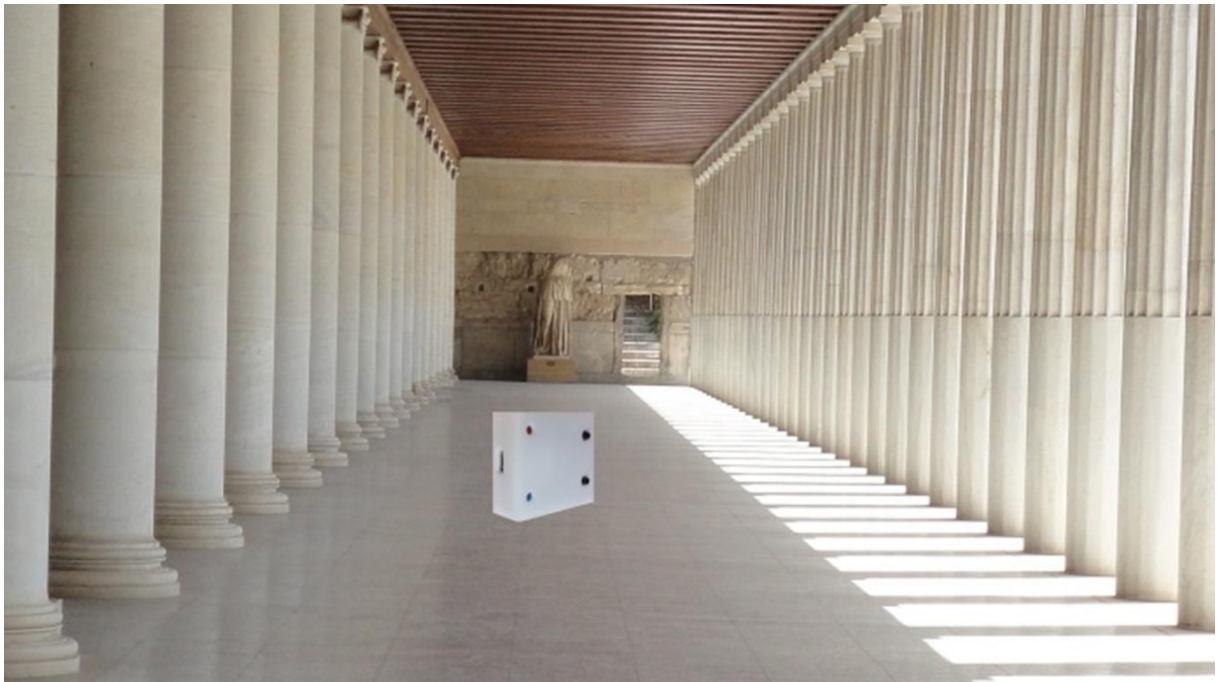


LeMonolith

A New Paradigm For Secure Element



LeMonolith Dev Kit (LEM) v0.2

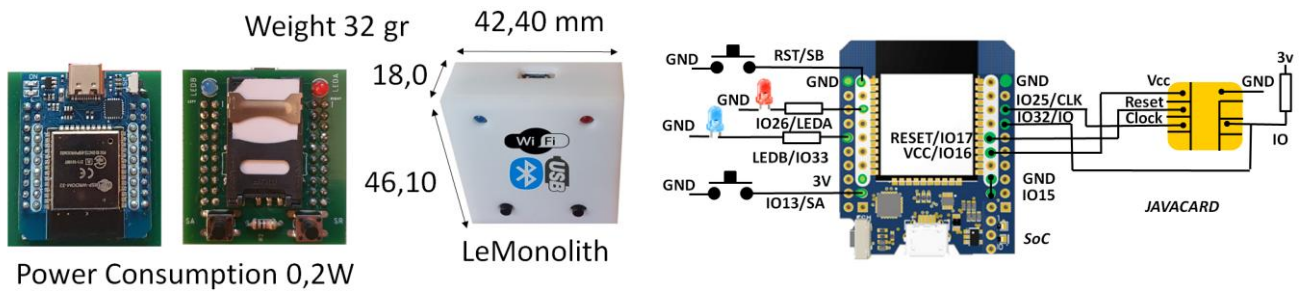
Table des matières

1 About LeMonolith	5
2 Initialization of the LEM DevKit	6
3 Loading software	6
3.1 ESP32 software loader.....	6
3.2 Secure Element Software Loader	7
3.2.1 Load Applications in SE.....	7
3.2.2 Load TLS-IM application in SE.....	7
3.2.2 List Applications stored in Secure Element	7
4 Working mode selection	8
4.1 Using push buttons.....	8
4.2 Using Serial Terminal.....	8
5 USB SHELL Commands.....	11
6 USB Bluetooth operations	11
7 Wi-Fi Operations.....	13
7.1 Example of OPENSSL command line.....	13
7.2 TLS-SE App commands	14
7.3 TLS-SE Application Certification Procedure (ACP).....	15
7.4 TLS-SE Session Authentication Procedure (SAP)	15
7.5 TLS-SE-IO commands	15
8 Bluetooth Operations.....	16
8.1 Serial Bluetooth terminal	17
8.2 Bluetooth CryptoToken App for Android	18
9 Bluetooth TLS-PSK (BTPSK).....	19
9.1 Testing Bluetooth TLS-PSK.....	19
10 LeMonolith (LEM) Dev Kit Tests	20
10.1 USB Operations	20
10.1.2 COM_List.bat.....	20
10.1.3 COM_Find.bat	20
10.1.4 TERM_hyperterminal.bat	20
10.1.5 TERM_terminal.bat	20
10.1.6 USB_GP_list.bat.....	20
10.1.7 USB_GP_delete.bat	20
10.1.8 USB_GP_install.bat.....	20

10.1.9 USB_KEYSTORE_Genkey00.bat	20
10.2 Wi-Fi Operations.....	21
10.2.1 SSL_openssl.bat	21
10.2.2 SSL_wolfssl.bat	21
10.2.3 SSL_wolfssl_MFA.bat.....	21
10.2.4 SSL_wolfssl_PCSC.bat	21
10.2.5 KEYSTORE_NET_Load_Key.bat	21
10.2.6 KEYSTORE_NET_Load_Key_SC.bat	21
10.2.7 KEYSTORE_NET_Load_Key_MFA.bat	21
10.2.8 KEYSTORE_NET_test_sign.bat	21
10.3 Wi-Fi Operations with TLS-IM	21
10.3.1 TLSIM_GP_USB_LOADER_IM.bat	21
10.3.2 TLSIM_GP_USB_DELETE_IM.bat	21
10.3.3 TLSIM_GP_USB_PERSO_IM.bat.....	22
10.3.4 TLSIM_LEM_Client_PSK_AESGCM_reader.bat.....	22
10.3.5 TLSIM_LEM_Client_PKI_AESGCM_echo.bat	22
10.3.6 TLSIM_LEM_Client_PSK_AESCCM_tlse.bat	22
10.3.7 TLSIM_SERVER_LOCAL_PSK_AESCCM.bat	22
10.3.8 TLSIM_SERVER_LEM_CONNECT_PCSC.bat	22
10.3.9 TLSIM_SERVER_LEM_CONNECT_SERIAL.bat.....	22
10.4 USB BLUETOOTH Tests	22
11 Secure Element Certification Procedure over Wi-Fi.....	22
11.1 Loading Authority Certification Key (CA).....	22
11.1.1 TLS-IM Smartcard	22
11.1.2 TLS-IM MFA Token.....	22
11.2 SE_NET_Cert_SOFT.bat	23
11.3 SE_NET_Cert_SC.bat.....	23
11.4 SE_NET_Cert_MFA.bat	23
12 Secure Element Authentication Session Procedure over Wi-Fi	23
12.1 SE_NET_auth_SOFT.bat.....	23
12.2 SE_NET_auth_SC.bat	23
12.3 SE_NET_auth_MFA.bat	23
13 IOSE Tests	23
13.1 IOSE_Server_WIN32.bat.....	23

13.2 IOSE_Server_Console.bat	23
13.3 IOSE_RACS_List.bat	24
13.4 IOSE_RACS_Console	24
13.5 IOSE_GP_list.bat.....	24
13.6 IOSE_GP_delete.....	24
13.7 IOSE_GP_install	24
13.8 IOSE_Openssl.bat	24
13.9 IOSE_KEYSTORE_test_sign.bat	24
13.10 IOSE_Cert_SOFT.bat	24
13.11 IOSE_Cert_SC.bat	24
13.12 IOSE_Cert_MFA.bat.....	24
13.13 IOSE_auth_SOFT.bat	24
13.14 IOSE_auth_SC.bat.....	24
13.15 IOSE_auth_MFA.bat	24
14 Ethereum Transactions over Wi-Fi.....	24
14.1 Ethereum transaction parameters	25
14.2 ETH_gasview.bat	25
14.3 ETH_NET_Make_Transaction.bat.....	25
14.4 ETH_Transaction_Send.bat	25
14.5 ETH_Transaction_View.bat	25
15 Software	25
15.1 Software components	25
15.2 How to build LeMonolith.....	26
16 Online technical resources	26
16.1 TLS for Secure Element, TLS-SE	26
16.2 TLS for secure element input output TLS-SE-IO	26
16.3 TLS identity module, TLS-IM.....	26
16.4 Remote APDU Server (RACS).....	26
16.5 Internet of Secure Element (IOSE).....	26

1 About LeMonolith



LeMonolith has two main components: an ESP32 D1 Mini board and a javacard. Its main goal is to demonstrate security nano-servers based on secure elements. An introduction to online TLS-SE secure element is available on youtube 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).

APP	APP	APP	APP	GPSHELL	Terminal	LeMonolith	
TLS	APDU	CMD	TLS	APDU	CMD	USB	
PSK		SHELL	Client	winscard.dll	SHELL	winscard.dll	
RFCOMM			TCP/IP	SERIAL		RACS	TLS
Bluetooth			Wi-Fi	USB		TCP/IP (IoSE)	
LeMonolith							

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 USB interface
- Smartcard use through the TLS for Secure Element (TLS-SE) Wi-Fi interface
- Smartcard as TLS identity module (TLS-IM) for Wi-Fi TLS server
- Smartcard use from Bluetooth interface, including mobile application for Android and TLS-SE services over Bluetooth

2 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 loSE server, comment the line:

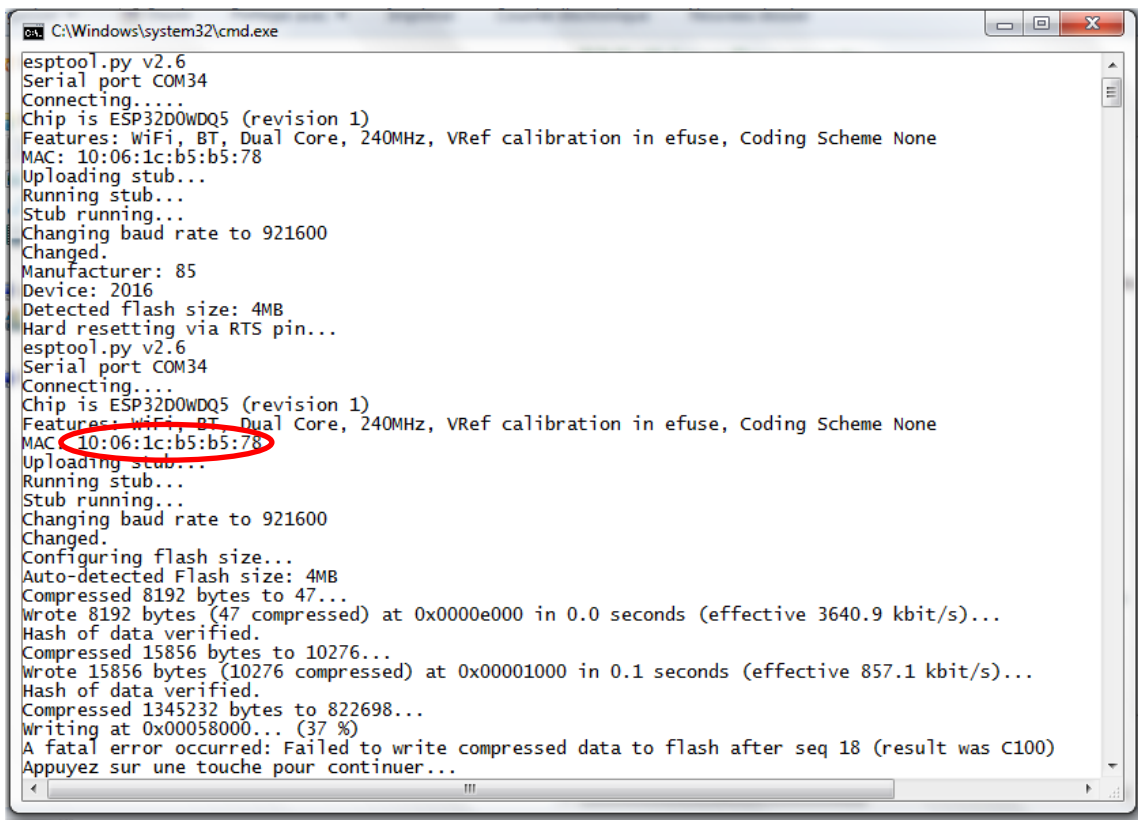
```
REM set MYIP=192.168.1.35
```

Execute MAKE.bat...the USB serial port in which is plugged LeMonolith is automatically detected.

3 Loading software

3.1 ESP32 software loader

Goto /ESP32loader, Execute loader2.bat for version 2 (loader.bat for version 1)



```
C:\Windows\system32\cmd.exe
esptool.py v2.6
Serial port COM34
Connecting....
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
MAC: 10:06:1c:b5:b5:78
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Manufacturer: 85
Device: 2016
Detected flash size: 4MB
Hard resetting via RTS pin...
esptool.py v2.6
Serial port COM34
Connecting....
Chip is ESP32D0WDQ5 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
MAC: 10:06:1c:b5:b5:78
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 8192 bytes to 47...
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (effective 3640.9 kbit/s)...
Hash of data verified.
Compressed 15856 bytes to 10276...
Wrote 15856 bytes (10276 compressed) at 0x00001000 in 0.1 seconds (effective 857.1 kbit/s)...
Hash of data verified.
Compressed 1345232 bytes to 822698...
writing at 0x00058000... (37 %)
A fatal error occurred: Failed to write compressed data to flash after seq 18 (result was C100)
Appuyez sur une touche pour continuer...
```

In this example the Wi-Fi Mac address is 10:06:1C:B5:B5:78

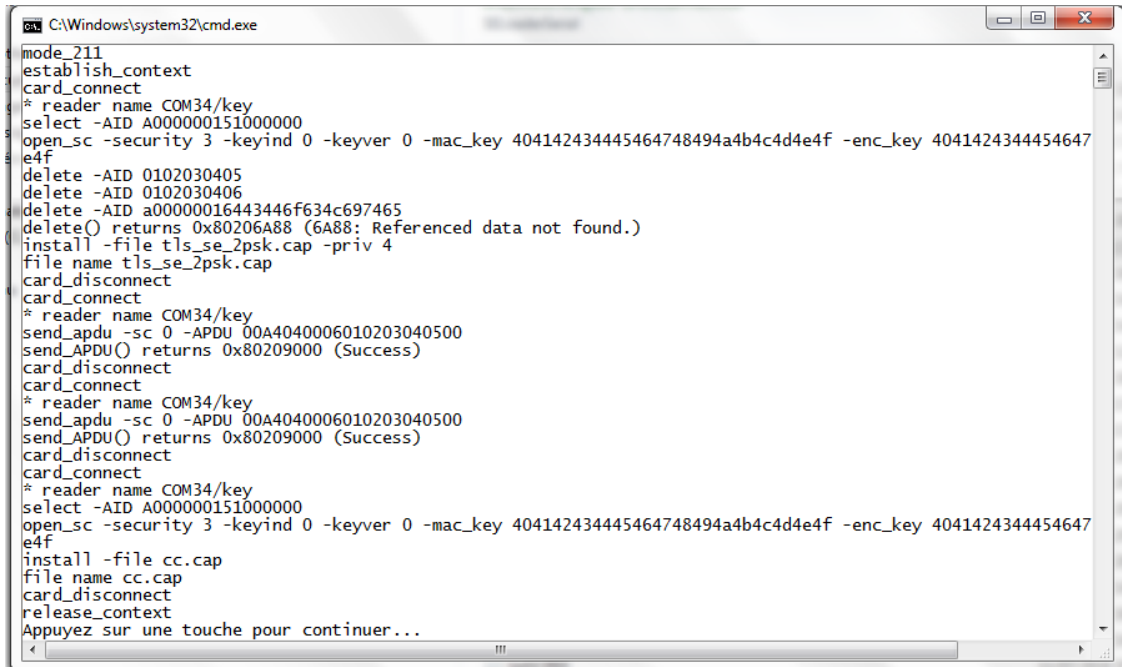
The Bluetooth address is obtained by adding 2 to the Wi-Fi address (10:06:1C:B5:B5:7A)

3.2 Secure Element Software Loader

Go to /

3.2.1 Load Applications in SE

Execute USB_GP_install.bat, which installs tls_se and cc.



```
C:\Windows\system32\cmd.exe
mode_211
establish_context
card_connect
* reader name COM34/key
select -AID A000000151000000
open_sc -security 3 -keyind 0 -keyver 0 -mac_key 404142434445464748494a4b4c4d4e4f -enc_key 4041424344454647
e4f
delete -AID 0102030405
delete -AID 0102030406
delete -AID a00000016443446f634c697465
delete() returns 0x80206A88 (6A88: Referenced data not found.)
install -file tls_se_2psk.cap -priv 4
file name tls_se_2psk.cap
card_disconnect
card_connect
* reader name COM34/key
send_apdu -sc 0 -APDU 00A4040006010203040500
send_APDU() returns 0x80209000 (Success)
card_disconnect
card_connect
* reader name COM34/key
send_apdu -sc 0 -APDU 00A4040006010203040500
send_APDU() returns 0x80209000 (Success)
card_disconnect
card_connect
* reader name COM34/key
select -AID A000000151000000
open_sc -security 3 -keyind 0 -keyver 0 -mac_key 404142434445464748494a4b4c4d4e4f -enc_key 4041424344454647
e4f
install -file cc.cap
file name cc.cap
card_disconnect
release_context
Appuyez sur une touche pour continuer...
```

3.2.2 Load TLS-IM application in SE

Execute TLSIM_GP_USB_LOADER_IM.bat

To delete the application execute TLSIM_GP_USB_DELETE_IM.bat

To download PSK key and demonstration CERTIFIC execute TLSIM_GP_USB_PERSO_IM.bat

3.2.2 List Applications stored in Secure Element

Execute List_USB_GP_list.bat



```
C:\Windows\system32\cmd.exe
mode_211
establish_context
card_connect
* reader name COM85/key
select -AID A000000151000000
open_sc -security 3 -keyind 0 -keyver 0 -mac_key 404142434445464748494a4b4c4d4e4f -enc_key 404142434445464748494a4b4c4d4
e4f
get_status -element 40 -noStop
AID | State | Privileges | Version | Linked Security Domain
-- | -- | -- | -- | --
010203040500 TLS-SE | Selectable | | 0000 | a000000151000000
| | Default Selected / Card Reset | |
-- | -- | -- | -- | --
010203040601 CC | Selectable | | 0000 | a000000151000000
-- | -- | -- | -- | --
010203040700 TLS-IM | Selectable | | 0000 | a000000151000000
card_disconnect
release_context
```

4 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
- Bluetooth, for application with smartphone

There are two ways to select the working mode:

- By using the two push buttons
- By using a serial terminal

4.1 Using push buttons



- Hold ACK button
- Press shortly RESET button
- Release ACK button when 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

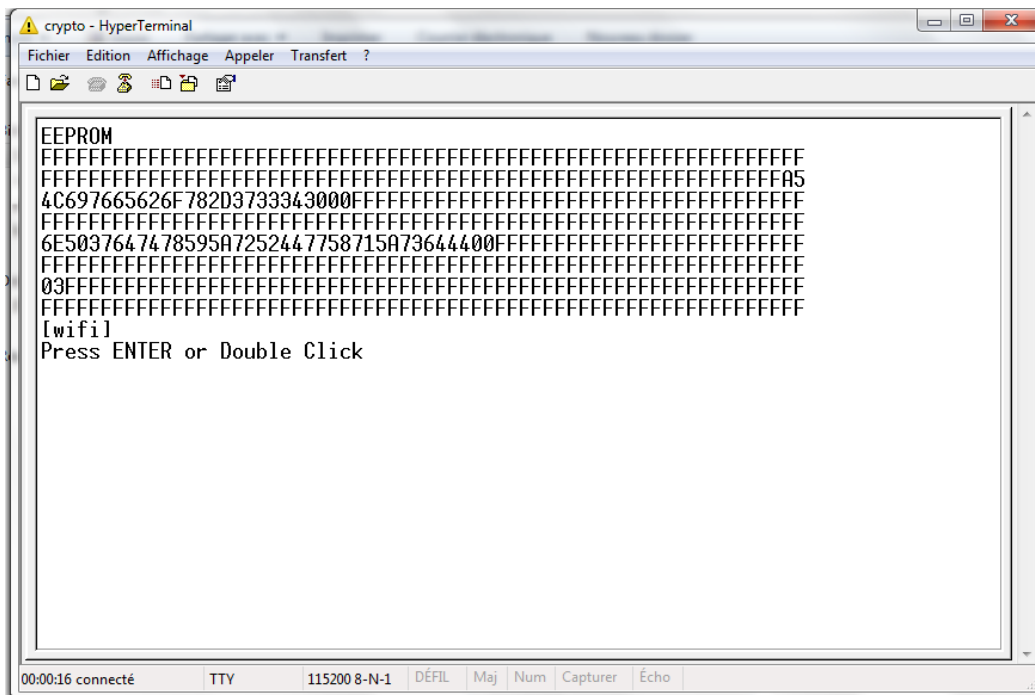
Mode	Blue LED	Red LED
USB	On	On
Bluetooth	On	Off
Wi-Fi	Off	On
Bluetooth USB	Blinking	Blinking
Bluetooth TLS-PSK	Blinking	Off
Wi-Fi + TLS-IM	Off	Blinking

4.2 Using Serial Terminal

The serial baudrate is 115200, with 8 bits, and no parity

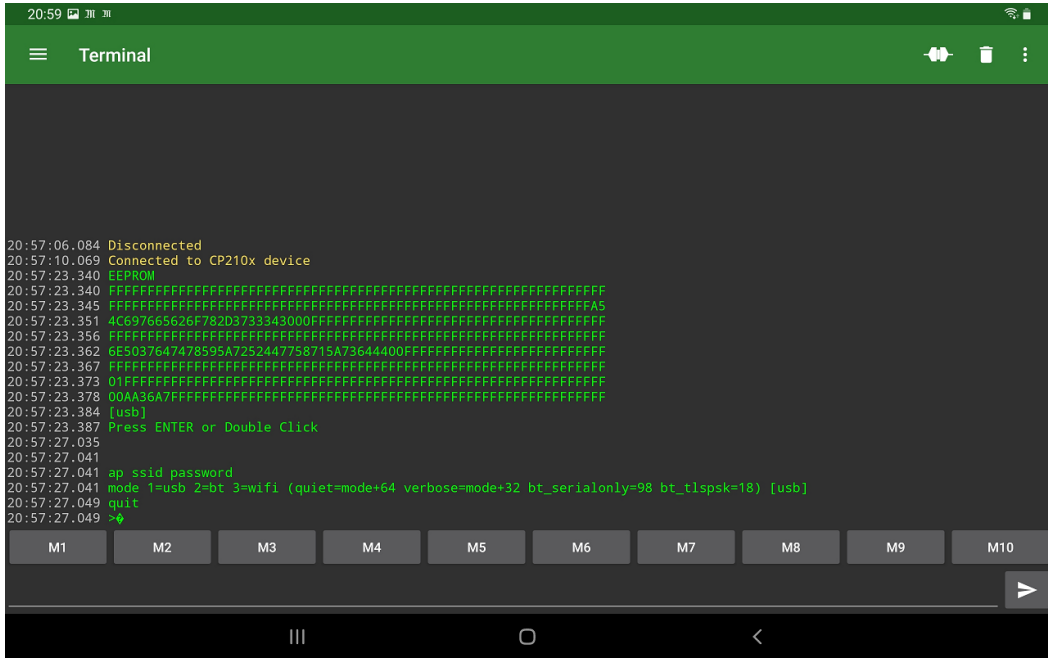
LeMonolith SDK provides the old version of windows HYPERTERMINAL and a dedicated terminal

Execute TERM_terminal.bat (for dedicated terminal) OR TERM_hypercenterminal.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 RESET button
- Release RESET button when blue LED is blinking

The following lines are displayed:

```
EEPROM
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
737369730808696400FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
706173776F72640808080873776F726400FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
[usb]
Press ENTER or Double Click
```

In the above example the 4th line is the Wi-Fi SSID and 6th line is the password, the first byte of the 8th line is the working mode (01=usb)

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

```
ap ssid password
mode 1=usb 2=bt 3=wifi (quiet=mode+64 verbose=mode+32 bt_serialonly=98
bt_tlpspk=18 tlsim=51) [usb]
quit
>
```

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
>
```

To select a mode type : mode number, and press ENTER

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

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 USB SHELL Commands

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	Un-power 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

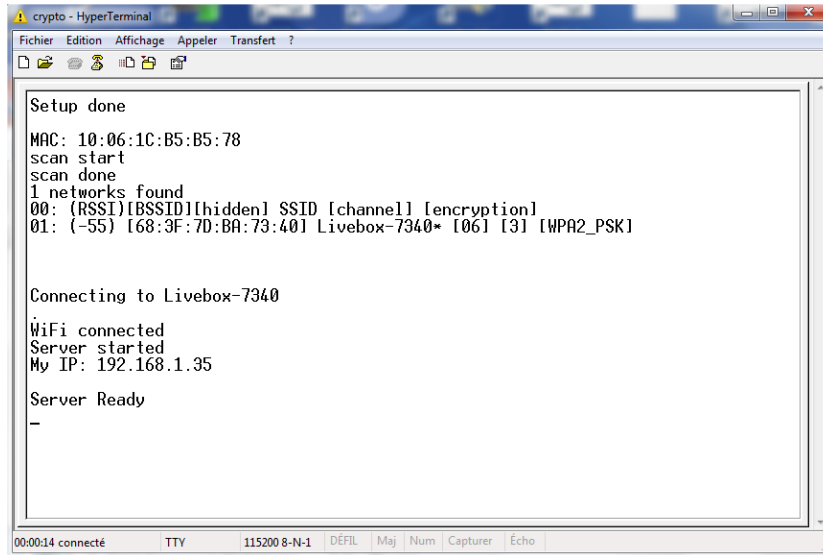
6 USB Bluetooth operations

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 ₂i _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)

7 Wi-Fi Operations



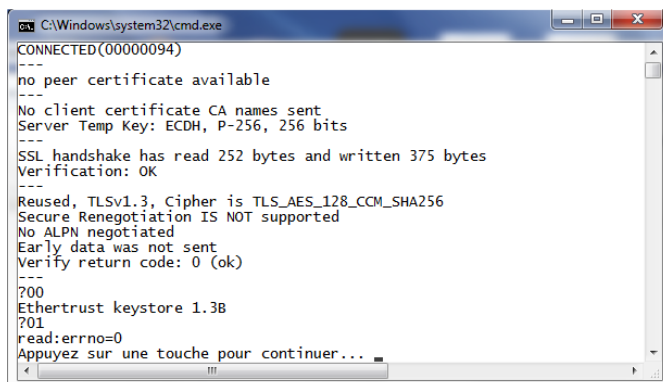
```
crypto - HyperTerminal
Fichier Edition Affichage Appeler Transfert ?
Setup done
MAC: 10:06:1C:B5:B5:78
scan start
scan done
1 networks found
00: (RSSI)[BSSID][hidden] SSID [channel] [encryption]
01: (-55) [68:3F:7D:BA:73:40] Livebox-7340* [06] [3] [WPA2_PSK]

Connecting to Livebox-7340
WiFi connected
Server started
My IP: 192.168.1.35
Server Ready
-
```

- 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.

7.1 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
```



```
C:\Windows\system32\cmd.exe
CONNECTED(00000094)
---
no peer certificate available
---
No client certificate CA names sent
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 252 bytes and written 375 bytes
Verification: OK
---
Reused, TLSv1.3, Cipher is TLS_AES_128_CCM_SHA256
Secure Renegotiation IS NOT supported
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
---
?00
Ethertrust keystore 1.3B
?01
read:errno=0
Appuyez sur une touche pour continuer...
```

7.2 TLS-SE App commands

Command	Comment
?00	Version
?01	Disconnect
?01text	Echo text
?0A	Get 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
?A1	Get PSK2
?AA[OldPSK,NewPSK]	Set PSK OldPSK=64HexaDigit NewPSK=64HexaDigits
?FF[hexa digits]	Echo(Hexadecimal value)
cxy Cxy	Clear key index=xy hexadecimal
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
rxxy	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
lxy	Read record number xy hexadecimal

7.3 TLS-SE Application Certification Procedure (ACP)

TLS-SE *Application Certification Procedure* creates a pair of public/private key upon instantiation. The certification procedure reads the public key and writes a certificate (ECDSA signature of public key).

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

7.4 TLS-SE Session Authentication Procedure (SAP)

The *Session Authentication Procedure* makes the proof that remote node knows the TLS-SE private key 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 ECDSASEPrivKey(SHA2(HandshakeSecret || Random))
?0C7F69B857C6C2675BC8D5238E3E8BFC4C633FB5E39DD07F4760F508084FD1B482
>>304402206D2E688731C2673F977BD49B37D6CEC966323E966E34DE426D424AC5506F4A4B0
2203F5462E3D0AA7A1ED410ADDB29AE7C980EFC0136028FDF8533843D6A3C854ABF
```

7.5 TLS-SE-IO commands

Command	Comment
#on	Blue LED on
#off	Blue LED off
#on\$[decimal value]	LED on value=0=>blue, value=1=>red
#off\$[decimal value]	LED 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)

8 Bluetooth Operations

APP	APP	APP	APP	GPSHELL	Terminal	LeMonolith	
TLS PSK	APDU	CMD SHELL	TLS Client	APDU	CMD SHELL	USB	
				winscard.dll		winscard.dll	
RFCOMM			TCP/IP	SERIAL		RACS	TLS
Bluetooth			Wi-Fi	USB		TCP/IP (IoSE)	
LeMonolith							

Command	Comments
Empty	Return "ERROR No Command!"
echo	Return "OK"
user	Only for Bluetooth Serial
eth keyindex.PIN	Only for Bluetooth serial Ethereum address (20 bytes) associated to keyindex PIN user 4 decimal digis 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	Power smartcard, select CC-SE-APP
on2	Power smartcard
off	Unpower smartcard
A [hexadecimal digits]	Only for USB debug. Send APDU
prompt	Prompt (>) is displayed

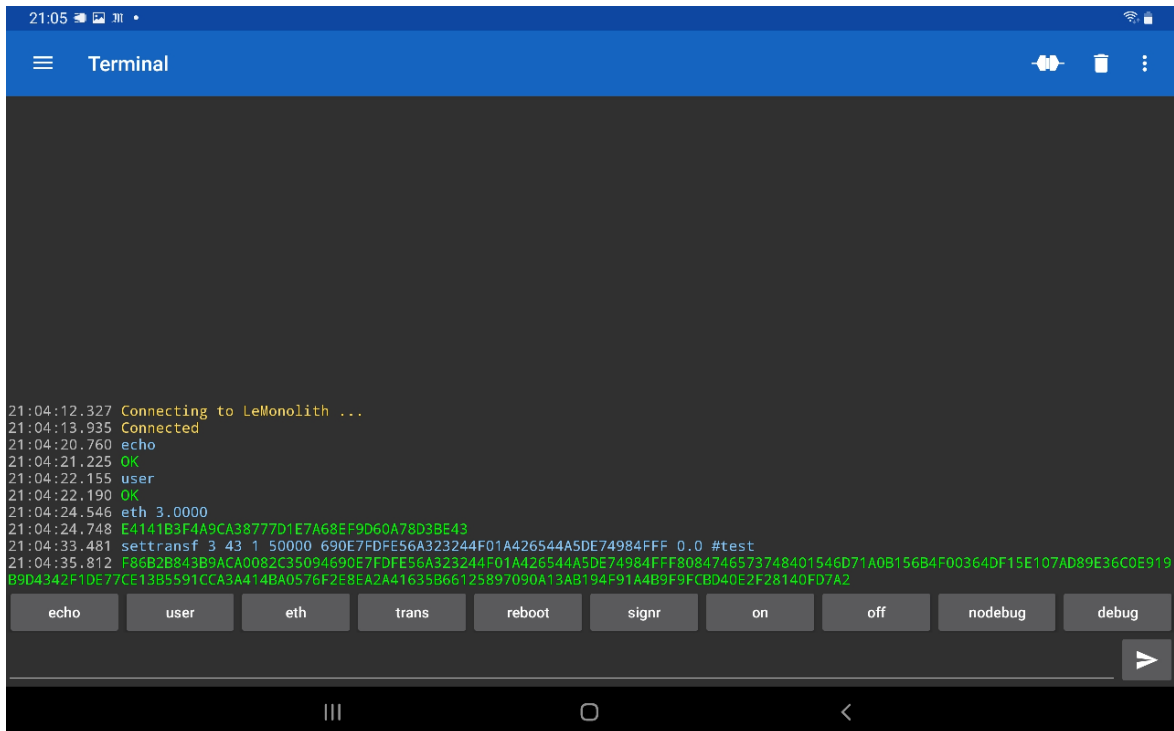
noprompt	Prompt (>) is not displayed
user PIN	Only for USB debug Start smartcard, select CC-SE App, and present user PIN code (four decimal digits, default 0000)
changeuser oldpin newpin	Modify user PIN(4 decimal digits)
changeuser2 oldpin newpin	Modify user2 PIN(4 decimal digits)
changeadm oldpin newpin	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-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 ₂i _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

8.1 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

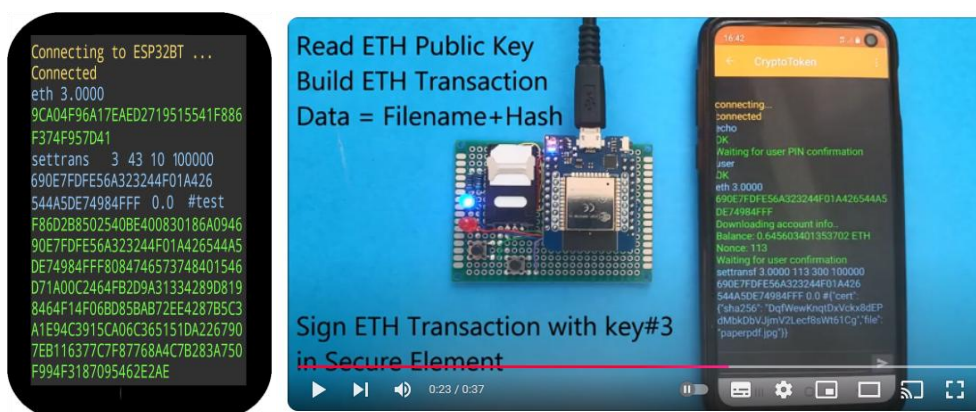
LeMonolith is compatible with the Serial Bluetooth Terminal application, with baudrate=9600, end of line CR LF, and local echo



8.2 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



9 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 LED blinking: smartcard access

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

```
>>> ??? [length 0005]
17 03 03 00 35
>>> TLS 1.3 [length 0001]
16
>>> TLS 1.3, Handshake [length 0024], Finished
14 00 00 20 b9 37 09 5c 65 52 1f 34 be 68 e
72 79 3c d6 c4 85 40 a0 0f 95 89 a9 d7 45 d
7c 15 10 17
write to 0x27ca80 [0x28b2a0] (64 bytes => 64 (0
0000 - 14 03 03 00 01 01 17 03-03 00 35 9f 94 0
0010 - a9 04 d9 a3 5f 18 1b 5f-b1 44 20 3b 42 9
0020 - 3e 65 65 64 47 60 72 61-Dc 63 28 90 84 6
0030 - fe 45 6f 19 d0 56 50 48-c9 a7 2c 46 98 d
---
no peer certificate available
---
No client certificate CA names sent
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 252 bytes and written 37
Verification: OK
---
Reused, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA2
Secure Renegotiation IS NOT supported
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
---
700
>>> ??? [length 0005]
17 03 03 00 16
>>> TLS 1.3 [length 0001]
17
write to 0x27ca80 [0x28618b] (27 bytes => 27 (0
0000 - 17 03 03 00 16 f4 63 d4-d6 04 af a3 81 6
0010 - c4 6d 1d a1 4d 33 b7 c2-e9 02 d1
read from 0x27ca80 [0x28203b] (5 bytes => 5 (0x
0000 - 17 03 03 00 2b
<<< ??? [length 0005]
17 03 03 00 2b
read from 0x27ca80 [0x282040] (43 bytes => 43 (
0000 - 19 70 b5 fa 67 d6 cb 9a-3b 4a 10 7a 3b 9
0010 - dd 1d 6c 42 b4 1b 48 ab-5f 22 63 d4 f2 a
0020 - 46 99 67 d9 4e 04 61 63-f9 a5 9f
<<< TLS 1.3 [length 0001]
17
Ethertrust keystore 1.38

Server TLS1.3 Ready...
Connexion from 127.0.0.1
Rx Thread Ready
311 bytes received on NetRecv
311 bytes sent on Serial
Rx Header (161)
166 bytes received on serial
Rx Header (23)
28 bytes received on serial
Rx Header (53)
58 bytes received on serial
252 bytes NetSend
6 bytes received on NetRecv
6 bytes sent on Serial
58 bytes received on NetRecv
58 bytes sent on Serial
27 bytes received on NetRecv
27 bytes sent on Serial
Rx Header (43)
48 bytes received on serial
48 bytes NetSend

File Edit Setup Control Window Help
Recv: 5
Recv: (53) 58
RxNET_BT
17030300359F9403F77FA904D9A35F181B5FB144203B4299E2743E6565644760
72610C6328908467102AFE456F19D0565048C9A72C4698DEBC59
Tx: 00D800033A17030300359F9403F77FA9
04D9A35F181B5FB144203B4299E2743E
656564476072610C6328908467102AFE
456F19D0565048C9A72C4698DEBC59
TxTTL: 4
00403FDF
RxTTL: 4
004002D3
Rx [217ms]:
9001
TLS OPEN
Recv: 5
Recv: (22) 27
RxNET_BT
1703030016F463D4D604AFA381623FA1C46D1DA14D33B7C2E902D1
Tx: 00D800031B1703030016F463D4D604AF
A381623FA1C46D1DA14D33B7C2E902D1
TxTTL: 4
00002025
RxTTL: 4
00003291
Rx [82ms]:
170303002B1970B5FA67D6CB9A3B4A10
7A3B984EE3DD1D6C42B41B48AB5F2263
D4F2A6747E469967D94E046163F9A59F
9000
TxNET_BT
170303002B1970B5FA67D6CB9A3B4A107A3B984EE3DD1D6C42B41B48AB5F2263
D4F2A6747E469967D94E046163F9A59F
Sent: 48
Recv: []
```

A video is available on youtube see <https://www.youtube.com/watch?v=Wyl4OxVTzHM>

10 LeMonolith (LEM) Dev Kit Tests

10.1 USB Operations

Select the USB mode.

APP	APP	APP	APP	GPShell	Terminal	LeMonolith	
TLS PSK	APDU	CMD SHELL	TLS Client	APDU	CMD SHELL	USB	
				winscard.dll		winscard.dll	
RFCOMM		TCP/IP	SERIAL			RACS	TLS
Bluetooth		Wi-Fi	USB			TCP/IP (IoSE)	
LeMonolith							

10.1.2 COM_List.bat

List COM port.

10.1.3 COM_Find.bat

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

10.1.4 TERM_hypterterminal.bat

Start hyperterminal.

10.1.5 TERM_terminal.bat

Start terminal.

10.1.6 USB_GP_list.bat

List javacard applications stored in the secure element.

10.1.7 USB_GP_delete.bat

Delete all javacard applications stored in the secure element.

10.1.8 USB_GP_install.bat

Install and tls_se_2psk.cap (TLS-SE App) and cc.cap (Crypto Currency App) in the secure element

10.1.9 USB_KEYSTORE_Genkey00.bat

Create a key at index 0 in TLS-SE App.

10.2 Wi-Fi Operations

Select the Wi-Fi mode.

APP	APP	APP	APP	GPShell	Terminal	LeMonolith	
TLS PSK	APDU	CMD SHELL	TLS Client	APDU	CMD SHELL	USB	
				winscard.dll		winscard.dll	
RFCOMM			TCP/IP	SERIAL		RACS	TLS
Bluetooth			Wi-Fi	USB		TCP/IP (IoSE)	
LeMonolith							

10.2.1 SSL_openssl.bat

Open a TLS session with OPENSLL.

10.2.2 SSL_wolfssl.bat

Open a TLS session with WOLFSSL.

10.2.3 SSL_wolfssl_MFA.bat

Open a TLS session with Multi Form Authentication (MFA) TLS-IM token

10.2.4 SSL_wolfssl_PCSC.bat

Open a TLS session with TLS-IM smartcard.

10.2.5 KEYSTORE_NET_Load_Key.bat

Load a key from \keystore\eth\mypp.txt at index 3, using PSK=\keystore\eth\mysk.txt

10.2.6 KEYSTORE_NET_Load_Key_SC.bat

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

10.2.7 KEYSTORE_NET_Load_Key_MFA.bat

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

10.2.8 KEYSTORE_NET_test_sign.bat

Perform ECDSA signatures with KEY at index 3.

10.3 Wi-Fi Operations with TLS-IM

10.3.1 TLSIM_GP_USB_LOADER_IM.bat

Load TLS-IM software in the secure element, under USB mode

10.3.2 TLSIM_GP_USB_DELETE_IM.bat

Delete TLS-IM software in the secure element, under USB mode

10.3.3 TLSIM_GP_USB_PERSO_IM.bat

Load (under USB mode) PSK key in the secure element and a certificate used for echo demonstration

10.3.4 TLSIM_LEM_Client_PSK_AESGCM_reader.bat

Connect to LeMonolith under Wi-Fi TLS-IM mode, with the ciphersuite AESGCM and pre-shared-key (PSK). The USB SHELL commands (detailed in section 5) are available.

10.3.5 TLSIM_LEM_Client_PKI_AESGCM_echo.bat

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

10.3.6 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 7.2 are available.

10.3.7 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.

10.3.8 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 PC/SC emulation)

10.3.9 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).

10.4 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.

11 Secure Element Certification Procedure over Wi-Fi

Select the Wi-Fi mode.

11.1 Loading Authority Certification Key (CA)

11.1.1 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

11.1.2 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.

11.2 SE_NET_Cert_SOFT.bat

This script generate a certificate for LeMonolith, with software credentials

11.3 SE_NET_Cert_SC.bat

This script generate a certificate for LeMonolith, with TLS-IM smartcard

11.4 SE_NET_Cert_MFA.bat

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

12 Secure Element Authentication Session Procedure over Wi-Fi

Select the Wi-Fi mode.

12.1 SE_NET_auth_SOFT.bat

This script opens an authenticated session with LeMonolith.

12.2 SE_NET_auth_SC.bat

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

12.3 SE_NET_auth_MFA.bat

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

13 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 a TLS-SE TLS1.3 PSK server, with the server name COMX001.

APP	APP	APP	APP	GPSHELL	Terminal	LeMonolith	
TLS PSK	APDU	CMD SHELL	TLS Client	APDU	CMD SHELL	USB	
				winscard.dll		winscard.dll	
RFCOMM			TCP/IP	SERIAL		RACS	TLS
Bluetooth			Wi-Fi	USB		TCP/IP (IoSE)	
LeMonolith							

13.1 IOSE_Server_WIN32.bat

This script starts the IoSE server for windows.

13.2 IOSE_Server_Console.bat

This script starts the IoSE server in console mode.

13.3 IOSE_RACS_List.bat

This script list the secure elements plugged to the IOSE server. The monolith has two SEIDs 0 and 999 (default)

13.4 IOSE_RACS_Console

This script starts a RACS console

13.5 IOSE_GP_list.bat

This scripts lists application stored in the javacard.

13.6 IOSE_GP_delete

This script deletes all applications stored in the javacard.

13.7 IOSE_GP_install

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

13.8 IOSE_Openssl.bat

This script opens a TLS-PSK session with OPENSSL (127.0.0.1:8888)

13.9 IOSE_KEYSTORE_test_sign.bat

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

13.10 IOSE_Cert_SOFT.bat

This script generates a certificate for SE with software credentials

13.11 IOSE_Cert_SC.bat

This script generates a certificate for SE with a TLS-IM smartcard

13.12 IOSE_Cert_MFA.bat

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

13.13 IOSE_auth_SOFT.bat

This script opens an authenticated session with LeMonolith

13.14 IOSE_auth_SC.bat

This script opens an authenticated session with LeMonolith and requires a TLS-IM smartcard

13.15 IOSE_auth_MFA.bat

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

14 Ethereum Transactions over Wi-Fi

Select the Wi-Fi mode.

To understand Ethereum API and get a free token visit: <https://etherscan.io/apis>

14.1 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"
```

14.2 ETH_gasview.bat

This script starts the URL <https://sepolia.beaconcha.in/gasnow>, which gives SEPOLIA GAS price

14.3 ETH_NET_Make_Transaction.bat

This script makes a SEPOLIA transaction.

14.4 ETH_Transaction_Send.bat

This script sends a SEPOLIA transaction.

14.5 ETH_Transaction_View.bat

The script shows the last SEPOLIA transaction.

15 Software

15.1 Software components

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

Arduino 1.8.9 IDE

Select the board: WEMOS D1 MINI ESP32

Sketch: monolith.ino

Dedicated Libraries: Sclib5c, Cryptoecc, ripemd160, sha256, btools

Imported Arduino Libraries: crypto, BigNumber

Arduino Standard Library: WiFi, EEPROM

15.2 How to build LeMonolith

Copy libraries : ScLib5c, Cryptoecc, ripemd160, sha256, btools, crypto, BigNumber in the Arduino library repertory.

Compile lemonolith.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 lemonolith.ino again, no error should be notified.

16 Online technical resources

16.1 TLS for Secure Element, TLS-SE

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

16.2 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>

16.3 TLS identity module, TLS-IM

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

16.4 Remote APDU Server (RACS)

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

16.5 Internet of Secure Element (IOSE)

IETF draft Internet of Secure Elements, <https://datatracker.ietf.org/doc/html/draft-urien-coinrg-iose-08>