
TrustFLEX Step by Step Guide

Accessory Authentication

Table of Contents

1	<i>Introduction</i>	3
1.1	<i>Getting started with Jupyter Notebook Tutorials</i>	3
1.1.1	Starting Jupyter Notebook.....	3
1.2	<i>Jupyter Notebook Basics</i>	4
1.2.1	The Notebook dashboard	4
1.3	<i>Introduction to Jupyter Notebook GUI</i>	4
2	<i>Jupyter Notebook Tutorials</i>	6
3	<i>Resource Generation Notebook</i>	7
4	<i>Use Case Prototyping</i>	16
4.1	<i>Running Accessory-Authentication example on Jupyter Notebook:</i>	16
4.2	<i>Running Accessory-Authentication on Embedded platform</i>	20
4.2.1	MPLAB:	20
4.2.2	Atmel Studio (Deprecated)	23
4.3	<i>Crypto Auth Trust Platform Factory reset</i>	26
5	<i>FAQ</i>	27

1 Introduction

This document gives a detailed walk through of the Accessory Authentication use case implementation. If familiar with Jupyter Notebook, can skip this section and move to Section 2.

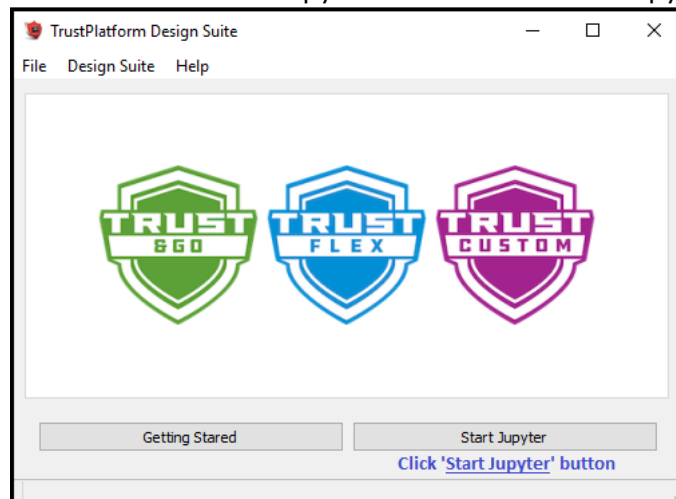
1.1 Getting started with Jupyter Notebook Tutorials

Jupyter Notebook is open source web application which allows you to create documents that contain code that you can execute in place as well as narrative text. It provides GUI elements, ability to execute code in place, ability to add images and gives it the look and feel that normal code files lack.

Jupyter notebooks are mainly used to explain/evaluate code in an interactive way.

1.1.1 Starting Jupyter Notebook

Jupyter notebook can be launched from Trust Platform GUI Main window. Run START -> Trust Platform x.x.x icon. Click on 'Start Jupyter' button to launch Jupyter local server.



Clicking on Start Jupyter should be web browser tab like below,



1.2 Jupyter Notebook Basics

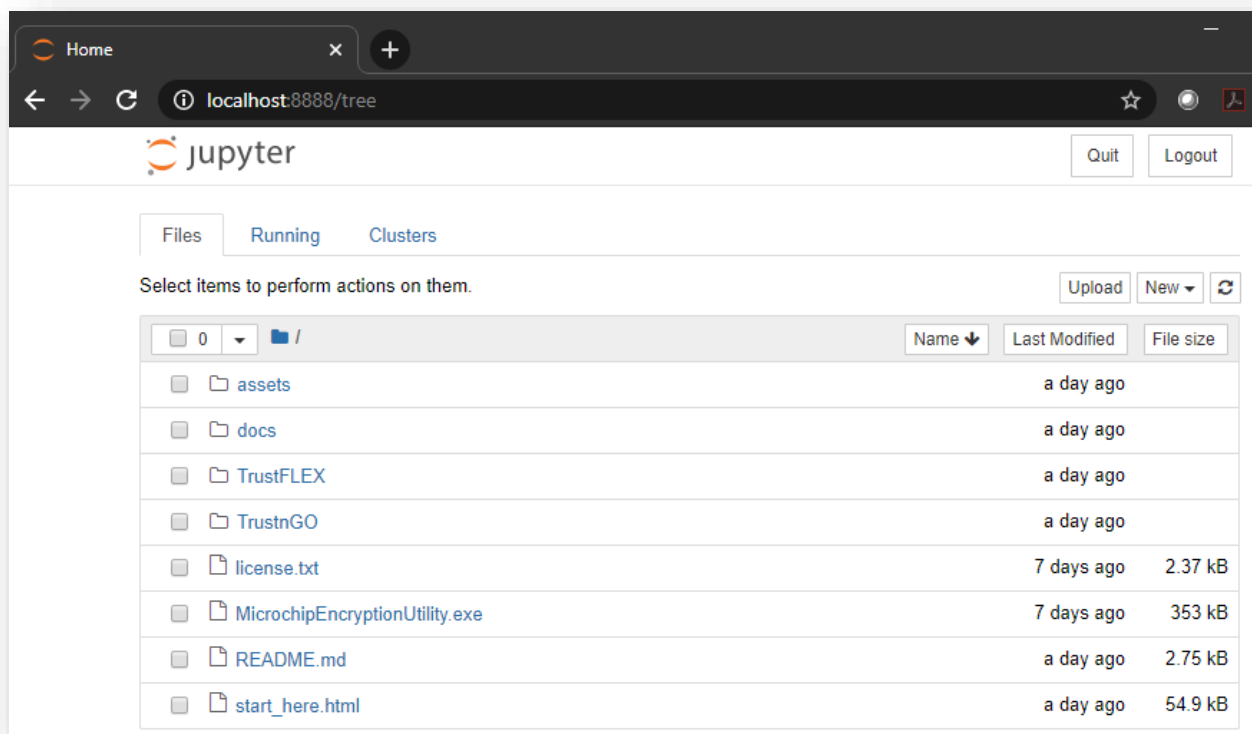
It is recommended to become familiar with Jupyter basic concepts with the online documentation, <https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html>

Some of the content is duplicated here for convenience. The online documentation should always be used as a reference.

1.2.1 The Notebook dashboard

When you first start the notebook server, your browser will open to the notebook dashboard. The dashboard serves as a home page for the notebook. Its main purpose is to display the notebooks and files in the current directory.

For example, here is a screenshot of the Jupyter dashboard. The top of the notebook list displays clickable breadcrumbs of the current directory. By clicking on these breadcrumbs or on sub-directories in the notebook list, you can navigate your file system.

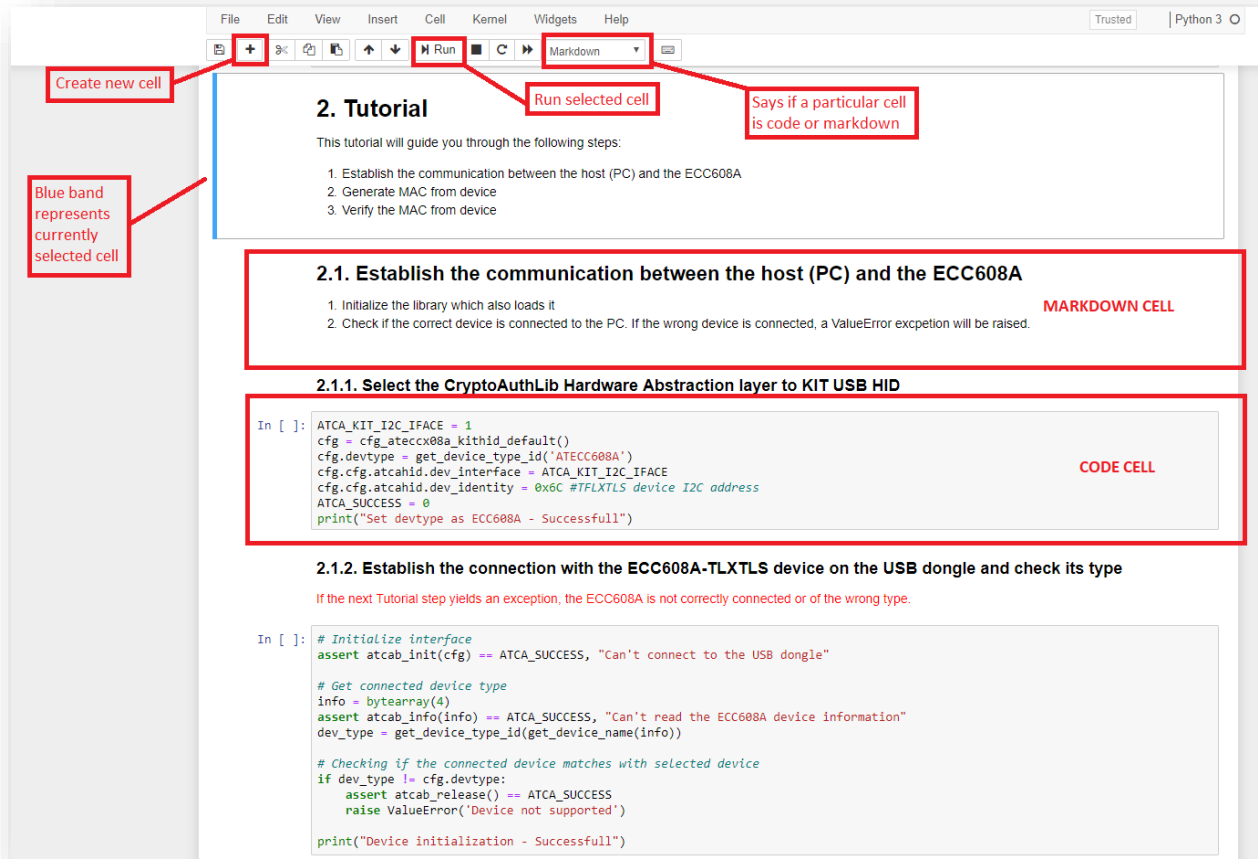


1.3 Introduction to Jupyter Notebook GUI.

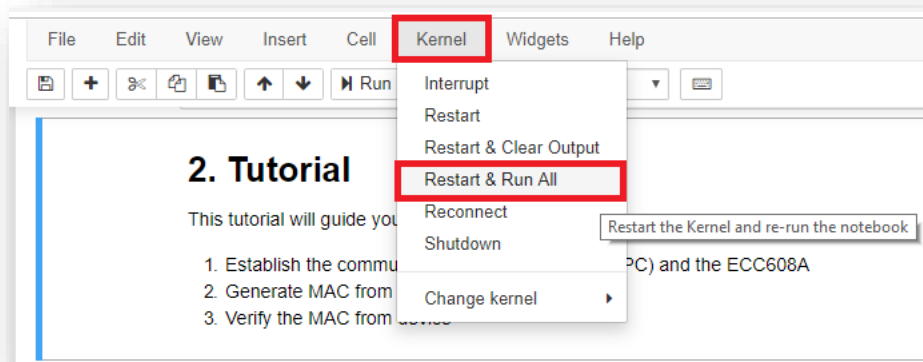
Jupyter Notebooks contain cells where you can either write code or markdown text. Notebooks contain multiple cells, some set as code and others markdown. Code cells contain code that can be executed live, and markdown contains text and images to explain the code.

Below image shows some options in a typical Jupyter Notebook. Individual cells can be executed by pressing on the RUN button as shown in the below image.

All cells in the Notebook can be executed in order by **Kernel->Restart & Run All**.



To run all cells in sequence.



2 Jupyter Notebook Tutorials

The Trust Platform Design Suite comes with Notebook Tutorials to easily prototype popular use cases for TrustFLEX. Here is the list of Jupyter Notebook Tutorials.

Jupyter Notebook Tutorials	Relative Path	Applicable Devices
Manifest Generation	TrustnGO\00_resource_generation\TNGTLS_manifest_file_generation.ipynb	Trust&GO
GCP Connect	TrustnGO\05_cloud_connect\notebook\gcp\TNGTLS_GCP_connect.ipynb	Trust&GO
AWS Connect	TrustnGO\05_cloud_connect\notebook\aws\TNGTLS_aws_connect.ipynb	Trust&GO
Azure Connect	TrustnGO\05_cloud_connect\notebook\azure\TNGTLS_azure_connect.ipynb	Trust&GO
Resource Generation	TrustFLEX\00_resource_generation\TFLXTLS_resource_generator.ipynb	TrustFLEX
Accessory Authentication	TrustFLEX\01_accessory_authentication\notebook\TFLXTLS_accessory_authentication.ipynb	TrustFLEX
Firmware Validation	TrustFLEX\02_firmware_validation\notebook\TFLXTLS_firmware_validation.ipynb	TrustFLEX
IP Protection	TrustFLEX\04_ip_protection\notebook\TFLXTLS_IP_protection.ipynb	TrustFLEX
Secure Public Key Rotation	TrustFLEX\05_public_key_rotation\notebook\TFLXTLS_public_key_rotation.ipynb	TrustFLEX
Asymmetric authentication	08_asymmetric_authentication\notebook\TFLXTLS_asymmetric_authentication.ipynb	TrustFLEX
GCP Connect	TrustFLEX\10_cloud_connect\notebook\gcp\TFLXTLS_GCP_connect.ipynb	TrustFLEX
AWS Custom PKI	TrustFLEX\10_cloud_connect\notebook\aws\TFLXTLS_aws_connect.ipynb	TrustFLEX
Azure Connect	TrustFLEX\10_cloud_connect\notebook\azure\TFLXTLS_azure_connect.ipynb	TrustFLEX

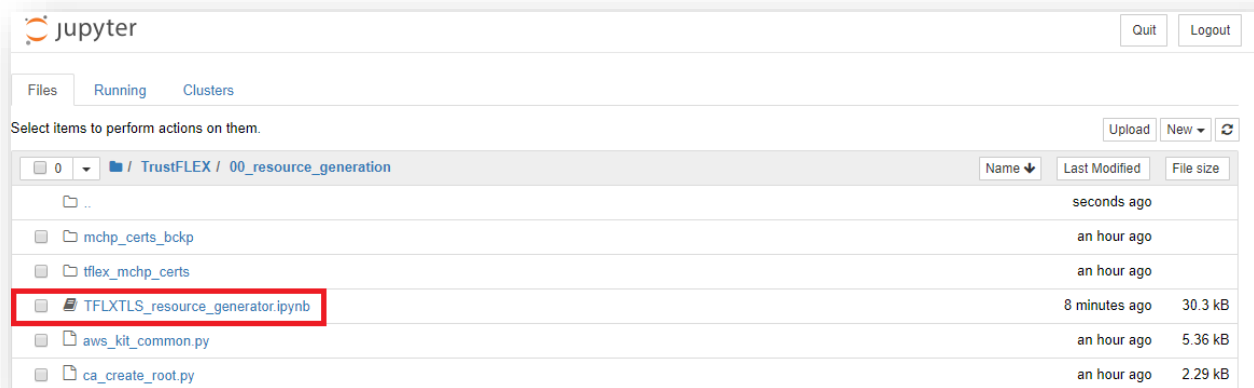
3 Resource Generation Notebook

TFLXTLS device is one of the three devices available in the Crypto Auth Trust Platform Board.

TrustFLEX devices come with pre-programmed certificates in slots 10, 11 and 12, also slots 0-4 have pre-generated private keys, other than the mentioned slots all the other slots have no data in them.

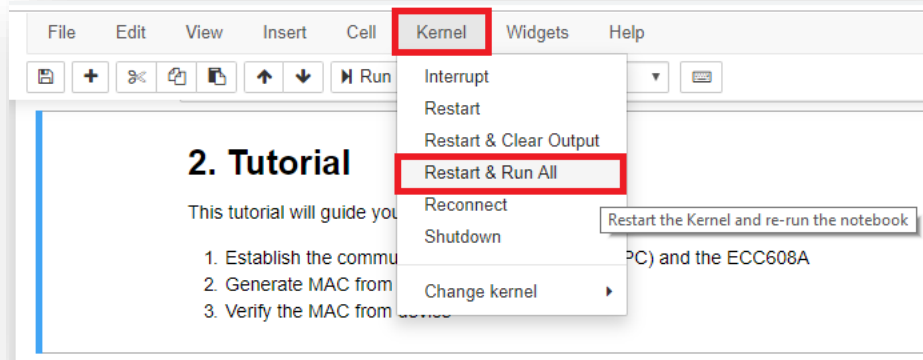
The Resource Generator Notebook will create development keys and certificates for all slots that can be further customized. Keys and Certificate chains are stored in the PC filesystem. These keys should never be used for production purposes as their generation is not handled in a secure environment. These development keys will be later used by the other notebooks to implement the various pre-defined use cases.

Within the Jupyter Dashboard, navigate **TrustFLEX\00_resource_generation** folder to open **TFLXTLS_resource_generator.ipynb** notebook



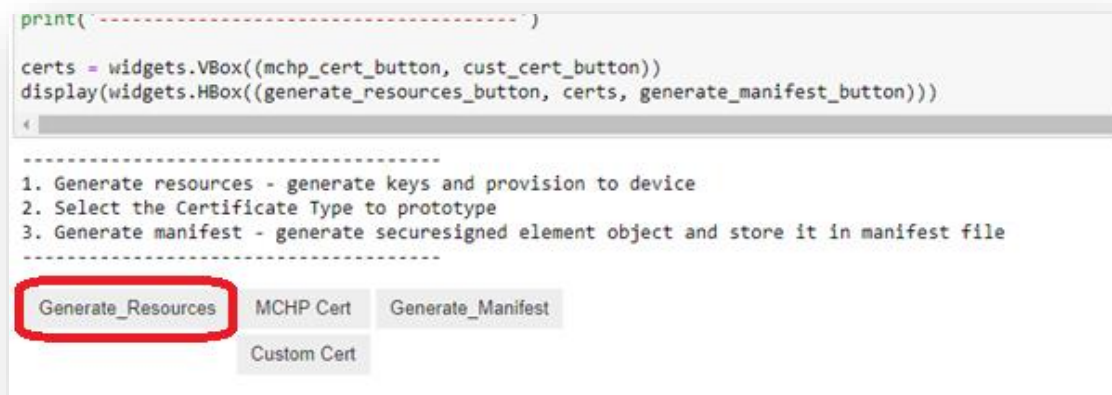
Run all cells of the Crypto Resource Generator Notebook: Kernel->Restart & Run All

Note: Before executing the cells on Crypto Trust Platform, its required to have factory default program running on SAMD21 of Trust Platform. Refer to [Crypto Auth Trust Platform Factory reset](#) section for reloading default program.



Crypto Resource Generator notebook is common for all the use case which comes with option to load the signer certificate and device certificate. The Notebook will generate several keys and certificates. Make sure you have an error free output before continuing to the next steps of the training. Following are 3 different things can be performed,

1. Generating resources to general key slots



The output log should resemble this:

Slot 0 is a private key slot, no action required
Slot 1 is a private key slot, no action required
Slot 2 is a private key slot, no action required
Slot 3 is a private key slot, no action required
Slot 4 is a private key slot, no action required

Slot 6 is a secret key, created slot_6_secret_key.pem and programmed

NOTE: While writing symmetric key into secure element it has to be encrypted with IO protection key. So here, Slot 6 (IO protection key) is written before slot 5 (Symmetric key)

Slot 5 is a secret key, created slot_5_secret_key.pem and programmed

Slot 7 is a secureboot digest slot, slot can only be written through secureboot command

Slot 8 is a general purpose slot of size 416 bytes, no action required

Slot 9 is a secret key, created slot_9_secret_key.pem and programmed

Slot 10 is a certificate slot, no action required now, will be updated as part of Generate Certificates

Slot 11 is a certificate slot, no action required now, will be updated as part of Generate Certificates

Slot 12 is a certificate slot, no action required now, will be updated as part of Generate Certificates

Slot 13 is a public key slot, created slot_13_ecc_key_pair.pem and programmed

Slot 14 is a public key slot, created slot_14_ecc_key_pair.pem and programmed

Slot 15 is a public key slot, created slot_15_ecc_key_pair.pem and programmed

Key generation - Success

2. Generating MCHP or Custom Certificates

On selecting Custom certificates, it prompts to enter the organization name, enter the name that will be used as an Organization Name in the certificate template. The name length is limited to 24 characters.

-
1. Generate resources - generate keys and provision to device
 2. Select the Certificate Type to prototype
 3. Generate manifest - generate securesigned element object and store it in manifest file
-

Generate_Resources **MCHP Cert** Generate_Manifest
Custom Cert

Slot 0 is a private key slot, no action required
Slot 1 is a private key slot, no action required
Slot 2 is a private key slot, no action required
Slot 3 is a private key slot, no action required
Slot 4 is a private key slot, no action required
Slot 6 is a secret key, created slot_6_secret_key.pem and programmed

NOTE: While writing symmetric key into secure element it has to be encrypted with IO protection key) is written before slot 5 (Symmetric key)

Slot 5 is a secret key, created slot_5_secret_key.pem and programmed
Slot 7 is a secureboot digest slot, slot can only be written through secureboot command
Slot 8 is a general purpose slot of size 416 bytes, no action required
Slot 9 is a secret key, created slot_9_secret_key.pem and programmed
Slot 10 is a certificate slot, no action required now, will be updated as part of Generate
Slot 11 is a certificate slot, no action required now, will be updated as part of Generate
Slot 12 is a certificate slot, no action required now, will be updated as part of Generate
Slot 13 is a public key slot, created slot_13_ecc_key_pair.pem and programmed
Slot 14 is a public key slot, created slot_14_ecc_key_pair.pem and programmed
Slot 15 is a public key slot, created slot_15_ecc_key_pair.pem and programmed

Key generation - Success

Org Name:

**Type Org Name and Press Enter to
continue Custom Certs processing**

The output log should resemble this:

Custom Certs processing...
Device contains custom device and signer certificates
Building new root certificate
Building new signer csr certificate
Building new signer certificate
Read device serial number...OK (SN: 01233E8A1491F2A601)

Read device public key from slot 0...OK (Public Key: CF1988BC3A6C252026FE70FB34397AD85A39AE811C722BFA6E5EC1E9CDA9133B3F0E91FD3877F25B8C893B311BAF0203CB5100C4CDABEBAFDAF3EBD550B00125)

Generating device certificate...OK (saved to device_01233E8A1491F2A601.crt)

Saving signer certificate to device...OK

Saving device certificate to device...OK

Thing ID eabc56113c70227a18c0a62f7c285fc68d75f9cd

Custom certificate generation and provisioning - SUCCESS

Validate root certificate...OK

-----BEGIN CERTIFICATE-----

MIIBYjCCAW+gAwIBAgIQeoueybRh8XWwzOkoixtW1jAKBggqhkJOPQQDAjA7MQ0wCwYDVQQKDAR0ZXN0MSowKAYDVQQDDCFDcnlwdG8gQXV0aGVudGljYXRpb24gUm9vdCBDQSAwMDIwIBcNMjAwNzAxMDgwNTE5WhgPMjA2MDA2MjEwODA1MTlaMDsxDTALBgNVBAoMBHRlc3QxKjAoBgNVBAMMIUNyeXB0byBBdXRoZW50aWNhdGlvbiBSb290IENBIDAwwMjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABFf6qcSyPv8iY0uccoTXSISstaz0ECCUxXUoqky8Xo40vsOCbPPt5QtlvNHnyy8tAbwza6DsAiz2sGLzDI5hQhqjUzBRMB0GA1UdDgQWBRRHVPQoljiq65JOG4vu5l32JzmkSTAfBgNVHSMEGDAWgBRHVPQoljiq65JOG4vu5l32JzmkSTAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0kAMEYCIQCB7FKx5K33xK9E0PsWGKZRaaQxxSRypC66y4hVqWVmmMAIhAMIG22zNUKPHCcHQxfQssYH5LfR5SVE+WC3Hyxem/EVj

-----END CERTIFICATE-----

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

7a:8b:9e:c9:b4:61:f1:75:b0:cc:e9:28:8b:1b:56:d6

Signature Algorithm: ecdsa-with-SHA256

Issuer: O=test, CN=Crypto Authentication Root CA 002

Validity

Not Before: Jul 1 08:05:19 2020 GMT

Not After : Jun 21 08:05:19 2060 GMT

Subject: O=test, CN=Crypto Authentication Root CA 002

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:57:fa:a9:c4:b2:3e:ff:22:63:4b:9c:72:84:d7:

4a:54:ac:b5:ac:f4:10:20:94:c5:75:28:aa:4c:bc:

5e:8e:34:be:c3:82:6c:f3:ed:e5:0b:65:bc:d1:e7:

cb:2f:2d:01:bc:33:6b:a0:ec:02:2c:f6:b0:62:f3:
0c:8e:61:42:1a
ASN1 OID: prime256v1
NIST CURVE: P-256
X509v3 extensions:
X509v3 Subject Key Identifier:
47:54:F4:28:96:38:AA:EB:92:4E:1B:8B:EE:E6:5D:F6:27:39:A4:49
X509v3 Authority Key Identifier:
keyid:47:54:F4:28:96:38:AA:EB:92:4E:1B:8B:EE:E6:5D:F6:27:39:A4:49

X509v3 Basic Constraints: critical
CA:TRUE

Signature Algorithm: ecdsa-with-SHA256
30:46:02:21:00:81:ec:52:b1:e4:ad:f7:c4:af:44:d0:fb:16:
18:a6:51:69:a4:31:c5:24:72:a4:2e:ba:cb:88:55:a9:65:66:
30:02:21:00:c9:46:db:6c:cd:50:a3:c7:71:c1:d0:c5:f4:2c:
b1:81:f9:2d:f4:79:49:51:3e:58:2d:c7:cb:17:a6:fc:45:63

Validate signer certificate...OK

-----BEGIN CERTIFICATE-----

MIIB3TCCAYKgAwIBAgIQV/RpeXxWfquIIYFCFTDc/TAKBggqhkJOPQQDAjA7MQ0w
CwYDVQQKDAR0ZXN0MSowKAYDVQQDDCFDcnlwdG8gQXV0aGVudGljYXRpb24gUm9v
dCBDQSAwMDIwIBcNMjAwNzAxMDgwMDAwWhgPMjA0MDA3MDEwODAwMDBaMDsxDTAL
BgNVBAoMBHRlc3QxKjAoBgNVBAMMIUNyeXB0byBBdXRoZW50aWNhdGlvb1BTaWdu
ZXIgaRkZGRjBZMBMGBByqGSM49AgEGCCqGSM49AwEHA0IABCEubbOfXDakettxvfKu
kfG5UhQNDHrPrZiURytSZmQ8p38VacZ682akSAC6XQYDzhly5/504eAHBCuN5rOt
vnOjZjBkMA4GA1UdDwEB/wQEAwIBhjASBgNVHRMBAf8ECDAGAQH/AgEAMB0GA1Ud
DgQWBBRycA/sc+NWXwp0wLudepyPtQtzFzAfBgNVHSMEGDAWgBRHVPQoljiq65JO
G4vu5I32JzmkSTAKBggqhkJOPQQDAgNJADBGAiEA1ThacjiYboKYh69+NIIQKiX2
wb7Jztq8zMsY61H/NKYCIQDQc2TQfOI9HBDUoDzUtTZNgIksElkU7ysiSgBhumAA
zQ==

-----END CERTIFICATE-----

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

57:f4:69:79:7c:56:7e:ab:88:21:81:42:15:30:dc:fd

Signature Algorithm: ecdsa-with-SHA256

Issuer: O=test, CN=Crypto Authentication Root CA 002

Validity

Not Before: Jul 1 08:00:00 2020 GMT

Not After : Jul 1 08:00:00 2040 GMT

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

77:19:29:06:cc:14:4f:e7:b8:75:28:4b:ea:da:74:d2

Signature Algorithm: ecdsa-with-SHA256

Issuer: O=test, CN=Crypto Authentication Signer FFFF

Validity

Not Before: Jul 1 06:00:00 2020 GMT

Not After : Jul 1 06:00:00 2048 GMT

Subject: O=test, CN=sn01233E8A1491F2A601

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:cf:19:88:bc:3a:6c:25:20:26:fe:70:fb:34:39:

7a:d8:5a:39:ae:81:1c:72:2b:fa:6e:5e:c1:e9:cd:

a9:13:3b:3f:0e:91:fd:38:77:f2:5b:8c:89:3b:31:

1b:af:02:03:cb:51:00:c4:cd:ab:eb:af:da:f3:eb:

d5:50:b0:01:25

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Basic Constraints: critical

CA:FALSE

X509v3 Key Usage: critical

Digital Signature, Key Agreement

X509v3 Subject Key Identifier:

EA:BC:56:11:3C:70:22:7A:18:C0:A6:2F:7C:28:5F:C6:8D:75:F9:CD

X509v3 Authority Key Identifier:

keyid:72:70:0F:EC:73:E3:56:5F:0A:74:C0:BB:9D:7A:9C:8F:B5:0B:73:17

Signature Algorithm: ecdsa-with-SHA256

30:44:02:20:03:67:fd:0a:ea:c7:09:b0:ad:1b:2b:71:8c:90:

a5:62:74:a3:80:31:2f:31:a8:78:26:63:7c:9e:68:d0:50:1b:

02:20:45:9d:ee:bb:88:4c:ee:87:a7:6a:c2:b7:50:62:f8:01:

eb:ea:93:c5:f2:f2:7a:2d:64:c2:81:5c:7d:59:c7:bc

3. Generating Manifest file

```
-----  
1. Generate resources - generate keys and provision to device  
2. Select the Certificate Type to prototype  
3. Generate manifest - generate securesigned element object and store it in manifest file  
-----
```

Generate_Resources

MCHP Cert

Generate_Manifest

Custom Cert

The output log should resemble this:

```
-----  
Generating manifest data...OK (saved to TFLXTLS_devices_manifest.json)  
-----
```

The Notebook will also generate a manifest file to be uploaded into the public cloud of your choice (Google GCP, AWS IoT and Microsoft Azure).

After running this Notebook, it generates the required resources and program data zone with required secrets, keys and certificates. For this use case, secret key only required, and which is loaded into TrustFLEX device (ATECC608B) in slot 5.

4 Use Case Prototyping

This hands-on lab is intended to demonstrate the usage of host to authenticate Accessory device. It uses symmetric authentication where both host and accessory device shares a common secret.

This process uses a challenge-response model. In this model, host authenticates the accessory device based on MAC response. MAC is calculated on the accessory device to prove that it holds the secret key that is shared by the host. Then the calculated MAC will be verified by the host to authenticate the accessory.

MAC calculation on accessory includes device serial number, nonce (number used once) and shared secret key. By including serial number and nonce, host can get unique MAC from each accessory every time, thereby avoiding the replay attacks.

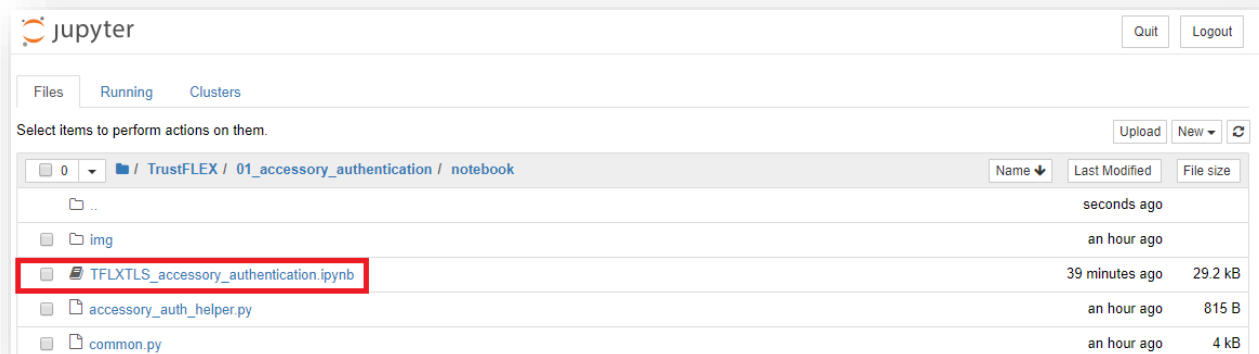
This lab is developed by simulating TrustFLEX device as Accessory and host to authenticate the accessory. In both TrustFLEX Slot5 and host has the same shared secret key.

The resource generation for TrustFLEX device will load a prototyping symmetric key to Slot5 of TrustFLEX device.

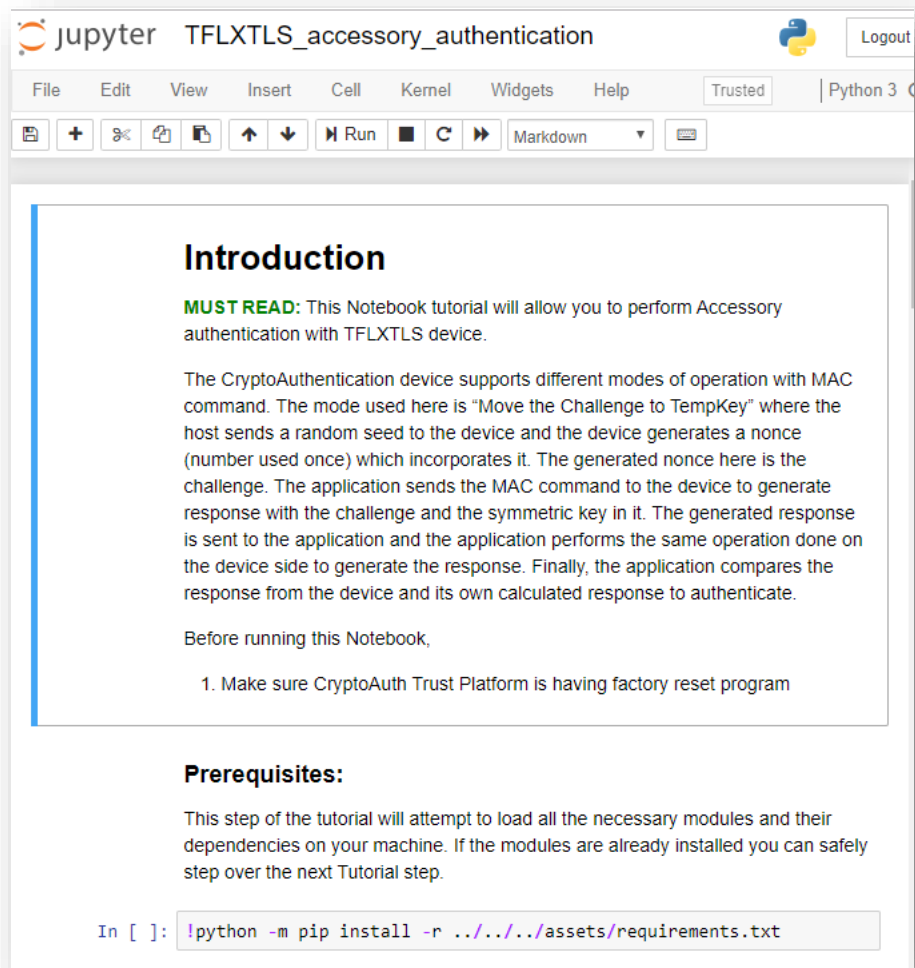
Following sections provides detail steps to execute the usecase both on Jupyter Notebook and on Embedded project

4.1 Running Accessory-Authentication example on Jupyter Notebook:

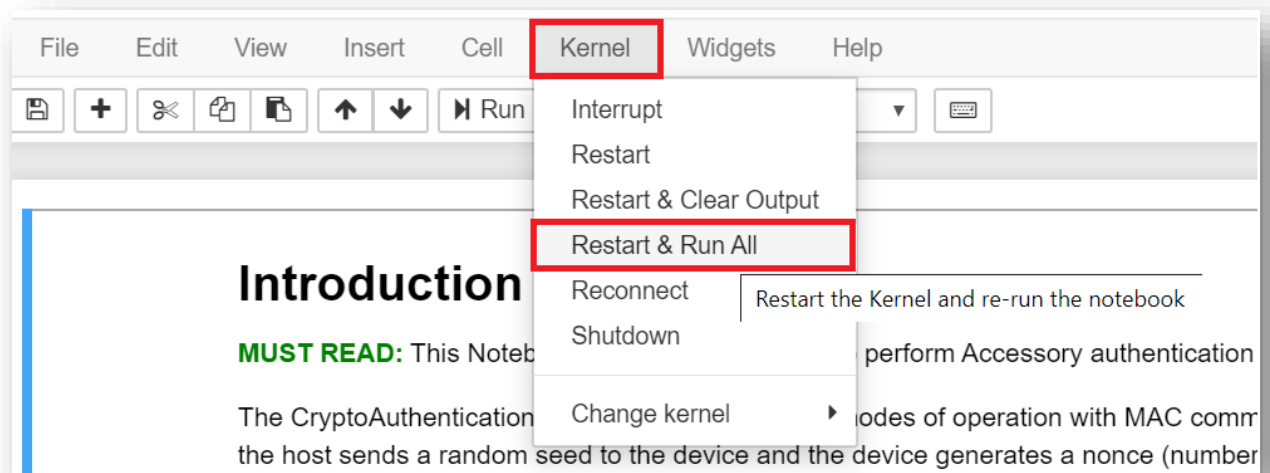
1. From the Jupyter Home page, navigate to **TrustFLEX\01_accessory_authentication\notebook\TFLXTLS_accessory_authentication.ipynb** notebook file and open it.



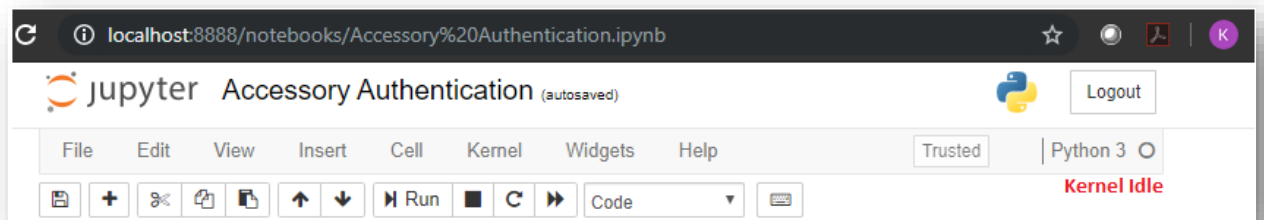
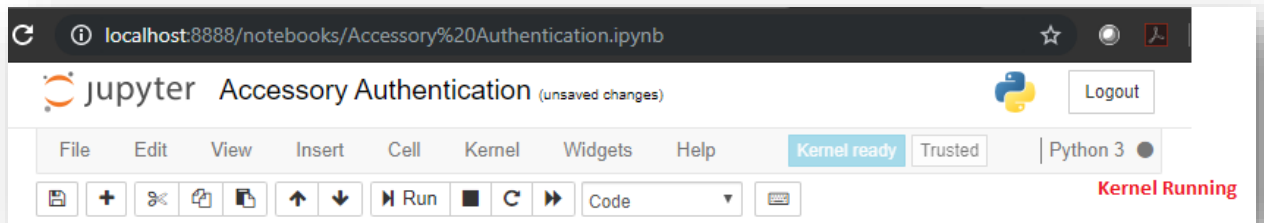
Opening the notebook from Jupyter home page should load the following on the browser,



2. Run All Cells by using Kernel -> Restart & Run All



- It may take a while to complete, wait for the kernel to complete all processing i.e. from Kernel Running to Kernel Idle state (Check circle above **RED** text)



- Navigate through different cells output for the description of the step and result from the execution.

- There are 2 major steps in this lab

Generate MAC from Accessory (TrustFLEX)

Code block of this step generates a random challenge and expects Accessory to provide the MAC for this challenge. Accessory calculates the MAC value by including its serial number, shared secret and the challenge received from host.

Calculate Nonce

To calculate MAC, nonce is considered. Nonce is calculated based on the challenge which is initiated by host. Then calculated nonce will be stored in tempkey on TrustFLEX and on host.

Calculate MAC on accessory device

MAC is calculated on accessory device is to prove that the accessory device has access and holds the secret key which is shared by host. Accessory device MAC is calculated with accessory device serial number, calculated nonce and secret key.

Then calculated MAC will send to the host to authenticate the accessory device.

Below screenshot display the accessory device MAC.

```
In [3]: seed_in = bytearray(20)
rand_out = bytearray(32)
nonce = bytearray()
device_mac = bytearray(32)

# Generate the nonce in device and return the random number
assert atcab_nonce_rand(seed_in,rand_out) == ATCA_SUCCESS, "Random nonce from device failed"

# Calculate the nonce value on the host side
nonce.extend(rand_out[0:32])
nonce.extend(seed_in[0:20])
nonce.append(0x16)
nonce.append(0)
nonce.append(0)
digest = hashes.Hash(hashes.SHA256(), backend=default_backend())
digest.update(bytes(nonce))
nonce = digest.finalize()

# Calculate the mac in device with its symmetric diversified key in slot
assert atcab_mac(MAC_MODE,SHARED_SECRET_SLOT,0,device_mac) == ATCA_SUCCESS, "MAC from Accessory device failed"
print("MAC Received from Accessory device:")
print(pretty_print_hex(device_mac, indent='  '))
```

```
MAC Received from Accessory device:
 3A CA F8 7B BF FF 25 9B 5B DA FA AD 60 C0 3B CB
 67 C6 DD C2 76 00 9D D7 E4 2E 8E 11 0E 6D 48 35
```

Verify the MAC with host device

Code block of this step generates Verify MAC button. Clicking the button performs checkmac operation to verify the MAC received from Accessory is corresponds to host challenge, Accessory serial number and shared secret key. If any of this mismatch, the checkmac operation fails indicating accessory is not authentic.

```
#uncomment following line to try wrong mac
#host_mac = bytes(b'00000000000000000000000000000000')

print('MAC calculated on host:')
print(pretty_print_hex(host_mac, indent='  '))

if (device_mac == host_mac):
    print('\nAccessory device authenticated successfully!')
    mac_verify.button_style = 'success'
else:
    mac_verify.button_style = 'danger'
    print('\Accessory device not authenticated...')

tooltip = 'Click to perform MAC-Response Verify'
mac_verify = widgets.Button(description = 'Verify MAC', tooltip=tooltip)
mac_verify.on_click(mac_mac_resp_verify)
display(mac_verify)
```

Verify MAC

```
MAC calculated on host:
 3A CA F8 7B BF FF 25 9B 5B DA FA AD 60 C0 3B CB
 67 C6 DD C2 76 00 9D D7 E4 2E 8E 11 0E 6D 48 35
```

Accessory device authenticated successfully!

6. In Jupyter notebook, run cells till the end of notebook, you will see a "Verify MAC" button will appear. Press the button, it will turn green if accessory device gets authenticated by MCU or it will turn red.

Pressing the button, turns it Green or Red. Green indicates that the device is authenticated by host and Red indicates the authentication is failed.

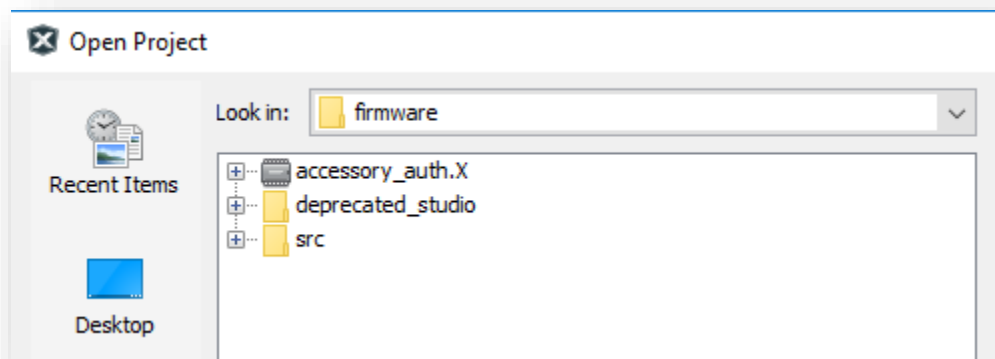
4.2 Running Accessory-Authentication on Embedded platform

This usecase can also be executed on Embedded platform. Once the resources are generated, both Atmel Studio and MPLAB projects provided can be used to run the application on Crypto Auth Trust Platform.

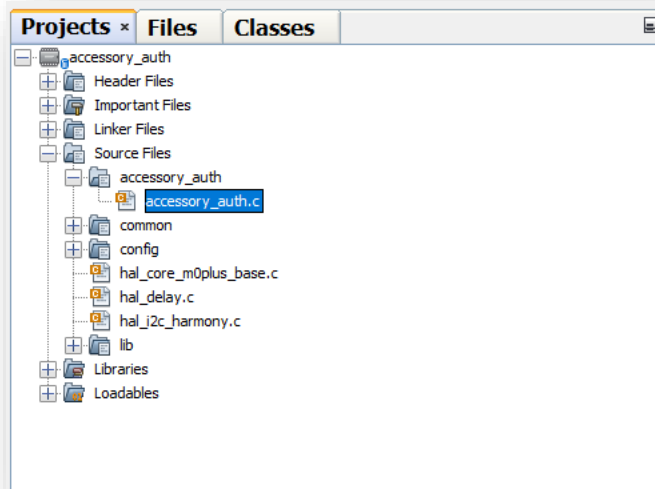
Note: This usecase requires resource generation notebook executed prior to using embedded projects.

4.2.1 MPLAB:

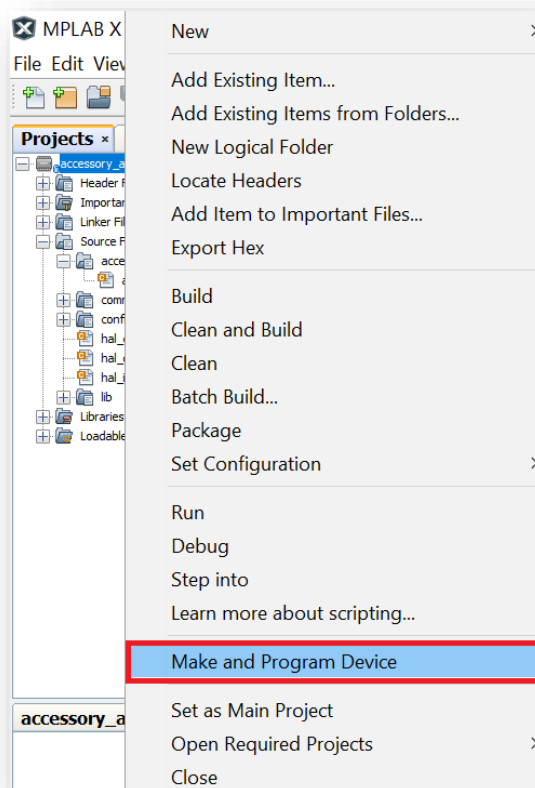
1. Open **accessory_auth.X** project by navigating to MPLAB -> File -> Open Project -> **TrustFLEX\01_accessory_authentication\firmware\accessory_auth.X**



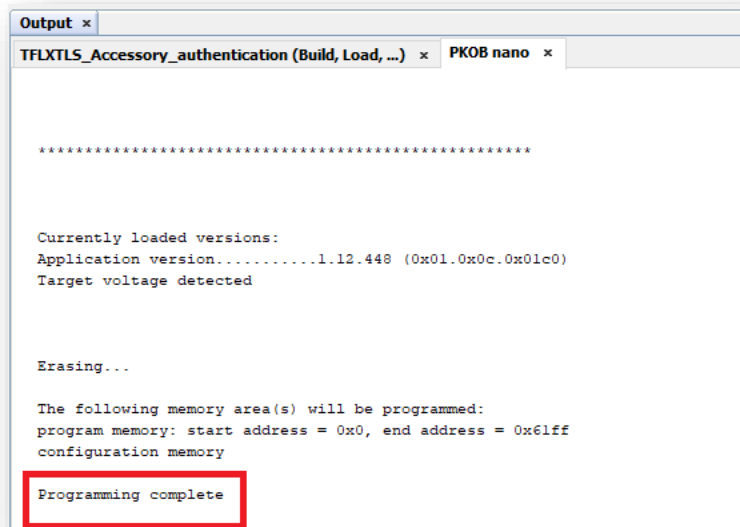
2. The application source code **accessory_auth.c** is available at **TrustFLEX\01_accessory_authentication\firmware\accessory_auth.c**. Other supporting files can be found under **assets\dependencies**



3. Program the Crypto Trust platform by navigating to **accessory_auth -> Make and Program Device**



This step may take some time, wait for MPLAB to program the device. Once it is done programming you will see "**Programming complete**" message in Output Window.



```
*****

Currently loaded versions:
Application version.....1.12.448 (0x01.0x0c.0x01c0)
Target voltage detected

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x61ff
configuration memory

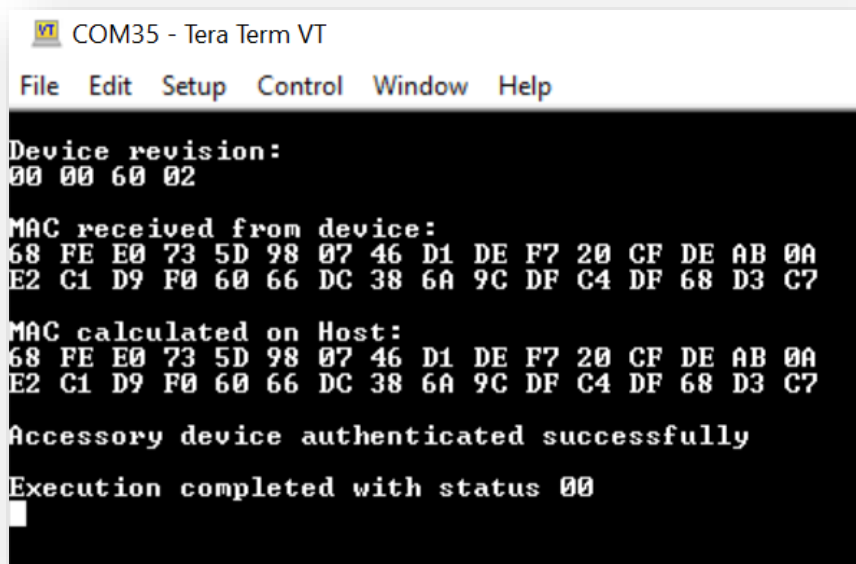
Programming complete
```

Once the programming is done, the firmware will do Accessory-Authentication operation. Depending on the Accessory-Authentication operation's output, the Crypto Auth Trust Platform board's status LED will blink at different rates.

If Accessory-Authentication operation **succeeds**, LED blinks once every second.

If Accessory-Authentication operation **fails**, LED blinks five times every second.

It is also possible to view the Console messages by using applications like TeraTerm. Open the application with the COM related to Crypto Auth Trust Platform with 115200-8-N-1 settings



```
VT COM35 - Tera Term VT
File Edit Setup Control Window Help

Device revision:
00 00 60 02

MAC received from device:
68 FE E0 73 5D 98 07 46 D1 DE F7 20 CF DE AB 0A
E2 C1 D9 F0 60 66 DC 38 6A 9C DF C4 DF 68 D3 C7

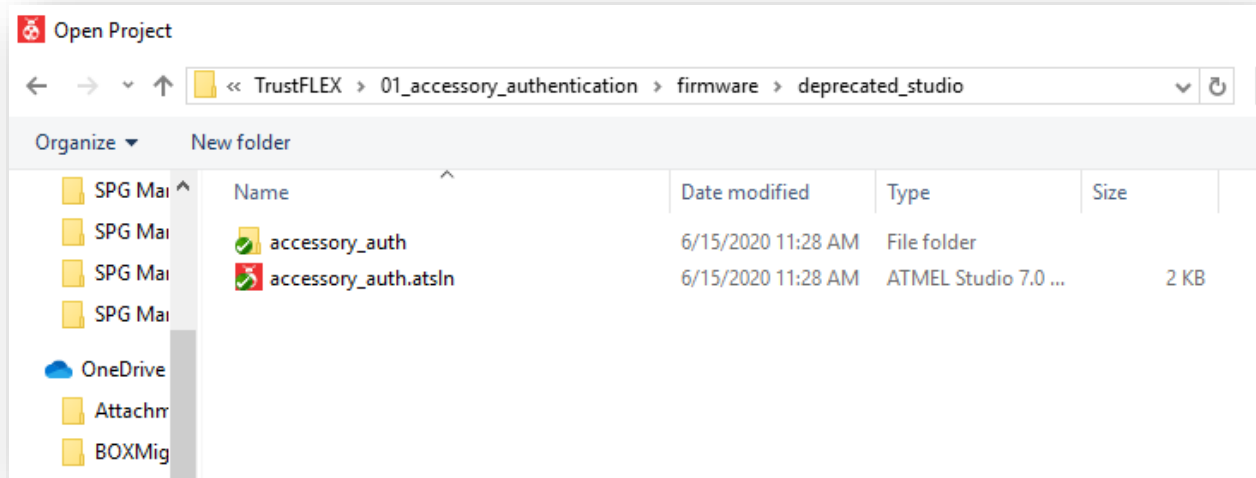
MAC calculated on Host:
68 FE E0 73 5D 98 07 46 D1 DE F7 20 CF DE AB 0A
E2 C1 D9 F0 60 66 DC 38 6A 9C DF C4 DF 68 D3 C7

Accessory device authenticated successfully

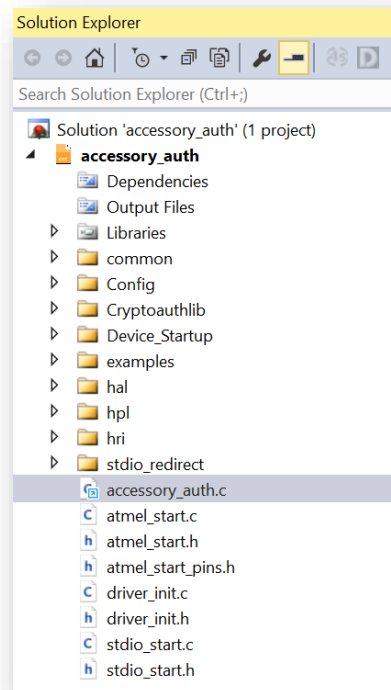
Execution completed with status 00
```

4.2.2 Atmel Studio (Deprecated)

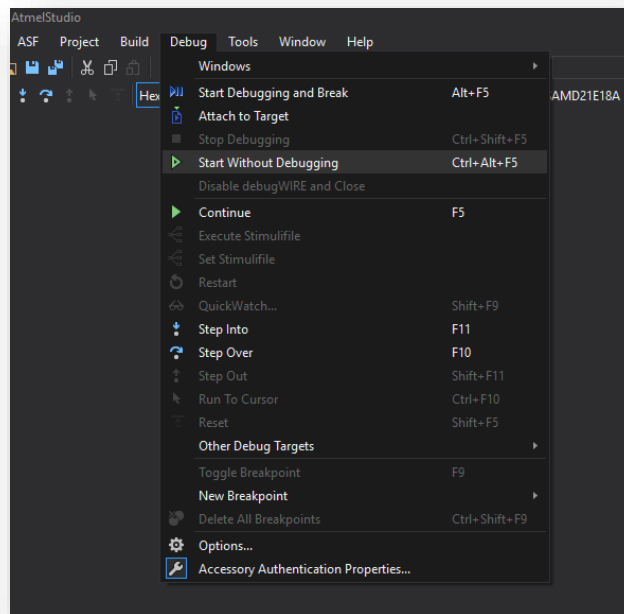
1. Open **accessory_auth.atsln** project by navigating to Atmel Studio -> File -> open -> **TrustFLEX\01_accessory_authentication\firmware\accessory_auth.atsln**



2. The application source code accessory_auth.c is available at **TrustFLEX\01_accessory_authentication\c\accessory_auth.c**. Other supporting files can be found under **assets\dependencies**



3. Program the Crypto Trust platform by navigating to **Debug -> Start Without Debugging**



This step may take some time, wait for Atmel Studio to compile and program the device.

Once the programming is done, the firmware will do Accessory-Authentication operation. Depending on the Accessory-Authentication operation's output, the Crypto Auth Trust Platform board's status LED will blink at different rates.

If Accessory-Authentication operation **succeeds**, LED blinks once every second.
If Accessory-Authentication operation **fails**, LED blinks five times every second.

It is also possible to view the Console messages by using applications like TeraTerm. Open the application with the COM related to Crypto Auth Trust Platform with 115200-8-N-1 settings

COM35 - Tera Term VT

File Edit Setup Control Window Help

Device revision:

00 00 60 02

MAC received from device:

68 FE E0 73 5D 98 07 46 D1 DE F7 20 CF DE AB 0A
E2 C1 D9 F0 60 66 DC 38 6A 9C DF C4 DF 68 D3 C7

MAC calculated on Host:

68 FE E0 73 5D 98 07 46 D1 DE F7 20 CF DE AB 0A
E2 C1 D9 F0 60 66 DC 38 6A 9C DF C4 DF 68 D3 C7

Accessory device authenticated successfully

Execution completed with status 00

4.3 Crypto Auth Trust Platform Factory reset

Once any of the embedded project is loaded to Crypto Auth Trust Platform, the default program that enables interaction with Trust Platform tools will be erased.

Before using the Platform with any other notebook or tools on PC, its required to reprogram the default .hex file. Default hex file is available in cloned repository at **assets\Factory_Program.X\CryptoAuth_Trust_Platform.hex**

If Trust Platform GUI is provided with MPLAB X IDE installation location, notebooks can program the Factory reset hex file if its not available by default.

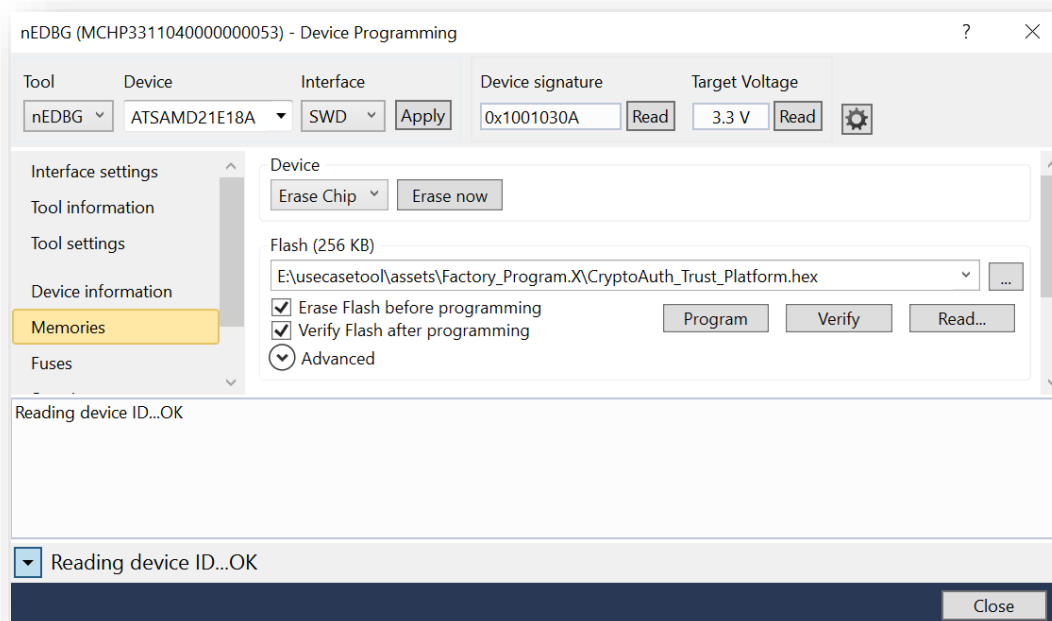
This can also be done manually by MPLAB and Atmel Studio

To reprogram using MPLAB:

1. Open **assets\Factory_Program.X** project in MPLAB IDE
2. Program the Crypto Trust platform by navigating to **CryptoAuth_Trust_Platform_Factory_Program -> Make and Program Device**

To reprogram using Atmel Studio:

1. Navigate to AtmelStudio -> Tools -> Device Programming
2. Select Tool as nEDBG and Apply
3. Go to Memories and navigate to above path under Flash dropdown
4. Check both Erase Flash and Verify Flash
5. Click on Program



Now, Crypto Auth Trust Platform contains factory programmed application that enables interactions with Notebooks and/or PC tools.

5 FAQ

1. What are the reasons for “**AssertionError: Can't connect to the USB dongle**” error?

There are many possibilities like,

1. Crypto Trust Platform is having different application than factory reset firmware. Refer to “Crypto Auth Trust Platform Factory reset” section any usecase TrustFLEX Guide for reloading it
2. Check the switch positions on Crypto Trust Platform and/or ATECC608B Trust board
 - a. Correct Trust device should be connected and only one device of that type is allowed on the I2C bus. Multiple devices with same address results in error
3. Check USB connections to Crypto Trust Platform

2. How to reload factory default application to Crypto Trust Platform?

Refer to “Crypto Auth Trust Platform Factory reset” section any usecase TrustFLEX Guide for reloading it.

3. Why does my C projects generates No such file or directory with ../../../00_resource_generation/?

C project generates this error when the resources are not generated prior to using embedded projects. Running the resource generation notebook ensures these files and secrets are generated.

4. Before running any use case notebook and/or C project, why is it mandate to execute resource generation?

When resource generation notebook is executed, it generates and programs the required resources like secrets, keys and certificates. These are only prototyping keys and cannot be used for production. These keys will be used part of Usecase notebooks and C projects

5. How to know the resources being used in a use case?

Refer to individual Usecase description html for details on transaction diagrams, resources being used and other details. The resources required for given use case is mentioned in INFER CRYPTOGRAPHIC ASSETS section.

6. When should I select Custom certificates while doing resource generation?

Custom certificates are required when user wants to have their own root, signer instead of MCHP provided. The difference would be organization name, common name and validity are configurable

7. How to know whether C project is executing on Trust Platform or not after programming?

Once the programming is done, the firmware will do use case operation. Depending on the use case operation's output, the Crypto Trust Platform board's status LED will blink at different rates.

If use case operation succeeds, LED blinks once every second. If it fails, LED blinks five times every second.

It is also possible to view the Console messages by using applications like TeraTerm.
Open the application with the COM related to Crypto Trust Platform with 115200-8-N-1 settings

The Microchip Web Site

Microchip provides online support via our web site at <http://www.microchip.com/>. This web site is used as

a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at <http://www.microchip.com/>. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support.

Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: <http://www.microchip.com/support>

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY

OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE.

Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq,

Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB,

OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST,

SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight

Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming,

ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient

Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE,

Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2018, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN:

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California

and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.



Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com Atlanta Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 Austin, TX Tel: 512-257-3370 Boston Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 Chicago Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 Dallas Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 Detroit Novi, MI Tel: 248-848-4000 Houston, TX Tel: 281-894-5983 Indianapolis Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 Los Angeles Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 Raleigh, NC Tel: 919-844-7510 New York, NY Tel: 631-435-6000 San Jose, CA Tel: 408-735-9110 Tel: 408-436-4270 Canada - Toronto Tel: 905-695-1980 Fax: 905-695-2078	Australia - Sydney Tel: 61-2-9868-6733 China - Beijing Tel: 86-10-8569-7000 China - Chengdu Tel: 86-28-8665-5511 China - Chongqing Tel: 86-23-8980-9588 China - Dongguan Tel: 86-769-8702-9880 China - Guangzhou Tel: 86-20-8755-8029 China - Hangzhou Tel: 86-571-8792-8115 China - Hong Kong SAR Tel: 852-2943-5100 China - Nanjing Tel: 86-25-8473-2460 China - Qingdao Tel: 86-532-8502-7355 China - Shanghai Tel: 86-21-3326-8000 China - Shenyang Tel: 86-24-2334-2829 China - Shenzhen Tel: 86-755-8864-2200 China - Suzhou Tel: 86-186-6233-1526 China - Wuhan Tel: 86-27-5980-5300 China - Xian Tel: 86-29-8833-7252 China - Xiamen Tel: 86-592-2388138 China - Zhuhai Tel: 86-756-3210040	India - Bangalore Tel: 91-80-3090-4444 India - New Delhi Tel: 91-11-4160-8631 India - Pune Tel: 91-20-4121-0141 Japan - Osaka Tel: 81-6-6152-7160 Japan - Tokyo Tel: 81-3-6880-3770 Korea - Daegu Tel: 82-53-744-4301 Korea - Seoul Tel: 82-2-554-7200 Malaysia - Kuala Lumpur Tel: 60-3-7651-7906 Malaysia - Penang Tel: 60-4-227-8870 Philippines - Manila Tel: 63-2-634-9065 Singapore Tel: 65-6334-8870 Taiwan - Hsin Chu Tel: 886-3-577-8366 Taiwan - Kaohsiung Tel: 886-7-213-7830 Taiwan - Taipei Tel: 886-2-2508-8600 Thailand - Bangkok Tel: 66-2-694-1351 Vietnam - Ho Chi Minh Tel: 84-28-5448-2100	Austria - Wels Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 Denmark - Copenhagen Tel: 45-4450-2828 Fax: 45-4485-2829 Finland - Espoo Tel: 358-9-4520-820 France - Paris Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 France - Saint Cloud Tel: 33-1-30-60-70-00 Germany - Garching Tel: 49-8931-9700 Germany - Haan Tel: 49-2129-3766400 Germany - Heilbronn Tel: 49-7131-67-3636 Germany - Karlsruhe Tel: 49-721-625370 Germany - Munich Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 Germany - Rosenheim Tel: 49-8031-354-560 Israel - Ra'anana Tel: 972-9-744-7705 Italy - Milan Tel: 39-0331-742611 Fax: 39-0331-466781 Italy - Padova Tel: 39-049-7625286 Netherlands - Drunen Tel: 31-416-690399 Fax: 31-416-690340 Norway - Trondheim Tel: 47-7289-7561 Poland - Warsaw Tel: 48-22-3325737 Romania - Bucharest Tel: 40-21-407-87-50 Spain - Madrid Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 Sweden - Gothenberg Tel: 46-31-704-60-40 Sweden - Stockholm Tel: 46-8-5090-4654 UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820