Name: Nur A Farabi
ID: 12455
Course: CS535
Week: 12
Homework Link:
https://npu85.npu.edu/~henry/npu/classes/python/https/slide/exercise_https.html
Q5 ==> Project Part 3: Asymmetric Key Crypto & Digital Certificate

Step 1:
First of all we need a cryptography library for python.

```
PS C:\Users\nfara> python -V
Python 3.8.2
PS C:\Users\nfara> pip install cryptography
Collecting cryptography
  Downloading https://files.pythonhosted.org/packages/00/fc/ed8cf3e3d3817707c11da167a3478f9cb834afed5e8af450516752bb7df8/cryptography-3.0-cp38-cp38-win_amd64.whl (1.5MB)
     |████████████████████████████████| 1.5MB 819kB/s
Collecting six>=1.4.1 (from cryptography)
  Using cached https://files.pythonhosted.org/packages/ee/ff/48bde5c0f013094d729fe4b0316ba2a24774b3ff1c52d924a8a4cb04078a/six-1.15.0-py2.py3-none-any.whl
Collecting cffi!=1.11.3,>=1.8 (from cryptography)
  Downloading https://files.pythonhosted.org/packages/40/ad/eb98b5ec6129ffdabedca218ded2c529d59b935dac7cc6108366e379de96/cffi-1.14.1-cp38-cp38-win_amd64.whl (178kB)
     |████████████████████████████████| 184kB 3.3MB/s
Collecting pycparser (from cffi!=1.11.3,>=1.8->cryptography)
  Using cached https://files.pythonhosted.org/packages/ae/e7/d9c3a176ca4b02024debf82342dab36efadfc5776f9c8db077e8f6e71821/pycparser-2.20-py2.py3-none-any.whl
Installing collected packages: six, pycparser, cffi, cryptography
Successfully installed cffi-1.14.1 cryptography-3.0 pycparser-2.20 six-1.15.0
WARNING: You are using pip version 19.2.3, however version 20.2.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
PS C:\Users\nfara> python -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/bd/b1/56a834acdbe23b486dea16aaf4c27ed28eb292695b90d01dff96c96597de/pip-20.2.1-py2.py3-none-any.whl (1.5MB)
     |████████████████████████████████| 1.5MB 1.1MB/s
Installing collected packages: pip
  Found existing installation: pip 19.2.3
    Uninstalling pip-19.2.3:
      Successfully uninstalled pip-19.2.3
Successfully installed pip-20.2.1
PS C:\Users\nfara>
```

Step2: Setup environment
1.

```
nfarabi@DESKTOP-F0RIEOJ:~$ sudo apt-get update
[sudo] password for nfarabi:
```

2.

```
Fetched 5388 kB in 8s (696 kB/s)
Reading package lists... Done
nfarabi@DESKTOP-F0RIEOJ:~$ sudo apt-get install build-essential libssl-dev libffi-dev python-dev
Reading package lists... Done
```

3.

```
nfarabi@DESKTOP-F0RIEOJ:~$ sudo apt install python3-pip
[sudo] password for nfarabi:
Reading package lists... Done
```

4.

```
nfarabi@DESKTOP-F0RIEOJ:~$ virtualenv -p python3 env3

Command 'virtualenv' not found, but can be installed with:

sudo apt install virtualenv

nfarabi@DESKTOP-F0RIEOJ:~$ sudo apt install virtualenv
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
nfarabi@DESKTOP-F0RIEOJ:~$ python --version
Python 2.7.17
nfarabi@DESKTOP-F0RIEOJ:~$ virtualenv -p /usr/bin/python2.7 venv
Running virtualenv with interpreter /usr/bin/python2.7
New python executable in /home/nfarabi/venv/bin/python2.7
Also creating executable in /home/nfarabi/venv/bin/python
Installing setuptools, pkg_resources, pip, wheel...done.
nfarabi@DESKTOP-F0RIEOJ:~$
```

Step3:

Python code

```
# pki_helpers.py
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
from datetime import datetime, timedelta
from cryptography import x509
from cryptography.x509.oid import NameOID
from cryptography.hazmat.primitives import hashes
def generate_private_key(filename: str, passphrase: str):
private_key = rsa.generate_private_key(
public_exponent=65537, key_size=2048, backend=default_backend()
)
utf8_pass = passphrase.encode("utf-8")
algorithm = serialization.BestAvailableEncryption(utf8_pass)
with open(filename, "wb") as keyfile:
keyfile.write(
private_key.private_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PrivateFormat.TraditionalOpenSSL,
encryption_algorithm=algorithm,
)
)
return private_key
def generate_public_key(private_key, filename, **kwargs):
subject = x509.Name(
[
x509.NameAttribute(NameOID.COUNTRY_NAME, kwargs["country"]),
x509.NameAttribute(
NameOID.STATE_OR_PROVINCE_NAME, kwargs["state"]
),
x509.NameAttribute(NameOID.LOCALITY_NAME, kwargs["locality"]),
x509.NameAttribute(NameOID.ORGANIZATION_NAME,
kwargs["org"]),
x509.NameAttribute(NameOID.COMMON_NAME,
kwargs["hostname"]),
]
)
# Because this is self signed, the issuer is always the subject
issuer = subject
# This certificate is valid from now until 30 days
valid_from = datetime.utcnow()
valid_to = valid_from + timedelta(days=30)
# Used to build the certificate
builder = (
x509.CertificateBuilder()
.subject_name(subject)
.issuer_name(issuer)
.public_key(private_key.public_key())
.serial_number(x509.random_serial_number())
.not_valid_before(valid_from)
.not_valid_after(valid_to)
)
# Sign the certificate with the private key
public_key = builder.sign(
private_key, hashes.SHA256(), default_backend()
)
with open(filename, "wb") as certfile:
certfile.write(public_key.public_bytes(serialization.Encoding.PEM))
return public_key
```