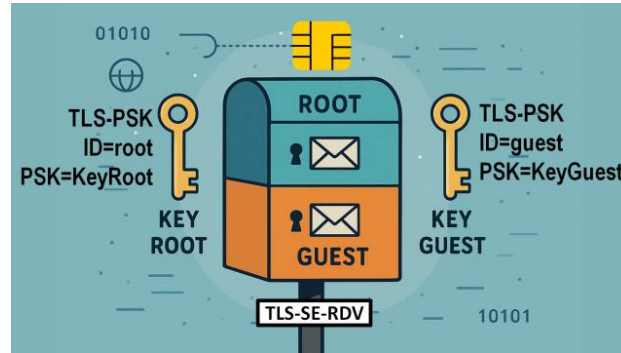


Trusted Asynchronous Communications for a Safer Internet



<https://datatracker.ietf.org/doc/draft-urien-tls-se-rdv/>

Pascal.Urien@ethertrust.com

Pascal.Urien@telecom-paris.fr

About Asynchronous Communications



Security Issues:

- Communication Security
 - TLS1.3 PSK
- Storage Trust
 - Secure Element

Two entities are not online at the same time. One entity post a content to a storage server, and latter the second one read or use this content

Some Architectures:

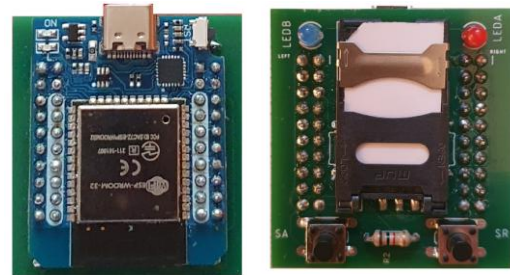
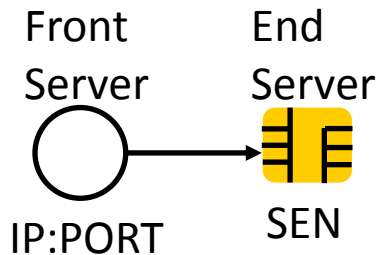
- eMail (STMP)
- Instant Messaging (Signal)
- Publish & Subscribe
- MQTT (Message Queuing Telemetry Transport)

About TLS for Secure Element (TLS-SE)

- Secure element are widely used by bank card, SIM modules, electronics passports. Ten billions SE are produced every year.
- TLS-SE is an embedded TLS server for SE. It realizes a transparent bridge between TLS1.3 and ISO7816. It works with pre-shared keys with out-of-band provisioning.
- Internal resources are identified by uniform resource identifier (URI)
 - **schemeS://SEN:PSK(ID)@IP:PORT/?query**
 - scheme is the syntax used over TLS
 - S means secured by TLS
 - SEN is the TLS Server Name used by the TLS-SE application (15 bytes)
 - PSK is the pre-shared-key (256 bits)
 - ID is the psk-identity
 - IP is the front TLS server IP address
 - PORT is the front TLS server port
 - query is a request expressed according to a scheme

Architectures for TLS-SE

- Secure Element is an end TLS server identified by a name, it has no IP connectivity.
- A TLS front server is needed that supports internet connectivity
- There are two kinds of TLS-SE servers
 - Single TLS-SE end server
 - Grid of TLS-SE end servers



**schemeS://SEN:PSK(ID)
@IP:PORT/?query**

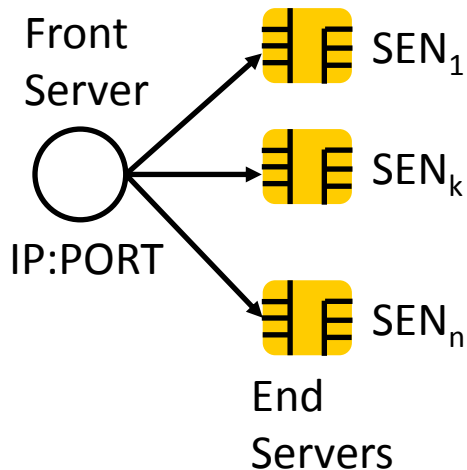


Illustration of TLS-SE Packet Exchange

```
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_identity Client identity -psk [PSK in hexadecimal]
```

RECV(*Client Hello*, 264 bytes)

Tx: **00D80001F0**

```
1603030103010000FF03035EB531B2CD19D5C6F39EEFD7F28517DBF  
158409F363D0B9E3EB5FF15C9C3F38A0000021304010000D4002D00  
03020001002B0003020304000D001E001C060305030403020308060  
80B0805080A08040809060105010401020100330047004500170041  
040E53B725DDC74E89CBBF4F09305DA6A0FC980F36805EBB1F4D5D9  
0071EAAE3D306F2AD081466E107D637D994433D86E59910F01512B4  
B91B2A54B0E867BE6ED8000A00060004001800170000000D000B000  
0086B6579312E636F6D0029003A0015000F436C69656E745F696465  
6E746974790000000000021203E10A0F4E3E33CDD
```

Rx: **9000** (101ms)

Tx: **00D8000218**

A7CED6506D869118F1BA27A81CE7F2D0907186310160E1E0

SEND(*Server Hello*, 134 bytes)

```
Rx:16030300810200007D03033CE7FF18A259156D3FFBFEFE90FC120BD73  
3E4B2BCB59BDC4443F5DFD11F5A94700130400005500290002000000330  
045001700410455C7C555C1F5B114BE8DAD00108081B4BCD027053C197A  
40E5254B9DA96FE99C0AF44515333C04EDB0CFE98DA589B0588C036D670  
540968E519786599BD28DDB002B00020304 9F1C (417ms)
```

SEND(*Server Encrypted Extensions*, 28 bytes)

Tx: **00C000001C**

```
Rx:17030300179D92BAB6772F673F070F2083C647C4977CEF6EF2BCF393  
9F3A (40 ms)
```

SEND(*Server Encrypted Finished*, 58 bytes)

Tx: **00C000003A**

```
Rx:1703030035CF85AB53BB6DBBDE7F45625222BF9436732E95B5866E1750D62F3  
6732513AE174CB31F7E3F43A48933E2CA19DA6C5AF24C905FA371 9000 (98ms)
```

RECV(*Client Encrypted Finished*, 58 bytes)

Tx: **00D8000033A**

```
17030300356806FD1AD7A828616F7950F92760C8F62056A6C7CD849681C9AD1767  
6F7DF8DFAD651158772212DE646D598934C5C4F2C9BEA505B0
```

Rx: **9001** (216 ms)

TLS13 session is open (872ms 542 bytes)

RECV(*Encrypted Message*) = CMD Write (69 clear bytes)

Tx: **00D8000035B**

```
17030300564829BD8A423DA235AA8F52B66A4EF17852AEF7BD47C8F9EFE526FF56  
6C8D52644B41B9A44330D17CD96EE6106C1E147925346C1A59D46AA3E0A217070D  
3CEB0FF98D12E061E6B324578FD2892C607BB99E039B3C7E69
```

Rx: 1703030015162D4E994E19F58C7A3A32287ABEAD92000D13C0E7 **9000**
(123 ms)

RECV(*Encrypted Message*) = CMD Close (5 clear bytes)

Tx: **00D8000031B**

```
1703030016C6AE5FCCCF95CCCF3CA444ABFA566F439CFF231D4BAA
```

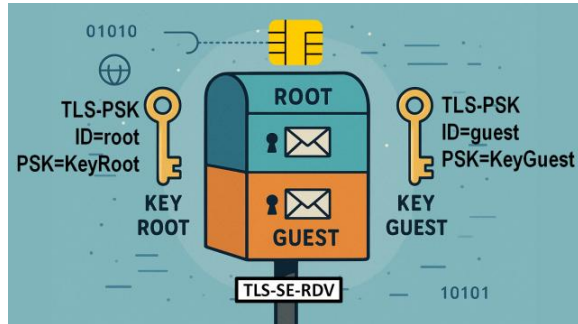
Rx: **6D16** (41ms)

TLS13 session is closed

red: server name (key1.com), **black** psk-identity (Client_identity), **green** binder, **blue** DH public keys
purple ISO7816 header

TLS-SE Rendez-Vous

- A TLS-SE server manages a root account and one or several guest accounts
- Root creates guest accounts with different privileges, for example to perform actions such as:
 - Read write operations
 - Cryptographic procedures, typically key generators



```
openssl s_client -tls1_3 -servername key1.com -connect 192.168.1.128_CCM_SHA256 -no_ticket -psk_identity root -psk "0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20"
Appuyez sur une touche pour continuer...
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 364 bytes
Verification: OK
---
Reused, TLSv1.3, Cipher is TLS_AES_128_GCM_SHA256
Secure Renegotiation IS NOT supported
No ALPN negotiated
Early data was not sent
Verify return code: 0 (ok)
---
?00
Ethertrust keystore 1.3C
?06guest
guest
?A0FF02030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20
FF02030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20
?A501
OK
Z001234
OK
I00
1234
?02
read:errno=0
```

Annotations for the terminal output:

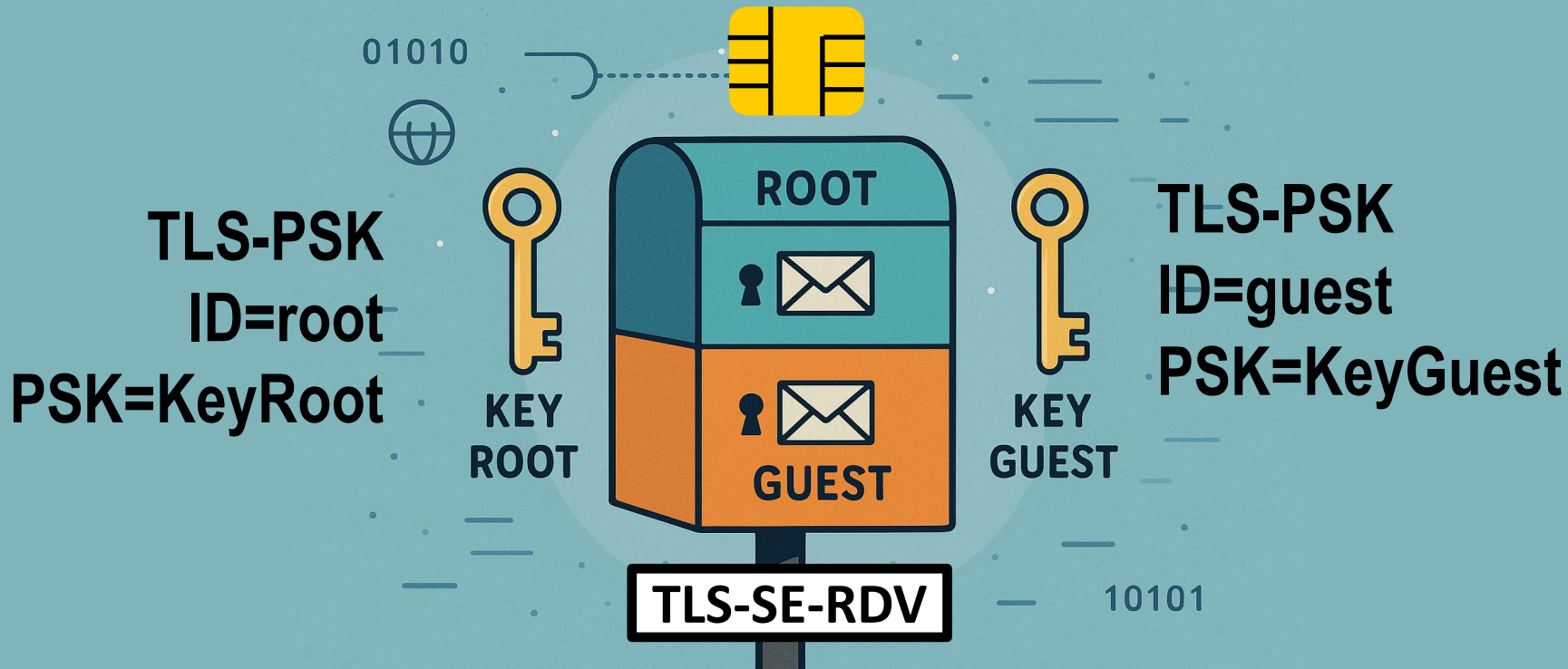
- create guest account (points to ?00)
- set guest PSK (points to ?06guest)
- enable guest account (points to ?A501)
- write "1234" data in record 0 (points to Z001234)
- disconnect (points to ?02)

<https://www.youtube.com/watch?v=bMAbxLbGvW4>

Call For Collaboration

- We would like to meet people interested by trusted internet TLS-SE server, in order to define:
 - Interface for TLS-SE resources, like CBOR (Concise Binary Object Representation),
 - Basic network services for TLS-SE-RDV
 - A minimal framework for accessing shared resources

Thank You For your Attention



<https://datatracker.ietf.org/doc/draft-urien-tls-se-rdv/>