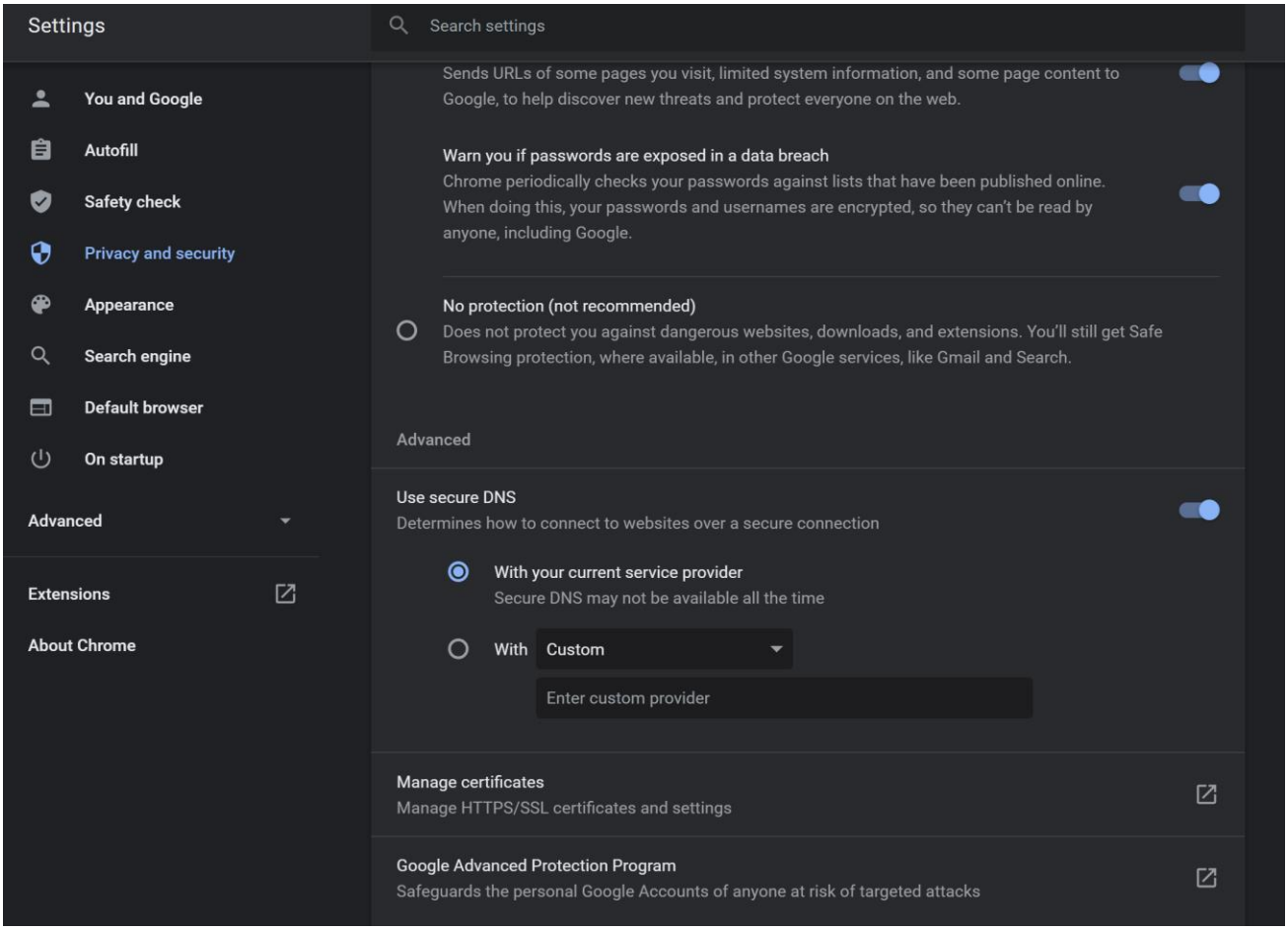
Name: Chang Pao Herr
Student ID: 19544
Course Term: Summer 2020 – CS535 - NETWORK SECURITY FUNDAMENTAL
Instructor: Dr. Chang Henry

Week# 12 Homework#: 10
Due Date: 7/28/2020 11:30:00 PM
Homework Subject: Project HTTPS (III)

Part#3 Question #5: Asymmetric Key Crypto & Digital Certificate
References:
Asymmetric Key Crypto
Answers - 2020 Summer

Chrome setting/configure:



List of codes need to complete this project.

symmetric_server.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS535/cs535_...

File Edit Format Run Options Window Help

```
# symmetric_server.py
import os
from flask import Flask
from cryptography.fernet import Fernet

SECRET_KEY = os.environb[b"SECRET_KEY"]
SECRET_MESSAGE = b"fluffy tail"
app = Flask(__name__)

my_cipher = Fernet(SECRET_KEY)

@app.route("/")
def get_secret_message():
    return my_cipher.encrypt(SECRET_MESSAGE)
```

symmetric_client.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS535/cs535_...

File Edit Format Run Options Window Help

```
# symmetric_client.py
import os
import requests
from cryptography.fernet import Fernet

SECRET_KEY = os.environb[b"SECRET_KEY"]
my_cipher = Fernet(SECRET_KEY)

def get_secret_message():
    response = requests.get("http://127.0.0.1:5683")

    decrypted_message = my_cipher.decrypt(response.content)
    print(f"The codeword is: {decrypted_message}")

if __name__ == "__main__":
    get_secret_message()
```

pki_helpers.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS535/cs535_p3Q5/pki_helpers.py (3.8.2)

File Edit Format Run Options Window Help

```
# 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
```

pki_helpers gen_public_key.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS5...

File Edit Format Run Options Window Help

```
from pki_helpers import generate_private_key, generate_public_key
>>> private_key = generate_private_key("ca-private-key.pem", "secret_password")
>>> private_key
>>> generate_public_key( ... private_key,
... filename="ca-public-key.pem",
... country="US",
... state="Maryland",
... locality="Baltimore",
... org="My CA Company",
... hostname="my-ca.com",
... )
```

*pki_helpers gen_csr.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS535/cs535_p3Q5/pki_helpers gen_c...

File Edit Format Run Options Window Help

```
# pki_helpers.py
def generate_csr(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"]),
]
)
# Generate any alternative dns names
alt_names = []
for name in kwargs.get("alt_names", []):
alt_names.append(x509.DNSName(name))
san = x509.SubjectAlternativeName(alt_names)
builder = (
x509.CertificateSigningRequestBuilder()
.subject_name(subject)
.add_extension(san, critical=False)
)
csr = builder.sign(private_key, hashes.SHA256(), default_backend())
with open(filename, "wb") as csrfile:
csrfile.write(csr.public_bytes(serialization.Encoding.PEM))
return csr
```

pki_helpers gen_private_key.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS...

File Edit Format Run Options Window Help

```
from pki_helpers import generate_csr, generate_private_key
>>> server_private_key = generate_private_key( ... "server-private-key.pem", "se
... )
>>> server_private_key
>>> generate_csr( ... server_private_key,
... filename="server-csr.pem",
... country="US",
... state="Maryland",
... locality="Baltimore",
... org="My Company",
... alt_names=["localhost"],
... hostname="my-site.com",
... )
```

pki_helpers sign_scr.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS535/cs535_p3Q5/pki_helpers sign_scr.py (3.8.2)

File Edit Format Run Options Window Help

```
# pki_helpers.py
def sign_csr(csr, ca_public_key, ca_private_key, new_filename):
valid_from = datetime.utcnow()
valid_until = valid_from + timedelta(days=30)
builder = (
x509.CertificateBuilder() .subject_name(csr.subject)
.issuer_name(ca_public_key.subject)
.public_key(csr.public_key())
.serial_number(x509.random_serial_number())
.not_valid_before(valid_from)
.not_valid_after(valid_until)
)
for extension in csr.extensions:
builder = builder.add_extension(extension.value, extension.critical)
public_key = builder.sign(
private_key=ca_private_key, algorithm=hashes.SHA256(), backend=default_backend(),
)
with open(new_filename, "wb") as keyfile:
keyfile.write(public_key.public_bytes(serialization.Encoding.PEM))
```

C_client.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS535/cs535_p3Q5/C_...

File Edit Format Run Options Window Help

```
# client.py
import os
import requests
def get_secret_message():
response = requests.get("https://localhost:5683")
print(f"The secret message is {response.text}")
if __name__ == "__main__":
get_secret_message()
```

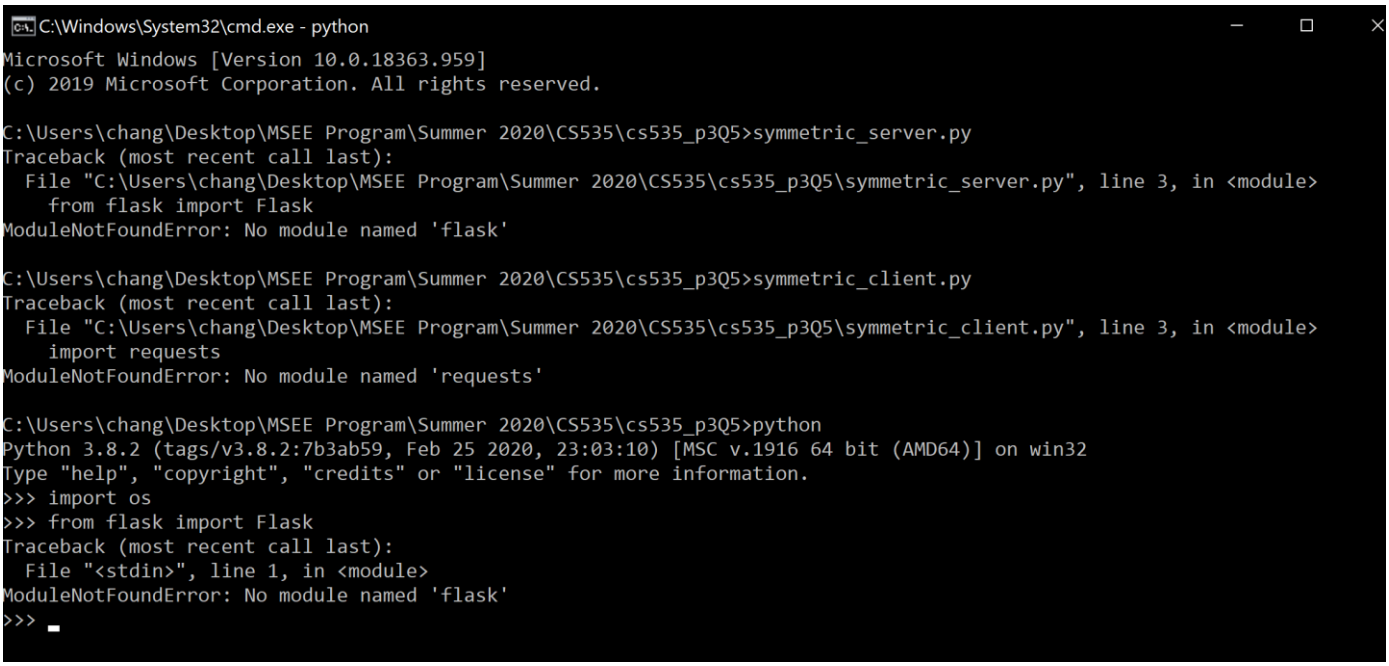
client get_secret_mes.py - C:/Users/chang/Desktop/MSEE Program/Summer 2020/CS535/cs...

File Edit Format Run Options Window Help

```
# client.py
def get_secret_message():
response = requests.get("http://localhost:5683", verify="ca-public-key.pem")
print(f"The secret message is {response.text}")
```

Execute the code:

I have too many attempts and not success. Disappointed I can not get this part to work. I do know that I missed something but do not know what it is. Below are few results of being trying.



```
C:\Users\chang>\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5
The system cannot find the path specified.

C:\Users\chang>cd desktop

C:\Users\chang\Desktop>cd msee program

C:\Users\chang\Desktop\MSEE Program>cd summer 2020

C:\Users\chang\Desktop\MSEE Program\Summer 2020>cs535
'cs535' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\chang\Desktop\MSEE Program\Summer 2020>cd cs535

C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535>cd cs535_p3q5

C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5>$ ls C_client.py
'$' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5>C_client.py
  File "C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5\C_client.py", line 5
    response = requests.get("https://localhost:5683")
    ^
IndentationError: expected an indented block

C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5>c_client.py
  File "C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5\C_client.py", line 5
    response = requests.get("https://localhost:5683")
    ^
IndentationError: expected an indented block

C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5>symmetric_server.py
Traceback (most recent call last):
  File "C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5\symmetric_server.py", line 3, in <module>
    from flask import Flask
ModuleNotFoundError: No module named 'flask'

C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5>pki_helpers.py
  File "C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5\pki_helpers.py", line 10
    private_key = rsa.generate_private_key(
    ^
IndentationError: expected an indented block

C:\Users\chang\Desktop\MSEE Program\Summer 2020\CS535\cs535_p3Q5>
```