TLS 1.3 integration for STSAFE TPM 2.0

Introduction

The objective of this application note is to describe how to integrate TLS 1.3 for TPM 2.0 with a Raspberry Pi.

For this purpose, it explains how to install packages and describe all the commands to obtain the TLS environment. To integrate the TPM with Linux, refer to [STSAFE-TPM integration].

Reference documents

The table below lists the documents referenced in this application note.

Table 1. Reference documents

Reference	Title
[TPM2-TSS]	GitHub repository containing projects for specific use cases: GitHub - tpm2-software/tpm2-tss: OSS implementation of the TCG TPM2 Software Stack (TSS2)
[OpenSSL-engine]	OpenSSL® engine project accessible from: GitHub - tpm2-software/tpm2-tss-engine: OpenSSL Engine for TPM2 devices
[TPM2-abrmd]	GitHub repository containing projects for specific use cases: GitHub - tpm2-software/tpm2-abrmd: TPM2 Access Broker & Resource Management Daemon implementing the TCG spec.
[TPM2-openssl]	GitHub repository containing projects for specific use cases: GitHub - tpm2-software/tpm2-openssl: OpenSSL Provider for TPM2 integration
[TLS]	TLS recommandations: anssi-guide-recommandations_de_securite_relatives_a_tls-v1.2.pdf
[STSAFE-TPM integration]	Integrating the STSAFE-TPM trusted platform modules with Linux® - Application note, STMicroelectronics
[OpenSSL]	Official website for OpenSSL: OpenSSL
[TPM2-Tools]	GitHub repository for the Trusted Platform Module (TPM2.0) tools based on tpm2-tss

1 Overview of TLS 1.3

TLS (transport layer security) is a cryptographic protocol that ensures secure communication over a computer network by encrypting data and verifying the identity of the parties involved.

TLS 1.3 is the latest version of the TLS protocol, offering significant improvements in security and performance compared to TLS 1.2.

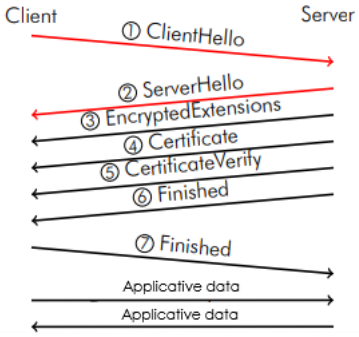
The TLS 1.3 platform supports the following high-level features:

- Reduced handshake latency with fewer round-trips
- Removal of outdated and insecure cryptographic algorithms
- Mandatory forward secrecy for enhanced privacy
- Simplified cipher suite selection

TLS 1.3 ensures authentication, confidentiality, and integrity of exchanged data.

All the messages which are necessary to initiate a TLS 1.3 session can be seen. in the figure below:

Figure 1. TLS 1.3 initiation



1. The client initiates a request by sending a `ClientHello` message, which contains the supported cipher suites and extensions.
2. The server responds with a `ServerHello` message that includes the selected cipher suite and the necessary extensions for the cryptographic key exchange.
3. The server sends an `EncryptedExtensions` message, which contains other extensions (not required for the key exchange) for this session whose cryptographic parameters do not depend on the key exchange.
4. The server sends a `Certificate` message, which notably contains its public key within a digital certificate.
5. The server authenticates itself to the client by sending a `CertificateVerify` message containing data signed with the private key corresponding to the previously sent public key.
6. The server sends a `Finished` message.
7. The client then sends its own `Finished` message.

2 Development environment setup and configuration

2.1 Prerequisites

Install this package before starting:

- TPM2 access broker and ressource manager [TPM2-abrmd]
- TPM software stack 2.0 [TPM2-TSS]
- TPM2 tools [TPM2-Tools]
- TPM2 TSS engine [OpenSSL-engine]
- TPM2 Openssl [OpenSSL]

2.2 Installing OpenSSL Engine

Check the installation of tpm2-tss-engine with the following code:

```
openssl engine -t -c tpm2tss
```

Figure 2. Test for TSS engine

```
root@raspberrypi:/home/tpm# openssl engine -t -c tpm2tss
(tpm2tss) TPM2-TSS engine for OpenSSL
 [RSA, RAND]
     [ available ]
```

Check your current version of OpenSSL:

```
openssl version
```

Figure 3. OpenSSL version

```
root@raspberrypi:/home/tpm# openssl version
OpenSSL 3.0.16 11 Feb 2025 (Library: OpenSSL 3.0.16 11 Feb 2025)
```

Check the installation of the providers of openssl:

```
openssl list -providers
```

Figure 4. OpenSSL providers

```
root@raspberrypi:/home/tpm/Documents# openssl list -providers
Providers:
  default
    name: OpenSSL Default Provider
    version: 3.0.16
    status: active
  tpm2
    name: TPM 2.0 Provider
    version: 1.3.0
    status: active
```

If tpm2 is not available after running this command, you need to create a file “providers.cnf” with the following content:

```
openssl_conf = openssl_init
[openssl_init]
providers = provider_sect
[provider_sect]
default = default_sect
tpm2 = tpm2_sect
[default_sect]
activate = 1
[tpm2_sect]
activate = 1
```

To make this file permanent in your OpenSSL configuration, make the following command:

```
export OPENSSL_CONF = /path/to/providers.cnf
```

2.3 Create directory structures

```
mkdir tls_tpm
mkdir -p tls_tpm/pki/{csr,certs,crl,newcerts,private}
mkdir -p tls_tpm/tpm2/{csr,certs,keys}
touch tls_tpm/pki/openssl.cnf tls_tpm/pki/index.txt tls_tpm/pki/index.txt.attry
```

In the file openssl.cnf, paste the following code:

```
[ ca ]
# `man ca`
default_ca = CA_default
[ CA_default ]
# Directory and file locations.
dir = ./tls_tpm/pki
certs = $dir/certs
crl_dir = $dir/crl
new_certs_dir = $dir/newcerts
database = $dir/index.txt
serial = $dir/serial
RANDFILE = $dir/private/.rand
# The root key and root certificate.
private_key = $dir/private/rootCA.key
certificate = $dir/private/rootCA.crt
# For certificate revocation lists.
crlnumber = $dir/crlnumber
crl = $dir/crl/intermediate.crl
crl_extensions = crl_ext
default_crl_days = 30
# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha256
name_opt = ca_default
cert_opt = ca_default
default_days = 375
preserve = no
policy = policy_loose
```

```
[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
```

```
[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
```

```
[ req ]
# Options for the `req` tool (`man req`).
default_bits = 2048
distinguished_name = req_distinguished_name
string_mask = utf8only
# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha256
# Extension to add when the -x509 option is used.
x509_extensions = v3_ca
```

```
[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name
localityName = Locality Name
0.organizationName = Organization Name
organizationalUnitName = Organizational Unit Name
commonName = Common Name
emailAddress = Email Address
# Optionally, specify some defaults.
countryName_default = US
stateOrProvinceName_default = California
localityName_default = Milpitas
0.organizationName_default = STMicroelectronics
organizationalUnitName_default = Security
emailAddress_default =
```

```
[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

```
[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
```

```
[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
```

```
[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always
```

```
[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

3 Key and certificate management with OpenSSL 3

3.1 Generate root CA key and certificate

Create rootCA key pair:

```
:openssl genrsa -out ./tls_tpm/pki/private/rootCA.key 2048
```

Self-signed rootCA certificate:

```
openssl req -config ./tls_tpm/pki/openssl.cnf -key ./tls_tpm/pki/private/rootCA.key -new -x50
9 -days 365 -sha256 -extensions v3_ca -out ./tls_tpm/pki/private/rootCA.crt
echo 1000 > ./tls_tpm/pki/serial
```

Figure 5. Root CA certificate information

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:US
State or Province Name [California]:California
Locality Name [Milpitas]:Milpitas
Organization Name [STMicroelectronics]:STMicroelectronics
Organizational Unit Name [Security]:Security
Common Name []:www.st.com
Email Address [security@st.com]:
```

Read the rootCA:

```
openssl x509 -in ./tls_tpm/pki/private/rootCA.crt -noout -text
```

Figure 6. root CA certificate

```
root@raspberrypi:/home/tpm/Documents# openssl x509 -in ./tls_tpm/pki/private/rootCA.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            17:46:4e:45:f1:3b:de:aa:30:0d:18:33:52:b8:f1:7b:38:2b:44:1d
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com, emailAddress = security@st.com
        Validity
            Not Before: Jul  1 15:32:40 2025 GMT
            Not After : Jul  1 15:32:40 2026 GMT
        Subject: C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com, emailAddress = security@st.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c0:b9:ce:d2:99:5e:68:c1:7a:49:6a:ac:3b:26:
                    3e:82:07:13:2e:b2:ab:af:98:87:4e:2d:82:c2:6e:
                    4b:76:dd:0e:05:14:28:46:d2:a8:f7:f8:70:6f:78:
                    90:8b:1a:e3:a6:3f:e1:5a:75:82:2c:51:45:b3:a9:
                    7c:f9:51:a7:86:1b:32:fe:07:73:5b:0f:fe:dc:4a:
                    13:9c:eb:4d:36:c4:e9:af:fe:07:b3:ec:f3:c2:0e:
                    d2:09:b0:3e:3d:bc:80:6b:6c:d8:fd:23:ae:24:17:
                    30:6c:5f:59:b3:7b:35:8c:65:98:ba:f6:93:61:70:
                    5b:54:ac:db:61:61:eb:c1:00:fa:38:a5:1a:48:a0:
                    4b:c9:61:3c:d2:03:cc:72:e2:89:37:ee:68:b7:95:
                    d4:5b:4e:87:43:6c:e8:1a:93:67:fc:2f:b9:54:b0:
                    e6:d1:09:72:6a:6b:b7:6a:73:4f:32:79:e5:33:20:
                    68:5d:63:2f:d8:13:18:a1:04:4f:12:35:d9:90:3c:
                    84:30:28:94:4b:45:04:d6:4c:24:07:c7:ea:ee:2c:
                    05:69:b8:d8:1f:6d:c6:2c:3a:45:52:fa:cb:15:93:
                    dc:af:f7:a9:c5:af:15:78:6c:02:97:fc:3c:c9:24:
                    7f:b1:60:ef:13:56:67:5f:f1:1c:60:e5:b3:dc:de:
                    dc:27
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                98:2E:0D:4F:84:82:A6:00:B0:7F:F1:35:39:23:7A:8A:6D:FB:64:A9
            X509v3 Authority Key Identifier:
                98:2E:0D:4F:84:82:A6:00:B0:7F:F1:35:39:23:7A:8A:6D:FB:64:A9
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        af:96:02:a5:73:de:65:de:9b:84:93:f1:e9:4c:7b:81:7c:55:
        18:17:23:5f:87:44:58:af:a5:fc:f0:a2:79:7f:d0:cc:92:52:
        85:f2:aa:93:5c:46:bc:0a:f5:13:30:84:0a:e1:9f:e3:d4:a4:
        95:be:b8:0d:30:24:98:11:da:0d:3e:18:05:00:16:cb:93:3a:
        64:5f:c9:68:66:8d:f4:7e:34:37:77:87:9b:cc:be:d9:0b:24:
        66:bc:3c:ef:8b:1a:e3:06:80:66:29:51:cb:51:ee:91:43:47:
        a6:3b:b5:3f:73:cb:c1:2b:42:2f:fd:ea:8c:f0:cc:d7:61:a7:
        d8:8b:44:46:97:ee:92:4d:87:6a:35:d8:ab:73:84:67:72:4d:
        a7:de:4a:2e:87:41:e9:c7:5e:f9:55:69:76:65:c8:20:e4:ca:
        b8:42:ad:d4:35:5c:70:a0:08:59:df:3d:94:23:94:cc:ef:1b:
        1d:8a:de:cf:75:73:85:95:b2:72:c1:f2:0f:c6:2d:94:02:a7:
        73:b4:eb:51:ff:0f:5a:4d:fb:7b:e5:76:f2:30:a4:f4:5e:80:
        b9:98:90:3d:61:38:7d:25:82:c4:18:31:98:ea:a3:73:8c:ad:
        2a:b3:6d:ae:0e:b7:83:18:33:61:a9:2f:f2:e0:a6:5c:e7:b0:
        45:aa:da:d6
```

3.2 Generate intermediate CA key and CSR

Generate Intermediate CA private key:

```
openssl genrsa -out ./tls_tpm/pki/private/intCA.key 2048
```

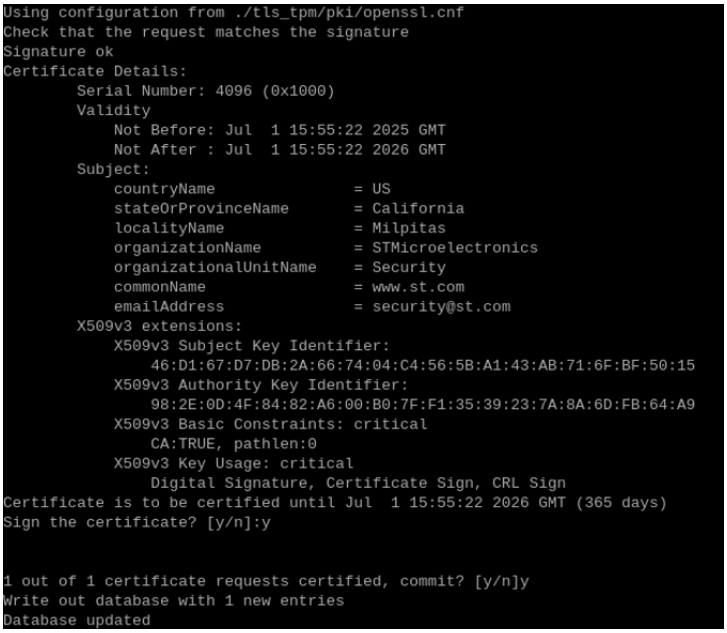
Create Intermediate CA CSR:

```
openssl req -config ./tls_tpm/pki/openssl.cnf -extensions v3_intermediate_ca -new -sha256 -ke
y ./tls_tpm/pki/private/intCA.key -out ./tls_tpm/pki/csr/intCA.csr
```

3.3 Sign intermediate CA certificate with root CA

```
openssl ca -config ./tls_tpm/pki/openssl.cnf -extensions v3_intermediate_ca -days 365 -notext
 -md sha256 -in ./tls_tpm/pki/csr/intCA.csr -out ./tls_tpm/pki/private/intCA.crt
```

Figure 7. Certificate sign



Verify the certificate chain:

```
openssl verify -verbose -x509_strict -CAfile ./tls_tpm/pki/private/rootCA.crt ./tls_tpm/pki/p
rivate/intCA.crt
```

A message is received if the verification is successful.

Reading the intermediate CA certificate:

```
openssl x509 -in ./tls_tpm/pki/private/intCA.crt -noout -text
```

Figure 8. Intermediate CA certificate

```
root@raspberrypi:/home/tpm/Documents# openssl x509 -in ./tls_tpm/pki/private/intCA.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4096 (0x1000)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com, emailAddress = security@st.com
        Validity
            Not Before: Jul  1 15:55:22 2025 GMT
            Not After : Jul  1 15:55:22 2026 GMT
        Subject: C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com, emailAddress = security@st.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b3:73:bf:af:a6:2c:43:9f:13:e2:90:52:15:18:
                    59:da:8d:39:90:0e:4e:c2:18:ea:30:33:5c:44:8a:
                    d8:ec:37:c4:a3:58:3a:0a:6f:e7:52:a9:7e:ef:7d:
                    f2:aa:4c:68:03:94:f9:38:60:bb:2d:1a:6f:db:6d:
                    b4:2d:2c:0b:7d:6f:4a:67:d2:68:17:ea:fc:58:bb:
                    7c:af:37:a4:3c:f2:21:5d:6c:33:05:dc:54:00:3c:
                    77:e1:79:27:6c:f8:4f:4d:25:fd:92:4a:03:69:27:
                    c6:72:41:99:af:99:9e:8a:a3:11:86:0c:fc:10:53:
                    c2:37:8b:f2:75:07:97:69:f7:ee:fe:a1:fc:40:2e:
                    ca:d4:cc:27:e4:ee:fe:ab:5f:98:65:e0:dc:9b:93:
                    2e:f8:54:7f:04:74:73:2b:93:9e:e6:b8:5c:9e:d4:
                    51:aa:b8:38:1a:07:8f:af:c1:98:12:67:b8:40:b8:
                    d1:c2:3a:de:03:55:56:32:55:1e:d7:f6:55:45:e2:
                    a2:fb:1b:a6:03:38:2c:f5:d4:d2:b8:87:ea:a4:ff:
                    c3:ff:f5:30:a2:c1:f9:da:be:93:46:b9:d5:e5:1a:
                    3d:09:41:e9:77:38:e3:39:40:ea:f8:12:fd:a3:6c:
                    fc:98:3b:48:92:9d:95:b2:73:66:32:d2:9b:3b:f4:
                    d5:4d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                46:D1:67:D7:DB:2A:66:74:04:C4:56:5B:A1:43:AB:71:6F:BF:50:15
            X509v3 Authority Key Identifier:
                98:2E:0D:4F:84:82:A6:00:B0:7F:F1:35:39:23:7A:8A:6D:FB:64:A9
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Key Usage: critical
                Digital Signature, Certificate Sign, CRL Sign
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        06:16:d0:51:72:6f:db:96:df:71:15:2f:e7:59:14:b5:4e:f2:
        33:3f:1b:93:3b:1b:27:9c:ee:64:26:04:f3:4d:a2:cf:e1:d8:
        b5:db:28:80:d0:f2:f4:23:c7:42:57:f8:f4:be:22:75:07:79:
        5c:6f:59:11:86:43:81:67:c6:9a:35:85:04:b9:17:bf:59:45:
        5f:26:28:86:3c:ca:06:5c:5c:10:6a:c6:8b:bf:cc:59:91:8d:
        6f:d5:ac:91:51:b1:3b:3a:7e:80:dd:5d:de:57:48:71:52:0b:
        68:ca:96:4f:2e:8b:16:ba:d5:09:18:aa:0c:0c:12:d4:1a:a5:
        43:8d:79:35:29:6c:9e:ba:ed:50:bf:0a:48:78:e2:69:77:85:
        1d:85:f5:0b:22:f2:2d:2c:c0:47:0e:6a:86:10:7f:54:25:99:
        97:2c:42:2d:69:f5:59:ed:83:d8:37:23:b7:a7:30:66:c2:1b:
        2c:16:72:34:f3:35:17:a5:aa:68:61:d7:c5:18:a1:77:f7:cc:
        04:b4:07:5d:b7:3d:14:5d:d7:b4:5e:8a:f8:f1:b9:fd:04:cd:
        42:00:e2:a4:7d:af:66:e6:56:53:ca:ba:05:f1:c9:29:e2:b7:
        6f:21:cb:91:4b:0e:a2:bb:28:f3:45:42:4f:51:fa:51:82:9d:
        43:07:3e:d0
```

3.4 Generate client key pair in STSAFE and create CSR

Generate client key:

```
tpm2tss-genkey -a rsa -s 2048 ./tls_tpm/tpm2/keys/client.key -p abc
```

Create client CSR using OpenSSL:

```
openssl req -keyform tss -provider tpm2 -config ./tls_tpm/pki/openssl.cnf -key ./tls_tpm/tpm2
/keys/client.key -new -out ./tls_tpm/tpm2/csr/client.csr -passin pass:abc
```

Read the client CSR:

```
openssl req -in ./tls_tpm/tpm2/csr/client.csr -noout -text
```

Figure 9. Client CSR

```
root@raspberrypi:/home/tpm/Documents# openssl req -in ./tls_tpm/tpm2/csr/client.csr -noout -text
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:9a:7b:06:56:dd:69:01:d9:d9:74:8b:8e:f7:a5:
                    80:af:f5:68:97:0d:46:f9:dd:1e:91:28:6d:7e:5c:
                    97:d1:dd:d8:d7:03:56:f4:ea:3f:fe:88:17:33:6f:
                    54:e8:97:02:0c:e5:46:fd:e6:26:a6:3c:88:23:93:
                    ec:c0:68:85:17:f2:7d:2a:2f:76:c3:bf:5c:7c:eb:
                    04:69:a3:1b:2c:51:af:5d:93:e0:c6:4a:59:29:44:
                    c6:4d:cb:61:bc:39:41:74:47:6c:58:25:15:fa:83:
                    82:da:9c:5b:e5:83:aa:ff:17:a0:6f:23:71:72:1e:
                    d5:b5:60:c1:76:b9:01:57:9c:21:af:a9:32:03:e2:
                    e4:d2:89:e1:b5:b0:ff:10:84:3d:e9:84:88:89:da:
                    4a:f6:a3:36:fc:99:48:d1:da:56:78:29:0e:e2:b1:
                    ad:6f:11:3f:ea:b0:a9:88:13:7f:c3:01:e3:25:01:
                    47:2b:f9:3c:ba:04:db:47:1b:61:a5:b8:c1:f6:47:
                    99:e0:15:58:2b:74:c5:d6:21:43:c6:0d:f7:ee:38:
                    40:ef:39:ea:ef:5a:9a:79:84:04:3d:d7:0d:82:2d:
                    73:15:16:2b:db:6f:7f:4c:4c:ea:82:1f:c7:81:b0:
                    fe:83:4d:33:39:fe:7c:0d:ca:e0:ef:57:d8:e9:9e:
                    1f:85
                Exponent: 65537 (0x10001)
        Attributes:
            (none)
            Requested Extensions:
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        21:37:88:46:7f:60:44:ef:92:27:40:05:66:aa:c8:f6:08:82:
        a7:0e:a9:ea:1e:19:6c:f8:88:5b:04:d9:02:66:b4:cf:ba:f4:
        bf:bf:58:83:b9:b7:a7:1d:26:74:a0:53:10:99:c3:96:c9:26:
        39:c6:f7:ea:60:77:ce:cd:c2:f5:7a:aa:f1:87:69:2d:a4:6f:
        db:4d:60:a1:d2:65:13:67:bf:46:e4:66:49:47:ad:6c:e0:7a:
        b8:d3:c6:01:0a:fc:d8:c9:ce:fd:95:5a:47:19:f6:52:3f:79:
        7b:8b:84:9a:c8:1e:4a:66:a6:bd:ec:ce:fd:06:12:49:b3:33:
        b8:ca:a8:9e:c1:c9:4a:8c:a2:b1:46:4b:e4:4b:c0:8f:72:5f:
        b4:a6:5d:9b:39:e2:27:ea:5d:ef:e0:ef:fc:a8:06:32:94:6c:
        02:a9:14:97:ba:78:1e:1a:0c:14:f0:d3:94:f8:e4:f5:d0:03:
        e7:24:32:e1:a9:92:47:72:72:1d:8c:c8:93:98:3b:62:e8:3b:
        f8:82:61:46:9b:66:3c:eb:e5:18:d9:8d:35:a9:b0:ee:f7:74:
        d8:21:d2:a4:a2:c7:59:75:b1:4b:f2:f5:ac:7e:4d:92:fb:46:
        36:a0:f0:2e:ed:61:db:d6:f3:88:d0:f0:fd:0d:34:44:3a:52:
        dc:77:bc:83
```

3.5 Sign client certificate with intermediate CA

Create a file “v3.ext” with the following content:

```
basicConstraints=CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
```

Then run:

```
openssl x509 -req -in ./tls_tpm/tpm2/csr/client.csr -CA ./ tls_tpm /pki/private/intCA.crt -CA
key ./tls_tpm /pki/private/intCA.key -CAcreateserial -out ./tls_tpm/tpm2/certs/client.crt -da
ys 365 -extfile v3.ext
```

Figure 10. Signing client CSR with intermediate CA

```
Certificate request self-signature ok
subject=C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com
```

Verification of the certificate chain:

```
openssl verify -CAfile ./tls_tpm/pki/private/rootCA.crt -untrusted ./tls_tpm/pki/private/intC
A.crt ./tls_tpm/tpm2/certs/client.crt
```

Read the client certificate:

```
openssl x509 -in ./tls_tpm/tpm2/certs/client.crt -noout -text
```

Figure 11. **Client certificate**

```
root@raspberrypi:/home/tpm/Documents# openssl x509 -in ./tls_tpm/tpm2/certs/client.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            69:0b:51:39:73:a2:af:db:63:0a:bf:ea:06:cd:28:f4:e5:66:71:c6
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com, emailAddress = security@st.com
        Validity
            Not Before: Jul  1 16:07:15 2025 GMT
            Not After : Jul  1 16:07:15 2026 GMT
        Subject: C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = www.st.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:9a:7b:06:56:dd:69:01:d9:d9:74:8b:8e:f7:a5:
                    80:af:f5:68:97:0d:46:f9:dd:1e:91:28:6d:7e:5c:
                    97:d1:dd:d8:d7:03:56:f4:ea:3f:fe:88:17:33:6f:
                    54:e8:97:02:0c:e5:46:fd:e6:26:a6:3c:88:23:93:
                    ec:c0:68:85:17:f2:7d:2a:2f:76:c3:bf:5c:7c:eb:
                    04:69:a3:1b:2c:51:af:5d:93:e0:c6:4a:59:29:44:
                    c6:4d:cb:61:bc:39:41:74:47:6c:58:25:15:fa:83:
                    82:da:9c:5b:e5:83:aa:ff:17:a0:6f:23:71:72:1e:
                    d5:b5:60:c1:76:b9:01:57:9c:21:af:a9:32:03:e2:
                    e4:d2:89:e1:b5:b0:ff:10:84:3d:e9:84:88:89:da:
                    4a:f6:a3:36:fc:99:48:d1:da:56:78:29:0e:e2:b1:
                    ad:6f:11:3f:ea:b0:a9:88:13:7f:c3:01:e3:25:01:
                    47:2b:f9:3c:ba:04:db:47:1b:61:a5:b8:c1:f6:47:
                    99:e0:15:58:2b:74:c5:d6:21:43:c6:0d:f7:ee:38:
                    40:ef:39:ea:ef:5a:9a:79:84:04:3d:d7:0d:82:2d:
                    73:15:16:2b:db:6f:7f:4c:4c:ea:82:1f:c7:81:b0:
                    fe:83:4d:33:39:fe:7c:0d:ca:e0:ef:57:d8:e9:9e:
                    1f:85
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Subject Key Identifier:
                60:9A:09:78:46:3C:3D:3C:CF:17:43:30:0A:A9:0D:A5:DE:2A:83:5E
            X509v3 Authority Key Identifier:
                46:D1:67:D7:DB:2A:66:74:04:C4:56:5B:A1:43:AB:71:6F:BF:50:15
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        55:ce:8d:c7:2a:cc:f7:83:29:6b:53:ef:30:fd:c8:cf:0a:eb:
        cf:9f:41:fc:61:70:f2:77:b5:3f:78:09:8a:f5:62:a9:f5:6e:
        d7:41:e5:f4:97:f6:86:5b:26:94:d8:9c:8f:a0:aa:3f:de:3d:
        eb:3f:79:c4:1e:02:b8:8d:85:ab:b7:77:fa:97:e1:45:e7:57:
        0e:e3:df:4a:2f:82:9f:ee:82:af:be:13:7b:fd:e6:88:b3:5e:
        8b:87:24:95:76:6d:8b:32:8d:cb:c9:fa:cb:df:89:97:51:44:
        32:6d:4d:e3:44:24:60:cb:90:d1:55:ce:d1:2a:f8:72:d2:b7:
        d1:54:80:62:2e:e2:2b:54:79:48:28:68:ed:89:45:c9:e5:d7:
        28:af:f9:05:34:11:3e:82:1e:e8:7a:7f:df:b2:9b:2d:31:95:
        04:5e:c0:9e:08:4e:67:89:42:7d:0d:b2:e7:30:b3:30:49:65:
        f8:d2:51:97:5b:69:83:2d:3e:e2:98:37:f2:94:ef:a7:62:00:
        4c:03:23:bd:e2:65:47:05:2c:9a:70:48:5b:5b:9a:c7:56:bf:
        c6:ac:4e:8c:ac:60:20:06:97:de:cf:93:0e:ba:49:e2:3d:79:
        6b:85:ca:ef:2e:98:a8:0b:c9:a8:0a:73:11:f1:25:b4:d7:b2:
        31:70:a1:a8
```

3.6 Generate server key pair and create CSR

Generate server key:

```
tpm2tss-genkey -a rsa -s 2048 ./tls_tpm/tpm2/keys/server.key -p abc
```

Create server CSR:

```
openssl req -keyform tss -provider tpm2 -config ./tls_tpm/pki/openssl.cnf -key ./tls_tpm/tpm2
/keys/server.key -new -out ./tls_tpm/tpm2/csr/server.csr
```

3.7 Sign server certificate with intermediate CA

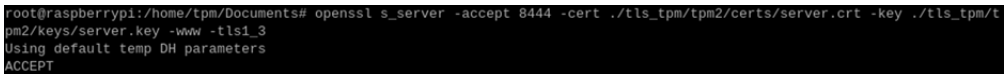
```
openssl x509 -req -days 365 -in ./tls_tpm/tpm2/csr/server.csr -CA ./tls_tpm/pki/private/intCA
.crt -CAkey ./tls_tpm/pki/private/intCA.key -CAcreateserial -out ./tls_tpm/tpm2/certs/server.
crt -extfile v3.ext
openssl verify -CAfile ./tls_tpm/pki/private/rootCA.crt -untrusted ./tls_tpm/pki/private/intC
A.crt ./tls_tpm/tpm2/certs/server.crt
```

4 Running a TLS 1.3 server and client

4.1 Start OpenSSL 3 TLS 1.3 server

```
openssl s_server -accept 8444 -cert ./tls_tpm/tpm2/certs/server.crt -key ./tls_tpm/tpm2/keys/
server.key -www -tls1_3 -keyform tss -provider tpm2
```

Figure 12. OpenSSL server



4.2 Connect OpenSSL 3 TLS 1.3 client

```
openssl s_client -connect 127.0.0.1:8444 -cert ./tls_tpm/tpm2/certs/client.crt -key ./tls_tpm
/tpm2/keys/client.key -tls1_3 -CAfile ./tls_tpm/pki/private/rootCA.crt -provider tpm2
```

5 Capturing and analyzing TLS traffic with TShark

Create new directory for TLS capture:

```
mkdir ./tls_tpm/tls_log
cd tls_tpm/tls_log
```

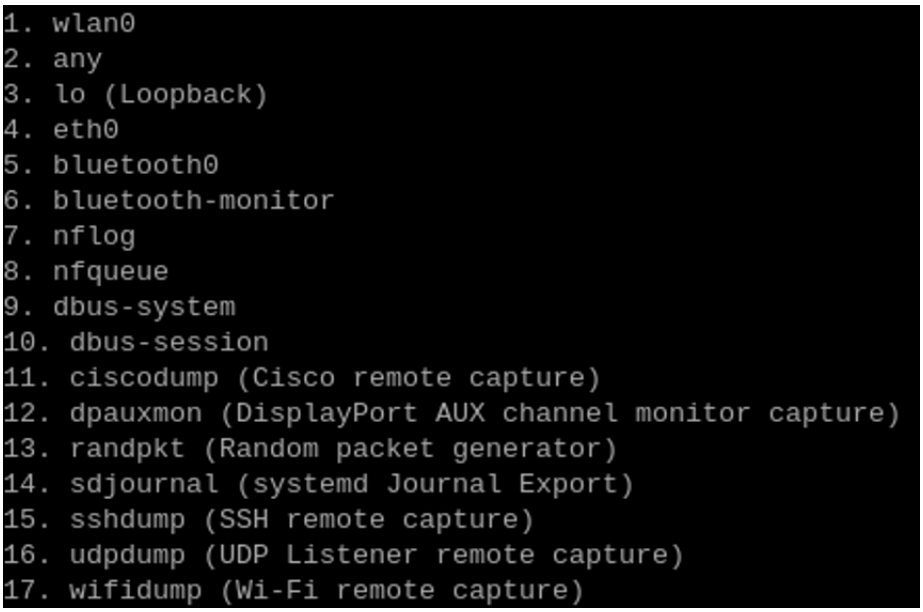
5.1 Install TShark

```
sudo apt-get update
sudo apt-get install tshark -y
tshark -v
```

5.2 List available network interfaces

```
sudo tshark -D
```

Figure 13. Network interfaces



5.3 Capture TLS traffic on loopback interface

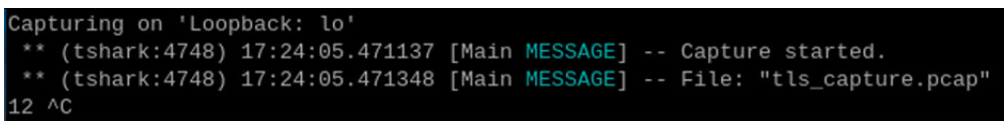
Open a new terminal and start capture:

```
sudo tshark -i 3 -w tls_capture.pcap
```

5.4 Run TLS client and server

Run the server and client commands to generate TLS traffic.
When it is done, stop the capture (Ctrl+C).

Figure 14. Capture TLS (12 captures)



5.5 Analyze captured TLS traffic

```
tshark -r tls_capture.pcap -V -x -o "tls.debug_file:ssldebug.log" -o "tls.desegment_ssl_records: TRUE" -o "tls.desegment_ssl_application_data: TRUE" -o "tls.keys_list:127.0.0.1,8444,http,server.pem" >> TLS_HandShake.log
```

Filter TLS handshake messages:

```
grep -A70 "TLS" TLS_HandShake.log
```

6 TLS algorithms

Table 2. **TLS1.3 algorithms**

Code TLS	Cryptographic suite
0x1302	TLS_AES_256_GCM_SHA384
0x1301	TLS_AES_128_GCM_SHA256
0x1304	TLS_AES_128_CCM_SHA256
0x1303	TLS_CHACHA20_POLY1305_SHA256

It is important to note that TLS 1.3 introduces a significantly different set of cryptographic algorithms compared to TLS 1.2. The cipher suites used in TLS 1.3 are streamlined and designed to provide stronger security guarantees and better performance by removing legacy and vulnerable algorithms.

For instance, when migrating from TLS 1.2 to TLS 1.3, the cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 is replaced by TLS_AES_256_GCM_SHA384 in TLS 1.3.

To run TLS1.3 with a specific cipher suite, the following command should be performed:

```
openssl s_server -accept 8444 - cert ./tls_tpm/tpm2/certs/server.crt -key ./tls_tpm/tpm2/keys
/server.key -www -tls1_3 -ciphersuites TLS_AES_256_GCM_SHA384
```

7 Uses cases

With the STSAFE-TPM, the algorithm secp521r1 is supported. Find the instruction to follow a TLS1.3 session with this algorithm in the section below.

7.1 ecdsa_sha512

Root CA generation:

```
openssl ecparam -name secp521r1 -genkey -noout -out tls_tpm/pki/private/rootCA.key
openssl req -config tls_tpm/pki/openssl.cnf -key tls_tpm/pki/private/rootCA.key -new -x509 -d
ays 365 -sha512 -extensions v3_ca -out tls_tpm/pki/private/rootCA.crt
```

Generation intermediate CA:

```
openssl ecparam -name secp521r1 -genkey -noout -out tls_tpm/pki/private/intCA.key
```

CSR signature:

```
openssl ca -config tls_tpm/pki/openssl.cnf -extensions v3_intermediate_ca -days 365 -notext -
md sha512 -in tls_tpm/pki/csr/intCA.csr -out tls_tpm/pki/private/intCA.crt
```

Verification of the certificate:

```
openssl verify -verbose -x509_strict -CAfile tls_tpm/pki/private/rootCA.crt tls_tpm/pki/priva
te/intCA.crt
```

Generation of the client key:

```
openssl ecparam -name secp521r1 -genkey -noout -out tls_tpm/tpm2/keys/client.key -provider tp
m2
```

Generation of the server key:

```
openssl ecparam -name secp521r1 -genkey -noout -out tls_tpm/tpm2/keys/server.key -provider tp
m2
export OPENSSL_CONF=openssl.cnf
```

Creation of the client CSR with the client key:

```
openssl req -keyform tss -provider tpm2 -config tls_tpm/pki/openssl.cnf -key tls_tpm/tpm2/key
s/client.key -new -out tls_tpm/tpm2/csr/client.csr -passin pass:abc
```

Signature of the client certificate:

```
openssl x509 -req -in tls_tpm/tpm2/csr/client.csr -CA tls_tpm/pki/private/intCA.crt -CAkey tl
s_tpm/pki/private/intCA.key -CAcreateserial -out tls_tpm/tpm2/certs/client.crt -days 365 -ext
file v3.ext -sha512
```

Verification of the client certificate:

```
openssl verify -CAfile tls_tpm/pki/private/rootCA.crt -untrusted tls_tpm/pki/private/intCA.cr
t tls_tpm/tpm2/certs/client.crt
```

Creation of the server CSR with the server key:

```
openssl x509 -req -in tls_tpm/tpm2/csr/server.csr -CA tls_tpm/pki/private/intCA.crt -CAkey tl
s_tpm/pki/private/intCA.key -CAcreateserial -out tls_tpm/tpm2/certs/server.crt -days 365 -ext
file v3.ext -sha512
```

Verification of the server certificate:

```
openssl verify -CAfile tls_tpm/pki/private/rootCA.crt -untrusted tls_tpm/pki/private/intCA.cr
t tls_tpm/tpm2/certs/server.crt
```

Run the OpenSSL server:

```
openssl s_server -accept 8444 -cert tls_tpm/tpm2/certs/server.crt -key tls_tpm/tpm2/keys/serv
er.key -www -tls1_3 -keyform tss -provider tpm2
```

Run the OpenSSL client:

```
openssl s_client -connect 127.0.0.1:8444  -cert ./tls_tpm/tpm2/certs/client.crt  -key ./tls_t
pm/tpm2/keys/client.key  -tls1_3  -CAfile ./tls_tpm/pki/private/rootCA.crt -provider tpm2
```

Figure 15. TLS client with ecdsa sha512 (part 1)

```
CONNECTED(00000008)
Can't use SSL_get_servername
depth=0 C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
verify error:num=26:unsuitable certificate purpose
verify return:1
depth=0 C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
verify error:num=21:unable to verify the first certificate
verify return:1
depth=0 C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
verify return:1
---
Certificate chain
 0 s:C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
   i:C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
   a:PKEY: id-ecPublicKey, 521 (bit); sigalg: ecdsa-with-SHA512
   v:NotBefore: Jul  4 00:33:22 2025 GMT; NotAfter: Jul  4 00:33:22 2026 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIC4TCCAkKgAwIBAgIUaQtROXOir9tjCr/qBs0o9OVmceswCgYIKoZIzj0EAwQw
dDELMAkGA1UEBhMCVVMxEzARBgNVBAgMCkNhbGlmb3JuaWExETAPBgNVBAcMCE1p
bHBpdGFzMRswGQYDVQQKDBJTVE1pY3JvZWx1Y3Ryb25pY3MxETAPBgNVBAsMCFN1
Y3VyaXR5MQ0wCwYDVQQDDAR0ZXN0MB4XDTI1MDcwNDAwMzMyMloXDTI2MDcwNDAw
MzMyMlowdDELMAkGA1UEBhMCVVMxEzARBgNVBAgMCkNhbGlmb3JuaWExETAPBgNV
BAcMCE1pbHBpdGFzMRswGQYDVQQKDBJTVE1pY3JvZWx1Y3Ryb25pY3MxETAPBgNV
BAsMCFN1Y3VyaXR5MQ0wCwYDVQQDDAR0ZXN0MIGbMBAGByqGSM49AgEGBSuBBAAj
A4GGAAQAFNsE89qKC88o3/7phJlaZAspGcN7nGTvndwKyWWJOEgvY7u7TLn+e2ox
YVvrOEawpcqiACRvk34TaJw9Kh/84MQBqiEyS7+Xyp8lIARTzM0vTBb2L7bj1MQJ
7vWwVxPEwo1xPaAEE6zw66lrlnh1ncfhDguW1c91lYqANjuisE0GtJmjbzBtMAkG
A1UdEwQCMAAwCwYDVR0PBAQDAgWgMBMGA1UdJQQMMAoGCCsGAQUFBwMCMB0GA1Ud
DgQWBBTNZIJ0DIOeIJrivDNxG4KOKniu4TAfBgNVHSMEGDAWgBTNOeY129KClIFT
ThZugelBiTGkrzAKBggqhkjOPQQDBAOBjAAwgYgCQgCpfiYVHH/S11zTHJ8iPvSN
zbnceHDI6LqB+AIB2fGUjNvk46HvQSOooNKzf1mK4xnsAyZCyHvT6JQ6hIykNglu
ugJCAPwCgBKOZY/Mn/ny0kjwJSWYpIMl6V0Vzgc5uv/q4a/R5tq2chbwXsNedqWO
tHLMYCuZcb82ajSN4qUtqQ7sijQ7
-----END CERTIFICATE-----
subject=C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
issuer=C = US, ST = California, L = Milpitas, O = STMicroelectronics, OU = Security, CN = test
---
No client certificate CA names sent
Peer signing digest: SHA512
Peer signature type: ECDSA
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 1180 bytes and written 305 bytes
Verification error: unable to verify the first certificate
---
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 521 bit
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 21 (unable to verify the first certificate)
---
---
Post-Handshake New Session Ticket arrived:
```

Figure 16. **TLS client with ecdsa sha512 (part 2)**

```
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: 0DC93389F103A844A01F9E9BD78F34D3280147A1A9BF2143E426EAD555BD7533
    Session-ID-ctx:
    Resumption PSK: CE6DBEF1FFDB29EC14868E5227314B135C3055E1DA37873B45741CAEA546BBADEDAF5ED937ABE8FD222914269A4CB4D7
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 7200 (seconds)
    TLS session ticket:
    0000 - 0c 50 02 b8 12 44 d0 5a-d4 80 1e 67 df 54 ce d5   .P...D.Z...g.T..
    0010 - 0b 3b 27 05 65 d3 cf 64-4f 32 a7 a2 75 10 1c c4   .;'.e..dO2..u...
    0020 - 49 f3 6a b5 c1 1e 41 48-0b 56 57 df 94 56 b3 8b   I.j...AH.VW..V..
    0030 - 15 54 a4 f2 e7 65 5b f8-81 7e 1d 65 cd e1 62 68   .T...e[..~.e..bh
    0040 - f4 73 3b c2 69 21 01 d0-72 08 47 f7 29 6e bb 92   .s;.i!..r.G.)n..
    0050 - b3 5b ae 73 7d b2 37 1b-e3 15 bc 49 5f 86 95 83   .[.s}.7....I_...
    0060 - d7 3c 65 8d ae 71 30 39-09 62 a7 11 ba 24 01 c7   .<e..q09.b...$..
    0070 - 43 9c e7 b3 e0 e8 bc 2a-9c 03 64 52 95 0e 56 f1   C......*..dR..V.
    0080 - 7c e6 78 13 be f4 c9 04-8b e6 50 64 f4 22 bb 8a   |.x.......Pd."..
    0090 - 14 7a d3 2d af 12 ed 5a-55 bb 95 2b e9 25 4d 94   .z.-...ZU..+.%M.
    00a0 - d9 00 d8 44 9c fd b7 90-65 d1 e2 39 01 87 ee ea   ...D....e..9....
    00b0 - 43 af 21 93 b8 72 5e e6-13 4b e4 8d 83 03 bb f8   C.!..r^..K......
    00c0 - b8 3d 9a 41 a2 c3 63 54-c2 da 0e e1 07 e4 72 6a   .=.A..cT......rj

    Start Time: 1751589209
    Timeout   : 7200 (sec)
    Verify return code: 21 (unable to verify the first certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: F01A75722D477C2C7CF0327FD70B07FCFCF99BFF03F7D2884A61C25F46FCCAE2
    Session-ID-ctx:
    Resumption PSK: AD7CC83F5D510DDBA004C97BD99F3B3E61F8D034644EBF14F26FDF55FE15FA08408D51F870BC98930C268F5C5D5B9E14
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 7200 (seconds)
    TLS session ticket:
    0000 - 0c 50 02 b8 12 44 d0 5a-d4 80 1e 67 df 54 ce d5   .P...D.Z...g.T..
    0010 - a0 2f 9f 10 2c 7c 8d 43-4e a4 56 2b f3 75 b1 7b   ./..,|.CN.V+.u.{
    0020 - 88 dc 40 74 4d 2b 3c 7d-be b0 64 68 f6 ec 7b 3e   ..@tM+<}..dh..{>
    0030 - e7 16 ce a9 54 95 54 eb-6e 39 d7 d1 92 a8 f5 f0   ....T.T.n9......
    0040 - f5 e2 98 0d 7b 82 8a a8-8b 67 6a 3c 83 8a 29 b8   ....{....gj<..).
    0050 - 90 d3 4f 71 00 91 4c 6a-4b de db a3 19 a1 24 04   ..Oq..LjK.....$.
    0060 - bb 2c 34 8a 32 ff 24 d8-8d 1d 8c b3 7b 89 c4 3f   .,4.2.$.....{..?
    0070 - 6f e6 85 ff 49 5f 10 63-38 b5 78 84 d2 6c fd 2f   o...I_.c8.x..l./
    0080 - f6 67 7f a5 ce 75 dc 85-6b da a6 b5 25 e3 ec 7d   .g...u..k...%..}
    0090 - 1c e2 ba 8b 45 a6 38 f8-7f f3 a0 d3 0e aa a7 73   ....E.8........s
    00a0 - 89 68 bb 58 8a 6b f5 81-52 be 23 df 2b c9 4b 68   .h.X.k..R.#.+.Kh
    00b0 - 90 8f 4c 90 68 20 a3 2d-7c b5 fd 23 c5 11 b1 63   ..L.h .-|..#...c
    00c0 - 36 a0 d9 8b 72 2e 1b 00-73 bd 54 93 e3 e7 ad f0   6...r...s.T.....

    Start Time: 1751589209
    Timeout   : 7200 (sec)
    Verify return code: 21 (unable to verify the first certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
```

Revision history

Table 3. **Document revision history**

Date	Revision	Changes
20-Nov-2025	1	Initial release.

Contents

List of figures

List of tables