management of secure element applications.

seid 127.0.0.1:7777/999
tx 1

cardconfig.txt file for RACS SHIM

6.7 Bluetooth Mode



Bluetooth mode works with the RFCOMM (*Radio Frequency Communication*) protocol, which provides serial port emulation at 9600 bauds. Two types of applications are available:

- Command shell (described in section 8), used from dedicated mobile applications.
- TLS-SE command shell (see section 10.1) used over TLS-PSK communications with the secure element.

7 LeMonolith Quick Testing

For these tests LeMonolith must work in the Wi-Fi TLS-IM mode. The default PSK key is used, i.e. *0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20*.
The ip address is stored in the file myip.txt.
All commands are located in the ./ssl repertory
The command *makecardconfig.bat* builds a configuration file (cardconfig.txt) needed for winscard.dll shim.

```
seid 192.168.1.35:444/key
psk 0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20
tx 1
```

Example of cardconfig.txt file

7.1 openssl.bat, WolfSSL.bat

These two commands open TLS1.3 session with the TLS-SE applications, and use the AES_128_CCM_SHA256 cipher suite. In this scenario the secure element is a TLS server.

7.2 openssl_AESGCM.bat, WolfSSL_AESCGM.bat

These two commands open TLS1.3 session with the LeMonolith server, secured by the TLS-IM application; they use the AES_128_GCM_SHA256 cipher suite. In this scenario LeMonolith is a smartcard reader over TLS.

7.3 openssl_PKI.bat

This command opens TLS1.3 session with the LeMonolith server, authenticated by a certificate and secured by the TLS-IM application. It uses the AES_128_GCM_SHA256 cipher suite. It echoes the received TLS payload.

7.4 ListSE.bat

This command lists applications hosted by the secure element. It executes gpshell.exe with the winscard.dll SHIM.

7.5 DeleteSE.bat

This command deletes TLS-SE and CC application in the secure element. It executes gpshell.exe with the winscard.dll SHIM.

7.6 LoadSE.bat

This command installs TLS-SE and CC application in the secure element. It executes gpshell.exe with the winscard.dll SHIM.

7.7 sign.bat

This command delivers a certificate to the TLS-SE application.

7.8 auth.bat

This command realizes an authenticated TLS session with the TLS-SE application. It is part of the attestation procedure that is used before modifying the PSK.

7.9 TLS13_Server_PSK_AESCCM.bat

This command starts a local TLS server, which is used to demonstrate TLS-IM security module.

7.10 connect_127_0_0_1_444_im_pcsc.bat

This command starts a TLS client secured by the TLS-IM module. It illustrates a remote use of TLS-IM application stored in LeMonolith

7.11 reader.exe, winsreader.exe

These tools work with PC/SC and winscard.dll SHIM, which enables transparent remote use of LeMonolith applications.

8 USB Command SHELL

The red LED is on when the smartcard is powered-on.
The red LED is blinking when a command is sent to smartcard.

Command	Comments
test [number]*	test for n ECDSA signatures
nodebug	nodebug mode
debug	Debug mode
F [frequency in KHz]	Get/Set smartcard clock frequency (recommended value 6000)
ta [value]	Get/Set TA byte for PTS protocol (recommended value 12 in hexa)
pts [value]	Get/Set TA byte, TA= 0x10 + pts (recommended value 2)
nopts	No PTS protocol (pts=0)
ptcol	Get working T=x protocol (0-=>T=0, 1=>T=1)
t0	Force T=0 protocol
t1	Force T=1 protocol
ifs [value]	Get/Set IFS value for T=1 protocol (recommended value 254)
retry [number]*	Get/Set retry number for T=1 protocol ((recommended value 3)
finject [value]*	- bit 0 (1), inject CRC error for next T=1 request - bit 1 (2), inject CRC error for next T=1 response
on	Power smartcard
off	Unpower smartcard
hist	Get historical bytes from ATR
A [APDU in hexadecimal]	Send ISO7816 APDU in ASCII hexadecimal

* not available for Wi-Fi TLS-IM mode

9 USB Bluetooth Command SHELL

The red LED is on when the smartcard is powered-on.

Command	Comments
Empty	Return “ERROR No Command!”
echo	Return “OK”
nodebug	nodebug mode
debug	debug mode
F [frequency in KHz]	Get/Set smartcard clock frequency (recommended value 6000)
ta [value]	Get/Set TA byte for PTS protocol (recommended value 12 in hexa)
pts [value]	Get/Set TA byte, TA= 0x10 + pts (recommended value 2)
nopts	No PTS protocol (pts=0)
ptcol	Get working T=x protocol (0-=>T=0, 1=>T=1)
t0	Force T=0 protocol
t1	Force T=1 protocol
user PIN	Start smartcard, select CC-SE App, and present user PIN code (four decimal digits, default 0000)
changeuser oldpin newpin	Modify user PIN(4 decimal digits)
changeuser2 oldpin newpin	Modify user2 PIN(4 decimal digits)
changeadm oldpin newpin	Modify administrator PIN
user2 PIN	Start smartcard, present user2 pin code (for read/write operations in non volatile memory only, default 0000)

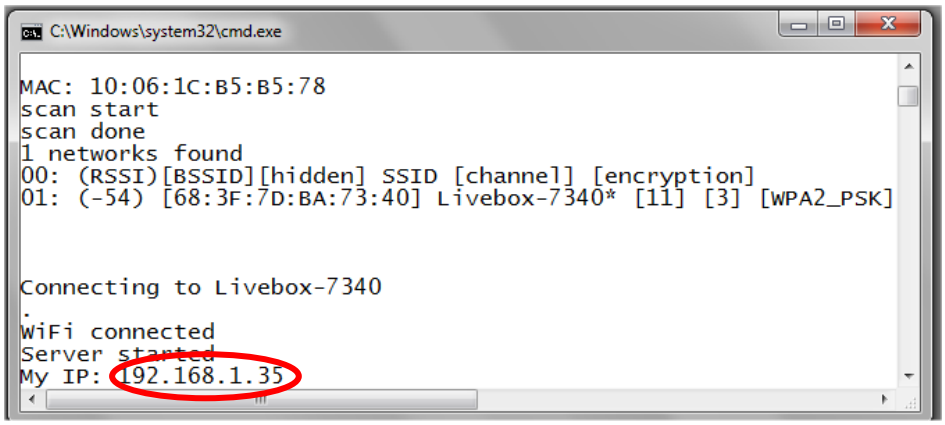
adm PIN	Start smartcard, select CC-SE App, and present user PIN code (eight decimal digits, default 00000000)
setlabel keyindex “text”	Associate a label to a keyindex
getlabel keyindex	Get keyindex label
~~recover keyindex~~	~~Compute recover parameter(0or 1) from a previous Ethereum transaction~~
check	Check a signed CC-SE App with the EtherTrust public key
~~content~~	~~Return the transaction buffer~~
tecc	Elliptic curve library test
binder 32bytes	Compute cryptographic binder for TLS 1.3
derive 32bytes	Compute handshake secret for TLS 1.3.
sign keyindex value	Compute ECDSA canonical signature for value (32 bytes)
signr keyindex value	Compute ECDSA canonical value and recover parameter for value (32 bytes)
status	Read CC-SE App status
read adr size	Read size bytes (maximum 256) in non volatile memory at adr (decimal)
write adr hexavalue	Write bytes (in hexa value) at at adr (decimal)
clear keyindex	Clear keyindex (1…15)
setseed keyindex hexavalue	Set BIP32 seed (up to 255 bytes in hexadecimal) for keyindex (1…15)
computekey keyindex path	Compute a key according to BIP32 with keyindex, path is a set of integers separated by ‘.’ ($i_1.i_2…..i_n$)
setpp keyindex privk	Set private and public key at keyindex using private key (privk)
setkey keyindex privk pubk	Set public key (pubk) and private key (privk) at keyindex
genkey keyindex	Generate a key at keyindex (1…15)
getpub keyindex	Read public key at keyindex (0,…,15)
getpriv keyindex	Read private key at keyindex (1,…,15)
getseed keyindex	Read BIP32 seed at keyindex
settransf param1= keyindex param2=Nonce (hexadecimal) param3=GasPrice in decimal GWEIs param4=GasLimit in decimal WEIs param5=Recipient Address (40 hexadecimal digits) param6=Amount in ETH floating point format(0.0) param7=Data #text or #$hexadecimal	settransf 1 45 10 100000 86F9E3E33BA7E42AB1128DA9291F675FA82546FF 0.0 #hello settransf 1 45 10 100000 86F9E3E33BA7E42AB1128DA9291F675FA82546FF 0.0 $1234 keyindex=1 nonce=45 GasPrice=10GWEI GasLimit=100000 amount=0.0 data=hello data=0x1234
btc keyindex [network ID]	BTC address with optional networked (0…255) associated to keyindex
hash160 keyindex	BTC hash160 address associated to keyindex
eth keyindex	Ethereum address (20 bytes) associated to keyindex
eip155 decimal-value	Set EIP155 ChainID value (1= mainnet, 11155111=Sepolia)
ap SSID PASSWORD	Set SSID and PASWORD for Wi-Fi mode

10 TLS-IM Wi-Fi Command Shell

The red LED is on when the smartcard is powered-on.
The red LED blinks when a command is sent to smartcard.

Command	Comments
nodebug	nodebug mode
debug	Debug mode
verbose 0/1	No verbose (0), or verbose (1)
F [frequency in KHz]	Get/Set smartcard clock frequency (recommended value 6000)
ta [value]	Get/Set TA byte for PTS protocol (recommended value is 12 in hexa)
pts [value]	Get/Set TA byte, TA= 0x10 + pts (recommended value 2)
nopts	No PTS protocol (pts=0)
ptcol	Get working T=x protocol (0-=>T=0, 1=>T=1)
t0	Force T=0 protocol
t1	Force T=1 protocol
ifs [value]	Get/Set IFS value for T=1 protocol (recommended value 254)
on	Power smartcard
off	Unpower smartcard
A [APDU in hexadecimal]	Send ISO7816 APDU in ASCII hexadecimal

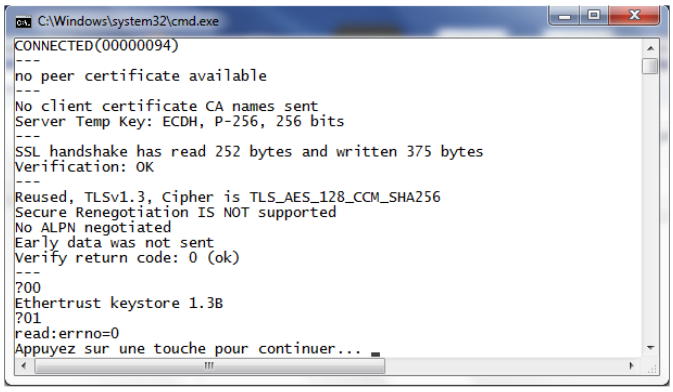
11 Wi-Fi Operations



- The blue LED is ON during Wi-Fi scan
- The blue LED is BLINKING when Wi-Fi is associated to an access point
- The blue LED is ON during TLS-PSK session establishment
- The red LED is ON when a TLS-PSK session is opened
- The red LED is BLINKING during access to smartcard.

11.1 Example of OPENSSL command line

openssl s_client -tls1_3 -connect IP:444 –servername key1.com –groups P-256 –cipher DHE –ciphersuites TLS_AES_128_CCM_SHA256 -no_ticket –psk 0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20



When Wi-Fi TLS-IM is activated the following command line gives access to an on-line smartcard reader interface:
openssl s_client -tls1_3 -connect IP:444 –servername key1.com –groups P-256 –cipher DHE –ciphersuites TLS_AES_128_GCM_SHA256 -no_ticket –psk 0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20

11.2 TLS-SE App commands

Command	Comment
?00	Version
?01	Disconnect
?01text	Echo text
?05[text]	Set root psk-identity
?06[text]	Set guest psk-identity
?0A	Get app ID
?0B	Get Certificate
?0C[64 hexa digits]	Authenticate(32 bytes)
?0D	Get Handshake Secret
?0E[64 hexa digits]	Set Certificate
?A0[64 hexa digits]	Set PSK2 (used with guest)
?A1	Get PSK2 (used with guest)
?A[2 hexa digits]	Bit 0 (01) enable guest psk-identity Bit 7 (80) check server name
?AA[OldPSK,NewPSK]	Set PSK OldPSK=64HexaDigit NewPSK=64HexaDigits
?FF[hexa digits]	Echo(Hexadecimal value]
cxy, Cxy	Clear key index=xy hexadecimal c for SECP256k1 C for SECP256r1
gxy	Generate SECP256k1, key index=xy hexadecimal
Gxy	Generate SECP256r1, key index=xy hexadecimal
sxy[64 hexa digits]	Sign value (up to 32 bytes), key index xy hexadecimal
pxy	Get public key, key index xy hexadecimal
rxy	Get private key, key index xy hexadecimal
Pxy(130 hexa digits]	Set public key (with prefix 04), 65 bytes at key index xy hexadecimal
Rxy[64 hexa digits] Xxy[64 hexa digits]	Set private key, 65 bytes at key index xy hexadecimal
txy[hexa digits] Txy[hexa digits]	Set BIP32 seed (up to 32 bytes) at key index xy hexadecimal
kxy[hexa digits] bxy[hexa digits]	Compute BIP32 key, at key index xy hexadecimal Path is a set of 32 bits value
vxy	Get BIP32 seed at key index xy hexadecimal
Zxy[hexa digits]	Write value (up to 32 bytes) in record number xy hexadecimal
Ixy	Read record number xy hexadecimal

11.3 TLS-SE Application Certification Procedure (ACP)

TLS-SE creates a pair of public/private key upon instantiation. The *Application Certification Procedure* (ACP) procedure reads the public key and writes a certificate (ECDSA signature of public key, two 32 bytes values R & S)).

```
// GetID= Get Application Public Key (over elliptic curve Secp256k1)
?0A
043288117A7871F1CC92E3204D444BD9E656C2047D4FCE189F2F3F22AF01B07D2665F0C5332
06333E37454A8D00A2803E07BFF7356ED6AE74D94D874334A022AEF
// Set Certificate= ECDSA_CAPrivKey(SHA2(AppPubKey))= 64 bytes= R || S
?0E55D20B301E6E6A543B8FF2DA1F7C42371042A88A556CF4ECD0E76BF9740C51C5D0CF9741
2BA12B6A8640BA48A90D3B6CA18C87981D7E95E0B7D3FEDEE068D2CF
OK
```

11.4 TLS-SE Session Authentication Procedure (SAP)

The *Session Authentication Procedure* makes the proof that remote node knows the TLS-SE private key (TLS-SE-PrivKey) and the TLS handshake secret.

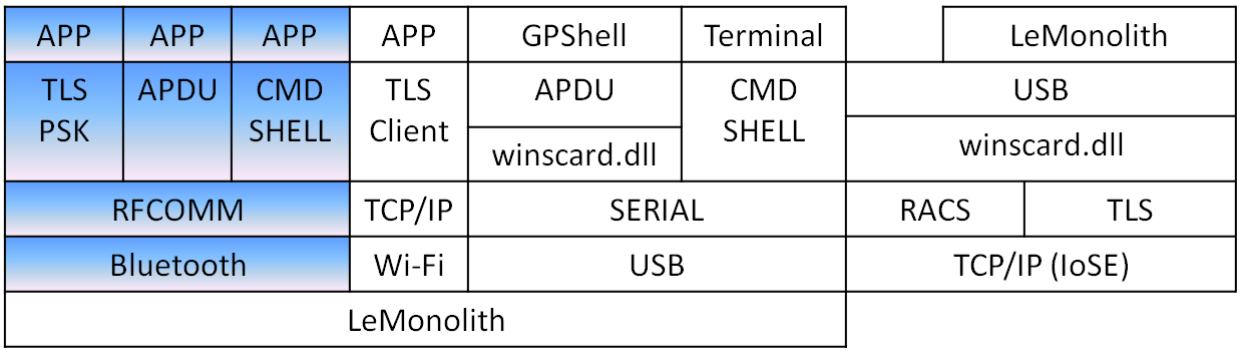
```
// GetID= Application Public Key
?0A
>>043288117A7871F1CC92E3204D444BD9E656C2047D4FCE189F2F3F22AF01B07D2665F0C53
3206333E37454A8D00A2803E07BFF7356ED6AE74D94D874334A022AEF
// Get TLS-SE Certificate
?0B
>>55D20B301E6E6A543B8FF2DA1F7C42371042A88A556CF4ECD0E76BF9740C51C5D0CF97412
BA12B6A8640BA48A90D3B6CA18C87981D7E95E0B7D3FEDEE068D2CF
// Authenticated Session Procedure
// Sign= Authenticate(32 bytes random)
// return ECDSA_TLS-SE-PrivKey(SHA2(HandshakeSecret || Random))
?0C7F69B857C6C2675BC8D5238E3E8BFC4C633FB5E39DD07F4760F508084FD1B482
>>304402206D2E688731C2673F977BD49B37D6CEC966323E966E34DE426D424AC5506F4A4B0
2203F5462E3D0AA7A1ED410ADDB29AE7C980EFC0136028FDF8533843D6A3C854ABF
```

11.5 TLS-SE-IO commands

TLS-SE-IO is a way to export command from secure element and then to return response. A TLS-SE-IO command is identified by a '#' prefix.

Command	Comment
#on	Blue LED on
#off	Blue LED off
#on$[decimal value]	LED on value=0=>blue, value=1=>red
#off$[decimal value]	KED on value=0=>blue, value=1=>red
#read	Read voltage on GPIO34
#read2	Read voltage on GPIO35
#vbat	Read voltage on GPIO35, corrected value, 2,5*input
#charge	Battery state (Full, High, Low, Critical)

12 Bluetooth Operations

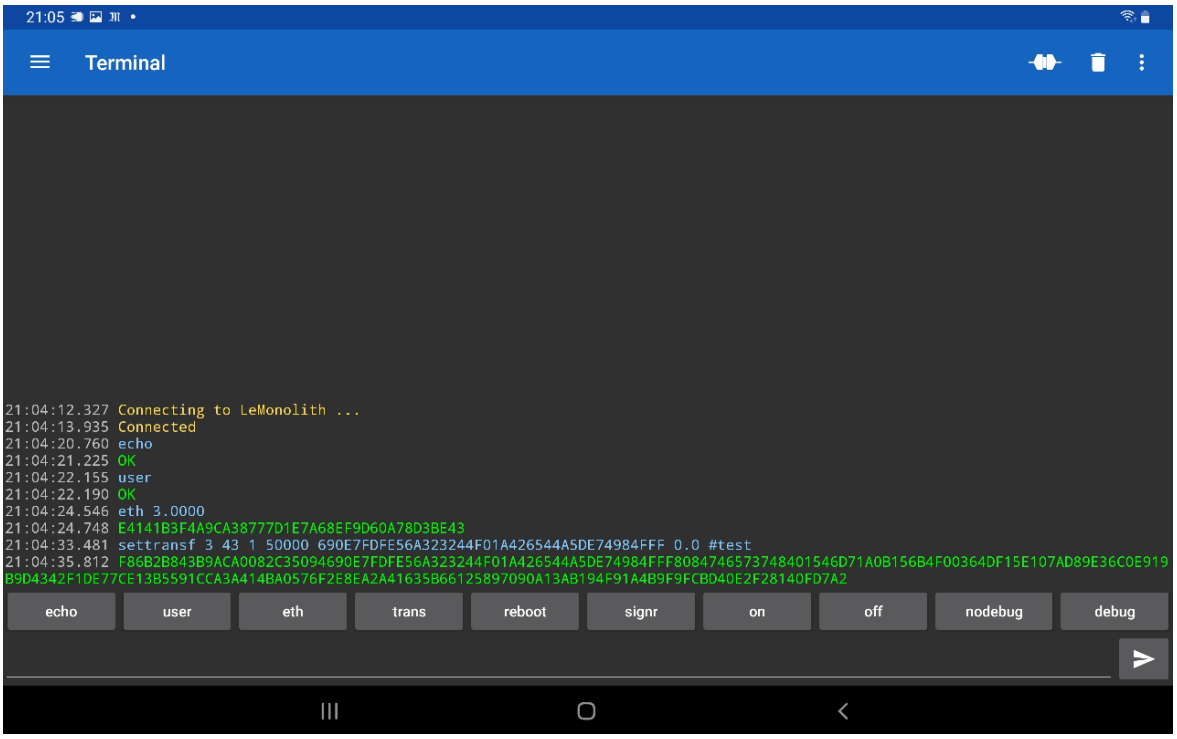


Command	Comments
Empty	Return "ERROR No Command!"
echo	Return "OK"
user	Only for Bluetooth Serial, return OK
eth keyindex.PIN	Only for Bluetooth serial Ethereum address (20 bytes) associated to key index PIN user 4 decimal digits PIN code
eip155 decimal-value	Set EIP155 ChainID value (1= mainnet, 11155111=Sepolia)
settransf param1= keyindex param2=Nonce (hexadecimal) param3=GasPrice in decimal GWEIs param4=GasLimit in decimal WEIs param5=Recipient Address (40 hexadecimal digits) param6=Amount in ETH floating point format(0.0) param7=Data #text or #$hexadecimal	settransf 1 45 10 100000 86F9E3E33BA7E42AB1128DA9291F675FA82546FF 0.0 #hello settransf 1 45 10 100000 86F9E3E33BA7E42AB1128DA9291F675FA82546FF 0.0 $1234 keyindex=1 nonce=45 GasPrice=10GWEI GasLimit=100000 amount=0.0 data=hello data=0x1234
Not Available for Bluetooth Serial compiled with the btstrict option	
nodebug	nodebug mode
debug	debug mode
pts [value]	Get/Set TA byte, TA= 0x10 + pts (recommended value 2)
nopts	No PTS protocol (pts=0)
ptcol	Get working T=x protocol (0-=>T=0, 1=>T=1)
t0	Force T=0 protocol
t1	Force T=1 protocol
on	Powert smartcard, select CC-SE-APP
on2	Power smartcard
off	Unpower smartcard
A [hexadecimal digits]	**Only for USB debug**. Send APDU
prompt	Prompt (>) is displayed
noprompt	Prompt (>) is not displayed
user PIN	**Only for USB debug** Start smartcard, select CC-SE App, and present user PIN code (four decimal digits, default 0000)

changeuser oldpin newpin	Modify user PIN(4 decimal digits)
changeuser2 oldpin newpin	Modify user2 PIN(4 decimal digits)
changeadm oldpin newpin	Modify administrator PIN
user2 PIN	Start smartcard, present user2 pin code (for read/write operations in non volatile memory only, default 0000)
adm PIN	Start smartcard, select CC-SE App, and present user PIN code (eight decimal digits, default 00000000)
setlabel keyindex "text"	Associate a label to a keyindex
getlabel keyindex	Get keyindex label
~~recover keyindex~~	~~Compute recover parameter(0or 1) from a previous Ethereum transaction~~
check	Check a signed CC-SE App with the EtherTrust public key
~~content~~	~~Return the transaction buffer~~
tecc	Elliptic curve library test
~~binder 32bytes~~	~~Compute cryptographic binder for TLS 1.3~~
~~derive 32bytes~~	~~Compute handshake secret for TLS 1.3.~~
sign keyindex value	Compute ECDSA canonical signature for value (32 bytes)
signr keyindex value	Compute ECDSA canonical value and recover parameter for value (32 bytes)
status	Read CC App status
read adr size	Read size bytes (maximum 256) in non volatile memory at adr (decimal)
write adr hexavalue	Write bytes (in hexa value) at at adr (decimal)
clear keyindex	Clear keyindex (1…15)
setseed keyindex hexavalue	Set BIP32 seed (up to 255 bytes in hexadecimal) for keyindex (1…15)
computekey keyindex path	Compute a key according to BIP32 with keyindex, path is a set of integers separated by '.' ($i_1.i_2.....i_n$)
setpp keyindex privk	Set private and public key at keyindex using private key (privk)
setkey keyindex privk pubk	Set public key (pubk) and private key (privk) at keyindex
genkey keyindex	Generate a key at keyindex (1…15)
getpub keyindex	Read public key at keyindex (0,…,15)
getpriv keyindex	Read private key at keyindex (1,…,15)
getseed keyindex	Read BIP32 seed at keyindex
btc keyindex [network ID]	BTC address with optional networked (0…255) associated to keyindex
hash160 keyindex	BTC hash160 address associated to keyindex

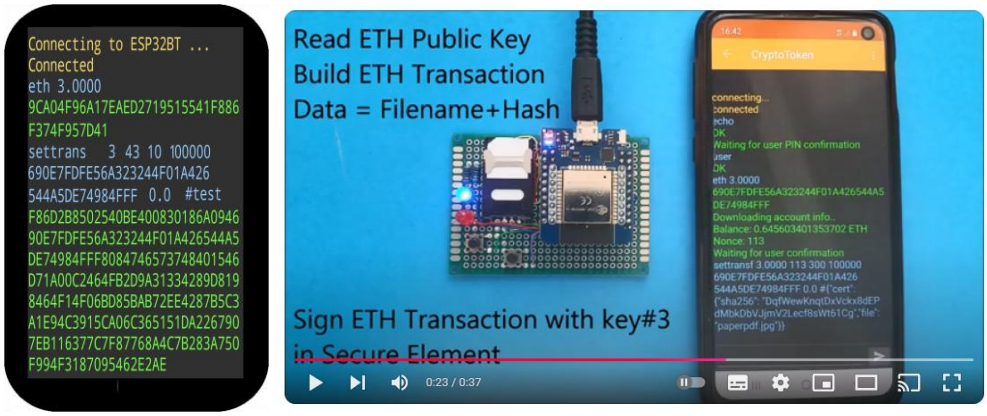
12.1 Serial Bluetooth terminal

The "Serial Bluetooth Terminal" is compatible with LeMonolith,
See https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal
LeMonolith is compatible with the *Serial Bluetooth Terminal* application, with baudrate=9600, end of line CR LF, and local echo.



12.2 Bluetooth CryptoToken App for Android

The Android application is located in /Android/monolith.apk. From a functional point of view it is similar to https://play.google.com/store/apps/details?id=pascal.urien.cryptoterminal.
This application realizes Ethereum transaction with LeMonolith, it signs a file located in the smartphone in the Ethereum blockchain. See the demonstration video on youtube https://www.youtube.com/watch?v=O8b_yfAkqRM



13 Bluetooth TLS-PSK (BTPSK)

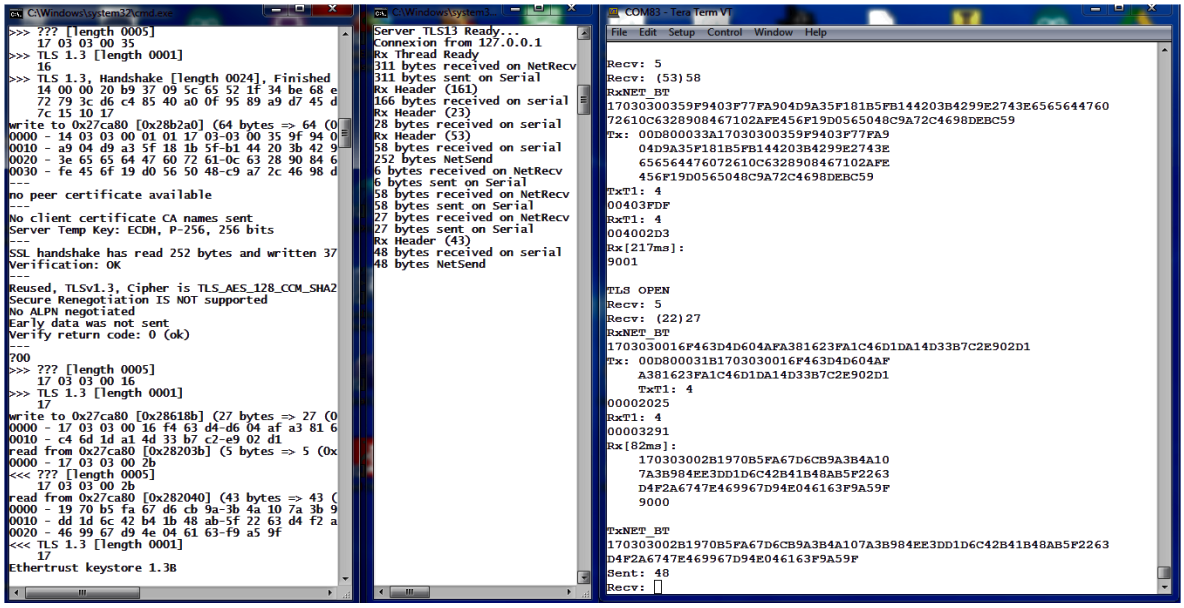
In this mode the TLS-SE server is running over Bluetooth RFCOMM in a transparent mode, i.e. TLS packets are exchanged in a transparent way over Bluetooth.

- Blue LED blinking: waiting for Bluetooth connection
- Blue LED on: Bluetooth connection established
- Blue LED off: TLS-PSK connection established
- Red LED on: smartcard powered
- Red LEB blinking: smartcard access

13.1 Testing Bluetooth TLS-PSK

A proxy application (proxy.bin) for LeMonolith realizes a bridge between TCP/IP socket and Bluetooth RFCOMM. It is available for *LeMonolith* device.

- Red LED on, the proxy is running
- Blue LED on, the Bluetooth connection is established with LeMonolith device

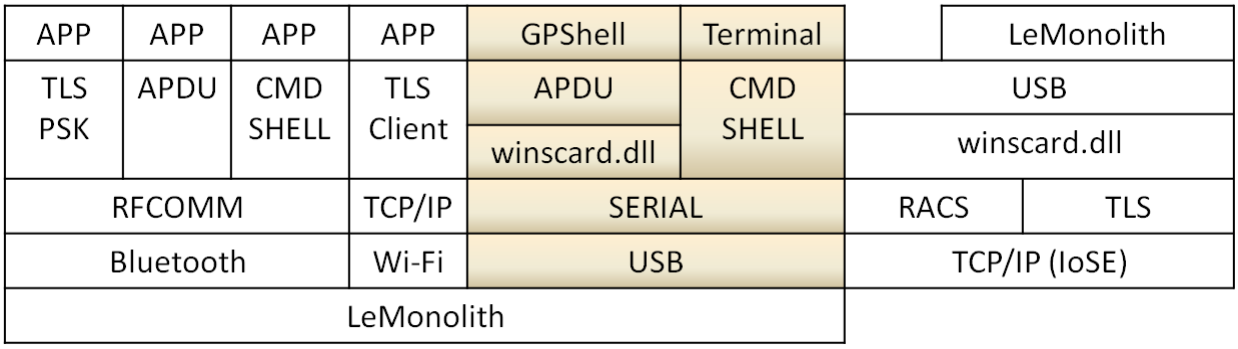


A video is available on youtube see https://www.youtube.com/watch?v=WyI4OxVTzHM

14 LeMonolith (LEM) Dev Kit Tests

14.1 USB Operations

Select the USB mode.



14.1.1 COM_List.bat

List COM port.

14.1.2 COM_Find.bat

Detect LeMonolith, and write COM port number in the file com.txt.

14.1.3 TERM_hyperterminal.bat

Start hyperterminal.

14.1.4 TERM_terminal.bat

Start terminal.

14.1.5 USB_GP_list.bat

List javacard applications stored in the secure element.

14.1.6 USB_GP_delete.bat

Delete **all** javacard applications (TLS-IM, TLS-SE, CC, TLS-IM0) stored in the secure element.

14.1.7 USB_GP_install.bat

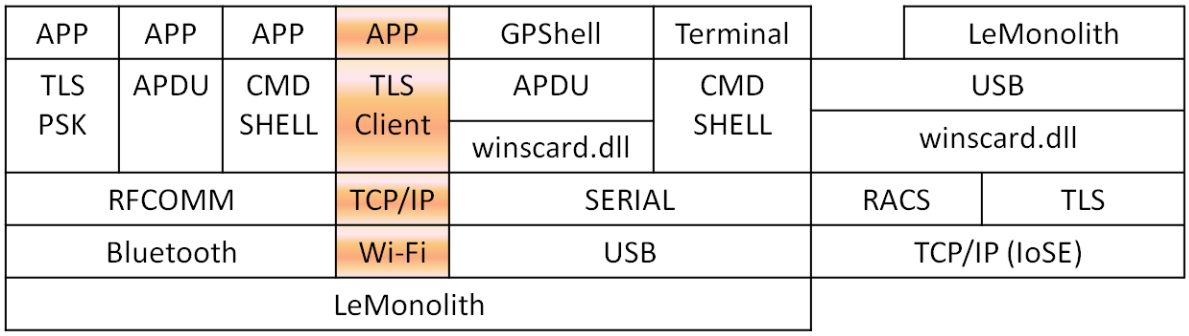
Install tls_se_guest.cap (TLS-SE App), cc.cap (Crypto Currency App), im.cap (TLS-IM app), and im0.cap (TLS-IM0) in the secure element

14.1.8 USB_KEYSTORE_Genkey00.bat

Create a key (for the curve SECP256k1) at index 0 in TLS-SE App.

14.2 Wi-Fi Operations

Select the Wi-Fi mode.



14.2.1 SSL_openssl.bat
Open a TLS session with OPENSSL.

14.2.2 SSL_wolfssl.bat
Open a TLS session with WOLFSSL.

14.2.3 SSL_openssl_guest.bat
Open a TLS session with OPENSSL using the guest psk-identity. The guest identity is enabled with the *?A501* command.

14.2.4 SSL_wolfssl_guest.bat
Open a TLS session with WOLFSSL using the guest psk-identity. The guest identity is enabled with the *?A501* command.

14.2.5 SSL_wolfssl_MFA.bat
Open a TLS session with the Multi Form Authentication (MFA) TLS-IM token

14.2.6 SSL_wolfssl_PCSC.bat
Open a TLS session with the TLS-IM smartcard.

14.2.7 KEYSTORE_NET_Load_Key.bat
Load a private key from keystore\eth\mypp.txt at index 3, using PSK=keystore\eth\mypsk.txt

14.2.8 KEYSTORE_NET_Load_Key_SC.bat
Load a private key from \keystore\eth\mypp.txt at index 3, using a TLS-IM smartcard.

14.2.9 KEYSTORE_NET_Load_Key_MFA.bat
Load a private key from \keystore\eth\mypp.txt at index 3, using a TLS-IM MFA Token.

14.2.10 KEYSTORE_NET_test_sign.bat
Perform ECDSA signatures with key at index 3.

14.3 Wi-Fi Operations with TLS-IM

14.3.1 TLSIM_GP_USB_LOADER_IM.bat
Load TLS-IM software in the secure element, **under USB mode**.

14.3.2 TLSIM_GP_USB_DELETE_IM.bat
Delete TLS-IM software in the secure element, **under USB mode**.

14.3.3 TLSIM_GP_USB_PERSO_IM.bat
Load (**under USB mode**) a PSK key in the secure element, and an X509 certificate used for echo demonstration.

14.3.4 TLSIM_GP_USB_LOADER_IM0.bat
Load TLS-IM0 software in the secure element, **under USB mode**.

14.3.5 TLSIM_GP_USB_DELETE_IM0.bat
Delete TLS-IM0 software in the secure element, **under USB mode**.

14.3.6 TLSIM_GP_USB_PERSO_IM0.bat
Load (**under USB mode**) a PSK key in the secure element.

14.3.7 TLSIM_LEM_Client_PSK_AESGCM_reader.bat
Connect to LeMonolith under Wi-Fi TLS-IM mode, with the ciphersuite AESGCM and the pre-shared-key (PSK). The USB SHELL commands (detailed in section 6) are available.

14.3.8 TLSIM_LEM_Client_PKI_AESGCM_echo.bat
Connect to LeMonolith under Wi-Fi TLS-IM mode, using the ciphersuite AESGCM and server authentication based on X509 certificate. Only an echo procedure is supported.

14.3.9 TLSIM_LEM_Client_PSK_AESCCM_tlsse.bat
Connect to LeMonolith under Wi-FI TLS-IM mode, with the ciphersuite AESGCM and pre-shared-key (PSK). TLS-SE App commands described in section 8.2 are available.

14.3.10 TLSIM_SERVER_LOCAL_PSK_AESCCM.bat
Start a TLS1.3 server with AESCCM ciphersuite and pre-shared-key (PSK), with localhost IP (127.0.0.1) and TCP port 444.

14.3.11 TLSIM_SERVER_LEM_CONNECT_PCSC.bat
Connect (under USB mode) to local TLS1.3 PSK server (127.0.0.1:444) with TLS-IM module (using PCSC emulation provided by the winscard.dll SHIM).

14.3.12 TLSIM_SERVER_LEM_CONNECT_SERIAL.bat
Connect (under USB mode) to local TLS1.3-PSK server (127.0.0.1:444) with TLS-IM module (using serial interface).

14.4 USB BLUETOOTH Tests

Select the USB_BLUETOOTH mode
Go in repertory /config
Start USB_TRANS.bat, which is an example of SEPOLIA (Ethereum) transaction generation.

15 Secure Element Certification Procedure over Wi-Fi

Select the Wi-Fi mode.

15.1 Loading Authority Certification Key (CA)

15.1.1 TLS-IM Smartcard

Go in the CertPCSC repertory, start init_ca_key_3.bat to download CA public/private keys in the TLS-IM smartcard.

15.1.2 TLS-IM MFA Token

Go in the /CertSerial repertory, start init_ca_key_3.bat to download CA public/private keys in the TLS-IM MFA token.

15.2 SE_NET_Cert_SOFT.bat

This script generates a certificate for LeMonolith, with software credentials.

15.3 SE_NET_Cert_SC.bat

This script generates a certificate for LeMonolith, with the TLS-IM smartcard.

15.4 SE_NET_Cert_MFA.bat

This script generates a certificate for LeMonolith, with TLS-IM MFA token

16 Secure Element Authentication Session Procedure (ASP) over Wi-Fi

Select the Wi-Fi mode.

16.1 SE_NET_auth_SOFT.bat

This script opens an authenticated session with LeMonolith.

16.2 SE_NET_auth_SC.bat

This script opens an authenticated session with LeMonolith, and requires a TLS-TM smartcard.

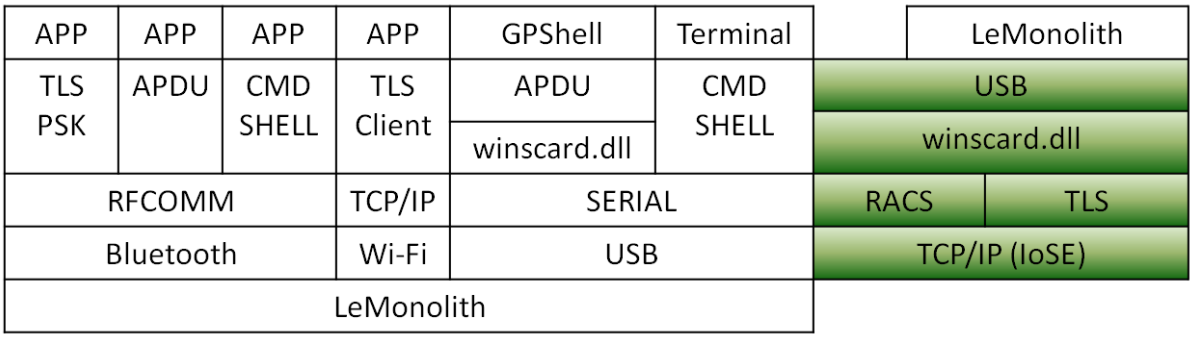
16.3 SE_NET_auth_MFA.bat

This script opens an authenticated session with LeMonolith, and requires a TLS-TM MFA token.

17 IOSE Tests

Select the USB mode.

The Internet of Secure Elements (IoSE) server starts two TCP daemons, RACS on port 7777 and TLS on port 8888. It uses LeMonolith as TLS-SE TLS1.3 PSK server, identified by the server name COMX001.



17.1 IOSE_Server_WIN32.bat
This script starts the IoSE server for windows.

17.2 IOSE_Server_Console.bat
This script starts the IoSE server in console mode.

17.3 IOSE_RACS_List.bat
This script lists the secure elements plugged to the IoSE server. LeMonolith is identified by two SEIDs 0 and 999 (default).

17.4 IOSE_RACS_Console
This script starts a RACS console.

17.5 IOSE_GP_list.bat
This script lists applications stored in the javacard.

17.6 IOSE_GP_delete
This script deletes **all** applications stored in the javacard.

17.7 IOSE_GP_install
This script Installs cc.cap (Crypto Currency App) and tls_se_2psk.cap (TLS-SE App) in the javacard.

17.8 IOSE_Openssl.bat
This script opens a TLS-PSK session with OPENSSL (127.0.0.1:8888).

17.9 IOSE_KEYSTORE_test_sign.bat
This script starts a test over TLS that performs ECDSA signatures, with key at index 0.

17.10 IOSE_Cert_SOFT.bat
This script generates a certificate for SE with software credentials.

17.11 IOSE_Cert_SC.bat
This script generates a certificate for SE with a TLS-IM smartcard.

17.12 IOSE_Cert_MFA.bat
This script generates a certificate for SE with a TLS-IM MFA token.

17.13 IOSE_auth_SOFT.bat
This script opens an authenticated session with LeMonolith.

17.14 IOSE_auth_SC.bat
This script opens an authenticated session with LeMonolith and requires a TLS-IM smartcard.

17.15 IOSE_auth_MFA.bat
This script opens an authenticated session with LeMonolith and requires a TLS-IM MFA token.

18 Ethereum Transactions over Wi-Fi
Select the Wi-Fi mode.
To understand the *Ethereum API* and get a free token, visit: https://etherscan.io/apis.

18.1 Ethereum transaction parameters
In file ./MAKE.bat
```
REM ETHEREUM TRANSACTION MAIN PARAMETERS
set GASPRICE=10
set APISERVER=api-sepolia.etherscan.io
set ETHSERVER=sepolia.etherscan.io
set TOKEN=0
set NETID=11155111
set ETHADR=62A52AC04BFB83723FF11295763E93B89D5DCB74
set ETHKEY=924121A5AAC0FAB04215B4A964D24681ACEC5D66ED61CD34F7770DAA37633F35
set ETHDATA="hello world"
```

18.2 ETH_gasview.bat
This script starts the URL https://sepolia.beaconcha.in/gasnow, which gives SEPOLIA GAS price.

18.3 ETH_NET_Make_Transaction.bat
This script makes a SEPOLIA transaction.

18.4 ETH_Transaction_Send.bat
This script sends a SEPOLIA transaction.

18.5 ETH_Transaction_View.bat
The script shows the last SEPOLIA transaction.

19 Software

19.1 Software components

Software components are located in the repertory ./ESP32Loader/monolith

- **Arduino IDE 1.8.9**
- **Select the board:** WEMOS D1 MINI ESP32
- **Sketch:** monolith2.ino
- **Dedicated Libraries**: ScLib5c, Cryptoecc, ripemd160, sha256, btools
- **Imported Libraries**: BigNumber
- **Arduino Standard Libraries**: WiFi, EEPROM Crypto

19.2 How to build LeMonolith

- Copy libraries: ScLib5c, Cryptoecc, ripemd160, sha256, btools, BigNumber, in the Arduino library repertory.
- Compile monolith2.ino, there is a library not found error (ScLib5c.a)
- Copy the file ScLib5c.located in the /ScLib5c/src/esp32 repertory in the Arduino build repertory (located in the Arduino preferences.txt file, *build.path*=)
- Compile monolith2.ino again, no error should be notified.

20 Online technical resources

20.1 TLS for Secure Element, TLS-SE

IETF draft TLS For Secure Element, https://datatracker.ietf.org/doc/html/draft-urien-tls-se-08

20.2 TLS for secure element input output TLS-SE-IO

IETF draft TLS for Secure Element Input Output, https://datatracker.ietf.org/doc/html/draft-urien-core-tls-se-io-02

20.3 TLS identity module, TLS-IM

IETF draft TLS Identity Module, https://datatracker.ietf.org/doc/html/draft-urien-tls-im-10

20.4 Remote APDU Server (RACS)

IETF Draft, Remote APDU Call Secure, https://datatracker.ietf.org/doc/html/draft-urien-core-racs-19

20.5 TLS for secure element Rendez-Vous TLS-SE-RDV

IETF draft, TLS For Secure Element Rendez Vous, https://datatracker.ietf.org/doc/draft-urien-tls-se-rdv/

21 ANNEXE

21.1 The Crypto Currency (CC) Application

The Crypto Currency smartcard (CCSC), of which AID is 010203040601 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 Read/Write non volatile memory (16KB), protected by a dedicated PIN (User2), is available for the storage of any sensitive information.

21.1.1 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	010203040601

Response

SW1	SW2
90	00

21.1.2 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

21.1.3 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

21.1.4 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 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

21.1.5 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

21.1.6 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= BigNumber (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)

21.1.7 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

21.1.8 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

21.1.9 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	Key Index [0,15]	00	Admin
00	81	SECP256k1 80 – Clear Keys & 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

21.1.10 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 modes 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

21.1.11 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, $IH_1/IH_2/\ldots/IH_n$. 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

21.1.12 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 *Key Dump* 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 P1 P2

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

21.1.13 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, or the user PIN for the *CardContent* option.
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

CLA	INS	P1	P2	P3	Payload	PIN required
00	86	FF CardContent	00	0 to 32	xx random bytes (r)	User or Admin

Response

Body	SW1	SW2	Comment
If P1=0 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
If P1=FF hash length 2 bytes hash value signature length 2 bytes signature value (ASN.1 encoded)	6C 90	xx 00	Read the signed card content sign(sha256(hash-value || r))
	63	80	PIN required
	69	85	Bad index

21.1.14 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

21.1.15 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

21.1.16 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

21.1.17 The GetCertificate command

This command reads the card certificate.
It requires the Admin or the User PIN.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).
The card certificate is the ECDSA SHA256 signature over the secp56k1 curve, of its public key (in full representation, i.e. 65 bytes) located at index 0. This certificate is the concatenation of two 32 bytes integers value r, s.

The experimental Ethertrust public key, on the curve secp256k1, is:
04
6099836D971593AAA2C1C32B6DB9EF9521041795E21CF1E7511DF3BD358F97DF
358B33A875E359CBE236163D6DBAEDFEC6C9393522C7EBC25A7CC85E1F0A7D67

Command

CLA	INS	P1	P2	P3	PIN required
00	8E	00	00	00	Admin or User

Response

Body	SW1	SW2	Comment
64 bytes certificate	90	00	Two 32 bytes integers r,s
Empty	63	80	PIN required

21.1.18 The SetCertificate command

This command sets the card certificate.
It requires the Admin PIN.
Upon success it returns the status word SW1, SW2 = 9000.

Two working mode are supported. The certificate (64 bytes) is imported or the certificate is internally computed from an embedded key (auto signing)

Command

CLA	INS	P1	P2	P3	Payload	PIN required
00	8F	00	00	40	64 bytes certificate	Admin
00	8F	FF	KeyIndex	00	Empty	Admin

Response

SW1	SW2	Comment
90	00	Certificate loaded or generated
63	80	PIN required
69	84	Certificate Already Set
69	85	Wrong Certificate Length
64	03	Private Key is not set

21.2 The TLS-IM Application

The TLS Identity Module application (extended) TLS-IM, of which AID is 010203040700 has two PINs, administrator, and user. The default values are 8 zeros (3030303030303030) for administrator and 4 zeros (30303030) for user.

Four P256 elliptic curve key pairs are available, and can be used for ECDSA signatures.

Procedures used for TLS1.3 PSK authentication are securely performed by the application.

A Read/Write non volatile memory (1KB), protected by a dedicated PIN (User), is available storage of X509 certificate or other purposes.

21.2.1 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	010203040700

Response

SW1	SW2
90	00

21.2.2 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 P1 P2

21.2.3 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 is 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 P1 P2

21.2.4 The ChangePin command

This command sets a PIN (UserPin,, AdminPin) to a new value
The P2 value is respectively 00, 01 for UserPin, , 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	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

21.2.5 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, 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 4 bytes.

b0=0,b1=2, version=b2.b3 (1.1)

21.2.6 The Write Command

This command writes data in the non volatile memory. This service is used for certificate store in the area [0000, 03FF]

It requires the previous checking of PINs UserPin 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 03FF 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	0400	Data Area	User or Admin

Response

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

21.2.7 The Read Command

This command reads data in the non volatile memory.
It requires the previous checking of PINs UserPin 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 03FF 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	0400	Data Area	User or Admin

Response

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

21.2.8 The Clear KeyPair command

This command MUST be used before any key setting or key generation operation.
This command clears the curve parameters.
The InitCurve command is needed to configure the Elliptic Curve

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	81	00	Key Index [0,3]	00	Admin

Response

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

21.2.9 The InitCurve command

This command initializes the elliptic curve SECP256r1 (P256) parameters

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	01 P256	Key Index [0,3]	00	Admin

Response

Data	SW1	SW2	Comment
	90	00	Key Reset Done
	63	80	PIN required
	69	85	Bad index
	64	01	Public Key is defined
	64	02	Private Key is defined
	6A	86	Incorrect P1P2

21.2.10 The Generate KeyPair Command

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	82	00	Key Index [0,3]	00	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

21.2.11 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 [0,3] FE for Cert Key FF for Ephemerous Key	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

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
	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

21.2.12 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 [0,3]	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

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

21.2.13 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 Raw Signature	Key Index [0,3] FE= Cert Key	Length 20	Data to be signed	Admin or User
00	80	21 Signature with SHA256	Key Index [0,3] FE= Cert Key	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

21.2.14 The Diffie Hellman Command

This command computes a DiffieHellman (DH) secret, and optionally generates an ephemerous pair of public/ private keys.
It requires no PIN.

Command

CLA	INS	P1	P2	P3	Data	PIN required
00	8A	01 P256 Curve	Key Index [0,3] FF for key generation	Length 41	Public Key Uncompress Format (04)	NO

Response

Body= Empty or 32 bytes DH	SW1	SW2	Comment
	90	00	No Error
	6D	50	DH Error

21.2.15 The GenerateRandom command

This command computes a set of random bytes.
It requires no PIN.

Command

CLA	INS	P1	P2	P3	PIN required
00	8B	00	00	Length	NO

Response

Body= P3 bytes or empty	SW1	SW2	Comment
	90	00	No Error
	69	85	Error

21.2.16 The HMAC Command

This command is used to compute keys used by TLS1.3
It requires Admin or User PIN.

Command

CLA	INS	P1	P2	P3	Data	PIN
00	85	00	02= Compute HMAC LengthKey [Key] LengthData [Data]	2+ LengthKey+ LengthData	32 bytes	Admin or User
		00 FF	0A= EXTRACT_EARLY P1=00 return ESK P1=FF return ESK, HSK, eBSK, rBSK, feBSK, frBSK	23	01 00 20 PSK (32 bytes)	Admin or User
		00 01	0B EXPAND_EARLY P1=0 "c e traffic" P1=1 "e exp master"	3+ DataLength	00 20 DataLength Data	Admin or User
		00	0C= HMAC_EBSK	Data Lentgh	Data	Admin or User
		00	0D=HMAC_RBSK	Data Length	Data	Admin or User
		00	0E=EXTRACT_HANDSHAKE (HMAC_HSK)	Data Length	Data (PubKey)	Admin or User

Response

Body 32 bytes 6x32 for = EXTRACT_EARLY with P1=FF	SW1	SW2	Comment
	90	00	No Error
	6A	86	Wrong P1P2

21.2.17 The GetCertificate command

This command reads the card certificate.
It requires the Admin or the User PIN.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).
The card certificate is the ECDSA SHA256 signature over the secp56k1 curve, of its public key (in full representation, i.e. 65 bytes) located at index 0. This certificate is the concatenation of two 32 bytes integers value r, s.

Command

CLA	INS	P1	P2	P3	PIN required
00	8E	00	00	00	Admin or User

Response

Body	SW1	SW2	Comment
64 bytes certificate	90	00	Two 32 bytes integers r,s
Empty	63	80	PIN required

21.2.18 The SetCertificate command

This command sets the card certificate.
It requires the Admin PIN.
Upon success it returns the status word SW1, SW2 = 9000.

Two working mode are supported. The certificate (64 bytes) is imported or the certificate is internally computed from an embedded key (auto signing)

Command

CLA	INS	P1	P2	P3	Payload	PIN required
00	8F	00	00	40	64 bytes certificate	Admin
00	8F	FF	KeyIndex	00	Empty	Admin

Response

SW1	SW2	Comment
90	00	Certificate loaded or generated
63	80	PIN required
69	84	Certificate Already Set
69	85	Wrong Certificate Length
64	03	Private Key is not set

21.3 The TLS-IM0 Application

The TLS Identity Module (minimal) application TLS-IM0, of which AID is 010203040800 has two PINs, administrator, and user. The default values are 8 zeros (3030303030303030) for administrator and 4 zeros (30303030) for user.

Procedures used for TLS1.3 PSK authentication are securely performed by the application.

21.3.1 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	010203040800

Response

SW1	SW2
90	00

21.3.2 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 P1 P2

21.3.3 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 is 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 P1 P2

21.3.4 The ChangePin command

This command sets a PIN (UserPin,, AdminPin) to a new value
The P2 value is respectively 00, 01 for UserPin, , 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	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

21.3.5 The HMAC Command

This command is used to compute keys used by TLS1.3
It requires Admin or User PIN.

Command

CLA	INS	P1	P2	P3	Data	PIN
00	85	00	02= Compute HMAC LengthKey [Key] LengthData [Data]	2+ LengthKey+ LengthData	32 bytes	Admin or User
		00 FF	0A= EXTRACT_EARLY P1=00 return ESK P1=FF return ESK, HSK, eBSK, rBSK, feBSK, frBSK	23	01 00 20 PSK (32 bytes)	Admin or User
		00 01	0B EXPAND_EARLY	3+ DataLength	00 20 DataLength	Admin or User

			P1=0 "c e traffic" P1=1 "e exp master"		Data	
		00	0C= HMAC_EBSK	Data Lentgh	Data	Admin or User
		00	0D=HMAC_RBSK	Data Length	Data	Admin or User
		00	0E=EXTRACT_HANDSHAKE (HMAC_HSK)	Data Length	Data (PubKey)	Admin or User

Response

Body 32 bytes 6x32 for = EXTRACT_EARLY with P1=FF	SW1	SW2	Comment
	90	00	No Error
	6A	86	Wrong P1P2

21.4 The TLS-SE Combi Application

This application is not a strict TLS-SE application, which should support only three commands RECV, SEND, and SELECT (optional). It provides some debug facilities and TLS-IM features.

21.4.1 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

21.4.2 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

21.4.3 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 is 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

21.4.4 The ChangePin command

This command sets a PIN (UserPin,, AdminPin) to a new value
The P2 value is respectively 00, 01 for UserPin andAdminPin.
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	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

21.4.5 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, 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 4 bytes.

b0=0,b1=2, version=b2.b3 (1.6)

21.4.6 The Write Command

This command writes data in the non volatile memory. This service is used for certificate store in the area [0000, 07FF]

It requires the previous checking of PINs UserPin 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 07FF 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	0800	Data Area	User or Admin

Response

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

21.4.7 The Read Command

This command reads data in the non volatile memory.
It requires the previous checking of PINs UserPin 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 07FF 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	0800	Data Area	User or Admin

Response

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

21.4.8 The Clear KeyPair 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 Elliptic Curve

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	81	00	Key Index [0,3]	00	Admin

Response

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

21.4.9 The InitCurve command

This command initializes the elliptic curve SECP256r1 (P256) parameters

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	01 P256	Key Index [0,3]	00	Admin

Response

Data	SW1	SW2	Comment
	90	00	Key Reset Done
	63	80	PIN required
	69	85	Bad index
	64	01	Public Key is defined
	64	02	Private Key is defined
	6A	86	Incorrect P1P2

21.4.10 The Generate KeyPair Command

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	82	00	Key Index [0,3]	00	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

21.4.11 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 [0,3] Ephemerous Key FF Cert Key FE	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 Forbidden for KeyIndex FE

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
	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

21.4.12 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 [0,3]	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

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

21.4.13 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 Raw Signature	Key Index [0,3] FE or 5 for Key Cert	Length 20	Data to be signed	Admin or User
00	80	21 Signature with SHA256	Key Index [0,3] FE or 5 for Key Cert	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

21.4.14 The Diffie Hellman Command

This command computes a DiffieHellman (DH) secret, and optionally generates an ephemerous pair of public/ private keys.
It requires no PIN.

Command

CLA	INS	P1	P2	P3	Data	PIN required
00	8A	01 P256 Curve	Key Index [0,3] FF for key generation	Length 41	Public Key Uncompress Format (04)	NO

Response

Body= Empty or 32 bytes DH	SW1	SW2	Comment
	90	00	No Error
	6D	50	DH Error

21.4.15 The GenerateRandom command

This command computes a set of random bytes.
It requires no PIN.

Command

CLA	INS	P1	P2	P3	PIN required
00	8B	00	00	Length	NO

Response

Body= P3 bytes or empty	SW1	SW2	Comment
	90	00	No Error
	69	85	Error

21.4.16 The HMAC Command

This command is used to compute keys used by TLS1.3
It requires Admin or User PIN.

Command

CLA	INS	P1	P2	P3	Data	PIN
00	85	00	02= Compute HMAC LengthKey [Key] LengthData [Data]	2+ LengthKey+ LengthData	32 bytes	Admin or User
		00 FF	0A= EXTRACT_EARLY P1=00 return ESK P1=FF return ESK, HSK, eBSK, rBSK, feBSK, frBSK	23	01 00 20 PSK (32 bytes)	Admin or User
		00 01	0B EXPAND_EARLY P1=0 "c e traffic" P1=1 "e exp master"	3+ DataLength	00 20 DataLength Data	Admin or User
		00	0C= HMAC_EBSK	Data Lentgh	Data	Admin or User
		00	0D=HMAC_RBSK	Data Length	Data	Admin or User
		00	0E=EXTRACT_HANDSHAKE (HMAC_HSK)	Data Length	Data (PubKey)	Admin or User

Response

Body	SW1	SW2	Comment
32 bytes 6x32 for = EXTRACT_EARLY with P1=FF	90	00	No Error
Empty	6A	86	Wrong P1 P2

21.4.17 The GetCertificate command

This command reads the card certificate.
It requires the Admin or the User PIN.
Upon success it returns the status word SW1, SW2 = 9000.
Otherwise it returns SW1=63, SW2=80 (PIN required).
The card certificate is the ECDSA SHA256 signature over the secp56k1 curve, of its public key (in full representation, i.e. 65 bytes) located at index 0. This certificate is the concatenation of two 32 bytes integers value r, s.

Command

CLA	INS	P1	P2	P3	PIN required
00	8E	00	00	00	Admin or User

Response

Body	SW1	SW2	Comment
64 bytes certificate	90	00	Two 32 bytes integers r,s
Empty	63	80	PIN required

21.4.18 The SetCertificate command

This command sets the card certificate.
It requires the Admin PIN.
Upon success it returns the status word SW1, SW2 = 9000.

Two working mode are supported. The certificate (64 bytes) is imported or the certificate is internally computed from an embedded key (auto signing)

Command

CLA	INS	P1	P2	P3	Payload	PIN required
00	8F	00	00	40	64 bytes certificate	Admin
00	8F	FF	KeyIndex	00	Empty	Admin

Response

SW1	SW2	Comment
90	00	Certificate loaded or generated
63	80	PIN required
69	84	Certificate Already Set
69	85	Wrong Certificate Length
64	03	Private Key is not set

21.4.19 The Command SEND

This command writes a TLS packet to the secure element, a returns a TLS message.
It requires no PIN.

Command

CLA	INS	P1	P2	P3	Data
00	D8	0 Normal Operation 1 if tls is open: Decrypt 2 if tls is open: Encrypt 3 if tls is open: Echo FF Rx Echo 1 in ClientHello PSK test 81 in ClientHelloPKI test	0:more 1: first 2: last 3: first&last	Data Length 00: Start/Reset TLS session	Data to be written

Response

Body	SW1	SW2	Comment
Optional message	90	00	OK
Optional message	90	01	TLS channel is opened
Optional message	90	02	TLS channel is closed
Optional message	61	length	More data (length bytes) to read
Optional message	9F	length	More data (length bytes) to read
Empty	6D	02	Write Error
Empty	6D	01	Read Error
Empty	6D	14	Decryption Error
Empty	6D	15	Encryption Error
Empty	6D	16	TLS Error

21.4.20 The RECV Command

This command reads a TLS packet
It requires no PIN.

Command

CLA	INS	P1	P2	P3
00	C0	0	0	Data Length

Response

Body	SW1	SW2	Comment
TLS Packet	90	00	No more date to read
TLS Packet	61	Length	More data (length bytes) to read
TLS Packet	9F	Length	More data (length bytes) to read

21.4.21 The TEST command

This command performs test operations.
It requires the Admin PIN.

Command

CLA	INS	P1	P2	P3	Data
00	C0	0	01 DH test	20	DiffieHellman
		0	02 Public Key test	41	PublicKey
		0	03 random test	20	Random value
		0	0A Server Name	length	ServerName
		0	FF Shell Call	length	Shell Command

Response

Body	SW1	SW2	Comment
Optional Body	90	00	OK
Empty	67	00	Wrong Length
Empty	6D	02	Error Write
Empty	63	80	PIN required
Empty	6A	86	Wrong P1 P2