Generating Certificates for SAM-IoT
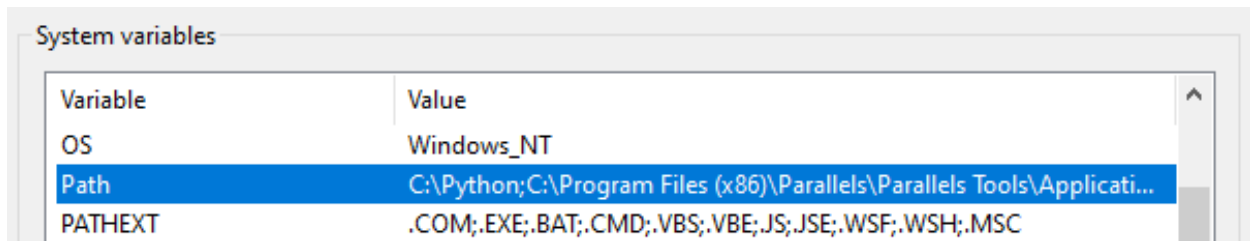
Software Installation Prerequisite

1. Download and install the latest Python release:
   https://www.python.org/downloads/

2. During installation setup, confirm the following boxes are checked:
   a. "Add Python x.x to Path"
   b. "pip (download and install other Python packages)"

3. Confirm that the Windows `Path` environment variable contains a pointer to the Python installation directory. For example, if Python was installed into `C:\Python`, the `Path` variable should include it

System variables	
Variable	Value
OS	Windows_NT
Path	**C:\Python;C:\Program Files (x86)\Parallels\Parallels Tools\Applicati...**
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

4. Open a command line window (e.g. Command Prompt or PowerShell)

   a. Navigate to the main "`sam-iot-provision`" folder
   ```
   > cd [path]\sam-iot-provision
   ```

   b. Install the required Python modules needed by the tool's scripts:
   ```
   > python -m pip install --upgrade pip
   > pip install -r py_modules.txt
   ```
   (if this step fails, you may need to install Microsoft Visual Studio from https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2019)

Generating Certificates for SAM-IoT

Generate the Certificates

Create the files that contain the CA certificate (`root-ca.crt`) & Signer certificate (`signer-ca.crt`). Navigate to the folder `\CertGen\cert` and execute the following scripts:

1. Create `root-ca.crt` & `root-ca.key`
```
> python ca_create_root.py
```

2. Create `signer-ca.csr` & `signer-ca.key`
```
> python ca_create_signer_csr.py
```

3. Create `signer-ca.crt`
```
> python ca_create_signer.py
```

Provisioning the ATWINC15x0

1. Connect the SAM-IoT board to the PC (a disk drive named "CURIOSITY" should appear on the desktop and/or in a file explorer window)
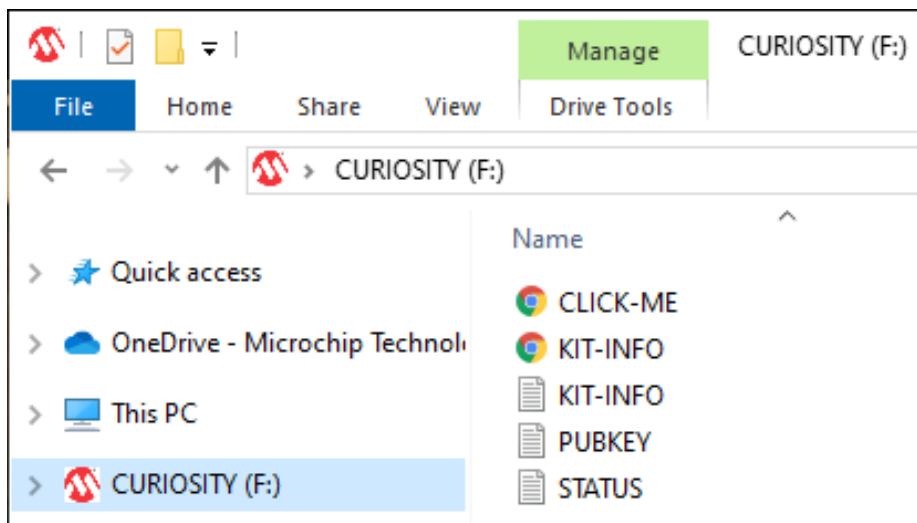
2. Using a terminal emulator program, control panel, etc. – note the COM port number corresponding to the SAM-IoT board connection (most likely the device name will show up as something like "USB Serial Device"). Do **not** initiate any serial connection at this time if using a terminal emulator, e.g.

Generating Certificates for SAM-IoT
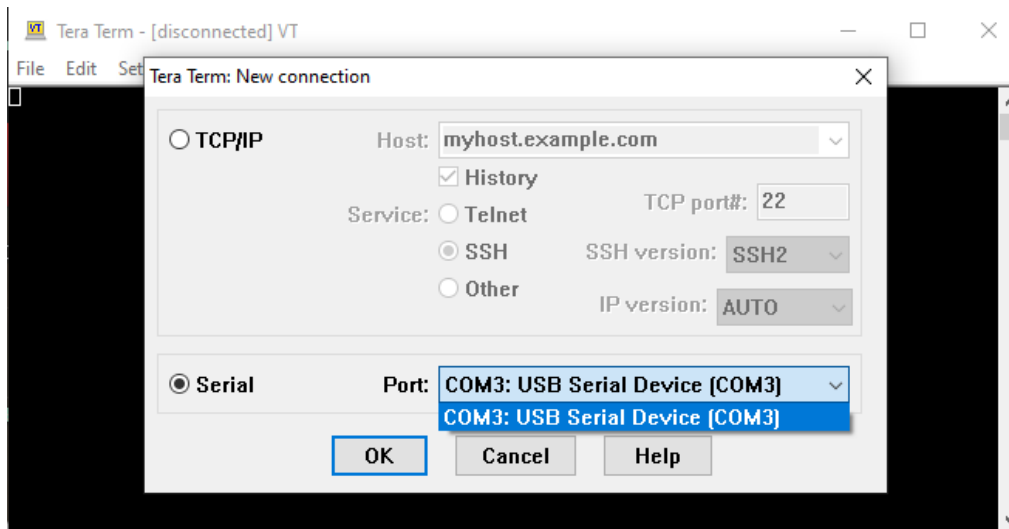
3. Drag & drop (i.e. copy) the "`SAM_IoT_CertGen.hex`" file to the "`CURIOSITY`" drive

NOTE
If the drag-and-drop operation fails, please refer to the instructions in "`Drag and Drop Troubleshooting for SAM-IoT.docx`". If drag and drop still does not work, use a DOS command line to manually copy the file to the Curiosity drive letter.

4. From the command line, go up one level to the ".`\CertGen\`" directory

5. Execute the following Python script:
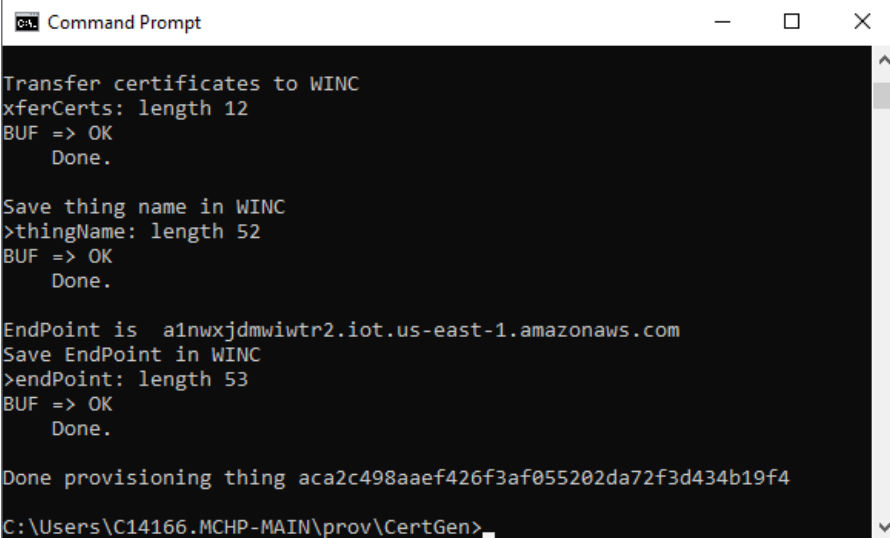   > `python provision_samiot.py com<#>`
   (e.g. `com<#>` = `com3`)

   This command should result in the following sequence of events:
   a. MCU sends a request to the ECC to create a device Certificate Signing Request (CSR) (`device.csr`)
   b. ECC signs the CSR with the signer's private key (`signer-ca.key`)
   c. ECC sends the device & signer certificates (`device.crt` & `signer-ca.crt`) back to the MCU
   d. MCU copies the 2 certs into the ATWINC15x0's flash memory

Generating Certificates for SAM-IoT

Conclusion

A total of 4 files are generated (`device.csr, device.crt, signer-ca.key, signer-ca.crt`) and 2 certificates (`device.crt` & `signer-ca.crt`) are stored in the ATWINC15x0

Generating Certificates for SAM-IoT

Sample Output

The following shows example certificates generated for a SAM-IoT board that has an ATECC608A which was pre-programmed with serial number `01239E946F011C66FE`

An online tool such as a Certificate Decoder can be used to display the pertinent information for a particular certificate by simply pasting the text of the certificate into a browser: https://www.sslshopper.com/certificate-decoder.html

Root CA

Certificate Information:

✅ **Common Name:** Root CA
✅ **Organization:** Microchip Technology Inc
✅ **Valid From:** August 5, 2020
✅ **Valid To:** July 30, 2045
✅ **Issuer:** Root CA, Microchip Technology Inc
✅ **Serial Number:** 48ea5888e05775d580f77ccb2c5b779b

Signer CA

Certificate Information:

✅ **Common Name:** Microsoft Signer
✅ **Organization:** Microsoft Inc
✅ **Valid From:** August 5, 2020
✅ **Valid To:** August 5, 2030
✅ **Issuer:** Root CA, Microchip Technology Inc
✅ **Serial Number:** 7f0a417a22f77741f567ab82024fe248

Device

Certificate Information:

✅ **Common Name:** sn01239E946F011C66FE
✅ **Organization:** Microchip Technology Inc
✅ **Valid From:** August 5, 2020
✅ **Valid To:** December 31, 3000
✅ **Issuer:** Microsoft Signer, Microsoft Inc
✅ **Serial Number:** 50cea2fdcf8be3d13fb56af2119a5adf

Generating Certificates for SAM-IoT

APPENDIX A: Background Information

1. MPLAB X Project "**SAM_IOT_WINC_PROV_Azure.X**" (located in the `SAM_IoT_Certs_Generator` folder)
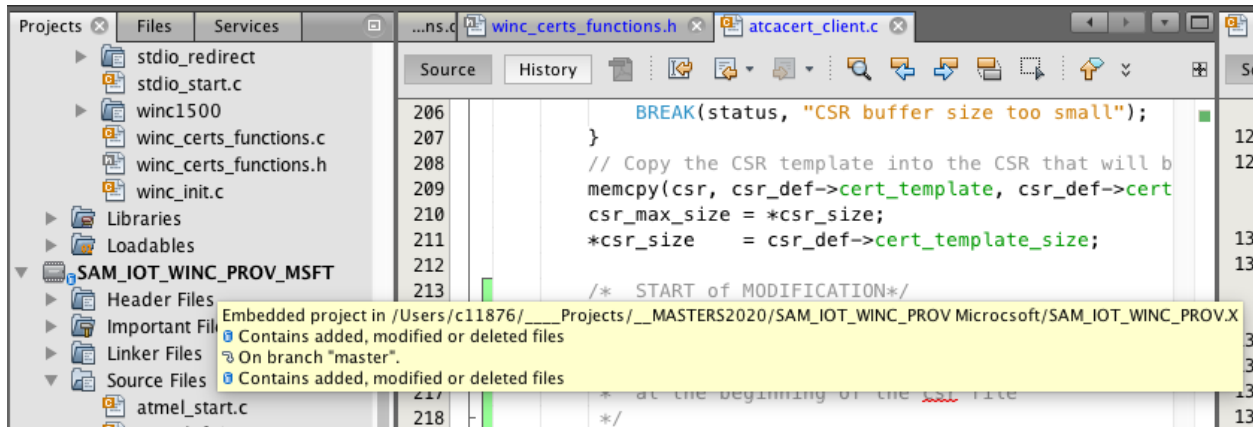
This ATSAMD21 MCU-based project is made to run on the SAM-IoT board when the ECC608A is already configured with PRIV/PUB keys and compressed CERT. This "helper utility app" will provision the ATWINC15x0 with either the original ECC device certificate or create a new one using the ECC PRIV/PUB keys. The HEX file generated for this project has been supplied for ease of programming the SAM-IoT board. The complete project with all the source code is provided in case any further customizations might be needed beyond what is currently supported.

2. Interaction between the main Python script and the **SAM_IOT_WINC_PROV_Azure** application running on the SAMD21

Python script (`provision_samiot.py`)		SAM_IOT_WINC_PROV_Azure
_Sends "<genCsr"	→	_Generate CSR from template "g_csr_def_3_device" _Derive public Key from private Key _Insert Public Key into CSR _Get CSR Digest

Generating Certificates for SAM-IoT

_Verify CSR signature _Scripts generate Device Cert by signing the CSR with SIGNER_CA Private Key _Save Device Cert into "device.crt"	←	_Sign Digest with Private Key _Insert Signed Digest into CSR _Return CSR to Python scripts
_Send ">caCert" with signer-ca.crt	→ ←	_Receive "caCert" and signer-ca.crt and stores it into buffer _Return SUCCESS
_Send ">devCert" with device.crt	→ ←	_Receive "devCert" with device.crt and stores it into buffer _Return SUCCESS
_Sends "xferCerts ["2"]	→	_Copy "caCert" and "devCert" into WINC Flash

Generating Certificates for SAM-IoT

APPENDIX B: Creating Custom Templates

The project needs to be built using the "ARM GNU Toolchain for 32-bit devices": https://www.microchip.com/mplab/avr-support/avr-and-arm-toolchains-c-compilers

There are 3 templates used in the provisioning project:

1. Signer Certificate template
   `g_cert_template_1_signer[]`
2. Device Certificate template
   `g_cert_template_2_device[]`
3. Device Certificate Signing Request template
   `g_csr_template_3_device[]`

To generate these templates, you need one of each (**signer-ca.csr**, **device.csr**, & **device.crt**). The **signer-ca.csr** was generated when you created the certificate earlier. For the **device.csr** and **device.crt** you will need to generate 2 samples.

To modify the Subject Common Name (CN) or the Subject Organization (O), look into the two (2) Python CSR generation for the following "`x509.oid.NameOID.ORGANIZATION_NAME`" and "`x509.oid.NameOID.COMMON_NAME`". Change the name for your specific value and then save the files.

```python
print('\nGenerating device CSR')
builder = x509.CertificateSigningRequestBuilder()
# Please note that the name of the signer is part of certificate definition in the SAM IoT firmware
# (g_cert_elements_1_signer). If this name is changed, it will also need to be changed in the firmware.
# The cert2certdef.py utility script can help with regenerating the cert_def_1_signer.c file after making changes.
builder = builder.subject_name(x509.Name([
    x509.NameAttribute(x509.oid.NameOID.ORGANIZATION_NAME, u'Microchip Technology Inc'),
    x509.NameAttribute(x509.oid.NameOID.COMMON_NAME, u'sn112233445566778899')]))
```

Don't forget to regenerate the **signer-ca.crt** if you change its OID in the **signer-ca.csr**

Now you need to generate the 2 samples for **device.csr** & **device.crt**. Navigate to the folder ".\CertGen\cert" and run the following scripts:

Generating Certificates for SAM-IoT

1. `python ca_create_device_csr.py`
    (this will create `device.csr`)

2. `python ca_create_device.py`
    (this will create `device.crt`)

At this point we should have 3 files: **`signer-ca.crt`,
`device.csr`**, and **`device.crt`** . Navigate to the folder
"`\CertGen\cert`" and execute the following scripts:

```
1. python cert2certdef.py –signer-cert signer-ca.crt > signer-ca.c
2. python cert2certdef.py –device-csrt device.csr > device-csr.c
3. python cert2certdef.py –device-cert device.crt > device-crt.c
```

Open the 3 C files and copy each corresponding file's contents into its
corresponding C file in the **`SAM_IOT_WINC_PROV`** project:

1. `signer-ca.c` → `cert_def_1_signer.c`
2. `device-csr.c` → `cert_def_3_device_csr.c`
3. `device-crt.c` → `cert_def_2_device.c`