Getting started with the STSW-STSA110-SSL software package

Introduction

This user manual explains how to open access to the STSAFE-A110 secure element with the STSAFE-A *OpenSSL*® software package (STSW-STSA110-SSL). This package provides a Linux® driver to the STSAFE-A110 solution.

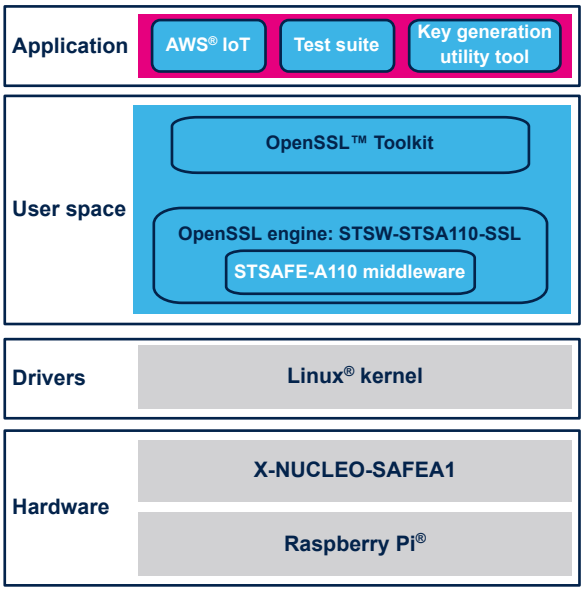
The STSAFE-A110 is a highly secure solution that acts as a secure element providing authentication and secure data management services to a local or remote host. It consists of a full turnkey solution with a secure operating system running on the latest generation of secure microcontrollers.

The STSW-STSA110-SSL software package can be used as an OpenSSL engine (hardware support) or a C library for any Linux application utilizing the STSAFE-A110 hardware

The software package contains:

- the OpenSSL engine as described in Section 5.1 Building the STSW-STSA110-SSL OpenSSL engine
- two test applications (see Section 5.2 Building the STSW-STSA110-SSL stsafe_engine_test_suite test suite and Section 5.3 Building the STSAFE-A key generation utility), which illustrate how to integrate the OpenSSL engine
- and an example, which demonstrates cloud connectivity with Amazon™ (see Section 5.4 STSAFE-A110 securing a connection with AWS IoT).

Figure 1. STSW-STSA110-SSL architecture



1 Features

The software is provided as source code under an ST software license agreement (SLA0088).

The STSW-STSA110-SSL software package:

- is compliant with the ENGINE cryptographic module support of *OpenSSL*®.
- connects to the user's AWS™ IoT account over a secure TLS connection.
- extends the OpenSSL toolkit's cryptographic features thanks to the use of the STSAFE-A110 solution.
- provides a test suite of 15 tests for:
  - Query functions to retrieve the product information
  - Envelop wrapping/unwrapping
  - ECDSA signature/verification
  - ECDH generation of ephemeral keys
  - Reading the CA certificate from the STSAFE-A110 device
  - Random number generation
  - Secure storage

  For the complete list of tests, refer to Section 5.2.1 stsafe_engine_test_suite.
- includes a key generation utility tool for the STSAFE-A110 solution

This product includes software developed by the OpenSSL Project for use in the OpenSSL toolkit (http://www.openssl.org/).

Note: OpenSSL is a registered trademark owned by the OpenSSL Software Foundation.

2 Setting up the development environment

To set up the development environment, follow the sequence below.

Step 1. Download the following from https://www.openssl.org/.

- `openssl-1.1.1g.tar.gz`
- `openssl-1.1.1g.tar.gz.sha256`

Step 2. Set up a work directory on the Raspberry Pi® such as:
`/home/pi/projects`

Step 3. Decompress the STSW-STSA110-SSL code into this directory, then build the code and install it in `/opt/openssl` so that it does not interfere with the existing system install of OpenSSL.

Step 4. Build the install:

```
./config --prefix=/opt/openssl --openssldir=/usr/local/ssl
make
make test
sudo make install
```

Step 5. On first use of the installed OpenSSL version, do as follows to set the path and the library path:

```
export PATH=/opt/openssl/bin:${PATH}
export LD_LIBRARY_PATH=/opt/openssl/lib
```

3 Code tree description

This section shows the tree structure of the STSW-STSA110-SSL software package.

```
STSAFE-A_OpenSSL_Engine
├── Documentation
├── Examples
│   ├── iot_openssl_test
│   ├── stsafe_engine_test_suite
│   └── stsafe_genkey
├── lib
│   ├── STSAFE_Axx0
│   ├── CoreModules
│   │   ├── Inc
│   │   └── Src
│   ├── _htmresc
│   ├── Interface
├── Licenses
├── inc
├── src
```

Note: The `lib` folder contains the STSAFE-A110's API interface, also called STSAFE-A110 middleware.
The `src` and `inc` folders contain the API of the STSW-STSA110-SSL OpenSSL engine.

4 Setting up the hardware environment

This section describes the hardware components needed for developing a secure application.

4.1 STSAFE-A1xx expansion board

The hardware environment includes the STSAFE-A1xx expansion board (X-NUCLEO-SAFEA1).
The figure below illustrates the board.

Figure 2. STSAFE-A1xx expansion board

4.2 Raspberry Pi® model board

A Raspberry Pi® model board is also required as part of the hardware environment.
Information about Raspberry Pi (RPi) boards is available at: https://www.raspberrypi.org/. The figure below provides an illustration of this type of board.

Figure 3. Raspberry Pi board

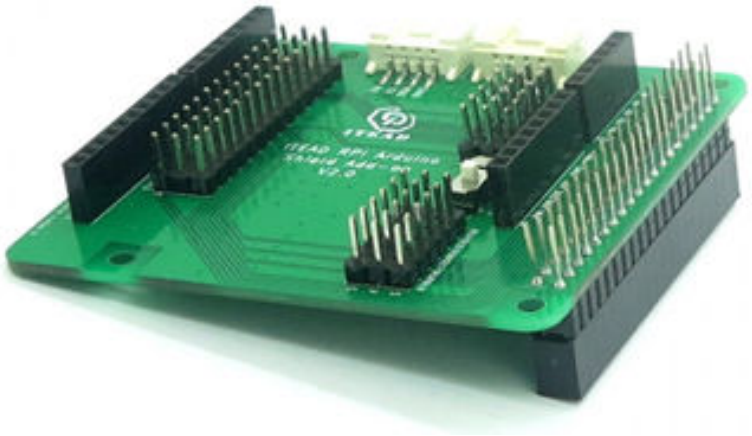
4.3 RPi to ARDUINO® connector shield add-on V2.0 (optional)

The use of this connector shield add-on in the hardware setup is optional.

An example of connector could be the ITEAD RPi ARDUINO shield add-on V2.0. The figure below depicts this connector shield.

Note: This board is optional because for prototyping, it is possible to connect the STSAFE-A expansion board (X-NUCLEO-SAFEA1) to the Raspberry Pi board using wires.

Figure 4. ITEAD RPi ARDUINO shield add-on V2.0



4.4 Hardware setup

This section describes the two hardware setup options: with or without the ARDUINO connector shield.

The first two figures illustrate the first option, that is with the ARDUINO connector shield, whereas the last image shows the setup with wires.

Figure 5. Example with the STSAFE-A1xx expansion board on an RPi board using the ARDUINO shield

Figure 6. **STSAFE-A1xx expansion board on an RPi board using the ARDUINO shield, with connections shown**

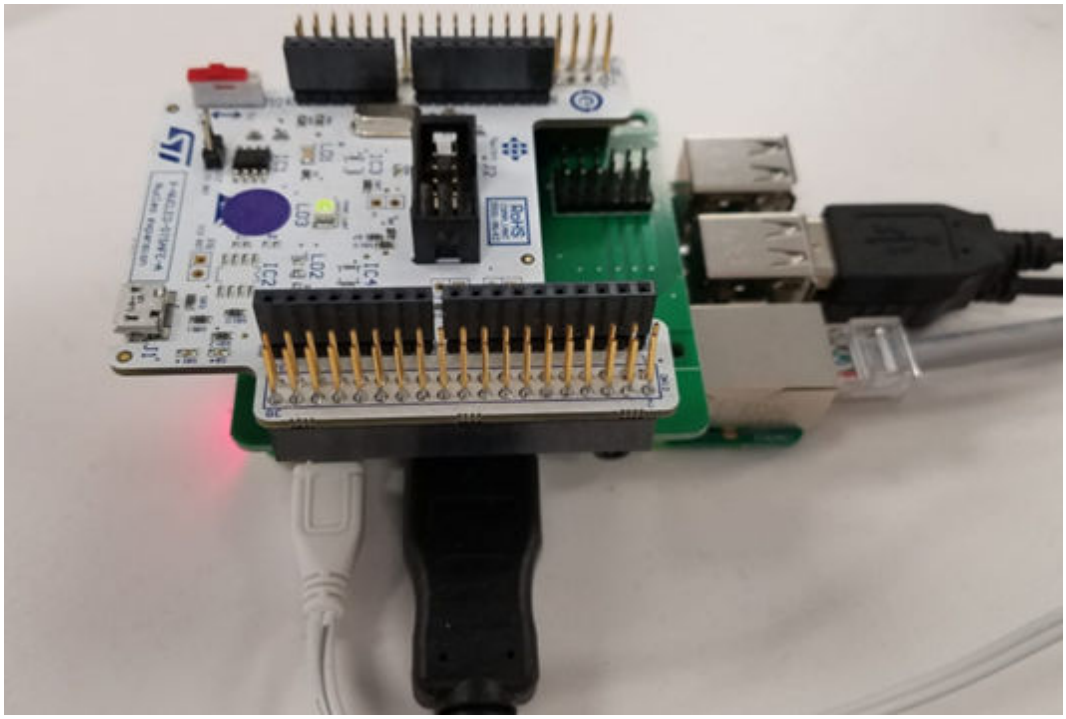
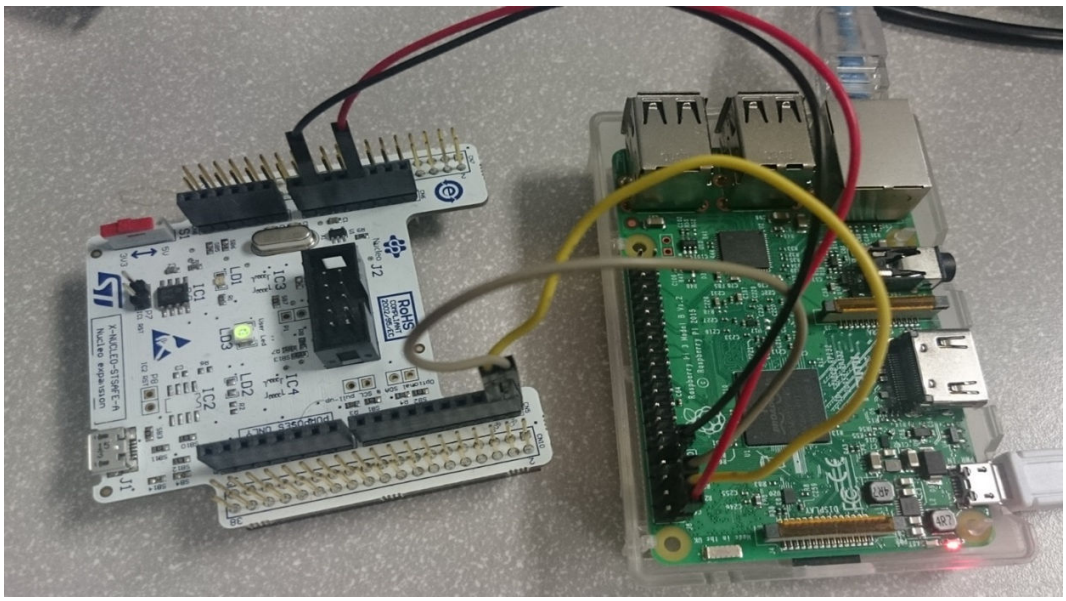


Figure 7. **STSAFE-A1xx expansion board on an RPi board using jumper wires**



5 Building the OpenSSL engine and example applications

5.1 Building the STSW-STSA110-SSL OpenSSL engine

To build the STSW-STSA110-SSL, copy the provided code to a suitable directory: `/home/pi/projects/STSAFE-A_OpenSSL_Engine/`. The `make.inc` file defines where the OpenSSL toolkit is installed.

By default the `make.inc` file has the following set for OpenSSL:

```
# Openssl directory setup
OPENSSL_INC = /opt/openssl/include
OPENSSL_LIB = /opt/openssl/lib
OPENSSL_BIN = /opt/openssl/bin
```

If OpenSSL is installed in the default directories, then either comment out this block or remove the directory name from the variables, the `make.inc` will then set the variables to:

```
OPENSSL_INC = /usr/include/openssl
OPENSSL_LIB = /usr/lib
OPENSSL_BIN = /usr/bin
```

To build the engine, run:

```
make
```

Then, to install the built library files:

```
sudo make install
```

After the build operation, create or modify the `openssl.conf.stsafe` file and save it:

```
openssl_conf = openssl_def

[openssl_def]
engines = engine_section

[engine_section]
Stsafe = Stsafe_section

[Stsafe_section]
dynamic_path = /opt/openssl/lib/engines-1.1/Stsafe.so
engine_id = Stsafe
default_algorithms = ALL
init = 1
```

To ease the usage of the OpenSSL engine, create an alias:

```
alias
stsafessl='OPENSSL_CONF=/home/pi/projects/STSafe_OpenSSL/OpenSSL/test/
openssl.conf.stsafe openssl'
```

To test the operation with the existing STSAFE-A engine, run:

```
$ stsafessl engine Stsafe

ENGINE> bind_helper: Engine id = Stsafe
ENGINE> bind_helper: ENGINE_set_id completed
ENGINE> bind_helper: ENGINE_set_name completed
ENGINE> bind_helper: ENGINE_set_init_function completed
ENGINE> bind_helper: ENGINE_set_RAND completed
ENGINE> bind_helper: ENGINE_set_ctrl_function completed
ENGINE> bind_helper: ENGINE_set_cmd_defns completed
stsafe_get_EC_methods called
EC_KEY_METHOD_set_sign.
EC_KEY_METHOD_set_verify.
EC_KEY_METHOD_set_keygen.
EC_KEY_METHOD_set_compute_key.
ENGINE> bind_helper: ENGINE_set_EC completed
ENGINE> bind_helper: ENGINE_set_load_pubkey_function completed
ENGINE> bind_helper: ENGINE_set_load_privkey_function completed
stsafe_pkey_meth_init called
stsafe_pkey_meth_init finished
ENGINE> bind_helper: stsafe_pkey_meth_init completed
ENGINE> bind_helper: ENGINE_set_pkey_meths completed
ENGINE> bind_helper: calling Engine_set_finish_function
ENGINE> bind_helper: ENGINE_set_finish_function completed
ENGINE> bind_helper: calling ENGINE_set_default
Using Openssl: OpenSSL 1.1.1g 21 Apr 2020
set host keys.
James STSAFE-A100 StSafeA_CreateHandle = 5, pStSafeA->InOutBuffer = 8000000,
        pStSafeA->InOutBuffer.LV.Data = 176aa84
StSafeA_GetDataBufferSize(): 523
About to call StSafeA_LocalEnvelopeKeySlotQuery: 12323, 12334, 22345
StSafeA_BuildCommandHeaderCMAC returns: 14
StSafeA_MAC_SHA_PreProcess called. Statuscode = 0
pStSafeA->HashObj.HashCtx = 0.
StSafeA_MAC_SHA_PostProcess called. Statuscode = 0
pStSafeA->HashObj.HashCtx = 0.
StSafeA_LocalEnvelopeKeySlotQuery: 0 slot 1: presence flag =1
---HostKeySlot = 7600312d, pStSafeA->InOutBuffer.LV.Data = 7edb464c
StSafeA_BuildCommandHeaderCMAC returns: 14
StSafeA_MAC_SHA_PreProcess called. Statuscode = 0
pStSafeA->HashObj.HashCtx = 0.
StSafeA_MAC_SHA_PostProcess called. Statuscode = 0
pStSafeA->HashObj.HashCtx = 0.
HostKeySlot->HostKeyPresenceFlag: 1
Main : stsafe_pairing success

**********^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Setting STSAFE-A110 host keys
set host keys.

**********vvvvvvvvvvvv***************
stsafe_pkey_meths called nid=0
ENGINE> bind_helper: ENGINE_set_default completed
(Stsafe) STSAFE-A110 engine for OpenSSL
stsafe_pkey_meths called nid=0
stsafe_pkey_meths called nid=408
stsafe_pkey_meths called nid=0
```

5.2 Building the STSW-STSA110-SSL stsafe_engine_test_suite test suite

This section explains the principle of the stsafe_engine_test_suite test suite and how it is run.

5.2.1 stsafe_engine_test_suite

The Examples directory of the STSW-STSA110-SSL software package includes a test application called stsafe_engine_test_suite, which can be used to conduct unitary tests for the OpenSSL engine. These tests can serve as an example of an application that utilizes STSAFE-A110 as well as the library directly.

List of testing features:

- Test 1 STSAFE Load Engine
- Test 2 STSAFE Engine Init
- Test 3 STSAFE Get Product Data
- Test 4 STSAFE Wrap Data
- Test 5 STSAFE Unwrap Data
- Test 6 STSAFE ECDSA Sign/Verify
- Test 7 STSAFE ECDH/Generate Ephemeral Keys
- Test 8 STSAFE Private Key Methods
- Test 9 STSAFE Random Number Generation
- Test 10 STSAFE Zone Data Read/update Test
- Test 11 STSAFE Query Test
- Test 12 STSAFE ECHO Test
- Test 13 Verify Password Test
- Test 14 Reset Test
- Test 15 Hibernate Test

For more details concerning what is tested within stsafe_engine_test_suite, refer to Section Appendix A stsafe_engine_test_suite execution log.

5.2.2 How to run the stsafe_engine_test_suite test suite

The stsafe_engine_test_suite test suite is located in the STSAFE-A_OpenSSL_Engine/Examples/stsafe_engine_test_suite directory.

A prerequisite to run the test suite is to compile it and link it to the OpenSSL engine. To do so, execute the following command:

```
cd <Directory where engine_is_installed>/STSAFE-A_OpenSSL_Engine/Examples/
stsafe_engine_test_suite

./make
./test_stafe_engine
```

The test suite is now ready to use! (See Section Appendix A stsafe_engine_test_suite execution log for the execution log of the test suite.)

stsafe_engine_test_suite comprises 15 tests (see previous section) that are run in sequence. This means that when a test is passed successfully, the suite jumps to the next test and so on. If a test fails, the test sequence is aborted.

The shared library that was built as specified in Section 5.1 Building the STSW-STSA110-SSL OpenSSL engine can also be used as a shared library for the test suite. The Stsafe.so and libStsafe.so files are located in the engines-1.1 directory.

The Examples directory provides an example of how to link the software library to the stsafe_engine_test_suite test suite. Refer to the makefile file for details.

The Examples directory also shows how to build an application using the shared library.

The STSAFE-A_OpenSSL_Engine\inc\stsafe_api.h file lists the available API functions.

5.3 Building the STSAFE-A key generation utility

The STSW-STSA110-SSL software package includes the source code to build a utility tool that allows keys to be generated in the STSAFE-A110 device, and makes them accessible through the STSW-STSA110-SSL OpenSSL engine. The source code of this utility tool is in the `STSAFE-A_OpenSSL_Engine/Examples/Stsafe_Genkey` directory.

To build the utility, use:

```
make
sudo make install
```

By default, the built utility is installed in `$(OPENSSL_BIN)/.`, and locally in `STSAFE-A_OpenSSL_Engine/Examples/Stsafe_Genkey/bin`.

Usage examples:

1. To get help:

```
$ ./stsafe_genkey --help
```

Usage: [options] `<filename>`

Arguments:

`<filename>`: storage for the public key

Options:

`-c, --curve`: curve for ECC (default: `nist_p256`)

`-h, --help`: print help

`-s, --slot`: slot to use for key generation (default slot 0)

`-v, --verbose`: print verbose messages

The available curves are:

- `nist_p256`
- `nist_p384`
- `brainpool_p256`
- `brainpool_p384`

The available private key slots are: 0, 1.

Note: Private key slot 255 is used for ephemeral keys.

2. To create a new EC key pair using private key slot 1:

```
./stsafe_genkey -s 1 Device-Pub.pem
```

This is the trace of the generation of a key pair:

```
pi@raspberrypi:~/projects/STSAFE-A_OpenSSL_Engine/Examples/stsafe_genkey/bin$ ./stsafe_genkey -s 1
Device-Pub.pem
ENGINE> bind_helper: Engine id = Stsafe
ENGINE> bind_helper: ENGINE_set_id completed
ENGINE> bind_helper: ENGINE_set_name completed
ENGINE> bind_helper: ENGINE_set_init_function completed
ENGINE> bind_helper: ENGINE_set_RAND completed
ENGINE> bind_helper: ENGINE_set_ctrl_function completed
ENGINE> bind_helper: ENGINE_set_cmd_defns completed
stsafe_get_EC_methods called
EC_KEY_METHOD_set_sign.
EC_KEY_METHOD_set_verify.
EC_KEY_METHOD_set_keygen.
EC_KEY_METHOD_set_compute_key.
ENGINE> bind_helper: ENGINE_set_EC completed
ENGINE> bind_helper: ENGINE_set_load_pubkey_function completed
ENGINE> bind_helper: ENGINE_set_load_privkey_function completed
stsafe_pkey_meth_init called
stsafe_pkey_meth_init finished
ENGINE> bind_helper: stsafe_pkey_meth_init completed
ENGINE> bind_helper: ENGINE_set_pkey_meths completed
ENGINE> bind_helper: calling Engine_set_finish_function
ENGINE> bind_helper: ENGINE_set_finish_function completed
ENGINE> bind_helper: calling ENGINE_set_default
Using Openssl : OpenSSL 1.1.1g 21 Apr 2020
STSAFE-A110 StSafeA_CreateHandle = 5, pStSafeA->InOutBuffer = 0x76b6a68c, pStSafeA-
>InOutBuffer.LV.Data = 0x76b6a430
************vvvvvvvvvvvv**************
stsafe_pkey_meths called nid=0
ENGINE> bind_helper: ENGINE_set_default completed
stsafe_cmd_ctrl in ACTION!!! cmd = 203
ENGINE> stsafe_cmd_ctrl: Setting STSAFE generate key slot to 1
STSAFE-EC> stsafe_ec_generate_key called.
STSAFE-EC> stsafe_ec_generate_key: Using Slot 1
STSAFE-EC> stsafe_ec_generate_key: Curve prime256v1 -> STSAFEA_NIST_P_256
STSAFE-EC> stsafe_ec_generate_key: X:Length 32 Data 0xdc 0x10 0xd3 0x99 0xb6 0xa2 0xbc 0x7f 0xbe 0x3c
0xe2 0x6b 0x0a 0x68 0x99 0x5d 0x69 0x7b 0x38 0xa3 0x60 0x49 0xdd 0xdd 0x74 0x49 0x7e 0x6d 0x00 0x0e
0xc1 0xf5
STSAFE-EC> stsafe_ec_generate_key: Y:Length 32 Data 0xe9 0x96 0xcf 0x62 0xd5 0x01 0x35 0x4c 0xb3 0x6d
0xf7 0xca 0xd9 0xc4 0xa2 0x3a 0x9b 0x05 0x86 0xba 0xad 0x0e 0x7b 0x3a 0x98 0x85 0xc1 0x50 0x12 0x0b
0xdb 0x5c
DC10D399B6A2BC7FBE3CE26B0A68995D697B38A36049DDDD74497E6D000EC1F5
E996CF62D501354CB36DF7CAD9C4A23A9B0586BAAD0E7B3A9885C150120BDB5C
```

5.4 STSAFE-A110 securing a connection with AWS IoT

5.4.1 Principle

This section describes how to use STSW-STSA110-SSL OpenSSL engine with the AWS IoT C-SDK for testing the available features of the STSW-STSA110-SSL OpenSSL engine and integration with AWS IoT.

This section shows a secure MQTT connection established using a TLS connection with AWS IoT.

The prerequisites are that both the STSW-STSA110-SSL engine library and the key generation utility have been built and installed (see Section 5.1 Building the STSW-STSA110-SSL OpenSSL engine and Section 5.3 Building the STSAFE-A key generation utility, respectively).

5.4.2 AWS IoT C-SDK

The SDK used is `AWS IoT Device SDK for Embedded C` version `v4_beta_deprecated` available from: https://github.com/aws/.

The AWS IoT device SDK for embedded C (C-SDK) is a collection of C source files under the MIT open source license that can be used in embedded applications to securely connect IoT devices to AWS IoT Core™. It contains MQTT client, HTTP client, JSON parser, AWS IoT device shadow, AWS IoT jobs and AWS IoT device defender libraries. This SDK is distributed in source form, and can be built into customer firmware along with application code, other libraries and an OS of your choice.

Note: This version is used because it provides support for OpenSSL when building for Linux.

Follow the instructions on the GitHub website to download and build the code:

```
cd <Your directory for installation>
git clone https://github.com/aws/aws-iot-device-sdk-embedded-C.git
cd aws-iot-device-sdk-embedded-C
git checkout v4_beta_deprecated
git pull
mkdir build
cd build
```

Do not start the compiling process yet. As part of the testing with or without an OpenSSL engine, the software needs three files; these are:

- `IOT_DEMO_ROOT_CA`, which is the `awsRootCA.pem` file
- `IOT_DEMO_CLIENT_CERT`, which is the `New-Device-Cert.pem` file
- `IOT_DEMO_PRIVATE_KEY`, which is the `temp.pem` file

5.4.3 Setting up the certificates and keys for AWS testing

Both the CA certificate and the **device certificate** are required to establish a TLS connection. In order to connect to the AWS server, both certificates need to be registered on your AWS IoT account. Refer to Section 5.4.3.3 Registering your CA certificate and device certificate on the AWS server for information on how to proceed.

5.4.3.1 Generating a CA root certificate and getting it registered

If you already have a CA certificate registered on the AWS server, then you can continue to use this CA certificate, and refer directly to the next section (Section 5.4.3.2 Device certificate generation).

Otherwise, create you own CA certificate by launching the command hereunder:

```
openssl ecparam -name prime256v1 -genkey -out root_CA_private.key
openssl req -x509 -new -nodes \
    -key root_CA_private.key \
    -sha256 -days 1024 \
    -out rootCA.pem
```

5.4.3.2 Device certificate generation

This generation method is used when the private key `slot 0` and `memory region 0` are matched. This is the default configuration of samples delivered with personalization SPL2 (see application note AN5435 available from https://www.st.com).

This method assumes that the X-NUCLEO-SAFEA1 expansion board is fresh out of the box and has only been paired, which happens when the STSW-STSA110-SSL software has just been loaded to the board or the provided tests are run. It should be noted that the keys used for pairing are fixed and well known in the software.

Step 1. Retrieve the existing certificate from the board by doing the following:

```
openssl engine Stsafe -t -post GET_DEVICE_CERT:Device-Cert.pem
```

Step 2. Retrieve the public key by doing the following:

```
openssl x509 -noout -in Device-Cert.pem -pubkey -out Device-Pub.pemopenssl
x509 -noout -in Device-Cert.pem -subject
```

The subject line will look as follows:

```
subject=C = FR, O = STMicroelectronics, CN = STSAFE-A110 EVAL2
```

Step 3. Create the following configuration files with the content below:

Filename: **`New-CSR.cfg`**

```
[ req ]
prompt = no
encrypt_key = no
string_mask = utf8only
default_md = sha256
distinguished_name = req_distinguished_name

[ req_distinguished_name ]
C = US
O = OEM
CN = < Copy CN Line From Above With No Spaces >
```

and

Filename: **`Cert-v3.cfg`**

```
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
```

Step 4. Create a temporary key pair and then create the new CSR. (Note: The CN field should not contain any space.) The `-subj` line can be left off if you use the filled in CN in the configuration file.

```
openssl ecparam -name prime256v1 -genkey -out temp.pem
openssl req -key temp.pem -new -config New-CSR.cfg -subj
"/CN=STSAFE-A110_EVAL2" -out New-Device-Cert.csr
```

Step 5. Create a new **device certificate** signed by the **OEM CA**.

```
openssl x509 -req -in New-Device-Cert.csr -CA rootCA.pem -CAkey
root_CA_private.key \
-days 500 -sha256 -CAcreateserial -force_pubkey \
Device-Pub.pem -extfile Cert-v3.cfg -out New-Device-Cert.pem
```

Now when running the AWS tests as detailed below, you can use the new `New-Device-Cert.pem` and `temp.pem` as your certificate and private key. (Note: The current version of OpenSSL requires these files.)

At the end of this Step, you have created a device certificate that use the private key stored in slot 0 of the STSAFE-A110 solution. The next step is to publish this certificate named `New-Device-Cert.pem` on your AWS account.

5.4.3.3 Registering your CA certificate and device certificate on the AWS server

The prerequisite is to have created and signed an AWS account on https://aws.amazon.com/iot/.

To use your own X.509 certificates, you must register a CA certificate with AWS IoT.

To do so, you must prove that you own the private key associated with the CA certificate by creating a private key verification certificate. Once done, the CA certificate can be used to sign device certificates.

Step 1. Register your CA certificate.

Go to https://aws.amazon.com/iot/, then navigate to `Certificate Authorities/Select a CA/Register a certificate` to register a CA certificate using the below procedure.

These steps a to f summarize how to create a private key verification certificate to enable you to use the previously created CA certificate.

Step 1a. Generate a key pair for the private key verification certificate.

To do so, use the following command:

```
openssl genrsa -out verificationCert.key 2048
```

Step 1b. Copy the registration code.

This code is delivered by AWS when you register your CA certificate on your AWS account at `AWS IoT / Certificate Authorities / Select a CA / Register a certificate`.

Registration code example: `123456789123456789123456789123456789`

Step 1c. Create a CSR with this registration code.

To do so, use the following command:

```
openssl req -new -key verificationCert.key -out verificationCert.csr
```

Then put the registration code in the `Common Name` field.

`Common Name` (e.g. server FQDN or YOUR name) []:
`123456789123456789123456789123456789`

Step 1d. Use the CSR that was signed with the CA private key to create a private key verification certificate.

To do so, use the following command:

```
openssl x509 -req -in verificationCert.csr -CA rootCA.pem -CAkey
root_CA_private.key -CAcreateserial -out verificationCert.crt -days
500 -sha256
```

Step 1e. Upload the CA certificate (`rootCA.pem`).

Step 1f. Upload the verification certificate (`verificationCert.crt`).

Step 2. Register your device certificate.

To do so, follow the following steps:

Step 2a. Go to the AWS website: https://aws.amazon.com/`AWS IoT/Certificate/Create a certificate/Get started/Use my certificate`.

Step 2b. Select the CA certificate previously published in 1..

Step 2c. Click on the **Next** button, then **Select** the previously created certificate named `New-Device-Cert.pem` and click on **Open**.

The device certificate is now published on your AWS account.

5.4.3.4 *Setting up AWS IoT C-SDK*

After creating or obtaining the required certificates and keys, the user can now set up the build tree.

Step 1. In the `ports/posix/posix.cmake` file, locate and change the following lines:

```
# Link OpenSSL
set( PLATFORM_DEPENDENCIES OpenSSL::SSL OpenSSL::Crypto )
```

to:

```
# Link OpenSSL.
set( STSAFE_ENGINE ${_OPENSSL_LIBDIR}/engines-1.1/Stsafe.so )
set( PLATFORM_DEPENDENCIES OpenSSL::SSL OpenSSL::Crypto ${STSAFE_ENGINE} )
```

Step 2. Make sure that the `demos/iot_config.h` file has your endpoint and certificate files/keys setup. The private key should be the temporary key created using the instructions given in the previous section.
For example:

```
/* Credential paths. May be overridden with command line options at
runtime. */
#define IOT_DEMO_ROOT_CA      "/<path to>/AmazonRootCA1.pem" /* Command
line: -r */
#define IOT_DEMO_CLIENT_CERT  "/<path to>/New-Device-Cert.pem" /* Command
line: -c */
#define IOT_DEMO_PRIVATE_KEY  "/<path to>/temp.pem" /* Command line: -k */
```

Note: *`AmazonRootCA1.pem` can be downloaded from: https://www.amazontrust.com/repository/.*

You need also to set the values of the defines statements below:

```
#define IOT_DEMO_SERVER "<the demo server>.amazonaws.com"
#define IOT_DEMO_PORT ( 8883 )
#define IOT_DEMO_USER_NAME "<your aws account>"
#define IOT_DEMO_PASSWORD "<your password>"
```

Note: *You need to change IOT_DEMO_PORT to 8833 to enable mutual authentication, 443 being for anonymous https connection. You also need to set the IoT demonstration server by adding IOT_DEMO_SERVER to your `AWS account/Settings/Endpoint`.*

Step 3. Replace the `ports/common/src/iot_network_openssl.c` file by the one in the `test\iot_openssl_test` directory. You can see how the STSW-STSA110-SSL software is used in the TLS set up there.

Step 4. Follow the instructions to set up the build. When you are prompted to run `cmake`, use the following command line:

```
cd build
cmake .. -DIOT_NETWORK_USE_OPENSSL=1 -DCMAKE_BUILD_TYPE=Debug
-DOPENSSL_CRYPTO_LIBRARY=/opt/openssl/lib/libcrypto.so
-DOPENSSL_INCLUDE_DIR=/opt/openssl/include
-DOPENSSL_SSL_LIBRARY=/opt/openssl/lib/libssl.so
```

The build is now ready to run `make`.

Step 5. Once the build has finished, change directory to `output/bin` and run:

```
./iot_demo_mqtt
```

You should now be connected to your AWS IoT account over a secure TLS connection.
See Section Appendix B Trace of the connectivity to AWS IoT for the trace of the connectivity to AWS IoT.

6 Function description

The STSW-STSA110-SSL software kit provides the functions that are described in the next sections:

6.1 ECDSA signing

The STSW-STSA110-SSL provides the `ECDSA_do_sign` function to perform ECDSA signature.

API function:

```
int stsafe_ecdsa_sign(
    int type,
    const unsigned char *dgst,
    int dlen,
    unsigned char *sig,
    unsigned int *siglen,
    const BIGNUM *kinv,
    const BIGNUM *r,
    EC_KEY *eckey);
```

Note: The `EVP_PKEY_sign_init` function should also be called to check the validity of the keys.

6.2 ECDSA verification

The STSW-STSA110-SSL provides the `ECDSA_do_verify` function to perform ECDSA signature verification.

API function:

```
int stsafe_ecdsa_verify(
    int type,
    const unsigned char *dgst,
    int dgst_len,
    const unsigned char *sigbuf,
    int sig_len,
    EC_KEY *eckey);
```

6.3 ECDH key establishment

The STSW-STSA110-SSL provides the `ECDH_compute_key` function to perform ECDH(E) ephemeral key establishment (for example to run the STSAFE-A110's Establish Key command).

API function:

```
int stsafe_engine_ecdh_compute_key(
unsigned char **out,
size_t *outlen,
const EC_POINT *pub_key,
const EC_KEY *ecdh);
```

6.4 EC key generation

The STSW-STSA110-SSL provides the `EC_KEY_generate_key` function to generate a new EC key pair.

API function:

```
int stsafe_ec_generate_key(EC_KEY *eckey);
```

6.5 Envelope wrapping/unwrapping

The STSAFE-A110 solution wraps/unwraps (AES encryption/decryption) a local envelope (data blob) in order to securely store a secret to any non-volatile memory (NVM), like local Flash memory or the STSAFE-A110 user data memory region.

The wrapping mechanism is used to protect a secret or plain text. The output of wrapping is an envelope encrypted with an AES key wrap algorithm that contains the secret or plain text to be protected.

The STSAFE-A110 solution supports two local envelope key slots (0,1). These key slots have to be populated before warp and unwrap functions can be used. The `stsafe_pairing()` function performs this process if the key slots have not already been populated.

The STSAFE-A110 solution provides the following APIs:

```
int stsafe_AES_wrap_key(
    unsigned char keyslot,
    unsigned char *out,
    const unsigned char *in,
    unsigned int inlen);

 int stsafe_AES_unwrap_key(
    unsigned char keyslot,
    unsigned char *out,
    const unsigned char *in,
    unsigned int inlen);
```

6.6 Host pairing

To protect communications over the I²C bus and pair the STSAFE-A110 solution with a host processor. The STSAFE-A110 supports two 128-bit keys called the *host MAC key* and the *host cipher key*.

Note: The host processor must store and protect these keys in NVM or file system in a protected manner. The pairing function synchronizes up the local keys with the STSAFE-A110 hardware.

These pairing keys can be programmed once only into the STSAFE-A110 solution, and cannot be read back from the STSAFE-A110.

The developer can choose to use their own keys for the programming (not enabled in the current release, refer to the `StSafeA_HostKeys_Init()` function in the X-CUBE-SAFEA1 code).

By default, for development, static known keys are used to aid debugging and development.

Note: The development static keys must NOT be used in a product.

Pairing API function available in the STSAFE-A110:

```
int32_t stsafe_pairing(void);
```

Note: This is not an STSW-STSA110-SSL function.

6.7 Secure storage

These functions are used to read or update (write) the STSAFE-A110 solution's memory regions (secure storage). When the data partition zone of the STSAFE-A110 solution has a one-way counter, these functions can also be used to decrease this counter.

```
int stsafe_read_zone(
    int zone_index,
    int offset,
    int length,
    unsigned char *data_buff);

int stsafe_update_zone(
    int zone_index,
    int offset,
    int length,
    unsigned char *data_buff);

int stsafe_zone_decrement(
    int zone_index,
    int offset,
    int amount,
    unsigned char *indata_buffer,
    int indata_length,
    unsigned char *outcounter);
```

Note: These are not STSW-STSA110-SSL functions.

6.8 Query

This feature is used to gather all the available attributes from the STSAFE-A110 solution. It is specified in Table 1. Control command parameters.

Several required query functions are provided to STSW-STSA110-SSL via the OpenSSL engine's support for the PKEY public/private key processing tool and for the Control command:

- `EC_KEY_set_private_key`
- `EC_KEY_set_public_key`
- `ENGINE_ctrl`

An API function will be provided to get the access of each attribute. This feature is not fully developed in this release.

```
EVP_PKEY* stsafe_load_pubkey(
    ENGINE *,
    Const char *,
    UI_METHOD *,
    void *);

EVP_PKEY* stsafe_load_privkey(
    ENGINE *e,
    const char *key_id,
    UI_METHOD *ui_method,
    void *callback_data);
```

6.9 Password verification

This function is used to perform password verification, and also to remember/feedback the number of remaining try attempts.

It is called through the Control command of STSW-STSA110-SSL or with the API function:

```
uint32_t stsafe_password_verification(
    const uint8_t *pInPassword,
    uint8_t *response);
```

6.10 Random generation

This function generates a random number using the STSAFE-A110 solution's random number generator.
It can be called through the STSW-STSA110-SSL with the API function:

```
int stsafe_get_random_bytes(
    unsigned char *buffer,
    int num);
```

6.11 Reset

This function resets the STSAFE-A110 solution and calls the `stsafe_init()` function to reinitialize the software.
It can be triggered through the Control command of the STSW-STSA110-SSL or with the API function:

```
    int stsafe_reset(void);
```

6.12 Hibernate

This function places the STSAFE-A110 solution in Hibernate mode.
It can be triggered through the Control command of STSW-STSA110-SSL or with the API function:

```
    int stsafe_hibernate(int wakeup);
```

The wakeup option has the following options:

```
STSAFEA_WAKEUP_FROM_I2C_START_OR_RESET 0x01
STSAFEA_WAKEUP_FROM_RESET 0x02
```

6.13 STSW-STSA110-SSL's Control command

STSW-STSA110-SSL provides the `ENGINE_ctrl` function to carry out various functions, mostly described in the previous sections. An API is also available for the direct call to these functions.

```
int stsafe_cmd_ctrl(
    ENGINE *e,
    int cmd,
    long i,
    void *p,
    void (*f)(void));
```

The table below gives the Control command parameters.

Table 1. Control command parameters

Command	ENGINE *e	int cmd	long i	void *p	void(*f)(void)
STSAFE_CMD_GET_PRODUCT_DATA	e	STSAFE_CMD_GET_PRODUCT_DATA	0	NULL for now, just print info	Null
STSAFE_CMD_GET_DEVICE_CERT	e	STSAFE_CMD_GET_DEVICE_CERT	0	File name to dump the cert	Null
STSAFE_CMD_SET_SIG_KEY_SLOT	e	STSAFE_CMD_SET_SIG_KEY_SLOT	Slot number	Null	Null
STSAFE_CMD_SET_GEN_KEY_SLOT	e	STSAFE_CMD_SET_GEN_KEY_SLOT	Slot number	Null	Null
STSAFE_CMD_SET_MEMORY_REGION	e	STSAFE_CMD_SET_MEMORY_REGION	Data partition zone number	Null	Null
STSAFE_CMD_WRITE_DEVICE_CERT	e	STSAFE_CMD_WRITE_DEVICE_CERT	0	File name to read from	Null
STSAFE_CMD_RESET	e	STSAFE_CMD_RESET	0	Null	Null
STSAFE_CMD_ECHO	e	STSAFE_CMD_ECHO	0	String to echo	Null
STSAFE_CMD_HIBERNATE	e	STSAFE_CMD_HIBERNATE	Wakeup code	Null	Null
STSAFE_CMD_VERIFYPASSWORD	e	STSAFE_CMD_VERIFYPASSWORD	0	Input/ Output byte string	Null
STSAFE_CMD_QUERY	e	STSAFE_CMD_QUERY	0	Item to query as string	Null

6.14 Examples of usage of STSW-STSA110-SSL commands from the CLI

Commands from the STSW-STSA110-SSL software have the general usage format shown below:

```
openssl engine Stsafe -t -post COMMAND:Value -post COMMAND:Value
```

To see the list of command and usage information, use:

```
openssl engine Stsafe -vvv
```

```
(Stsafe) STSAFE-A110 engine for OpenSSL
     PRODUCTINFO: Get STSAFE Product version
          (input flags): NO_INPUT
     GET_DEVICE_CERT: Get device certificate from hardware and store in the provided filename
          (input flags): STRING
     SET_SIG_KEY_SLOT: Set the slot that the engine will use for signature generation
(default 1)
          (input flags): NUMERIC
     SET_GEN_KEY_SLOT: Set the slot that the engine will use for key generation (default 255)
          (input flags): NUMERIC
     SET_MEMORY_REGION: Set the memory region to be used for writing of certifiate (default
1)
          (input flags): NUMERIC
     WRITE_CERTIFICATE: Write certificate given in filename (DER format) to memory region
          (input flags): STRING
     RESET_ENGINE: Reset the Stsafe to default and call the driver init function
          (input flags): NO_INPUT
     COMMAND_ECHO: Echo back the given string
          (input flags): STRING
     ENGINE_HIBERNATE: Put STSafe in Hibernate mode
          (input flags): NUMERIC
     ENGINE_VERIFYPASSWORD: Verify the password based on the password stored in the hardware
          (input flags): STRING
     ENGINE_QUERY: Query the requested setting on the STSAFE device
          (input flags): STRING
```

Note: Options for `ENGINE_QUERY` are shown below:

- *`DataPartition`*
- *`ProductData`*
- *`I2cParameter`*
- *`LifeCycleState`*
- *`HostKeySlot`*
- *`LocalEnvelopeKeySlot`*
- *`PublicKeySlot` (STSAFE-A110 feature - NOT supported at present)*
- *`CommandAuthorizationConfiguration`*

Examples

To retrieve the product information:

```
openssl engine Stsafe -t -post PRODUCTINFO
```

To set memory region and write certificate:

```
openssl engine Stsafe -t -post SET_MEMORY_REGION:4 -post
WRITE_CERTIFICATE:Test.der
```

To set memory region and get certificate from the memory region:

```
openssl engine Stsafe -t -post SET_MEMORY_REGION:4 -post
    GET_DEVICE_CERT:Test.pem
```

Echo string:

```
openssl engine Stsafe -t -post COMMAND_ECHO:"Hello engine"
```

6.15 STSW-STSA110-SSL command usage from an application using OpenSSL

The Control commands of STSW-STSA110-SSL can be used from within an application, for example the AWS IoT sample code described later uses these commands to set up the STSW-STSA110-SSL engine for a particular configuration. The following sections lists the most commonly used APIs.

6.15.1 Loading the engine

The first thing an application must do after normal OpenSSL initialization is to load the engine.

```
static ENGINE *stsafe_engine = NULL;

/* Initialize and load the engine for STSAFE-A110 */
stsafe_engine = ENGINE_by_id("Stsafe");
if (stsafe_engine == NULL) {
      // process error
}
```

6.15.2 Initializing the engine

Before the engine can be used by OpenSSL it needs to be initialized. This is done via a call to `ENGINE_init`, with the above obtained reference to the `stsafe_engine`.

```
// Initialize STSAFE ENGINE
if (! ENGINE_init(stsafe_engine)) {
    // process error
    ENGINE_free(stsafe_engine);
}
```

6.15.3 Setting up engine options for keys and memory regions

The engine can be told which key slots to use for key generation and signing, and the memory region to use. The defaults can be set in the build. This is done in:

```
stsafe_engine/Src/engine_init.c

long int stsafe_sig_key_slot = STSAFE_A_SLOT_0;
long int stsafe_gen_key_slot = STSAFE_A_SLOT_EPHEMERAL;
long int stsafe_memory_region = 0;
```

For the STSAFE-A110 SPL02 profile. This profile is used to configure generic samples of the STSAFE-A110 devices. See Application note AN5435 available from https://www.st.com for details.

Private key slots:

```
STSAFE_A_SLOT_0
STSAFE_A_SLOT_1
```

Ephemeral key slot:

```
STSAFE_A_SLOT_EPHEMERAL
```

Memory regions value :(`0..7`).

They can then be changed as needed in the application. Once these API calls are made, the settings will be used until they are changed by subsequent calls to the API.

```
// Set the key slots and secure memory region
if (! ENGINE_ctrl_cmd_string(stsafe_engine, "SET_SIG_KEY_SLOT", "1", 0)) {
    //  process error
}

if (! ENGINE_ctrl_cmd_string(stsafe_engine, "SET_GEN_KEY_SLOT", "255", 0)){
    // process error
    ENGINE_free(stsafe_engine);
}

if (! ENGINE_ctrl_cmd_string(stsafe_engine, "SET_MEMORY_REGION", "1", 0)) {
    // process error
    ENGINE_free(stsafe_engine);
}
```

6.15.4 Shutting down and releasing the engine

On application shutdown and during OpenSSL cleanup, the engine resources should be released as follows:

```
if (!
    ENGINE_finish(stsafe_engine)) {
    // process error
}

if (! ENGINE_free(stsafe_engine)) {
    // process error
}
```

6.15.5 Default values of the STSW-STSA110-SSL engine

The `engine init.c` file in the `src` folder contains the default values of the STSW-STSA110-SSL engine for the STSAFE-A110 SPL02 evaluation profile. For further details, see Application note AN5435 available from the ST website.

Private key associated with the **device certificate** (Slot 0):

```
stsafe_sig_key_slot = STSAFE_A_SLOT_0
```

Ephemeral key slot (Slot 255):

```
stsafe_gen_key_slot = STSAFE_A_SLOT_EPHEMERAL
```

Memory region where the **device certificate** is located:

```
stsafe_memory_region = 0
```

Appendix A stsafe_engine_test_suite execution log

```
============================================================
===== Pre Test Configuration =====
============================================================
============================================================
===== PASS Pre Test Configuration =====
============================================================

============================================================
===== Test 1 STSAFE Load Engine =====
============================================================
ENGINE> bind_helper: Engine id = Stsafe
ENGINE> bind_helper: ENGINE_set_id completed
ENGINE> bind_helper: ENGINE_set_name completed
ENGINE> bind_helper: ENGINE_set_init_function completed
ENGINE> bind_helper: ENGINE_set_RAND completed
ENGINE> bind_helper: ENGINE_set_ctrl_function completed
ENGINE> bind_helper: ENGINE_set_cmd_defns completed
stsafe_get_EC_methods called
EC_KEY_METHOD_set_sign.
EC_KEY_METHOD_set_verify.
EC_KEY_METHOD_set_keygen.
EC_KEY_METHOD_set_compute_key.
ENGINE> bind_helper: ENGINE_set_EC completed
ENGINE> bind_helper: ENGINE_set_load_pubkey_function completed
ENGINE> bind_helper: ENGINE_set_load_privkey_function completed
stsafe_pkey_meth_init called
stsafe_pkey_meth_init finished
ENGINE> bind_helper: stsafe_pkey_meth_init completed
ENGINE> bind_helper: ENGINE_set_pkey_meths completed
ENGINE> bind_helper: calling Engine_set_finish_function
ENGINE> bind_helper: ENGINE_set_finish_function completed
ENGINE> bind_helper: calling ENGINE_set_default
Using Openssl : OpenSSL 1.1.1g 21 Apr 2020
STSAFE-A110 StSafeA_CreateHandle = 5, pStSafeA->InOutBuffer = 0, pStSafeA-
>InOutBuffer.LV.Data = 0
StSafeA_GetDataBufferSize(): 523
About to call StSafeA_LocalEnvelopeKeySlotQuery: 367a68, 367ab8, 3661f8
StSafeA_LocalEnvelopeKeySlotQuery: 0 slot 0: presence flag =1
StSafeA_LocalEnvelopeKeySlotQuery: 0 slot 1: presence flag =1
---HostKeySlot = 3643b8, pStSafeA->InOutBuffer.LV.Data = 76f2c6ec
HostKeySlot->HostKeyPresenceFlag: 1
Main : stsafe_pairing success

**********^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Setting STSAFE-A110 host keys

**********vvvvvvvvvvvv***************
stsafe_pkey_meths called nid=0
ENGINE> bind_helper: ENGINE_set_default completed
============================================================
===== PASS Test 1 STSAFE Load Engine =====
============================================================
============================================================
===== Test 2 STSAFE Engine Init =====
============================================================
============================================================
===== PASS Test 2 STSAFE Engine Init =====
============================================================
============================================================
===== Test 3 STSAFE Get product Data =====
============================================================
stsafe_cmd_ctrl in ACTION!!! cmd = 200
STSAFE-A110 Product Information
------------------------------
MaskIdentification : aa4602
ST Product Number : a021e021c4e1d00139
```

```
InputOutputBufferSize : 507
AtomicityBufferSize : 64
NonVolatileMemorySize : 6376
TestDate : 37649
InternalProductVersionSize : 69
ModuleDate : 37713
FirmwareDeliveryTraceability : 000000
BlackboxDeliveryTraceability : 000000
PersoId : 000000
PersoGenerationBatchId : 000000
PersoDate : 000000
================================================================
===== PASS Test 3 STSAFE Get product Data =====
================================================================
================================================================
===== Test 4 STSAFE Wrap Data =====
================================================================
===== Generate 480 bytes of data to wrap
stsafe_AES_wrap_key called.
envelopeIn
0x30bd45487b91aef2
0xa5500d323fb4c63b
0xd9a557fa4cdad362
0xceba255a7d2de0ae
0xea25f665b6a4575c
0xf4648e3418546ff1
0xf9c6ec45a0bfa86f
0x79cdc9f7faa9a5e4
0xce9b49843fa0e033
0x056e671dc2d60fbb
0x9cfb013dbaa9ac34
0x76752b711ed055ec
0x6b9f70aa3f51dd44
0xbf4562821b713db8
0x6c3ef526e7a15a5e
0x1685cf34552420c0
0xc3906a03e14847a1
0x8da923a81a606086
0x9f55ad86f607e40c
0x8db340e2d860a39b
0xf10d9ed255e673e2
0x8f968baaf7eb3096
0x41dd1c37e5014472
0xb484548ce5f728d6
0x05c6a85aac1c3d3c
0xb2c8e6a9b3163ff4
0xf45c2cd95d704b11
0xf49f9ed997c6af9c
0x8c58f63974337526
0xfb5bd0af710fa365
0x6bcf3ec83f89da34
0x29780dc03ebd5cca
0x155203898678af81
0xd37f30458fd4aafa
0xa3e9c3e3729d179b
0x15245b53e1b71df6
0x0a217f90992f116d
0xae42b23d165c38b9
0x45fb9cb898b353ad
0xd8af00b9661db070
0x3e2f00d85e12450d
0x54f74a6a53822399
0x7dc0511573a4c24b
0x53c205bae0b52a1e
0xe42bf6433d3b5091
0x329afb861d1e1f9a
0xde70b05214729d68
0x35a22215574c333c
0x772a7fb465cf4598
```

```
0x69401e865f3d213d
0xadd18fc1432d2978
0xcf4b8d2798c1630f
0xebe2c450b109e81a
0x4a06a1a943c2e6f0
0x9376b2d6a3db4f72
0x27dc99bf9dfcce88
0xde92d98f9cc1aae6
0xc84b8f0b0d75fca0
0xebae768e89c501b0
0xa29a6f3f973ec875
================================================================
===== PASS Test 4 STSAFE Wrap Data =====
================================================================
================================================================
===== Test 5 STSAFE Unwrap Data =====
================================================================
stsafe_AES_unwrap_key called.
================================================================
===== PASS Test 5 STSAFE Unwrap Data =====
================================================================
================================================================
===== Test 6 STSAFE ECDSA Sign/Verify =====
================================================================
===== Setup for test
===== Read certificate from STSAFE
stsafe_cmd_ctrl in ACTION!!! cmd = 201
ENGINE> stsafe_cmd_ctrl: STSAFE_CMD_GET_DEVICE_CERT Device-Cert.pem
STSAFE> readCertificate: OPENSSL_malloc size is 523 bytes
STSAFE> readCertificate: certificateSize 402 numWrites 1 finalBytes 402
STSAFE> readCertificate: Read number 00 numBytesRead: 402
STSAFE> readCertificate: Chunk data :
3082018e30820134a003020102020b0209a021e021c4e1d00139300a06082a86
48ce3d040302304f310b300906035504061302e4c311e301c060355040a0c15
53544d6963726f656c656374726f6e696373206e763120301e06035504030c17
53544d205354534146452d412054455354204341203031302017d3230303930
323030303030305a180f323035303039303333030303030305a3046310b300906
035504061302465311b3019060355040a0c1253544d6963726f656c65637472
6f6e696373311a30180603550403 0c115354534146452d41313130204556414c
323059301306072a8648ce3d020106082a8648ce3d03010703420004f0f949ba
8040bb4033cad02ef6784f6490a3b199c0ba9a0f4e3def634af3d505bbb365fe
4453975b1257cab900b0436e5a31c27d5ac4ab8fc55337fb5c16e935300a0608
2a8648ce3d040302034800304502 2100d43ce253be5699e1644ffc23dd63dc23
289cf67832dfde9e59623df770709b5f02202d50e888c8ba3f0825d735813569
c71a9716ed60dbf8d9ff1b9cf6a0ed0d7335
Copying 402 bytes to 0x3657f8
STSAFE> readCertificate: Device certificate size: 402
STSAFE> readCertificate: Device certificate :
3082018e30820134a003020102020b0209a021e021c4e1d00139300a06082a86
48ce3d040302304f310b300906035504061302e4c311e301c060355040a0c15
53544d6963726f656c656374726f6e696373206e763120301e06035504030c17
53544d205354534146452d412054455354204341203031302017d3230303930
323030303030305a180f323035303039303333030303030305a3046310b300906
035504061302465311b3019060355040a0c1253544d6963726f656c65637472
6f6e696373311a30180603550403 0c115354534146452d41313130204556414c
323059301306072a8648ce3d020106082a8648ce3d03010703420004f0f949ba
8040bb4033cad02ef6784f6490a3b199c0ba9a0f4e3def634af3d505bbb365fe
4453975b1257cab900b0436e5a31c27d5ac4ab8fc55337fb5c16e935300a0608
2a8648ce 3d040302034800304502 2100d43ce253be5699e1644ffc23dd63dc2
3289cf67832dfde9e59623df77 0709b5f02202d50e888c8ba3f0825d7358135
69c71a9716ed60dbf8d9ff1b9cf6a0ed0d7335
STSAFE> readCertificate: Store the certificate to Device-Cert.pem
===== Certificate written to Device-Cert.pem
===== Generate digest
===== ECDSA sign

stsafe_engine_ecdsa_do_sign digest_len = 32
StSafeA_GenerateSignature : RLength=32 SLength=32
```

```
Input Hash size:32
09a64a87239d21c118b112d385574319ff396e42e0f8ed0a161ff9bb22fa9d0d

Signature R size:32
9404422966b2688a81dc359e6313ae2ecc63a26abc95c8d3799d2a74b597b574
Signature S size:32
730f9503375454148efdac6aef5e969657bb30596417b74bbaa7b417a9437164

===== ECDSA Verify Process
===== Open Device-Cert.pem file
===== Read certificate from Device-Cert.pem
===== Get public key from certificate
===== Do verification
stsafe_engine_ecdsa_do_verify called
StSafeA_VerifyMessageSignature called, StatusCode:0 SignatureValidity=1
stsafe ecdsa verfiy end! result 1
===== Verification Success
================================================================
===== PASS Test 6 STSAFE ECDSA Sign/Verify =====
================================================================
================================================================
===== Test 7 STSAFE ECDH/Generate Ephemeral Keys =====
================================================================
===== Setup for test
STSAFE-EC> stsafe_ec_generate_key called.
STSAFE-EC> stsafe_ec_generate_key: Using Slot 255
STSAFE-EC> stsafe_ec_generate_key: Curve prime256v1 -> STSAFEA_NIST_P_256
STSAFE-EC> stsafe_ec_generate_key: X:Length 32 Data 0x59 0x9f 0x2e 0x5b 0x24 0x48 0x10 0x88
0x07 0xd5 0x22 0x00 0x77 0x9e 0x18 0xcc 0x61 0x4d 0x01 0xa7 0x73 0x0f 0x84 0x27 0x06 0x38
0x02 0xbb 0x53 0x05 0x00 0x06
STSAFE-EC> stsafe_ec_generate_key: Y:Length 32 Data 0xc2 0x25 0xd0 0x07 0xbe 0xab 0x40 0x8c
0x39 0x91 0x66 0xd9 0x34 0x86 0xdc 0xa4 0x66 0xc6 0x77 0xa4 0x26 0xcb 0xe4 0xb2 0xb3 0x4b
0x3a 0x33 0x7d 0x04 0xef 0x80
599F2E5B2448108807D52200779E18CC614D01A7730F8427063802BB53050006
C225D007BEAB408C399166D93486DCA466C677A426CBE4B2B34B3A337D04EF80
stsafe_engine_ecdh_compute_key called
STSAFE-EC> stsafe_ecdh_compute_key: Using Slot 255
STSAFE-EC> stsafe_ecdh_compute_key: EC_POINT_point2oct len 65
Before calling StSafeA_EstablishKey.
STSAFE-EC> stsafe_ecdh_compute_key: StatusCode = 0, outlen = 32
================================================================
===== PASS Test 7 STSAFE ECDH/Generate Ephemeral Keys =====
================================================================
================================================================
===== Test 8 STSAFE Private Key Methods =====
================================================================
===== Setup for test
===== Read certificate from STSAFE
stsafe_cmd_ctrl in ACTION!!! cmd = 201
ENGINE> stsafe_cmd_ctrl: STSAFE_CMD_GET_DEVICE_CERT Device-Cert.pem
STSAFE> readCertificate: OPENSSL_malloc size is 523 bytes
STSAFE> readCertificate: certificateSize 402 numWrites 1 finalBytes 402
STSAFE> readCertificate: Read number 00 numBytesRead: 402
STSAFE> readCertificate: Chunk data :
3082018e30820134a003020102020b0209a021e021c4e1d00139300a06082a86
48ce3d040302304f310b300906035504061302e4c311e301c060355040a0c15
53544d6963726f656c656374726f6e696373206e763120301e06035504030c17
53544d205354534146452d412054455354204341203031302017d3230303930
323030303030305a180f323035303039303330303030305a3046310b300906
035504061302e4c311b3019060355040a0c1253544d6963726f656c65637472
6f6e696373311a30180603550403 0c115354534146452d41313130204556414c
323059301306072a8648ce3d020106082a8648ce3d03010703420004f0f949ba
8040bb4033cad02ef6784f6490a3b199c0ba9a0f4e3def634af3d505bbb365fe
4453975b1257cab900b0436e5a31c27d5ac4ab8fc55337fb5c16e935300a0608
2a8648ce3d040302034800304502210 0d43ce253be5699e1644ffc23dd63dc23
289cf67832dfde9e59623df770709b5f02202d50e888c8ba3f0825d735813569
c71a9716ed60dbf8d9ff1b9cf6a0ed0d7335
Copying 402 bytes to 0x368098
```

```
STSAFE> readCertificate: Device certificate size: 402
STSAFE> readCertificate: Device certificate :
3082018e30820134a003020102020b0209a021e021c4e1d00139300a06082a86
48ce3d040302304f310b300906035504061302e4c311e301c060355040a0c15
53544d6963726f656c656374726f6e696373206e763120301e06035504030c17
53544d205354534146452d412054455354204341203031302017 0d3230303930
323030303030305a180f323035303039303330303030305a3046310b300906
035504061302e4c311b3019060355040a0c1253544d6963726f656c65637472
6f6e696373311a301806035504030c115354534146452d413131302045564 14c
323059301306072a8648ce3d020106082a8648ce3d03010703420004f0f949ba
8040bb4033cad02ef6784f6490a3b199c0ba9a0f4e3def634af3d505bbb365fe
4453975b1257cab900b0436e5a31c27d5ac4ab8fc55337fb5c16e935300a0608
2a8648ce3d040302034800304502210 0d43ce253be5699e1644ffc23dd63dc23
289cf67832dfde9e59623df770709b5f02202d50e888c8ba3f0825d735813569
c71a9716ed60dbf8d9ff1b9cf6a0ed0d7335
STSAFE> readCertificate: Store the certificate to Device-Cert.pem
===== Certificate written to Device-Cert.pem
===== Load private key via Engine
stsafe_load_privkey called
STSAFE_PKEY> stsafe_load_pubkey_internal called
STSAFE_PKEY> stsafe_load_pubkey_internal pkey is NULL so allocate new one
stsafe_pkey_meths called nid=408
STSAFE_PKEY> stsafe_load_pubkey_internal StSafeA_Read Success CertificateSize = 402
STSAFE_PKEY> stsafe_load_pubkey_internal returns pkey
===== privkey of size 1
stsafe_pkey_meths called nid=408
stsafe_pkey_ec_init called
stsafe_pkey_ec_init ctx not NULL
stsafe_pkey_is_stsafe_key called
StSafeA_Read Success CertificateSize = 402

Input key (len = 65): 04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35

STSafe key (len = 65): 04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35
stsafe_pkey_is_stsafe_key return =1
STSAFE_PKEY> stsafe_load_pubkey_internal called
STSAFE_PKEY> stsafe_load_pubkey_internal pkey NOT NULL.
STSAFE_PKEY> stsafe_load_pubkey_internal returns pkey
stsafe_pkey_ec_init returned
stsafe_pkey_ec_sign_init called
===== Generate digest
stsafe_pkey_ec_sign called
stsafe_pkey_is_stsafe_key called
StSafeA_Read Success CertificateSize = 402

Input key (len = 65): 04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35

STSafe key (len = 65): 04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35
stsafe_pkey_is_stsafe_key return =1
stsafe_pkey_ec_sign ---1
stsafe_pkey_ec_sign ---signlen=256
stsafe_pkey_ec_sign ---tbslen=32
stsafe_engine_ecdsa_do_sign digest_len = 32
StSafeA_GenerateSignature : RLength=32 SLength=32


Input Hash size:32
6b321c0fe290496095a841962aa986dc4f8520693773d9f1fec295e95747405a

Signature R size:32
08544093a7f85396c3e66cfb4ab6de498ba1ca3a3da741ee7591a35aa9ab636a
```

```
Signature S size:32
3936eab7fd9e2e976aaff62dd85b2a008a8644b9ea60e7d7ace7255a7a800d93

===== Signing success
===== Prepare verification
===== Open Device-Cert.pem file
===== Read certificate from Device-Cert.pem
stsafe_engine_ecdsa_do_verify called
StSafeA_VerifyMessageSignature called, StatusCode:0 SignatureValidity=1
stsafe ecdsa verfiy end! result 1
================================================================
===== PASS Test 8 STSAFE Private Key Methods =====
================================================================
================================================================
===== Test 9 STSAFE Randon Number Generation =====
================================================================
Stsafe engine random length 5

STSAFE> stsafe_get_random_bytes: Success Random number = 0xb872fb05f6
================================================================
===== PASS Test 9 STSAFE Randon Number Generation =====
================================================================
================================================================
===== Test 10 STSAFE Zone Data Read/update Test =====
================================================================
ENGINE> stsafe_update_zone: Update Zone function called.
READ test : Updated data 100 bytes to Zone 0x6:
 be ef 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
 60 61 62 63
ENGINE> stsafe_update_zone: Update Zone function called.
READ test : Updated data 499 bytes to Zone 0x6:
 be ef 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
 b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
 c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
 d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
 e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
 f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff
 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
 b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
 c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
 d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
 e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
 f0 f1 f2
ENGINE> stsafe_read_zone: Read Zone function called.
```

```
READ test : Reading data 100 bytes from Zone 0x6:
 be ef 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
 60 61 62 63
ENGINE> stsafe_read_zone: Read Zone function called.
READ test : Reading data 499 bytes from Zone 0x6:
 be ef 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
 b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
 c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
 d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
 e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
 f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff
 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
 60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
 a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af
 b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
 c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
 d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
 e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
 f0 f1 f2
ENGINE> stsafe_zone_decrement: Decrement Zone counter function called.
READ test : Decrement Zone index 6 counter by 1, now it is: 26
================================================================
===== PASS Test 10 STSAFE Zone Data Read/update Test =====
================================================================
================================================================
===== Test 11 STSAFE Query Test =====
================================================================
===== Query DataPartition
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE-A1x0 Data Partition Information
--------------------------------------
Index : 00
ZoneType : 00
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x00
UpdateAccessCondition : 0x07
DataSegmentLength : 1000 bytes
Index : 01
ZoneType : 00
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x01
UpdateAccessCondition : 0x00
DataSegmentLength : 0700 bytes
Index : 02
```

```
ZoneType : 00
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x01
UpdateAccessCondition : 0x00
DataSegmentLength : 0600 bytes
Index : 03
ZoneType : 00
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x01
UpdateAccessCondition : 0x00
DataSegmentLength : 0600 bytes
Index : 04
ZoneType : 00
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x01
UpdateAccessCondition : 0x00
DataSegmentLength : 1696 bytes
Index : 05
ZoneType : 01
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x01
UpdateAccessCondition : 0x00
DataSegmentLength : 0064 bytes
Index : 06
ZoneType : 01
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x01
UpdateAccessCondition : 0x00
DataSegmentLength : 0064 bytes
Index : 07
ZoneType : 00
ReadAcChangeRight : 0x00
ReadAccessCondition : 0x00
UpdateAcChangeRight : 0x01
UpdateAccessCondition : 0x00
DataSegmentLength : 1578 bytes
===== Query ProductData
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE-A110 Product Information
-------------------------------
MaskIdentification : aa4602
ST Product Number : a021e021c4e1d00139
InputOutputBufferSize : 507
AtomicityBufferSize : 64
NonVolatileMemorySize : 6376
TestDate : 37649
InternalProductVersionSize : 69
ModuleDate : 37713
FirmwareDeliveryTraceability : 000000
BlackboxDeliveryTraceability : 000000
PersoId : 000000
PersoGenerationBatchId : 000000
PersoDate : 000000
===== Query I2cParameter
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE-A1x0 I2C Information
---------------------------
I2cAddress : 0x21
LowPowerModeConfig : 0x04
LockConfig : 0x01
===== Query LifeCycleState
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE-A1x0 Lifecycle Information
---------------------------------
```

```
LifeCycleStatus : 0x03
===== Query HostKeySlot
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE-A1x0 Host Key Slot Information
-------------------------------------
HostKeyPresenceFlag : 0x01
HostCMacSequenceCounter : 19
===== Query LocalEnvelopeKeySlot
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE-A1x0 Local Envelope Key Slot Information
-----------------------------------------------
NumberOfSlots : 2
SlotNumber : 0
PresenceFlag : 1
KeyLength : AES 128 bit
SlotNumber : 1
PresenceFlag : 1
KeyLength : AES 128 bit
===== Query PublicKeySlot
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE> queryPublicKeySlot: Function not supported at this time
===== Query CommandAuthorizationConfiguration
stsafe_cmd_ctrl in ACTION!!! cmd = 210
STSAFE-A1x0 Command Authorization Information
---------------------------------------------
ChangeRight : 0x00
CommandAuthorizationRecordNumber : 9
Record : 0
CommandCode : 0x08
CommandAC : 0x00
HostEncryptionFlags : 0x00
Record : 1
CommandCode : 0x09
CommandAC : 0x00
HostEncryptionFlags : 0x00
Record : 2
CommandCode : 0x0a
CommandAC : 0x00
HostEncryptionFlags : 0x00
Record : 3
CommandCode : 0x0e
CommandAC : 0x03
HostEncryptionFlags : 0x02
Record : 4
CommandCode : 0x0f
CommandAC : 0x03
HostEncryptionFlags : 0x01
Record : 5
CommandCode : 0x16
CommandAC : 0x01
HostEncryptionFlags : 0x00
Record : 6
CommandCode : 0x18
CommandAC : 0x01
HostEncryptionFlags : 0x00
Record : 7
CommandCode : 0x1b
CommandAC : 0x00
HostEncryptionFlags : 0x00
Record : 8
CommandCode : 0x1c
CommandAC : 0x00
HostEncryptionFlags : 0x00
============================================================
===== PASS Test 11 STSAFE Query Test =====
============================================================
============================================================
===== Test 12 STSAFE ECHO Test =====
============================================================
```

```
stsafe_cmd_ctrl in ACTION!!! cmd = 207
ENGINE> stsafe_cmd_ctrl: send the string to STSAFE A110 and send back the response from the
chip.
ENGINE> stsafe_cmd_ctrl: Echoed string len 14 content is: Pinging STSafe
ECHO CMD returns Pinging STSafe. Originally sent Pinging STSafe
================================================================
===== PASS Test 12 STSAFE ECHO Test =====
================================================================
================================================================
===== Test 13 Verify Password Test =====
================================================================
stsafe_cmd_ctrl in ACTION!!! cmd = 209
ENGINE> stsafe_cmd_ctrl: verify the password and return with status + remaining retries
count within the same string.
stsafe_password_verification called.
Result of veryfy password: length = 0, status = e0, remaining count = 249
Verify Password CMD returns status 0xe0. Retry count = 249, Originally sent password
Banana101
================================================================
===== PASS Test 13 Verify Password Test =====
================================================================
================================================================
===== Test 14 Reset Test =====
================================================================
stsafe_cmd_ctrl in ACTION!!! cmd = 206
ENGINE> stsafe_cmd_ctrl: Reseting STSAFE hardware to default state, and then re-init the
driver.
Using Openssl : OpenSSL 1.1.1g 21 Apr 2020
STSAFE-A110 StSafeA_CreateHandle = 5, pStSafeA->InOutBuffer = 0, pStSafeA-
>InOutBuffer.LV.Data = 0
StSafeA_GetDataBufferSize(): 523
About to call StSafeA_LocalEnvelopeKeySlotQuery: 36ab00, 3682e8, 3664d8
StSafeA_LocalEnvelopeKeySlotQuery: 0 slot 0: presence flag =1
StSafeA_LocalEnvelopeKeySlotQuery: 0 slot 1: presence flag =1
---HostKeySlot = 76b43ae0, pStSafeA->InOutBuffer.LV.Data = 76c78434
HostKeySlot->HostKeyPresenceFlag: 1
Main : stsafe_pairing success

**********^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Setting STSAFE-A110 host keys

**********vvvvvvvvvvvv***************
================================================================
===== PASS Test 14 Reset Test =====
================================================================
================================================================
===== Test 15 Hibernate Test =====
================================================================
stsafe_cmd_ctrl in ACTION!!! cmd = 208
ENGINE> stsafe_cmd_ctrl: Put the STSAFE in Hibernate state, wakeup mode 1.
================================================================
===== PASS Test 15 Hibernate Test =====
================================================================

================================================================
===== END OF TEST!!!! =====
================================================================
```

Appendix B Trace of the connectivity to AWS IoT

```
[INFO ][INIT][2020-12-08 10:32:22] SDK successfully initialized.
[INFO ][NET][2020-12-08 10:32:22] Network library initialized.
ENGINE> bind_helper: Engine id = Stsafe
ENGINE> bind_helper: ENGINE_set_id completed
ENGINE> bind_helper: ENGINE_set_name completed
ENGINE> bind_helper: ENGINE_set_init_function completed
ENGINE> bind_helper: ENGINE_set_RAND completed
ENGINE> bind_helper: ENGINE_set_ctrl_function completed
ENGINE> bind_helper: ENGINE_set_cmd_defns completed
stsafe_get_EC_methods called
EC_KEY_METHOD_set_sign.
EC_KEY_METHOD_set_verify.
EC_KEY_METHOD_set_keygen.
EC_KEY_METHOD_set_compute_key.
ENGINE> bind_helper: ENGINE_set_EC completed
ENGINE> bind_helper: ENGINE_set_load_pubkey_function completed
ENGINE> bind_helper: ENGINE_set_load_privkey_function completed
stsafe_pkey_meth_init called
stsafe_pkey_meth_init finished
ENGINE> bind_helper: stsafe_pkey_meth_init completed
ENGINE> bind_helper: ENGINE_set_pkey_meths completed
ENGINE> bind_helper: calling Engine_set_finish_function
ENGINE> bind_helper: ENGINE_set_finish_function completed
ENGINE> bind_helper: calling ENGINE_set_default
Using Openssl     : OpenSSL 1.1.1g  21 Apr 2020
STSAFE-A110 StSafeA_CreateHandle = 5, pStSafeA->InOutBuffer = 0x76aa768c, pStSafeA-
>InOutBuffer.LV.Data = 0x76aa7430

***********vvvvvvvvvvvv**************
stsafe_pkey_meths called nid=0
ENGINE> bind_helper: ENGINE_set_default completed
stsafe_cmd_ctrl in ACTION!!! cmd = 202
ENGINE> stsafe_cmd_ctrl: Setting STSAFE signature key slot to 0
stsafe_cmd_ctrl in ACTION!!! cmd = 203
ENGINE> stsafe_cmd_ctrl: Setting STSAFE generate key slot to 255
stsafe_cmd_ctrl in ACTION!!! cmd = 204
ENGINE> stsafe_cmd_ctrl: Setting STSAFE memory region to 0
[INFO ][MQTT][2020-12-08 10:32:22] MQTT library successfully initialized.
[INFO ][DEMO][2020-12-08 10:32:22] MQTT demo client identifier is iotdemo14654328 (length
15).
[INFO ][NET][2020-12-08 10:32:22] TCP connection successful.
[INFO ][NET][2020-12-08 10:32:22] Setting up TLS.
Stsafe engine random length 16

STSAFE> stsafe_get_random_bytes:  Success Random number = 0x04a98e6437dd0a381f3489e430ba1df5
Stsafe engine random length 32

STSAFE> stsafe_get_random_bytes:  Success Random number =
0x17739d10ad928948378dc1bf6bad5dd1ccd9432267a9a05b8895d48a6f79976c
Stsafe engine random length 32

STSAFE> stsafe_get_random_bytes:  Success Random number =
0x231cd8bbd961cbbb5b47e2ef74b93392a48374018b6e9a35681f7af59e257b1d
Stsafe engine random length 32

STSAFE> stsafe_get_random_bytes:  Success Random number =
0x12e7402d3f7f345fcfc9a3771b9926d2e4533cd6e0c5b2f6b9c0bc9a3397cdb8
[INFO ][NET][2020-12-08 10:32:22] Successfully imported root CA.
[INFO ][NET][2020-12-08 10:32:22] Successfully imported client certificate.
stsafe_load_privkey called
STSAFE_PKEY> stsafe_load_pubkey_internal called
STSAFE_PKEY> stsafe_load_pubkey_internal pkey is NULL so allocate new one
stsafe_pkey_meths called nid=408
STSAFE_PKEY> stsafe_load_pubkey_internal StSafeA_Read Success CertificateSize = 402
STSAFE_PKEY> stsafe_load_pubkey_internal returns pkey
[INFO ][NET][2020-12-08 10:32:22] Successfully imported client certificate private key.
```

```
Stsafe engine random length 32

STSAFE> stsafe_get_random_bytes:  Success Random number =
0x754e5e3159d01242ec27f212abd829d94610563beceeaf2099ffa43abb9fa129
Stsafe engine random length 32

STSAFE> stsafe_get_random_bytes:  Success Random number =
0xf832b59596a4e602ee75b4920e22f85153b8942180b56c81f6750ff58cc3ba0a
Stsafe engine random length 32

STSAFE> stsafe_get_random_bytes:  Success Random number =
0x598de25399b43e059a9dcfbc18edf4b313c2e6a29eb39b7063ad1aaea21539bd
stsafe_pkey_meths called nid=408
stsafe_pkey_ec_init called
stsafe_pkey_ec_init ctx not NULL
stsafe_pkey_is_stsafe_key called
stsafe_pkey_is_stsafe_key return =0
stsafe_pkey_ec_init returned
stsafe_pkey_meths called nid=408
stsafe_pkey_ec_init called
stsafe_pkey_ec_init ctx not NULL
stsafe_pkey_is_stsafe_key called
StSafeA_Read Success CertificateSize = 402

Input key (len = 65):  04 db 40 ea 57 ac 22 e4 fd 0b 6b 27 fb da c5 45 f4 46 4b 85 b1 de d6
5f 1d 6a e2 3e 1f 51 eb 26 91 55 dd 32 e4 2e 67 09 ac 71 4e 46 df 8c d7 c6 be 4e 30 32 81 5c
7d 28 85 28 43 0e d3 5e 81 45 6f

STSafe key (len = 65):  04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35
StSafeA_Read Success CertificateSize = 0
stsafe_pkey_is_stsafe_key return =0
stsafe_pkey_ec_init returned
STSAFE-EC> stsafe_ec_generate_key called.
STSAFE-EC> stsafe_ec_generate_key: Using Slot 255
STSAFE-EC> stsafe_ec_generate_key: Curve prime256v1 -> STSAFEA_NIST_P_256
STSAFE-EC> stsafe_ec_generate_key: X:Length 32 Data 0x37 0x90 0x67 0x5b 0xaa 0x9d 0xbd 0x9a
0xda 0xfd 0xd0 0xc0 0x42 0xf7 0x5f 0xac 0xbd 0x8e 0x3b 0x71 0xbf 0x38 0x33 0xcd 0xaa 0x98
0xfa 0x61 0x38 0x31 0x3e 0x70
STSAFE-EC> stsafe_ec_generate_key: Y:Length 32 Data 0x86 0xd2 0x2a 0x2c 0x22 0x52 0x01 0x0b
0x5e 0x3f 0x77 0xcb 0xd4 0xe5 0x02 0x66 0x01 0xef 0x9f 0x75 0x65 0x71 0xe8 0x29 0xed 0x39
0x60 0xa0 0xbe 0x33 0xe8 0xa8
3790675BAA9DBD9ADAFDD0C042F75FACBD8E3B71BF3833CDAA98FA6138313E70
86D22A2C2252010B5E3F77CBD4E5026601EF9F756571E829ED3960A0BE33E8A8
stsafe_pkey_meths called nid=408
stsafe_pkey_ec_init called
stsafe_pkey_ec_init ctx not NULL
stsafe_pkey_is_stsafe_key called
StSafeA_Read Success CertificateSize = 402

Input key (len = 65):  04 37 90 67 5b aa 9d bd 9a da fd d0 c0 42 f7 5f ac bd 8e 3b 71 bf 38
33 cd aa 98 fa 61 38 31 3e 70 86 d2 2a 2c 22 52 01 0b 5e 3f 77 cb d4 e5 02 66 01 ef 9f 75 65
71 e8 29 ed 39 60 a0 be 33 e8 a8

STSafe key (len = 65):  04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35
StSafeA_Read Success CertificateSize = 0
stsafe_pkey_is_stsafe_key return =0
stsafe_pkey_ec_init returned
stsafe_engine_ecdh_compute_key called
STSAFE-EC> stsafe_ecdh_compute_key: Using Slot 255
STSAFE-EC> stsafe_ecdh_compute_key: EC_POINT_point2oct len 65
Before calling StSafeA_EstablishKey.
STSAFE-EC> stsafe_ecdh_compute_key: StatusCode = 0, outlen = 32
stsafe_pkey_meths called nid=408
stsafe_pkey_ec_init called
stsafe_pkey_ec_init ctx not NULL
```

```
stsafe_pkey_is_stsafe_key called
StSafeA_Read Success CertificateSize = 402

Input key (len = 65):  04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35

STSafe key (len = 65):  04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35
stsafe_pkey_is_stsafe_key return =1
STSAFE_PKEY> stsafe_load_pubkey_internal called
STSAFE_PKEY> stsafe_load_pubkey_internal pkey NOT NULL.
STSAFE_PKEY> stsafe_load_pubkey_internal returns pkey
stsafe_pkey_ec_init returned
stsafe_pkey_ec_sign_init called
stsafe_pkey_ec_sign called
stsafe_pkey_is_stsafe_key called
StSafeA_Read Success CertificateSize = 402

Input key (len = 65):  04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35

STSafe key (len = 65):  04 f0 f9 49 ba 80 40 bb 40 33 ca d0 2e f6 78 4f 64 90 a3 b1 99 c0 ba
9a 0f 4e 3d ef 63 4a f3 d5 05 bb b3 65 fe 44 53 97 5b 12 57 ca b9 00 b0 43 6e 5a 31 c2 7d 5a
c4 ab 8f c5 53 37 fb 5c 16 e9 35
stsafe_pkey_is_stsafe_key return =1
stsafe_pkey_ec_sign ---1
stsafe_pkey_ec_sign ---signlen=72
stsafe_pkey_ec_sign ---tbslen=64
stsafe_engine_ecdsa_do_sign digest_len = 64
StSafeA_GenerateSignature : RLength=32 SLength=32


Input Hash size:64
bcfc04ccf4bc600f2afd77e11e16186147303c7c1a33310d848e1ebbebc7f02c
f68b563d2ca4e31859d23273405d041b5a0cacc148dcb6d11b7f32bc877765b8

Signature R size:32
5c5842c176973197ab35c7abff94fffdd218722a082d90cf194a1e1b6131b9f4
Signature S size:32
439642dbb44ef1b0548f28b7f832ce3c4c1b52bb3177e6dcdb6f2af15f143928

Stsafe engine random length 8

STSAFE> stsafe_get_random_bytes:  Success Random number = 0x70f762f960ac4b18
[INFO ][NET][2020-12-08 10:32:22] TLS handshake succeeded.
[INFO ][NET][2020-12-08 10:32:22] Peer certificate verified. TLS connection established.
```

Appendix C Glossary

Table 2. List of abbreviations and terms

Term	Meaning
AES	Advanced encryption standard
AWS™	Amazon Web Services®
CA	Certificate authority
CLI	Command-line interface
CN field	Common Name field
C-SDK	Software development kit for C
CSR	Certificate signing request
EC	Elliptic curve
ECC	Elliptic curve cryptography
ECDSA	Elliptic curve digital signature algorithm
ECDH	Elliptic curve Diffie–Hellman
HTTP	Hypertext transfer protocol
JSON	JavaScript object notation
MIT	Massachusetts Institute of Technology
MQTT	Message queuing telemetry transport
NVM	Non-volatile memory
OpenSSL®	*OpenSSL* is a robust, commercial-grade, and full-featured toolkit for the transport layer security (TLS) and secure sockets layer (SSL) protocols.
OS	Operating system
PKEY	Public or private key processing tool
RAND bytes	Random bytes
SDK	Software development kit
TLS	Transport layer security

Revision history

Table 3. **Document revision history**

Date	Version	Changes
10-Dec-2020	1	Initial release.

Contents

List of tables

List of figures