TLS Working Group
Internet Draft
Intended status: Experimental

June 24 2020

Expires: December 2020

P. Urien
Telecom Paris

Identity Module for TLS Version 1.3
draft-urien-tls-im-00.txt

Abstract

TLS 1.3 will be deployed in the Internet of Things ecosystem. In many IoT frameworks, TLS or DTLS protocols, based on pre-shared key (PSK), are used for device authentication. So PSK tamper resistance, is a critical market request, in order to prevent hijacking issues. If DH exchange is used with certificate bound to DH ephemeral public key, there is also a benefit to protect its signature procedure. The TLS identity module (im) MAY be based on secure element; it realizes some HKDF operations bound to PSK, and cryptographic signature if certificates are used. Secure Element form factor could be standalone chip, or embedded in SOC like eSIM.

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

TLS 1.3 [RFC8446] will be deployed in the Internet of Things ecosystem. In many IoT frameworks, TLS or DTLS protocols, based on pre-shared key (PSK), are used for device authentication. So PSK tamper resistance, is a critical market request, in order to prevent hijacking issues. If DH exchange is used with certificate bound to DH ephemeral public key, there is also a benefit to protect its signature procedure. The TLS identity module (im) MAY be based on secure element [ISO7816]; it realizes some HKDF [RFC5869] operations bound to PSK, and cryptographic signature if certificates are used. Secure Element form factor could be standalone chip or embedded in SOC like eSIM.

```
+-----------+      +----------+
| Processor |      | Identity |
|  TLS 1.3  +------+  Module  |
|           |      |    im    |
+-----------+      +----------+
```

Figure 1. TLS 1.3 Identity Module (im)

2 Protecting the Key Schedule for PSK

2.1 Context

According to [RFC8446] external PSKs MAY be provisioned outside of TLS.

The Early Secret (ESK) is computed according to relation:
ESK =HKDF-Extract(salt=0s,PSK) = HMAC(salt=0s,PSK)

The Binder Key (BSK) for outside provisioning is computed according to the relation:
BSK = Derive-Secret(ESK, "ext binder", "")

The Derived Secret (DSK) is computed according to the relation:
DSK= Derive-Secret(ESK, "derived", "")

2.2 Identity Module Procedures

The identity module MUST provide a KSGS (Keys Secure Generation and Storage) procedure, which computes and securely stores ESK, BSK and DSK keys.

This procedure MUST require administrative rights.

A set IMKP (Identity Module Key Procedures) of four procedures is required, in order to protect from public exposure ESK, BSK and DSK:

- CETS: Client Early Traffic Secret
- EEMS: Early Exporter Master Secret
- HEDSK: HKDF-Extract from Derived Secret Key
- HBSK: HMAC from Binder Key Secret

These procedures MAY require user rights.

2.3 KSGS: Keys Secure Generation and Storage

The Identity module MUST provide a KSGS procedure, requiring administrative rights, which computes and securely stores ESK, BSK, DSK

Input: salt and PSK
Output: Success or Failure

ESK, DSK, and BSK secret values are stored in the identity module

```
ESK= HMAC(salt=0s,PSK)
DSK= HMAC(ESK,Hash-Length || 0d746c73313320646572697665640001)
BSK= HMAC(ESK,Hash-Length || 10746c7331332065787420696e6465720001)
```

2.4 Identity Module Key Procedures (IMKP)

2.4.1 CETS: Client Early Traffic Secret

Input: Length, Message
Output: Client Early Traffic Secret or Failure

```
CETS(ClientHello) = Derive-Secret(ESK, "c e traffic", Message)
= HMAC(ESK, Length || 11746c73313320632065207472616666963 ||
Message || 01)
```

2.4.2 EEMS: Early Exporter Master Secret

Input: Length, Message
Output: Early Exporter Master Secret or Failure

```
EEMS(ClientHello) = Derive-Secret(ESK, "e exp master", Message)
= HMAC(ESK, Length || 12746c7331332065206578702006d6173746572 ||
Message || 01)
```

2.4.3 HEDSK: HKDF-Extract from Derived Secret Key

Input: DHE
Output: Handshake Secret or Failure

```
EDSK(DHE)= HKDF-Extract(DHE, DSK) = HMAC(DHE, DSK)
```

2.4.4 HBSK: HMAC from Binder Key Secret

Input: data
Output: HMAC(BSK, data) or Failure

HBSK(data) = HMAC(BSK, data)

3. Asymmetric Signature

The identity module MUST provide a GENKEY (GENKEY: Generate Key) procedure, in order to store or generate private asymmetric key and associated public key.

This procedure MUST require administrative rights.

The procedure GETPUB (GETPUB: Get Public Key) is required in order to read the public key value.

This procedure MAY require user rights.

The procedure SIGN (SIGN: Signature) is required in order to perform a raw signature for a digest value, computed from certificate.

This procedure MAY require user rights.

3.1 GENKEY

Input: None
Output: Success or Failure

A private key is generated and store in the identity module. A public key is computed from the private key.

3.2 GETPUB

Input: None
Output: Public Key Value or Failure

3.3 SIGN

Input: DigestValue
Output: Signature Value or Failure

4. Secure Element as Identity Module

Secure elements are defined according to [ISO7816] standards. They support hash functions (sha256, sha384, sha512) and associated HMAC procedures. They also provide DH procedures in Z/pZ* and elliptic curves. Open software can be released thanks to the Javacard standards, such as JC3.04, JC3.05.

Below is an illustration of binary encoding rules for secure element according to the T=1 ISO7816 protocol.

An ISO7816 command (TAPDU) 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 payload, or number of expected bytes

The response comprises a payload (up to 256 bytes) and a two bytes status word (SW1, SW2), 9000 meaning successful operation.

4.1 Administrative mode

The [ISO7816] command VERIFY (INS=0x20) SHOULD be used to enter the administrative mode

```
Tx: CLA=00 INS=20 P1=00 P2=Adm P3=PIN-Length [PIN-Value]
Rx: 9000
```

4.2 User Mode

The [ISO7816] command VERIFY SHOULD be used to enter the user mode

```
Tx: CLA=00 INS=20 P1=00 P2=User P3=PIN-Length [PIN-Value]
Rx: 9000
```

4.3 KSGS: Keys Secure Generation and Storage

Length= 2 + Salt-Length + PSK-Length

```
Tx: CLA=00 INS=TLS13 P1=0 P2=KSGS P3=Length Salt-Length [Salt-Value]
PSK-Length [PSK-Value]
Rx: 9000
```

This procedure computes and stores ESK, BSK and DSK.

4.3.1 Example

```
PSK=0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20

Tx: CLA=00 INS=85 P1=00 P2=0A P3=23 01 00 20
0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20
Rx:9000

ESK= HMAC-SHA256(0,PSK)
ESK=
23499E7EDF0FBE6BAA137DF0F23BECAEFA722AD19FC262855409DE8CD8B3C897

DSK= HMAC-SHA256(ESK,0020 0d746c7331332064657269766564 00 01)
DSK=
98EEAA27F7D77499E5FBC63A413CD8C395CAE42D850B65A5AE6A63807368A3F5

BSK = HMAC-SHA256(ESK,0020 10746c73313320657874206269e646572 00 01)
BSK=
4B6B423D2B92D840CC9A1A30D457BC5A4B10918587BBFF96380E91CE20A5FA2C
```

4.4 CETS: Client Early Traffic Secret

```
Length = 2 + Messages-Length
Hash-Length: the hash length (2 bytes)

Tx: CLA INS=TLS13 P1=CETS P2=ESK P3=Length Hash-Length Messages-
Length [Messages]
Rx:[Client Early Traffic Secret] 9000
```

4.4.1 Example

```
Tx: CLA=00 INS=85 P1=00 P2=0B P3=03 0020 00
Rx: 0738A2B6F6FAA2AF5CDD9B6F0F2B232F19B3256A5926EAC600B911F91E98D2D4
9000

Message= NULL = 0s
[Client Early Traffic Secret] =
HMAC-SHA256(ESK, 0020 11746c7331332063206520747261666963 00 01)
```

4.5 EEMS: Early Exporter Master Secret

```
Length = 2 + Messages-Length
Hash-Length: the hash length (2 bytes)

Tx: CLA INS=TLS13 P1=EEMS P2=ESK P3=Length Hash-Length Messages-
Length [Messages]
Rx: [Early Exporter Master Secret] 9000
```

4.5.1 Example

```
Tx: CLA=00 INS=85 P1=01 P2=0B P3=03 0020 00
Rx: 9B7FC6A8F854C16A301DFC566859931DB5EE9A22793142A0C67159C445E7BEAB
9000

Message= NULL = 0s
[Early Exporter Master Secret] =
HMAC-SHA256(ESK, 0020 12746c7331332065206578706d6173746572 00 01)
```

4.6 HEDSK: HKDF-Extract from Derived Secret Key

```
Tx: CLA INS=TLS13 P1=0 P2=HEDSK P3=Data-Length [Data]
Rx: [HMAC(Data,DSK)] 9000
```

4.6.1 Example

```
Tx: CLA=00 INS=85 P1=00 P2=0E P3=01 00
Rx: 3074777017FA405DB00BF0F4E24E5A3E0A5F8CE357472BEA4F442D7754E13BF2
900

DHE=NULL=0s
HMAC-256(DHE,DSK)= HMAC-256(0s,DSK)
```

5 IANA Considerations

TODO

6 Security Considerations

TODO

7 References

7.1 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.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

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

7.2 Informative References

8 Authors' Addresses

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