TLS Working Group P. Urien
Internet Draft Telecom Paris
Intended status: Experimental

March 27 2022
Expires: September 2022

Secure Element for TLS Version 1.3
draft-urien-tls-se-04.txt

Abstract

This draft presents ISO7816 interface for TLS1.3 stack running in secure element. It presents supported cipher suites and key exchange modes, and describes embedded software architecture. TLS 1.3 is the de facto security stack for emerging Internet of Things (IoT) devices. Some of them are constraint nodes, with limited computing resources. Furthermore cheap System on Chip (SoC) components usually provide tamper resistant features, so private or pre shared keys are exposed to hacking. According to the technology state of art, some ISO7816 secure elements are able to process TLS 1.3, but with a limited set of cipher suites. There are two benefits for TLS-SE; first fully tamper resistant processing of TLS protocol, which increases the security level insurance; second embedded software component ready for use, which relieves the software of the burden of cryptographic libraries and associated attacks. TLS-SE devices may also embed standalone applications, which are accessed via internet node, using a routing procedure based on SNI extension.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Status of this Memo

Copyright Notice

Table of Contents

1 Overview

This draft presents ISO7816 interface for TLS1.3 stack running in secure element (see Figure 1), it presents supported cipher suites and key exchange modes, and describes embedded software architecture. TLS 1.3 [RFC8446] is the de facto security stack for emerging Internet of Things (IoT) devices. Some of them are constraint nodes, with limited computing resources. Furthermore cheap System on Chip (SOC) components don't usually provide tamper resistant features, so private or pre shared keys are exposed to hacking. The identity module (im) detailed in [IM] protects identity credentials. The TLS identity module [IM] MAY be based on secure element [ISO7816]. According to the technology state of art, some secure elements are able to process TLS 1.3, but with a limited set of cipher suites. There are two benefits for TLS-SE; first fully tamper resistant processing of TLS protocol, which increases the security level insurance; second embedded software component ready for use, which relieves the software of the burden of cryptographic libraries and associated attacks.
Multiple TLS-SE devices, embedding standalone applications, can be hosted by an internet node. In this case SNI extension [RFC6066] MAY be used in order to select the right secure element (see Figure 2).

```
           +-----------+ recv +------------+ RECV +-----------+
           |     IP    + ---> |   TCP/IP   | ---> |  TLS 1.3  |
           |  Network  +------+ Constraint +------+  Secure   |
           |           | <--- |    Node    | <--- |  Element  |
           +-----------+ send +------------+ SEND +-----------+
                          |                   |
                   Network Interface    ISO7816 interface
```

Figure 1. TLS 1.3 Secure Element (TLS-SE)

```
                                                  +----------+
                                                  |  TLS-SE  |
                                  TLS-SE Name +---+  Secure  |
                                              |   |  Element |
                                              |   +----------+
                             SNI= TLS-SE Name |
           +-----------+      +------------+  |   +----------+
           |     IP    +      |   TCP/IP   |  |   |  TLS-SE  |
           |  Network  +------+    Node    +--+---+  Secure  |
           |           |      |SN Extension|  |   |  Element |
           +-----------+      +------------+  |   +----------+
                                              |
                                              |   +----------+
                                              |   |  TLS-SE  |
                                  TLS-SE Name +---+  Secure  |
                                                  |  Element |
                                                  +----------+
```

Figure 2. Routing procedure based on SNI for TLS-SE devices

2 About Secure Elements

Secure elements are defined according to [ISO7816] standards. They support hash functions (sha256, sha384, sha512) and associated HMAC procedures. They also provide signatures and DH procedures in Z/pZ* groups, or elliptic curves (for example secp256r1). Open software can be released thanks to JavaCard (JC) standards, such as JC3.04, or JC3.05. Most of secure elements use 8 bits Micro Controller Unit (MCU) and embedded cryptographic accelerator. Non volatile memory size is up to 100KB, and RAM size is up to 10KB.

Below is an illustration of binary encoding rules for secure element according to the T=0 ISO7816 protocol.
An ISO7816 request is a set of bytes comprising a five byte header and an optional payload (up to 255 bytes)
The header comprises the following five bytes:

- CLA, Class
- INS, Instruction code
- P1, P1 byte
- P2, P2 byte
- P3, length of the optional payload, or number of expected bytes

The response comprises an optional payload (up to 256 bytes) and a two bytes status word (SW1, SW2), SW1=90, SW2=00 (SW=9000) meaning successful operation.

The ISO7816 defines two main classes for data exchange (called transport protocol), T=0, and T=1.
The T=0 transport protocol is a byte stream; a payload can be included in request or response, but not in both.
The T=1 transport protocol is a frame stream; payload can be included both in request and response.

3 Software components for TLS-SE

```
                       +--------+
                       | Crypto +----------------+
   RECV +-----------+  |   Lib  |                |
   ---> |   Data    |  +--------+     +----------+----------+
   <--- | Exchange  |       +---------+       TLS Lib       |
   SEND +----+------+       |         | Make  Record Packet |
             |        +-----+-----+   | Check Record Packet |
             |        | TLS State |   +---------------------+
             +--------+  Machine  |
                      +-----------+
```

Figure 3. Software Components for TLS-SE

3.1 Cryptographic resources

Many secure elements support hash functions sha256, sha384 and sha512 used by TLS1.3. Associated HMAC, HKDF-Extract and Derive-Secret, MUST be implemented by a dedicated cryptographic library.

Many secure elements support the secp256r1 elliptic curve. Diffie-Hellman (DH)calculation are performed according to [IEEE1363] using the ECKAS-DH1 scheme with the identity map as the key derivation function, (KDF), so that the shared secret is the x-coordinate of the ECDH shared secret elliptic curve point represented as an octet string. ECDSA signature is also available for 256,384 and 512 hash size.

AES-128 is usually implemented, by not AES-CCM. So this AEAD algorithm SHOULD be implemented by a dedicated cryptographic library.

In summary, according to the state of art TLS-SE supports the secp256r1 EC group, associated ECDSA signature computing and checking, and EC-DHE key establishment. It also implements the AES-128-CCM-SHA256 cipher suite.

3.2 Data exchange

TLS record layer packets are received and sent from/to TCP/IP network thanks to well known socket procedures. TLS-SE processes these packets according to a dedicated state machine.

3.2.1 Receiving Record Packet

Dedicated ISO7816 requests (named RECV) push incoming record messages in secure element. A fragmentation mechanism splits the record packet in one a several ISO7816 requests, whose payload size is less than 255 bytes. A 2 bits fragmentation-flag field indicates the fragment status; bit F-First notifies the first fragment, and bit F-Last notifies the last fragment.

The ISO7816 RECV request COULD be encoded as
CLA=00, INS=D8, P1=0, P2=fragmentation-flag, P3=fragment-length
F-First=b01, F-Last=b10, F-More=b00

When application AEAD is opened a two bits flag (F-Encrypt, F-Decrypt) indicates the cryptographic operation:
- P2=b01= F-Decrypt, decryption
- P2=b10= F-Encrypt, encryption
- P2=b00= Standalone embedded application.

If F-Last is not set, the ISO7816 response is always 9000 when no error occurs. For the last fragment five cases may occur:
- sw-ok: no error, no record message returned, response = 9000.
- sw-open, no error, no record message returned, TLS application AEAD is opened, for example response =9001.
- sw-close: no error, , no record message returned, TLS application AEAD is closed, for example response =9002
- sw-error: error, no record message returned.
- sw-more(size): no error, a message or message fragment is ready. For example sw-more(size)= 61xy, in which xy is the size of the first fragment.

```
              TCP/IP Node                   Secure Element
                  |                               |
                  |-RECV(F-First, Frag#1)-------->|
                  |<-------------------sw-ok 9000-|
                  |-RECV(F-More,  Frag#i)-------->|
                  |<------------------sw-ok= 9000-|
                  |-RECV(F-Last,  Frag#n)-------->|
                  |<------------------sw-ok= 9000-|
                  |<----------------sw-open= 9001-|
                  |<---------------sw-close= 9002-|
                  |<----------sw-more(size)= 61xy-|
```

Figure 4. Receiving record packet, segmentation mechanism.

3.2.2 Sending Record Packet

A sending procedure starts by the reception of a sw-more(size) status, ending a response. This event may occur at the end of RECV procedure (see figure 6) or after TLS state machine reset (see figure 5).

A RECV(F-First, No-Frag) request resets the TLS state machine. For TLS client a sw-more(size) status is returned. For TLS server the sw-ok status is returned.

```
    TCP/IP Node                   Secure Element
        |                               |
        |-RECV(F-First, No-Frag)------->|=> Reset State Machine
        |<------------------sw-ok= 9000-| Server
        |<----------sw-more(size)= 61xy-| Client
```

Figure 5. Starting the SEND procedure after RESET request.

```
    TCP/IP Node                   Secure Element
        |                               |
        |-RECV(F-Last, Last-Fragment)-->| => End of Message
        |<----------sw-more(size)= 61xy-| Client
```

Figure 6. Starting SEND procedure after the end of RECV procedure.

The SEND(size) reads a record fragment, whose length is equal to size. It MAY be necessary to adjust the SEND size (see figure 7). Typically at the end of RECV procedure, the size indicated by the sw-more(size) status is an expected fragment length. In that case the status sw-retry status (for example 6Cxy) indicates the fragment size.

```
    TCP/IP Node                   Secure Element
        |                               |
        |-RECV(F-Last, Last-Frag)------>| => End of Message
        |<----------sw-more(size)= 61xy-|
        | SEND(size)------------------->|
        | <------- sw-retry(size')=6Czt-|
        | SEND(size')------------------>|
```

Figure 7. Adjusting SEND size.

The SEND(size) request is encoded as :
CLA=0, INS=C0, P1=0, P2=0, P3=size

The SEND procedure (see Figure 8) is a set of SEND requests, which read record packet fragments.

```
  TCP/IP Node                           Secure Element
    |                                         |
    |<--------------- ---sw-more(size#1)= 61xy-|
    |-SEND(size#1)--------------------------->|
    |<---------------Frag#1 || sw-more(size#2)-|
    |-SEND(size#i)--------------------------->|
    |<-----------Frag#i || sw-more(size#[i+1])-|
    |-SEND(size#n)--------------------------->|
    |<-------------------------Frag#n || sw-ok)-|=> SEND End
    |<------------Frag#n || sw-more(next-size)-|=> SEND Continue
    |<----------------------Frag#n || sw-open)-|=> Open
    |<----------------------Frag#n || sw-close-|=> Close
```

Figure 8. The SEND procedure

At the end of SEND procedure four events MAY occur:
- End of SEND procedure (status = sw-ok). No more record packets are available.
- SEND procedure to be continued (status = sw-more(size)). Another record packet is available.
- End of SEND procedure, application AEAD is ready for use (status = sw-open)
- End of SEND procedure, application AEAD is closed (status = sw-close)

3.2.4 RECV and SEND procedure for open application AEAD

When the application AEAD is opened RECV performs decryption and encryption operations (see figure 9).

For decryption operation (RECV(F-Decrypt)) the RECV procedure pushes the incoming record packet. The returned payload by the SEND procedure is the decrypted message ended by the protocol byte.

For encryption operation (RECV(F-Encrypt)) the RECV procedure pushes the content to encrypt ended by the associated protocol byte. The returned payload by the SEND procedure is a record packet, including the encrypted content.

```
 TCP/IP Node                                    Secure Element
    |                                                 |
    |-RECV(F-First, Frag#1)-------------------------->|
    |<------------------------------------sw-ok= 9000-|
    |-RECV(F-More,  Frag#i)-------------------------->|
    |<------------------------------------sw-ok= 9000-|
    |-RECV(F-Decrypt/F-Encrypt, F-Last, Frag#n)------>|
    |<--------------------------sw-more(size#1)= 61xy-|
    |-SEND(size#1)----------------------------------->|
    |<----------------------Frag#1 || sw-more(size#2)-|
    |-SEND(size#i)----------------------------------->|
    |<------------------Frag#i || sw-more(size#[i+1])-|
    |-SEND(size#n)----------------------------------->|
    |<-------------------------------Frag#n || sw-ok)-|=> SEND End
    |<-----------------------------Frag#n || sw-close-|=> Close
```

Figure 9. Decryption/Encryption operations.

3.3 TLS state machine

The state machine manages TLS flights, it determines the next record packet to be received and checked, and the next record packet to be built and sent. The number of states and their order is dependent on the TLS-SE role (client or server), and on the supported working mode (pre shared key, server with certificate, server and client with certificate). Figure 10 details an example of state machine for TLS-SE server, using pre-shared key. The ordered list of states comprises: S-Ready, S-Extensions, S-SFinished, S-ClientCCS, S-CFinished and S-Open.

```
          TCP/IP Node                   Secure Element
              |                               |
              |-RESET------------------------>|
              |<------------------------sw-ok-| state = S-Ready
       Client |-RECV(F-First,Frag#1)--------->|
       Hello  |<------------------------sw-ok-|
              |-RECV(F-Last,Frag#2)---------->|
              |                               | Check-ClientHello
              |                               | Make-ServerHello
              |<----------------sw-more(size)-| state= S-Extensions
              |-SEND(size)------------------->|
       Server |<--------Packet||sw-more(size)-|
       Hello  |                               |
              |-SEND(size)------------------->| Make-Extensions
    Server    |<--------Packet||sw-more(size)-| state= S-SFinished
    Encrypted |                               |
    Extension |-SEND(size)------------------->|
              |                               | Make-ServerFinished
     Server   |<----------------Packet||sw-ok-| state= S-ClientCCS
     Finished |                               |
              |                               |
    ClientCCS |-RECV(F-First,F-Last,Packet)-->| Check-ClientCCS
              |<------------------------sw-ok-| state= S-CFinished
              |                               |
              |                               |
    Client    |-RECV(F-First,F-Last,Packet)->-| Check-CFinished
    Finished  |<----------------------sw-open-| state= S-Open
              |                               |
       Packet |-RECV(F-Decrypt,Packet)------->| Decrypt Packet
              |<----------------sw-more(size)-| Clear Form Message
              |-SEND(size)------------------->|
              |<------ -Message||ptcol||sw-ok-|
              |                               |
      Message |-RECV(F-Encrypt,Message)------>| Encrypt
              |<----------------sw-more(size)-| Record Packet
              |-SEND(size)------------------->|
              |<------ -Record Packet ||sw-ok-|
              |                               |
```

Figure 10. TLS-SE server with pre-share key state machine

3.4 TLS library

The TLS-SE library is a set of procedures that check, according to the state machine, incoming record packets and build outgoing record packets. In figure 10 the TLS library comprises the following elements: Check-ClientHello, Check-ClientCCS, Check-ClientFinished, Make-ServerHello, Make-EncryptedExensions, and MakeServerFinished.

4 ISO7816 interface

The RECV and SEND binary encoding is shown by figure 11

name	CLA	INS	P1	P2	P3	Payload
RECV	00	D8	01= Decrypt 02= Encrypt	01= First 02= Last	Fragment Length 0= RESET	Yes
SEND	00	C0	00	00	Incoming Length	No

Figure 11. RECV and SEND ISO7816 requests binary encoding

The status word binary encoding is shown by figure 12. Two binary encoding of sw-more MUST be supported. In the T=0 context, SE operating system returns the 61xy status when a request including a payload, induces a response with a payload. The status 9Fxy is managed by the application in order to notify response size to be returned. The TLS-SE application MAY use 61xy status, but this could induce interoperability issues.

name	SW1	SW2
sw-ok	90	00
sw-more(size)	61 9F	size size
sw-retry(size)	6C + size	
sw-open	90	01
sw-close	90	02
sw-error	6D 6F	error number

Figure 12. ISO7816 status word binary encoding

5 ISO 7816 Use Case

An open implementation is available at [TLS-SE].

Below is an illustration of TLS-SE server, using a pre-shared key (PSK) with DHE over the secp256r1 curve, and the cipher suite AES-128-CCM-SHA256. The time consumed by handshake is about 1.4s.

```
PSK=
0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20
DHE=
037E6E633541EC03DB700A28E7DABB74F8E84D4A28E5F024B46F468A7821305D

RECV(Client Hello)

Tx: 00 D8 00 01 F0 16 03 03 00 F2 01 00 00 EE 03 03
    4E 65 53 05 52 AB 3E 83 14 0B 2F 9C 2F D7 BC 16
    F9 F5 C4 A9 86 CA 3F C8 8C 6E 8C D1 10 BB B1 57
    00 00 02 13 04 01 00 00 C3 00 2D 00 03 02 00 01
    00 2B 00 03 02 03 04 00 0D 00 1E 00 1C 06 03 05
    03 04 03 02 03 08 06 08 0B 08 05 08 0A 08 04 08
    09 06 01 05 01 04 01 02 01 00 33 00 47 00 45 00
    17 00 41 04 9A 1E 0A D8 40 88 D4 21 D1 55 D7 F2
    8F 78 4C 28 75 F5 19 CA 12 71 96 92 C4 07 8F B4
    35 42 57 E7 64 24 C1 BC 5D 89 0E F4 08 FD 25 8D
    24 F4 64 BB C3 F4 80 D3 BF 2C 23 A0 F9 2D A7 88
    0C 5B 44 53 00 0A 00 06 00 04 00 18 00 17 00 29
    00 3A 00 15 00 0F 43 6C 69 65 6E 74 5F 69 64 65
    6E 74 69 74 79 00 00 00 00 00 21 20 CC 05 4A 9F
    DE 70 E9 96 D6 01 69 61 F5 9A 78 20 D9 FC 6D ED
    4C C6 0A 7B 0D
Rx: 90 00  [47 ms]
Tx: 00 D8 00 02 07 4B 68 8F 4E B9 B2 CA

Rx: 61 86  [879 ms]

SEND(Server Hello)

Tx: 00 C0 00 00 86
Rx: 16 03 03 00 81 02 00 00 7D 03 03 5C 78 A4 E1 93
    34 D7 D9 64 B2 85 64 1B E4 76 63 94 39 1F 4A 15
    27 0A A4 C6 A0 C6 93 D9 E2 16 4D 00 13 04 00 00
    55 00 29 00 02 00 00 00 33 00 45 00 17 00 41 04
    25 C9 16 94 8B 39 51 D2 8E 88 70 F7 F5 4E 6C 31
    62 93 B1 65 55 2C 30 B2 5E 75 6C D8 FE AF DA A7
    67 D8 AD A7 BE 68 54 EA 3E A0 0B 4D CC 62 93 96
    38 07 68 29 3E D5 E6 0C 25 4A EA 12 C9 F8 99 7F
    00 2B 00 02 03 04 9F 1C  [32 ms]
```

```
SEND(Server Encrypted Extensions)

Tx: 00 C0 00 00 1C


Rx: 17 03 03 00 17 E6 04 4A 52 1A 50 B5 54 D8 73 5E
    00 F4 FD 66 BB B3 74 50 99 36 C8 08 9F 3A  [78 ms]

SEND(Server Encrypted Finished)

Tx: 00 C0 00 00 3A

Rx: 17 03 03 00 35 CB CA 03 3E E4 34 7E D2 0C 7C 24
    C1 8F 39 A2 74 39 24 47 78 BE 94 95 7A 31 EC 03
    D5 0C A8 1C 46 04 05 F2 83 3E 99 0D AD D6 66 63
    60 23 F8 5D 7B 77 0F 95 18 35 90 00  [185 ms]

RECV(Client Encrypted Finished)

Tx: 00 D8 00 03 3A 17 03 03 00 35 BC 29 18 D1 B8 4B
    C0 3F 6F 81 79 D9 7E FD 58 E3 76 EA 61 13 9C 3E
    40 0F 34 CD 94 CE C1 44 CB 76 70 7D DA 8A 54 69
    41 D9 80 CD 5D 52 8F E5 38 D8 52 92 20 54 5E
Rx: 90 01  [389 ms]

TLS13 session is open

Decryption of incoming Record Packet
RECV(Decrypt, Packet)
Tx: 00 D8 01 03 24 17 03 03 00 1F 56 E2 D5 B5 C4 A6
    E2 3E 54 56 5A C4 2D E9 99 F3 58 22 34 15 15 A7
    96 FD 0E B0 61 60 4C 52 87
Rx: 61 0F  [78 ms]

SEND(Message)

Tx: 00 C0 00 00 0F

Rx: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21 0D 0A 17 90
    00  [15 ms]
Rx: hello world! ptcol=17

Encryption of message

RECV(Encrypt,Message)
Tx: 00 D8 02 03 0F 68 65 6C 6C 6F 20 77 6F 72 6C 64
    21 0D 0A 17
```

```
Rx: 61 24  [79 ms]
```

```
SEND(Record Packet)

Tx: 00 C0 00 00 24
Rx: 17 03 03 00 1F 6F 78 FF 68 0F CA 9E 31 53 2C 96
    B3 FA D7 B0 51 1B 92 81 35 3D DB FE E9 18 A7 DF
    36 2F A5 27 90 00  [16 ms]
```

5 TLS-SE Name

According to ISO7816 standards, secure elements return upon reset a set of bytes called Answer to Reset (ATR). ATR comprises at least two bytes (TS, T0). The LSB nibble of T0 indicates the number of historical bytes (ranging from 0 to 15). Historical bytes (HB) are located at the end of ATR. Historical bytes can be programmed by standardized API, and therefore MAY be used for secure element naming.

6 Server Name Indication

According to [RFC6066] Server Name Indication extension is used to route TLS packets toward a virtual host.
Multiple TLS-SE devices, embedding standalone applications, can be hosted by an internet node. In this case SNI extension MAY be used in order to select the right secure element, whose name, typically stored in historical bytes, is determined from SNI.

7 IANA Considerations

This draft does not require any action from IANA.

8 Security Considerations

This entire document is about security.

9 References

9.1 Normative References

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, https://www.rfc-editor.org/info/rfc8446.

[RFC6066] Eastlake 3rd, D., "Transport Layer [RFC6066] Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011.

[ISO7816] ISO 7816, "Cards Identification - Integrated Circuit Cards with Contacts", The International Organization for Standardization (ISO).

[IEEE1363] IEEE, "IEEE Standard Specifications for Public Key Cryptography", IEEE Std. 1363-2000, DOI 10.1109/IEEE STD.2000.92292.

9.2 Informative References

[IM] Urien, P., "Identity Module for TLS Version 1.3", draft-urien-tls-im-04.txt, July 2021.

[TLS-SE] "tls-se.java", https://github.com/purien/TLS-SE

10 Authors' Addresses

Pascal Urien
Telecom Paris
19 place Marguerite Perey
91120 Palaiseau   Phone: NA
France   Email: Pascal.Urien@telecom-paris.fr