TrustFLEX Step by Step Guide
Secure Public key Rotation

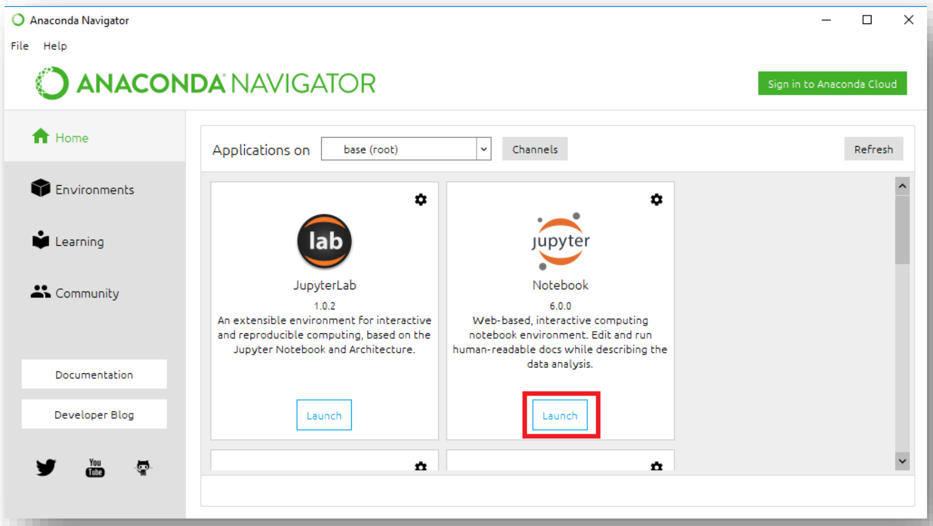
Table of Contents

1 Introduction
This document helps in running Secure Public Key rotation use case example provided in TrustFLEX package. 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 the Anaconda Navigator main window.



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

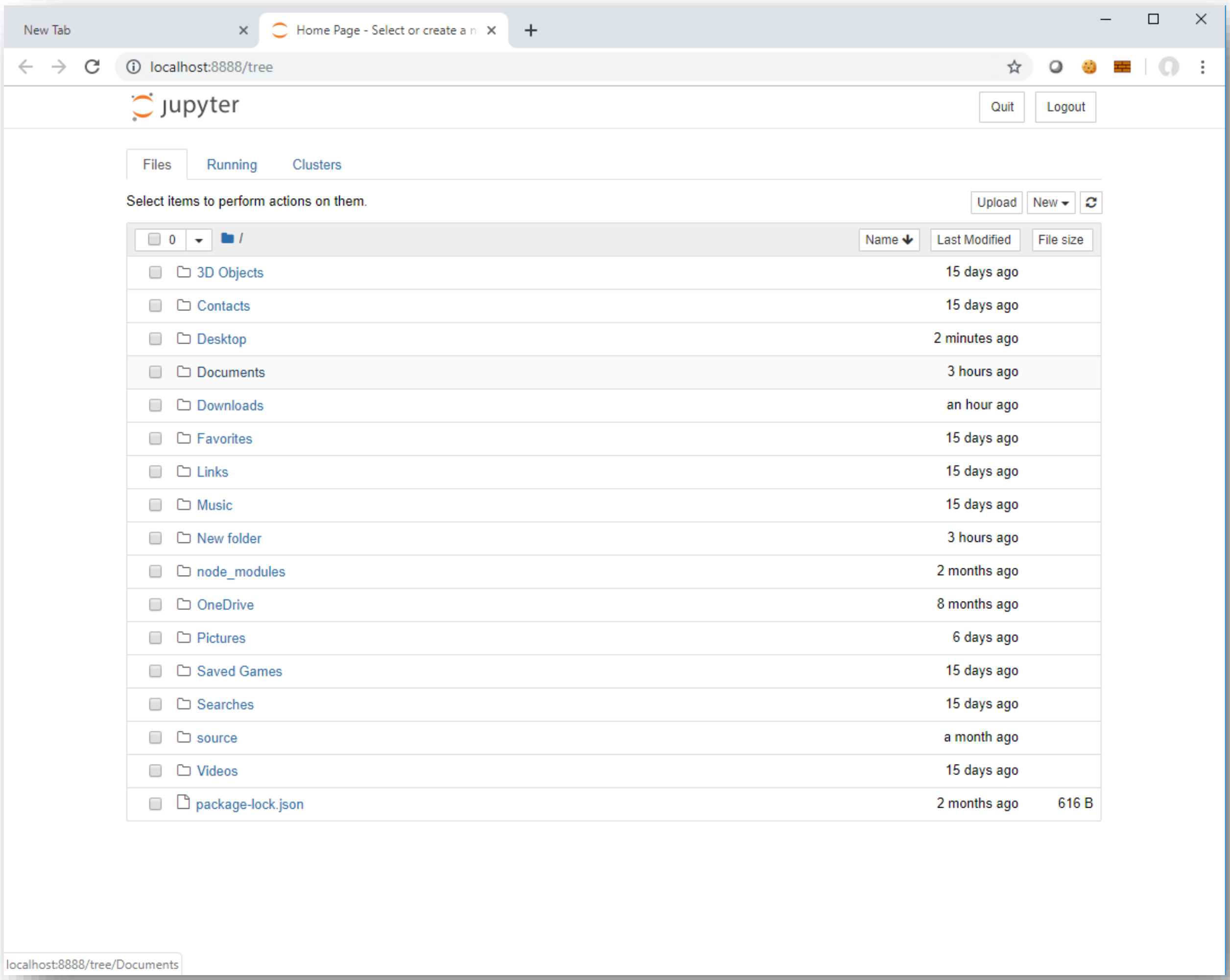
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

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.



Select items to perform actions on them.

Name	Last Modified	File size
3D Objects	15 days ago	
Contacts	15 days ago	
Desktop	2 minutes ago	
Documents	3 hours ago	
Downloads	an hour ago	
Favorites	15 days ago	
Links	15 days ago	
Music	15 days ago	
New folder	3 hours ago	
node_modules	2 months ago	
OneDrive	8 months ago	
Pictures	6 days ago	
Saved Games	15 days ago	
Searches	15 days ago	
source	a month ago	
Videos	15 days ago	
package-lock.json	2 months ago	616 B

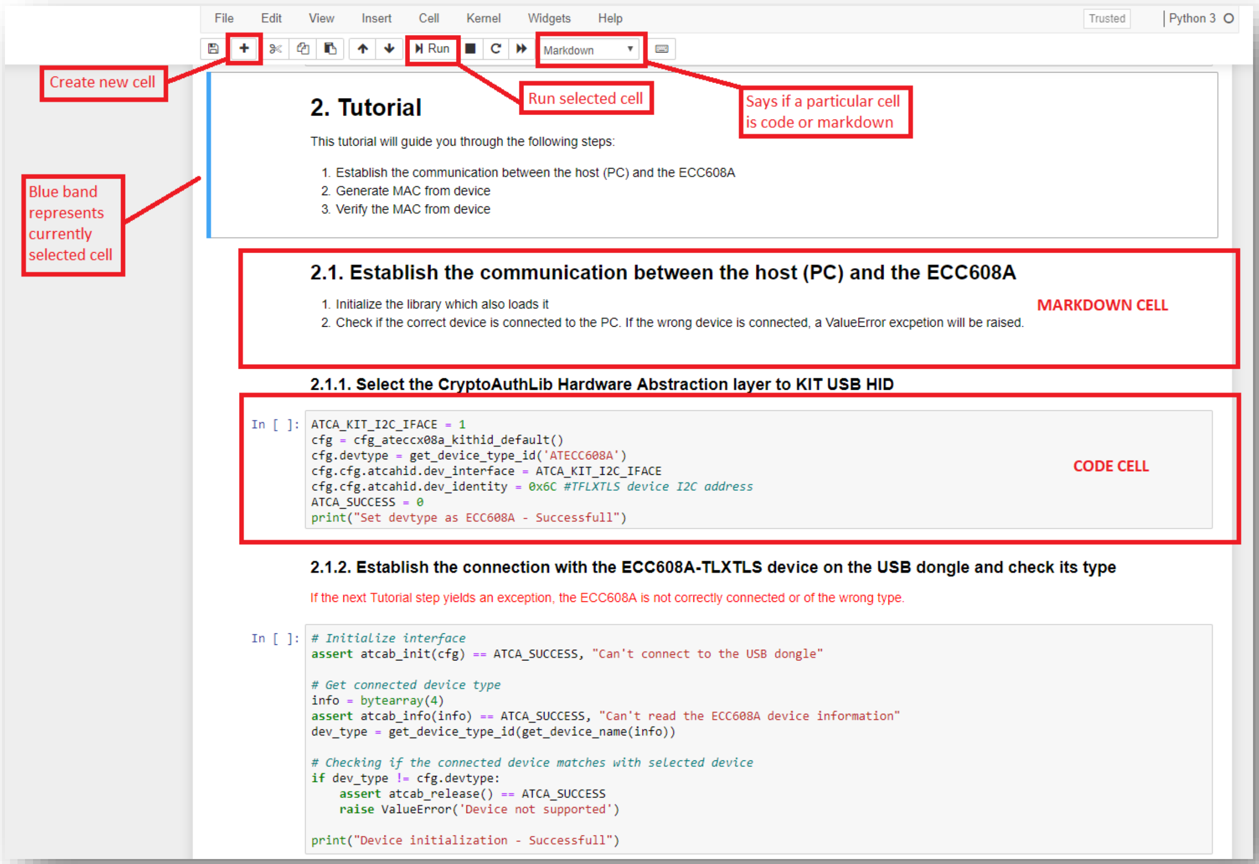
localhost:8888/tree/Documents

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**.



2. Tutorial

This tutorial will guide you through the following steps:

1. Establish the communication between the host (PC) and the ECC608A
2. Generate MAC from device
3. Verify the MAC from device

2.1. Establish the communication between the host (PC) and the ECC608A

1. Initialize the library which also loads it
2. Check if the correct device is connected to the PC. If the wrong device is connected, a ValueError excpetion will be raised.

2.1.1. Select the CryptoAuthLib Hardware Abstraction layer to KIT USB HID

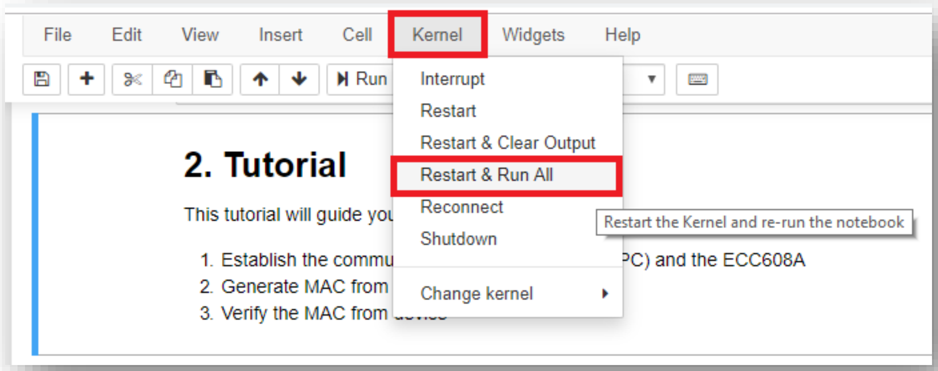
```
In [ ]: ATCA_KIT_I2C_IFACE = 1
        cfg = cfg_ateccx08a_kithid_default()
        cfg.devtype = get_device_type_id('ATECC608A')
        cfg.cfg.atcahid.dev_interface = ATCA_KIT_I2C_IFACE
        cfg.cfg.atcahid.dev_identity = 0x6C #TFLXTLS device I2C address
        ATCA_SUCCESS = 0
        print("Set devtype as ECC608A - Successfull")
```

2.1.2. Establish the connection with the ECC608A-TLXTLS device on the USB dongle and check its type

If the next Tutorial step yields an exception, the ECC608A is not correctly connected or of the wrong type.

```
In [ ]: # Initialize interface
        assert atcab_init(cfg) == ATCA_SUCCESS, "Can't connect to the USB dongle"

        # Get connected device type
        info = bytearray(4)
        assert atcab_info(info) == ATCA_SUCCESS, "Can't read the ECC608A device information"
        dev_type = get_device_type_id(get_device_name(info))

        # Checking if the connected device matches with selected device
        if dev_type != cfg.devtype:
            assert atcab_release() == ATCA_SUCCESS
            raise ValueError('Device not supported')

        print("Device initialization - Successfull")
```

To run all cells in sequence.



2 Jupyter Notebook Tutorials

The TrustPlatform Design Suite comes with a Notebook Tutorials to easily prototype popular use cases for TrustFLEX devices. Here is the Jupyter Notebook Tutorials.

Jupyter Notebook Tutorials	**Relative Path**	**Applicable devices**
Manifest Generation	TNGTLS_Manifest_Generation\notebooks\TNGTLS Manifest File Generation.ipynb	Trust&GO
Resource Generation	TFLXTLS_resource_generation\Crypto Resource Generator.ipynb	TrustFLEX
Accessory Authentication	TFLXTLS_Use_Cases\notebooks\accessory-authentication\Accessory Authentication.ipynb	TrustFLEX
AWS Custom PKI	TFLXTLS_Use_Cases\notebooks\aws-iot\aws-iot with ECC608A-TLFXTLS.ipynb	TrustFLEX
Firmware Validation	TFLXTLS_Use_Cases\notebooks\firmware-validation\Firmware Validation with ECC608A-TFLXTLS Tutorial.ipynb	TrustFLEX
IP Protection	TFLXTLS_Use_Cases\notebooks\ipprotection\IP Protection with ECC608A-TFLXTLS Tutorial.ipynb	TrustFLEX
Secure Public Key Rotation	TFLXTLS_Use_Cases\notebooks\public-key-rotation\Public Key Rotation with ECC608A-TFLXTLS Tutorial.ipynb	TrustFLEX

3 Resource Generation Notebook

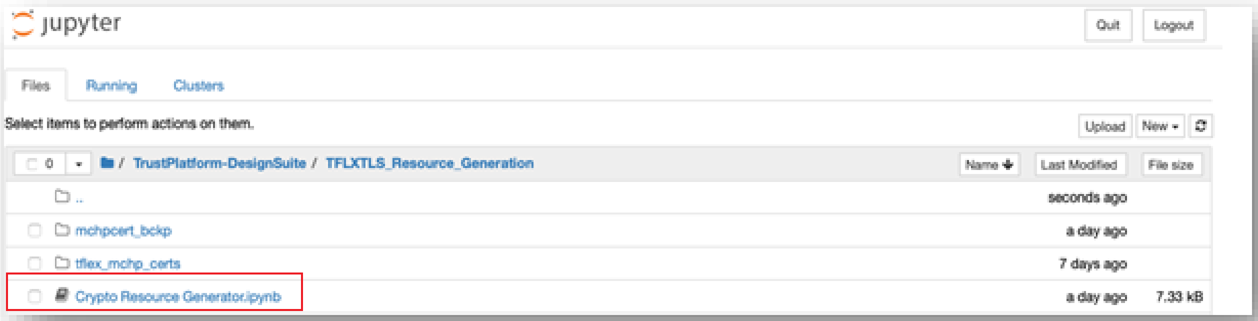
TFLXTLS device is one of the three devices available in the Trust Platform USB Dongle Board.

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.

By default, Jupyter starts in Users directory ($HOME for MacOS or Linux systems). For the remainder of this document, it will be assumed that the trust_platform folder is contained in Users directory. If this is not the case, please move trust_platform folder to your Users directory

Within the Jupyter Dashboard, navigate trust_platform\DesignTools\TFLXTLS_Resource_Generation folder to open Crypto Resource Generator.ipynb notebook



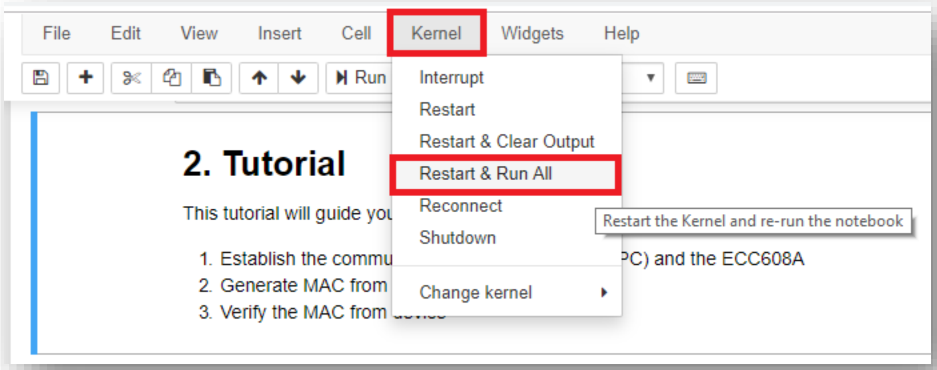
Select items to perform actions on them.

/ TrustPlatform-DesignSuite / TFLXTLS_Resource_Generation

Name	Last Modified	File size
..	seconds ago	
mchpcert_bckp	a day ago	
tflex_mchp_certs	7 days ago	
Crypto Resource Generator.ipynb	a day ago	7.33 kB

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

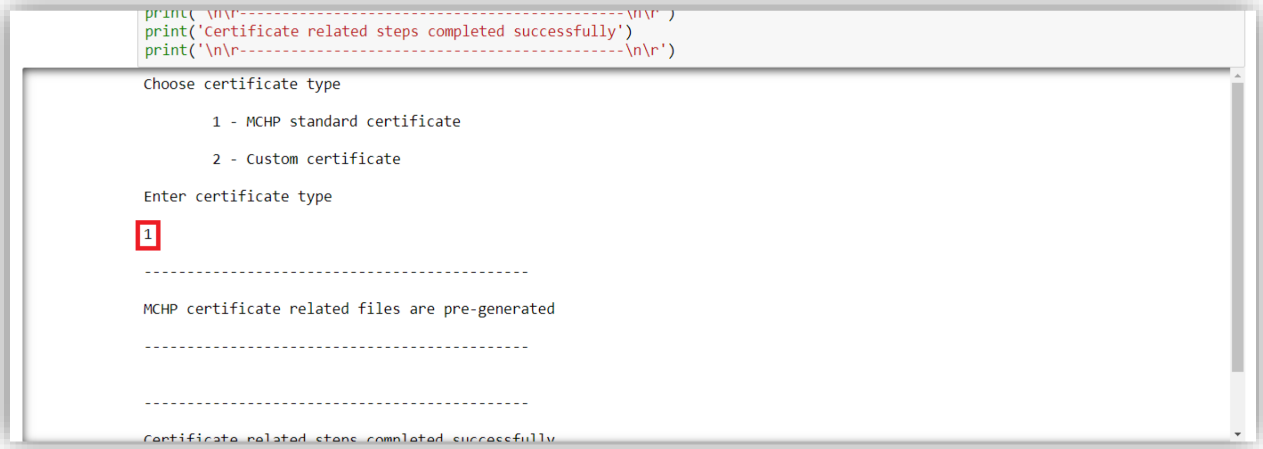
Note: Before executing the cells on Crypto Trust Platform, its required to have factory default program running on SAMD21 of Trust Platform. Refer to 4.3 CryptoAuth TrustPlatform Factory reset section for reloading default program.



Crypto Resource Generator notebook is common for all the use case which comes with option to load the signer certificate and device certificate.

So, it will execute and prompt you to choose between MCHP certificate and a custom certificate chain, enter '1' (for MCHP certificate) and press Enter key for this use case.

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.



```
print('\n\r--------------------------------------------\n\r')
print('Certificate related steps completed successfully')
print('\n\r--------------------------------------------\n\r')
```

```
Choose certificate type

        1 - MCHP standard certificate

        2 - Custom certificate

Enter certificate type

1

--------------------------------------------

MCHP certificate related files are pre-generated

--------------------------------------------


--------------------------------------------

Certificate related steps completed successfully
```

The output log should look like this.

```
Choose certificate type

        1 - MCHP standard certificate

        2 - Custom certificate

Enter certificate type

1

--------------------------------------------

MCHP certificate related files are pre-generated

--------------------------------------------


--------------------------------------------

Certificate related steps completed successfully

--------------------------------------------
```

The Notebook will also generate a manifest file to be uploaded into the public cloud of your choice (Google GCP, AWS IoT and soon to be supported 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, Validation Authority public key and a public key are loaded into TrustFLEX device in slot 13 and 14 respectively.

4 Use Case Prototyping

This hands-on lab is intended to demonstrate how to securely rotate public key in the TrustFLEX device. This uses the PubInvalid feature of the Slot configuration.

Secure Public Key Rotation is sequence of steps for host to update the public key in the TrustFLEX device securely. The basic flow to update the rotating public key is as follows:

1. Invalidate existing public key
2. Update (write) New rotating public key
3. Validate new rotating public key

To perform a validation/invalidation process, its required to have a validation authority. Typically, Validation Authority public key will be loaded to secure element and locked permanently.

This validation/invalidation process includes the authority signing the public key’s digest and generating the signature. Authority’s public key will be used for verifying the signature with second slot public key’s digest.

To update a new rotating public key in the TrustFLEX device, first we need to invalidate the existing public key. By invalidating it, the slot becomes available for write operation and a new public key can be written to it. However, the slot cannot be used for cryptographic operations until its validated.

TrustFLEX device two slots being used for key rotation sequence,

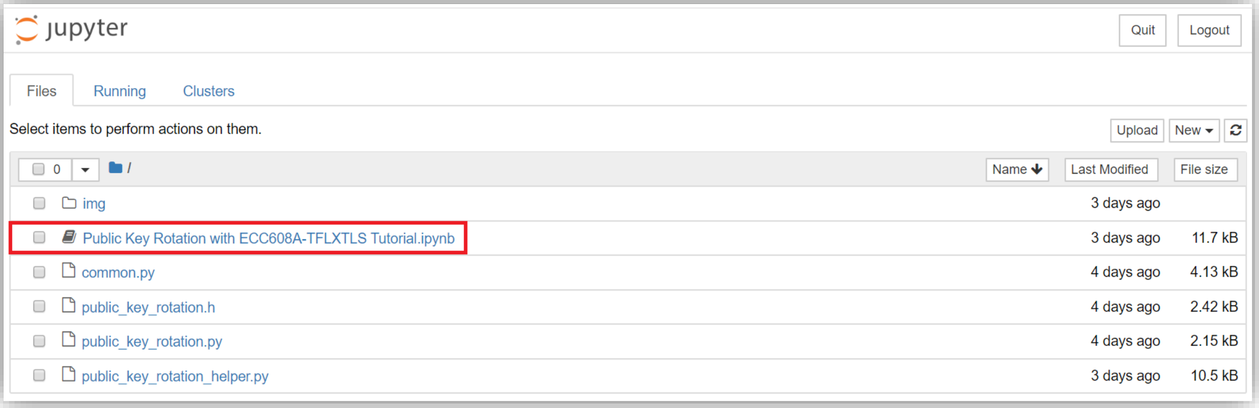
- Validation Authority public key – slot 13
- Rotating public key – slot 14

The resource generation for TrustFLEX device will generate and load prototyping validation authority and required public keys to TrustFLEX device.

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

4.1 Running Public Key Rotation example on Jupyter Notebook:

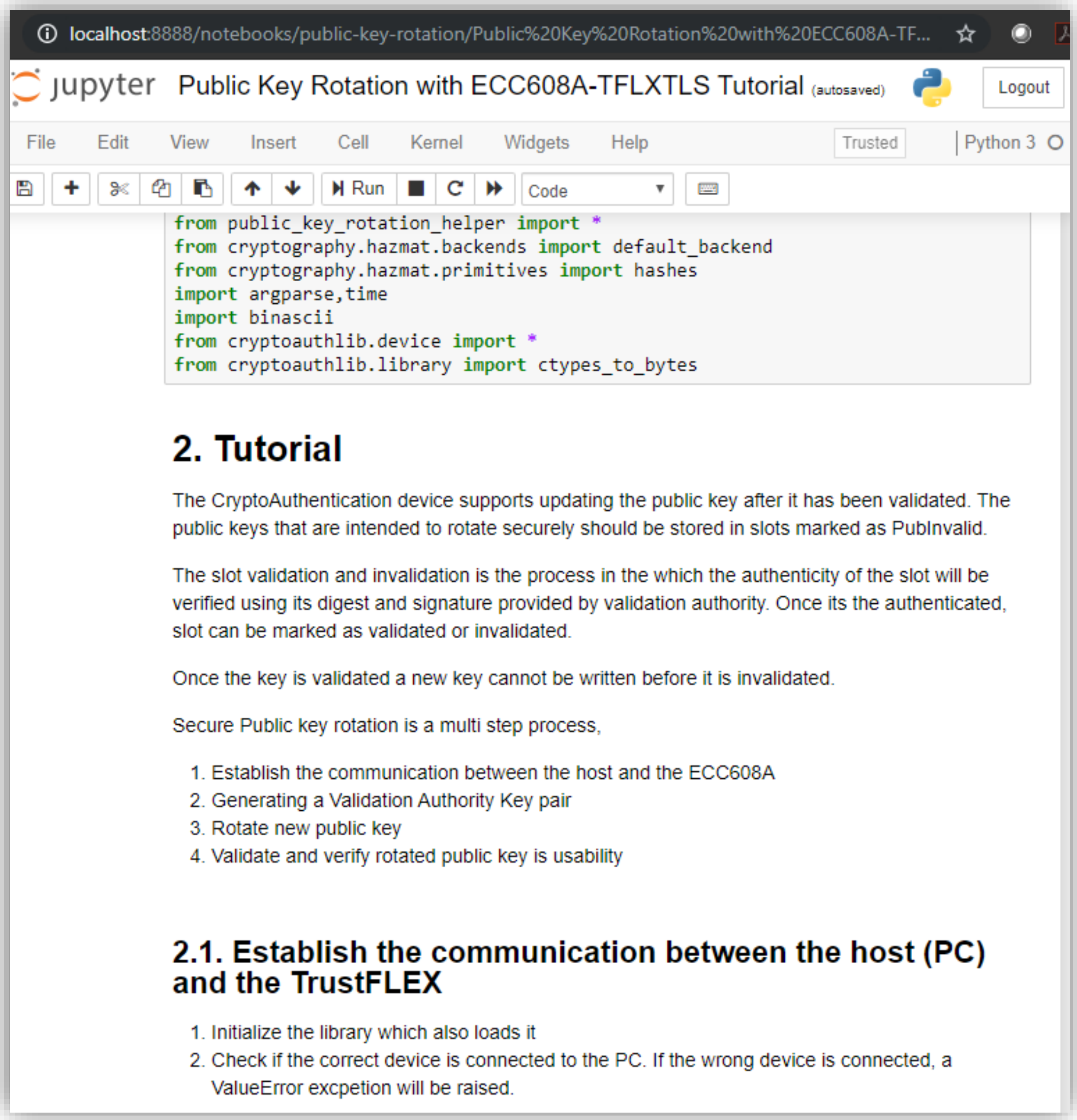
1. From the Jupyter Home page, navigate to **TFLXTLS_Use_Cases\notebooks\public-key-rotation\Public Key Rotation with ECC608A-TFLXTLS Tutorial.ipynb** notebook file and open it.



Select items to perform actions on them.

img	3 days ago	
Public Key Rotation with ECC608A-TFLXTLS Tutorial.ipynb	3 days ago	11.7 kB
common.py	4 days ago	4.13 kB
public_key_rotation.h	4 days ago	2.42 kB
public_key_rotation.py	4 days ago	2.15 kB
public_key_rotation_helper.py	3 days ago	10.5 kB

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


localhost:8888/notebooks/public-key-rotation/Public%20Key%20Rotation%20with%20ECC608A-TF...
jupyter Public Key Rotation with ECC608A-TFLXTLS Tutorial (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```python
from public_key_rotation_helper import *
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
import argparse,time
import binascii
from cryptoauthlib.device import *
from cryptoauthlib.library import ctypes_to_bytes
```

2. Tutorial

The CryptoAuthentication device supports updating the public key after it has been validated. The public keys that are intended to rotate securely should be stored in slots marked as PubInvalid.

The slot validation and invalidation is the process in the which the authenticity of the slot will be verified using its digest and signature provided by validation authority. Once its the authenticated, slot can be marked as validated or invalidated.

Once the key is validated a new key cannot be written before it is invalidated.

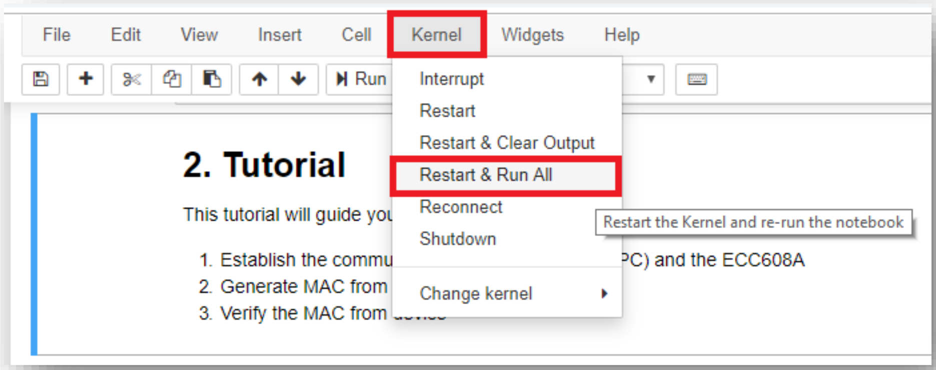
Secure Public key rotation is a multi step process,

1. Establish the communication between the host and the ECC608A
2. Generating a Validation Authority Key pair
3. Rotate new public key
4. Validate and verify rotated public key is usability

2.1. Establish the communication between the host (PC) and the TrustFLEX

1. Initialize the library which also loads it
2. Check if the correct device is connected to the PC. If the wrong device is connected, a ValueError excpetion will be raised.

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

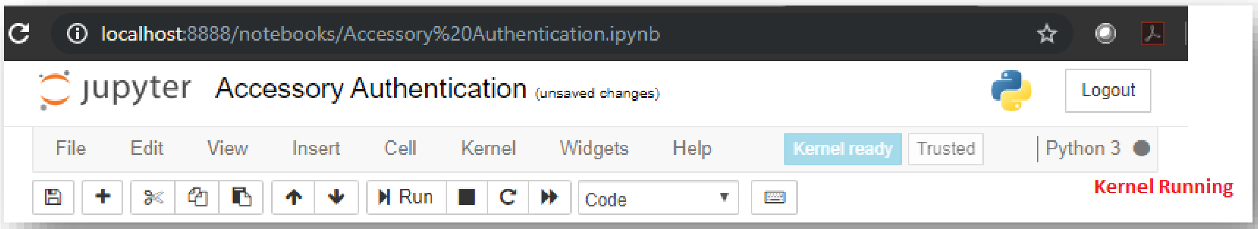

File Edit View Insert Cell Kernel Widgets Help
Run
Interrupt
Restart
Restart & Clear Output
Restart & Run All
Reconnect
Shutdown
Change kernel
Restart the Kernel and re-run the notebook

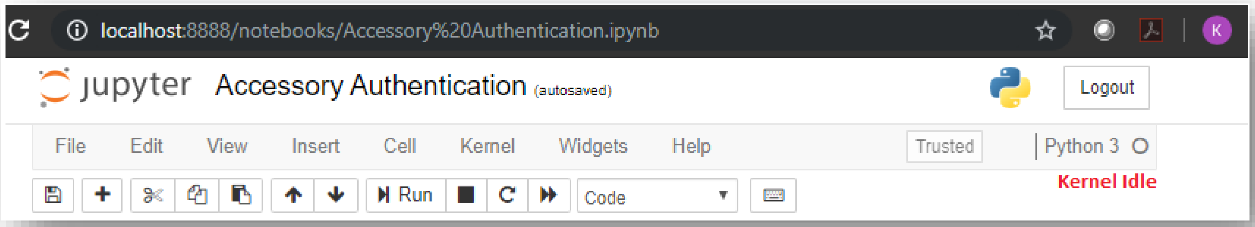
2. Tutorial

This tutorial will guide you

1. Establish the commu PC) and the ECC608A
2. Generate MAC from
3. Verify the MAC from device

3. 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)


localhost:8888/notebooks/Accessory%20Authentication.ipynb
jupyter Accessory Authentication (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Kernel ready
Trusted
Python 3
Run
Code
Kernel Running

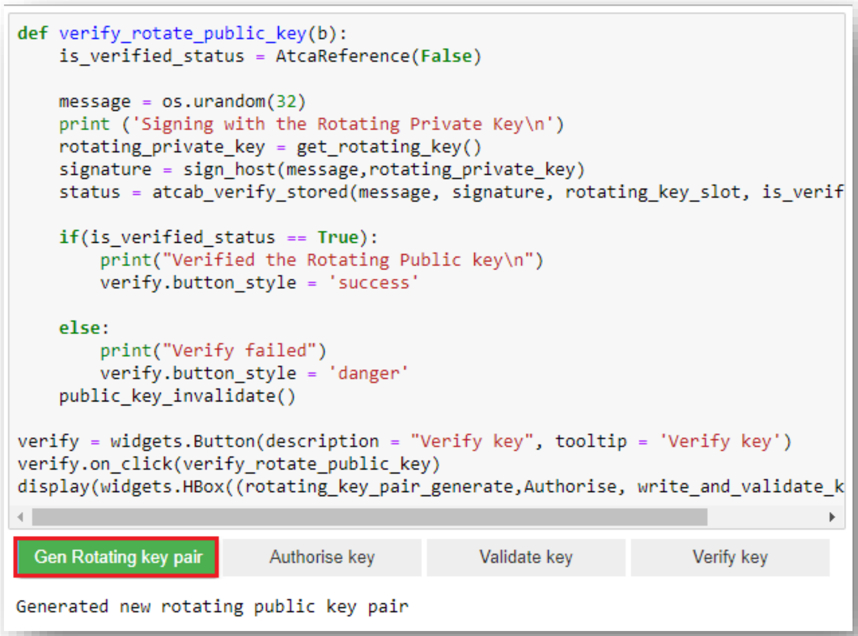

localhost:8888/notebooks/Accessory%20Authentication.ipynb
jupyter Accessory Authentication (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Kernel Idle

4. Navigate through different cells output for the description of the step and result from the execution.
5. There are 6 major steps in this lab
   This lab is setup to generate multiple button at the end of the example. These buttons perform the tasks listed below.

   <u>Generating a Validation Authority key pair</u>
   This step setups a temporary validation authority to perform slot validation and validation process. This is already taken care part of resource generation

   <u>Generating a new public key</u>
   This step generates a new key pair to rotate the existing public key in the slot. This will be the new public key attempted to load in to PubInvalid slot i.e. for secure public key rotation.

   Click on "Generate Key Pair" button at the bottom of the Notebook. This generates new key to update the existing slot. The button turns RED if there is any error in the execution.

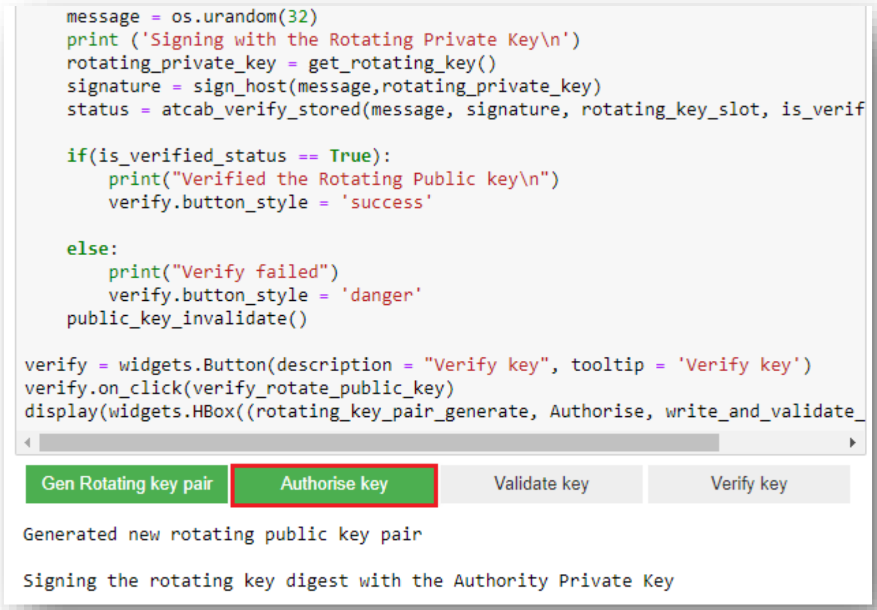

```
def verify_rotate_public_key(b):
    is_verified_status = AtcaReference(False)

    message = os.urandom(32)
    print ('Signing with the Rotating Private Key\n')
    rotating_private_key = get_rotating_key()
    signature = sign_host(message,rotating_private_key)
    status = atcab_verify_stored(message, signature, rotating_key_slot, is_verif

    if(is_verified_status == True):
        print("Verified the Rotating Public key\n")
        verify.button_style = 'success'

    else:
        print("Verify failed")
        verify.button_style = 'danger'
    public_key_invalidate()

verify = widgets.Button(description = "Verify key", tooltip = 'Verify key')
verify.on_click(verify_rotate_public_key)
display(widgets.HBox((rotating_key_pair_generate,Authorise, write_and_validate_k
```

```
Generated new rotating public key pair
```

Authorize public key

Before the new key to be used for writing into the slot, this should be validated by validation authority. In this step, validation authority calculates the Pubkey's digest and signs using its private key.

Click on "Authorize key" button at the bottom of the Notebook. This authorizes the new public key using validation authority. The button turns RED if there is any error in the execution.


```
    message = os.urandom(32)
    print ('Signing with the Rotating Private Key\n')
    rotating_private_key = get_rotating_key()
    signature = sign_host(message,rotating_private_key)
    status = atcab_verify_stored(message, signature, rotating_key_slot, is_verif

    if(is_verified_status == True):
        print("Verified the Rotating Public key\n")
        verify.button_style = 'success'

    else:
        print("Verify failed")
        verify.button_style = 'danger'
    public_key_invalidate()

verify = widgets.Button(description = "Verify key", tooltip = 'Verify key')
verify.on_click(verify_rotate_public_key)
display(widgets.HBox((rotating_key_pair_generate, Authorise, write_and_validate_
```

```
Generated new rotating public key pair

Signing the rotating key digest with the Authority Private Key
```

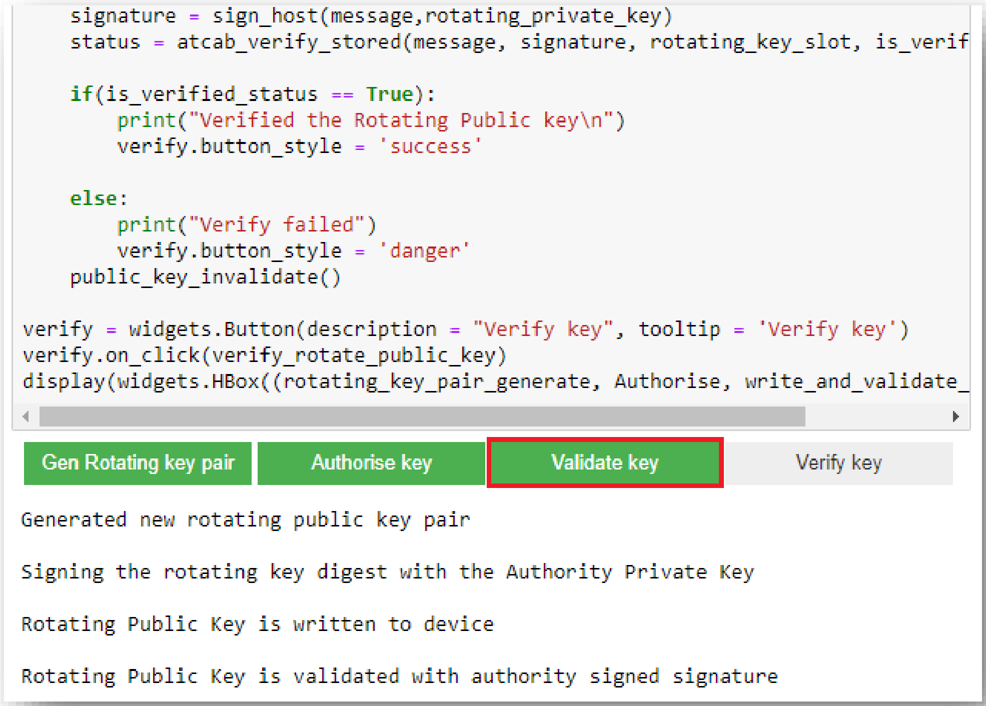
Update public key
This is the step where actual slot update happens. Before updating the existing slot should be invalidated using existing public key digest and signature provided by the validation authority. This signature should be of existing (old) public key, but not new public key.

Once the slot is invalidated, new public key can be overwritten to this slot. After writing it successfully, the slot remains in invalidated state and doesn’t allow any cryptographic operations.

Validate the rotating public key
This step does the slot validation after writing the new public key into PubInvalid slot i.e. slot14. Unless the slot is validated this cannot be used for cryptographic operations like Verify.

The process of validation involves the Public Key digest, Signature provided by validation authority. During this process, TrustFLEX device initiates internal Public key digest calculation on Slot14. Once the Digest is generated atcab_verify_validate will be issued with Slot number and Signature as parameters. On the successful match of digest and signature, the slot will be marked as PubInvalid. This restricts further writes to slot, but enables cryptographic operations using this slot.

Click on “Validate Key” button at the bottom of the Notebook to perform Public key authorization, updating to slot and validating the new public key. The button turns RED if there is any error in the execution.



```
    signature = sign_host(message,rotating_private_key)
    status = atcab_verify_stored(message, signature, rotating_key_slot, is_verif

    if(is_verified_status == True):
        print("Verified the Rotating Public key\n")
        verify.button_style = 'success'

    else:
        print("Verify failed")
        verify.button_style = 'danger'
    public_key_invalidate()

verify = widgets.Button(description = "Verify key", tooltip = 'Verify key')
verify.on_click(verify_rotate_public_key)
display(widgets.HBox((rotating_key_pair_generate, Authorise, write_and_validate_
```

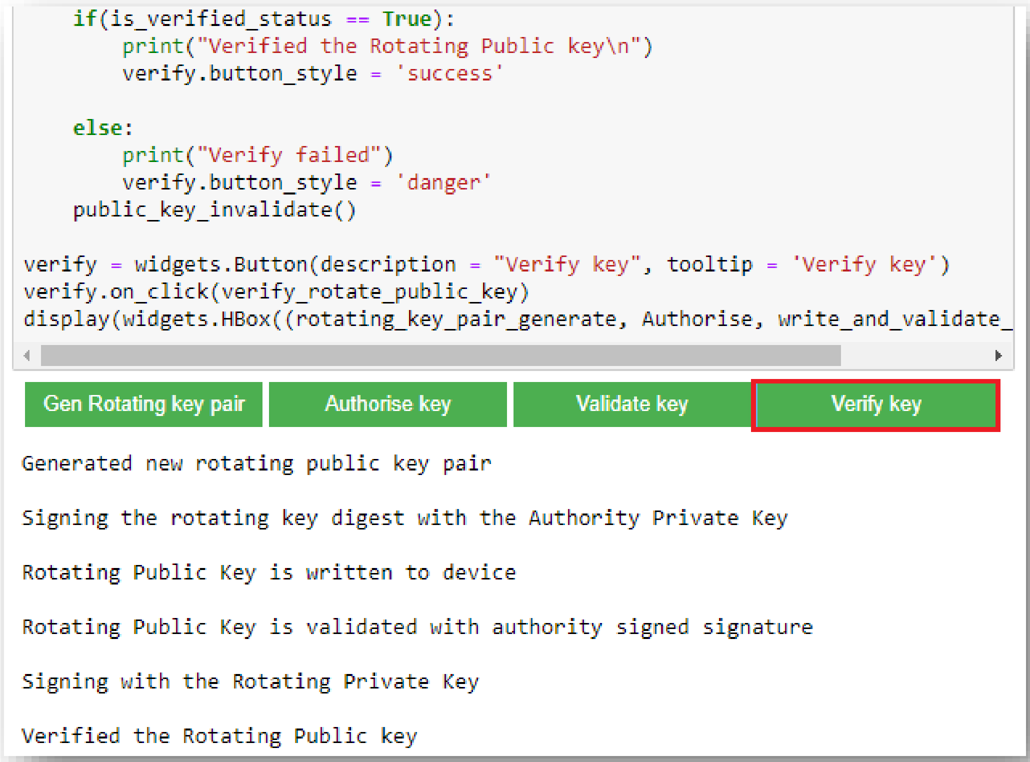
```
Generated new rotating public key pair

Signing the rotating key digest with the Authority Private Key

Rotating Public Key is written to device

Rotating Public Key is validated with authority signed signature
```

Verify the rotating public key

Once rotating public key is validated, then the rotating public key can be used for ECC operations. To verify that rotating public key availability for ECC operations, here we perform a sign and verify ECC operation.

First generate a temporary message digest and sign by private key corresponding to new/rotated public key. Then the signature is verified by using rotating public key. If it is verified, then the rotating public key is available for ECC operations or else not.

Click on “Verify Key” button at the bottom of the Notebook to perform verify operation using rotated key. The button turns green if rotated public key can be used for ECC operations or it will turn RED.



```
    if(is_verified_status == True):
        print("Verified the Rotating Public key\n")
        verify.button_style = 'success'

    else:
        print("Verify failed")
        verify.button_style = 'danger'
    public_key_invalidate()

verify = widgets.Button(description = "Verify key", tooltip = 'Verify key')
verify.on_click(verify_rotate_public_key)
display(widgets.HBox((rotating_key_pair_generate, Authorise, write_and_validate_
```

```
Generated new rotating public key pair

Signing the rotating key digest with the Authority Private Key

Rotating Public Key is written to device

Rotating Public Key is validated with authority signed signature

Signing with the Rotating Private Key

Verified the Rotating Public key
```

4.2 Running Public Key Rotation on Embedded platform

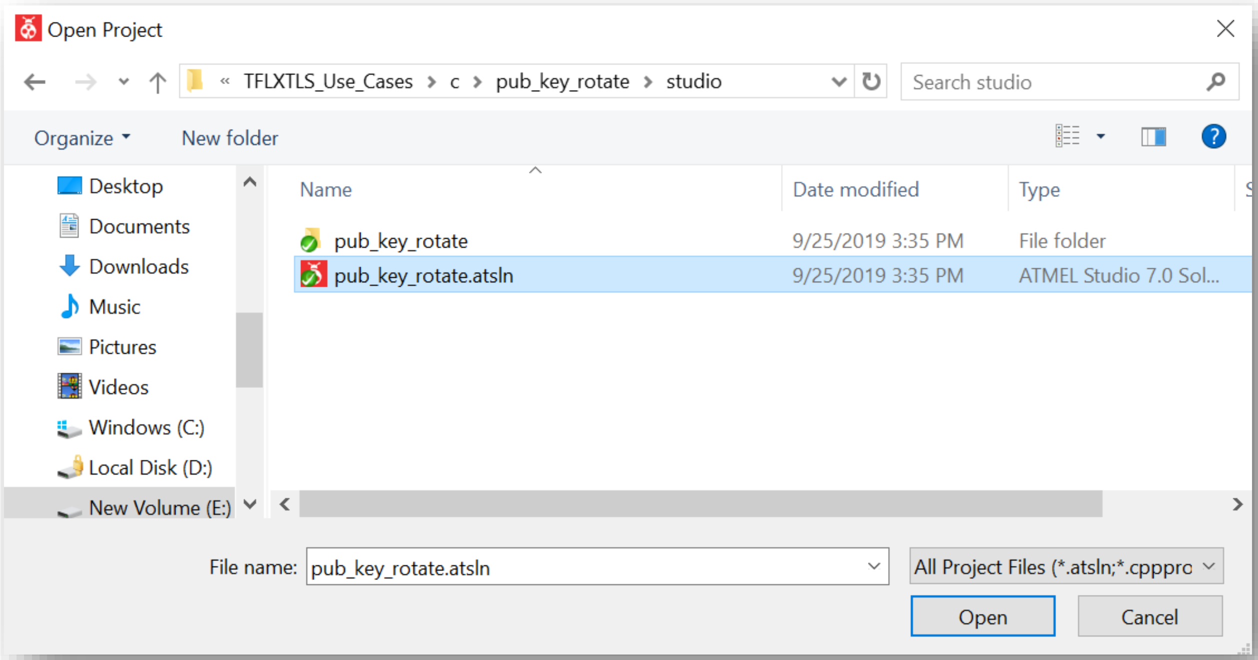
This usecase can also be executed on Embedded platform. Once the resources are generated, both Atmel Studio and MPLAB projects provided can be used to run the application on CryptoAuth Trust Platform.

This project can only perform Public key rotation steps, but not key generations for validation authority and new public key. It is **required** to use Public Key Rotation Notebook to generate new public key and get it authorized by validation authority. This notebook generates supporting data files like public keys, signatures and nonces for C projects.

Once the new public key is generated and authorized, these embedded projects can be used to rotate the current public key to the new one.

4.2.1 Atmel Studio:

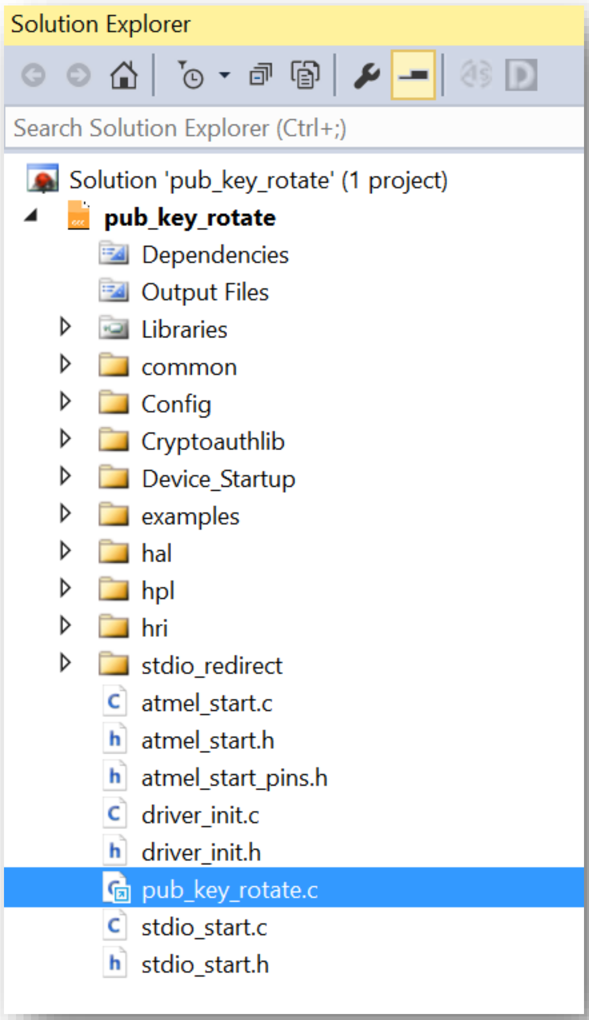
1. Open **pub_key_rotate.atsln** project by navigating to Atmel Studio -> File -> open-> **TFLXTLS_Use_Cases\c\pub_key_rotate\studio\pub_key_rotate.atsln**



Name	Date modified	Type
pub_key_rotate	9/25/2019 3:35 PM	File folder
pub_key_rotate.atsln	9/25/2019 3:35 PM	ATMEL Studio 7.0 Sol...

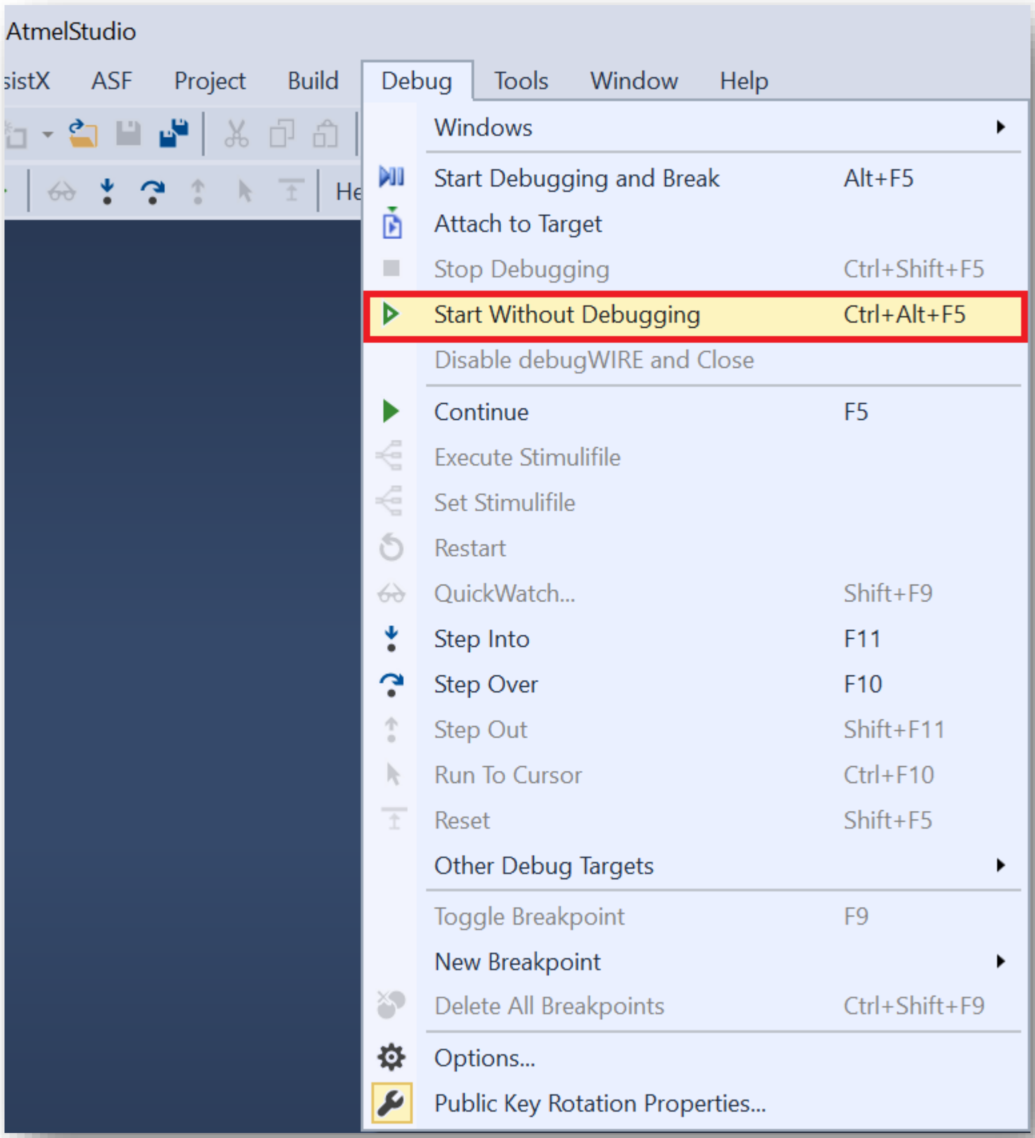
File name: pub_key_rotate.atsln

2. The application source code **pub_key_rotate.c** is available at **TFLXTLS_Use_Cases\c\pub_key_rotate\pub_key_rotate.c.** Other supporting files can be found under **TFLXTLS_Use_Cases\c\ dependencies**



Solution 'pub_key_rotate' (1 project)
- pub_key_rotate
  - Dependencies
  - Output Files
  - Libraries
  - common
  - Config
  - Cryptoauthlib
  - Device_Startup
  - examples
  - hal
  - hpl
  - hri
  - stdio_redirect
  - atmel_start.c
  - atmel_start.h
  - atmel_start_pins.h
  - driver_init.c
  - driver_init.h
  - pub_key_rotate.c
  - stdio_start.c
  - stdio_start.h

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

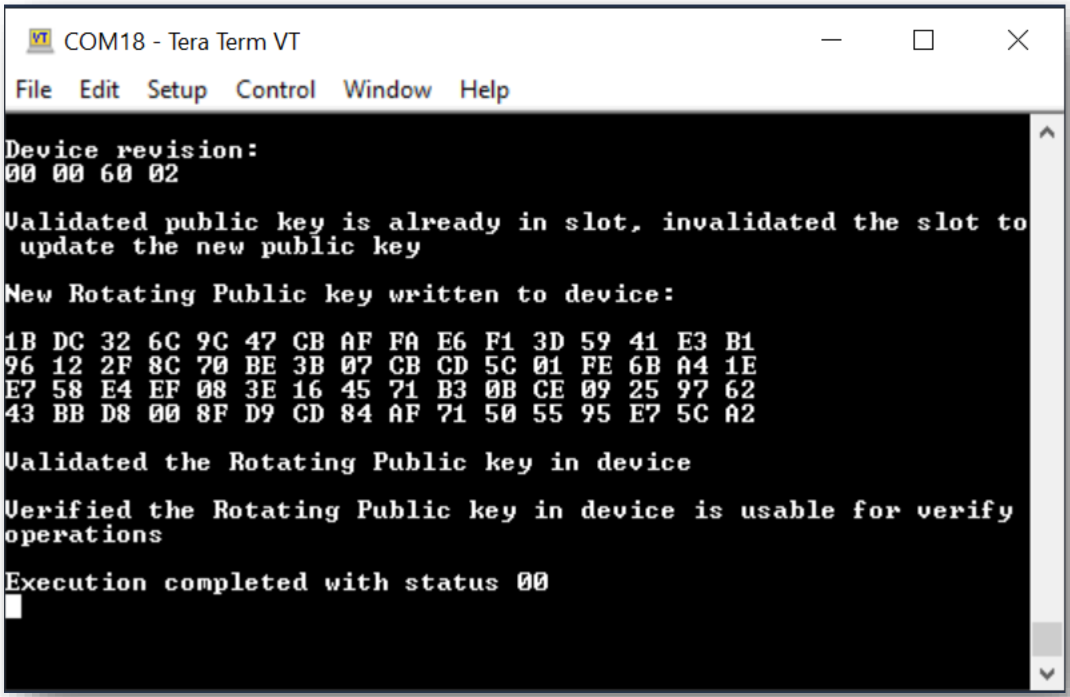


This step may take some time, wait for Atmel Studio to compile and program the device.

Once the programming is done, the firmware will do public key rotation operation. Depending on the public key rotation operation's output, the Cryptoauth Trust Platform board's Status LED will blink at different rates.

If public key rotation operation succeeds, LED blinks once every second.
If public key rotation operation fails, LED blinks five times every second.

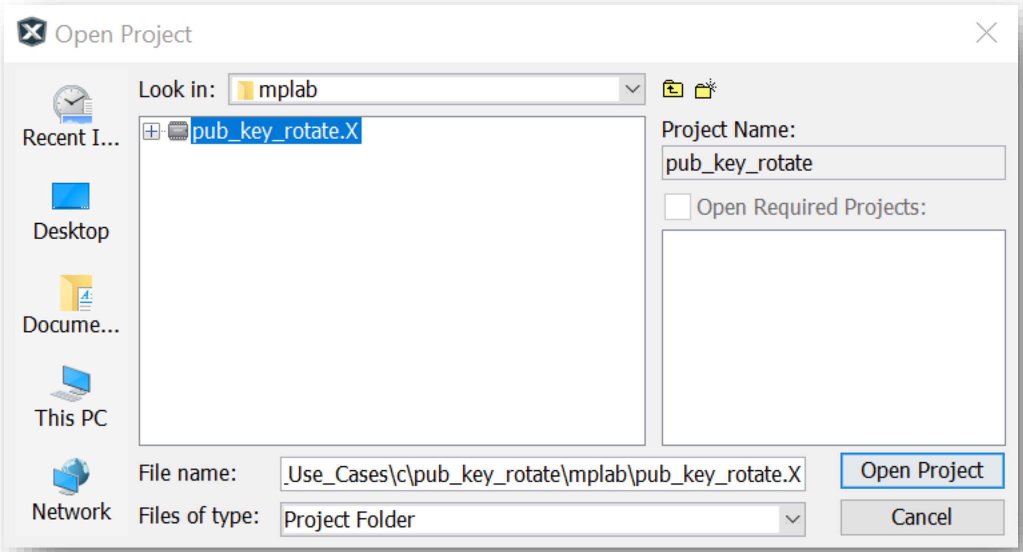
It is also possible to view the console messages by using applications like TeraTerm. Open the application with the COM related to CryptoAuth TrustPlatform with 115200-8-N-1 settings.



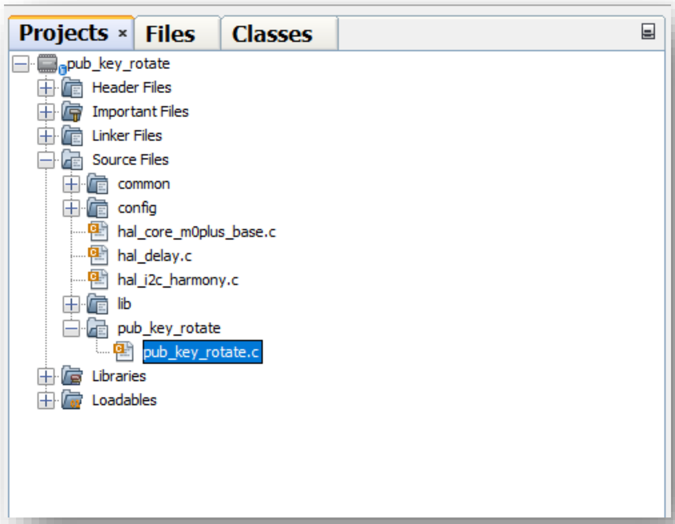
```
Device revision:
00 00 60 02

Validated public key is already in slot, invalidated the slot to
 update the new public key

New Rotating Public key written to device:

1B DC 32 6C 9C 47 CB AF FA E6 F1 3D 59 41 E3 B1
96 12 2F 8C 70 BE 3B 07 CB CD 5C 01 FE 6B A4 1E
E7 58 E4 EF 08 3E 16 45 71 B3 0B CE 09 25 97 62
43 BB D8 00 8F D9 CD 84 AF 71 50 55 95 E7 5C A2

Validated the Rotating Public key in device

Verified the Rotating Public key in device is usable for verify
operations

Execution completed with status 00
```

4.2.2 MPLAB:

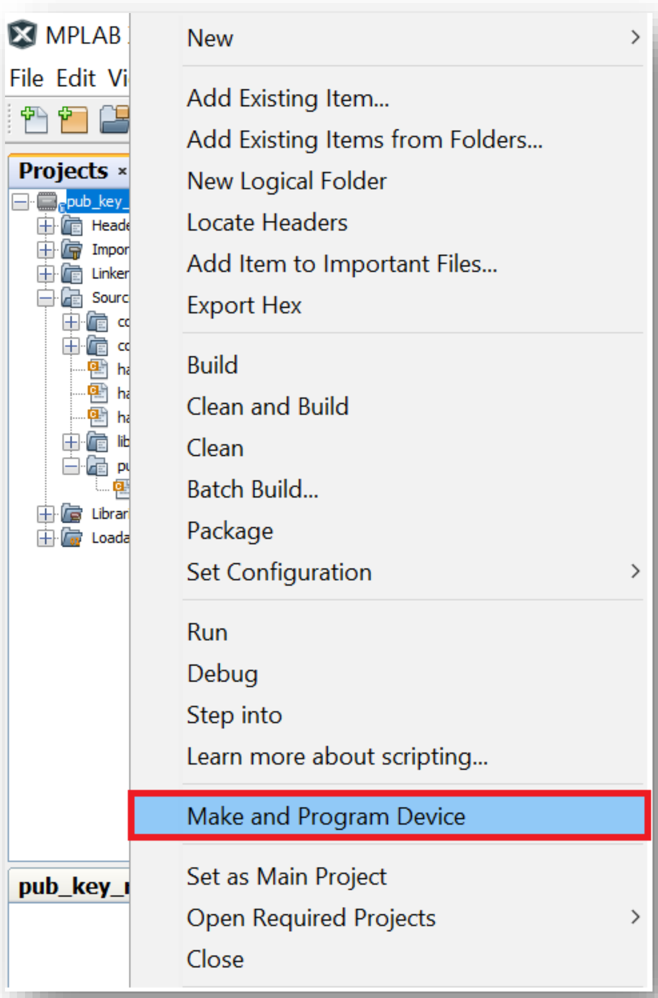
1. Open **pub_key_rotate.X** project by navigating to MPLAB -> File -> Open Project -> **TFLXTLS_Use_Cases\c\pub_key_rotate\mplab\pub_key_rotate.X**



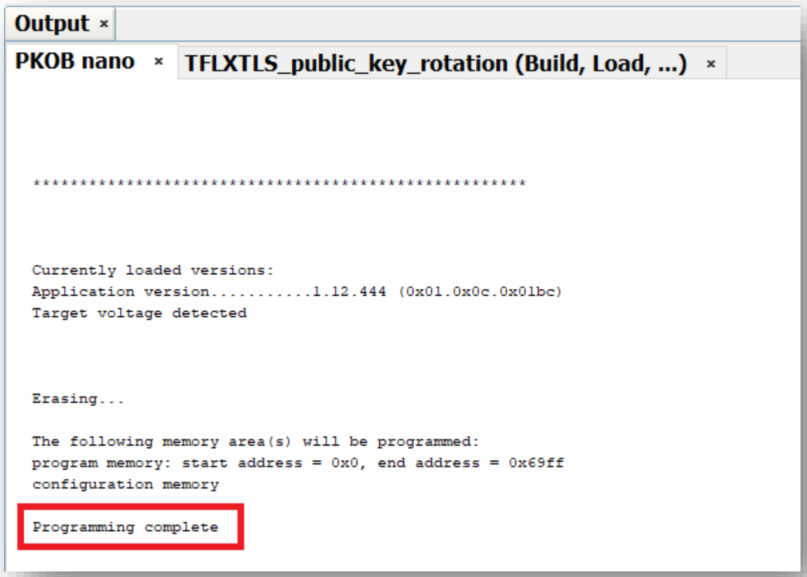
2. The application source code **pub_key_rotate.c** is available at **TFLXTLS_Use_Cases\c\pub_key_rotate\pub_key_rotate.c.** Other supporting files can be found under **TFLXTLS_Use_Cases\c\ dependencies**.



3. Program the Crypto Trust platform by navigating to **pub_key_rotate -> 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.

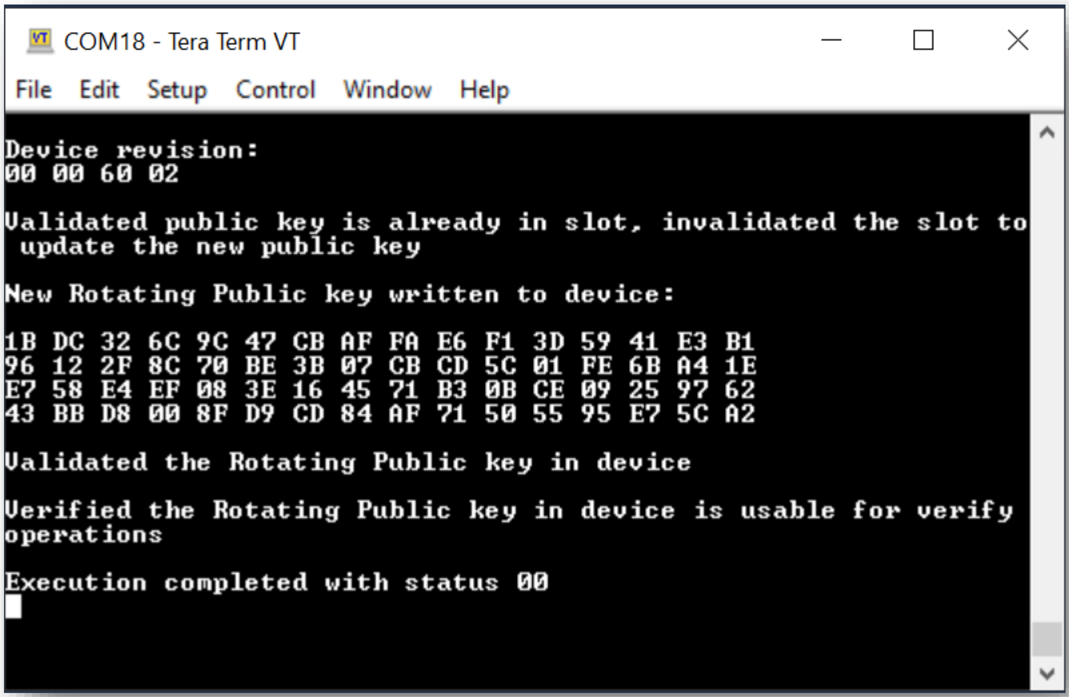


```
*****************************************************

Currently loaded versions:
Application version...........1.12.444 (0x01.0x0c.0x01bc)
Target voltage detected

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0x0, end address = 0x69ff
configuration memory

Programming complete
```

Once the programming is done, the firmware will do public key rotation operation. Depending on the public key rotation operation's output, the Cryptoauth Trust Platform board's Status LED will blink at different rates.

If public key rotation operation succeeds, LED blinks once every second.
If public key rotation 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 CryptoAuth TrustPlatform with 115200-8-N-1 settings



```
Device revision:
00 00 60 02

Validated public key is already in slot, invalidated the slot to
 update the new public key

New Rotating Public key written to device:

1B DC 32 6C 9C 47 CB AF FA E6 F1 3D 59 41 E3 B1
96 12 2F 8C 70 BE 3B 07 CB CD 5C 01 FE 6B A4 1E
E7 58 E4 EF 08 3E 16 45 71 B3 0B CE 09 25 97 62
43 BB D8 00 8F D9 CD 84 AF 71 50 55 95 E7 5C A2

Validated the Rotating Public key in device

Verified the Rotating Public key in device is usable for verify
operations

Execution completed with status 00
```

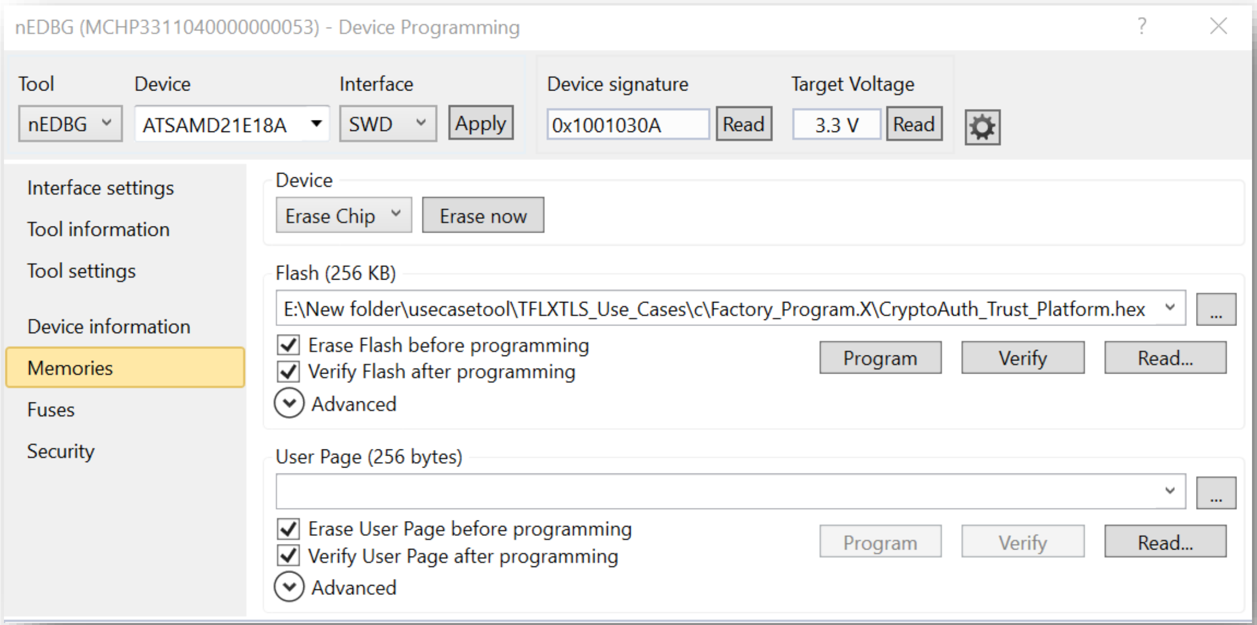
4.3 CryptoAuth TrustPlatform Factory reset

Once any of the embedded project is loaded to CrytoAuth TrustPlatform, the default program that enables interaction with TrustPlatform 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 at **TFLXTLS_Use_Cases\c\Factory_Program.X\CryptoAuth_Trust_Platform.hex**

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


nEDBG (MCHP3311040000000053) - Device Programming

Tool: nEDBG
Device: ATSAMD21E18A
Interface: SWD
Device signature: 0x1001030A
Target Voltage: 3.3 V

Device: Erase Chip

Flash (256 KB): E:\New folder\usecasetool\TFLXTLS_Use_Cases\c\Factory_Program.X\CryptoAuth_Trust_Platform.hex
- [x] Erase Flash before programming
- [x] Verify Flash after programming

User Page (256 bytes)
- [x] Erase User Page before programming
- [x] Verify User Page after programming

To reprogram using MPLAB:

1. Open **TFLXTLS_Use_Cases\c\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**

Now, Crypto 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 "CryptoAuth TrustPlatform Factory reset" section any usecase TrustFLEX Guide for reloading it
   2. Check the switch positions on Crypto Trust Platform and/or ATECC608A 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 "CryptoAuth TrustPlatform Factory reset" section any usecase TrustFLEX Guide for reloading it.

3. **Why does my C projects generates No such file or directory with ../../../ TFLXTLS_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

8. **Why is public key rotation project fails with error "Rotating Public key verification failed"?**
   There are many possibilities like,
   a. Signer registration is not done to the right account
   b. aws client region is select incorrectly
   c. WiFi credentials are not populated or in correct in C project
   d. aws-iot endpoint is not populated or in-correct in C project

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.

Legal Notice

Trademarks

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.

MICROCHIP

Worldwide Sales and Service

AMERICAS

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

ASIA/PACIFIC

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

ASIA/PACIFIC

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

EUROPE

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