Week 9 Practical: Digital certificates using *Cyptography*

Dr. Qublai Ali Mirza

University of Gloucestershire

qalimirza@glos.ac.uk

Overview

1. Recap
2. Environment setup
3. Accessing certificate information
4. Certificate generation
5. Digital signatures
6. Bringing it all together
7. Post-sessional work

UNIVERSITY OF GLOUCESTERSHIRE

Recap

- Last week we looked at how to go about using hashing and Elliptical Curve Cryptography
- This week we will be looking at digital signatures and signing using certificates
- But before doing so, 10 minutes to complete the post-sessional work from last week.

UNIVERSITY OF GLOUCESTERSHIRE

Environment setup

- For our practical session today, we will be using
  - OpenSSL for `Windows`
  - Cryptography module for `Python`

Installing OpenSSL

- The first tool that we need to install is the *OpenSSL* suite for *Windows*
- Provides a collection of tools for key and certificate generation
- To that end, download the package from here
- Then extract the zip file and drag the file path onto the command line

Installing Cryptography

- Next we need to install the *cryptography* module for `Python`
- To that end, we will set up our Anaconda environment as usual
- Then type in: `pip install cryptography pyopenssl` to install the package

Accessing certificate information

- For our first activity, we will be reading information from a security certificate
- To that end, type in the following one after the next:
  - `genrsa -out cert.key`
  - `openssl req -x509 -new -nodes -key cert.key -days 365 -out cert.crt`
- Once we have provided the required information, we will proceed to type in the following *Python* script

Accessing certificate information I

```
from cryptography import x509
from cryptography.hazmat.backends import default_backend

backend = default_backend()
with open('./Mycert.crt', 'rb') as f:
    crt_data = f.read()
    cert = x509.load_pem_x509_certificate(crt_data, backen

class Certificate:

    _fields = ['country_name',
               'state_or_province_name',
```

Accessing certificate information II

```
               'locality_name',
               'organization_name',
               'organizational_unit_name',
               'common_name',
               'email_address']

    def __init__(self, cert):
        assert isinstance(cert, x509.Certificate)
        self._cert = cert
        for attr in self._fields:
            oid = getattr(x509, 'OID_' + attr.upper())
            subject = cert.subject
```

Accessing certificate information III

```
            info = subject.get_attributes_for_oid(oid)
            setattr(self, attr, info)
```

```
cert = Certificate(cert)
for attr in cert._fields:
    for info in getattr(cert, attr):
        print("{}: {}".format(info._oid._name, info._value)
```

Certificate generation

- For our first activity, we will be looking at *certificate generation*
- But we need to first generate an X.509 private key.
- To that end, type in:
  - `openssl genrsa -out Mykey.pem 2048`
- Then type in the following `Python` code

Certificate generation I

```
from __future__ import print_function, unicode_literals

from datetime import datetime, timedelta
from OpenSSL import crypto

# load private key
ftype = crypto.FILETYPE_PEM
with open('Mykey.pem', 'rb') as f: k = f.read()
k = crypto.load_privatekey(ftype, k)

now    = datetime.now()
expire = now + timedelta(days=365)
```

Certificate generation II

```
# country (countryName, C)
# state or province name (stateOrProvinceName, ST)
# locality (locality, L)
# organization (organizationName, O)
# organizational unit (organizationalUnitName, OU)
# common name (commonName, CN)

cert = crypto.X509()
cert.get_subject().C  = "UK"
cert.get_subject().ST = "United Kingdom"
cert.get_subject().L  = "Cheltenham"
```

Certificate generation III

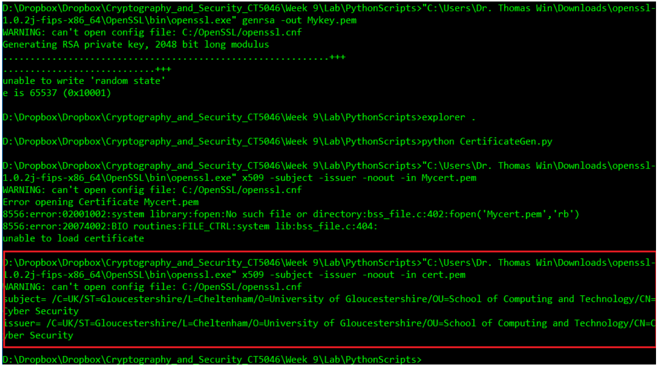
```
cert.get_subject().O  = "UoG"
cert.get_subject().OU = "Computing and Technology"
cert.get_subject().CN = "Cyber Security"
cert.set_serial_number(1000)
cert.set_notBefore(now.strftime("%Y%m%d%H%M%SZ").encode())
cert.set_notAfter(expire.strftime("%Y%m%d%H%M%SZ").encode(
cert.set_issuer(cert.get_subject())
cert.set_pubkey(k)
cert.sign(k, 'sha1')

with open('cert.pem', "wb") as f:
    f.write(crypto.dump_certificate(ftype, cert))
```

Obtaining certificate

- Once complete, type: `python CertificateGen.py` to execute
- Then type in:
  - `openssl x509 -subject -issuer -noout -in cert.pem`

Overall operation



```
D:\Dropbox\Dropbox\Cryptography_and_Security_CT5046\Week 9\Lab\PythonScripts>"C:\Users\Dr. Thomas Win\Downloads\openssl-
1.0.2j-fips-x86_64\OpenSSL\bin\openssl.exe" genrsa -out Mykey.pem
WARNING: can't open config file: C:/OpenSSL/openssl.cnf
Generating RSA private key, 2048 bit long modulus
...........................................................+++
...........................+++
unable to write 'random state'
e is 65537 (0x10001)

D:\Dropbox\Dropbox\Cryptography_and_Security_CT5046\Week 9\Lab\PythonScripts>explorer .

D:\Dropbox\Dropbox\Cryptography_and_Security_CT5046\Week 9\Lab\PythonScripts>python CertificateGen.py

D:\Dropbox\Dropbox\Cryptography_and_Security_CT5046\Week 9\Lab\PythonScripts>"C:\Users\Dr. Thomas Win\Downloads\openssl-
1.0.2j-fips-x86_64\OpenSSL\bin\openssl.exe" x509 -subject -issuer -noout -in Mycert.pem
WARNING: can't open config file: C:/OpenSSL/openssl.cnf
Error opening Certificate Mycert.pem
8556:error:02001002:system library:fopen:No such file or directory:bss_file.c:402:fopen('Mycert.pem','rb')
8556:error:20074002:BIO routines:FILE_CTRL:system lib:bss_file.c:404:
unable to load certificate

D:\Dropbox\Dropbox\Cryptography_and_Security_CT5046\Week 9\Lab\PythonScripts>"C:\Users\Dr. Thomas Win\Downloads\openssl-
1.0.2j-fips-x86_64\OpenSSL\bin\openssl.exe" x509 -subject -issuer -noout -in cert.pem
WARNING: can't open config file: C:/OpenSSL/openssl.cnf
subject= /C=UK/ST=Gloucestershire/L=Cheltenham/O=University of Gloucestershire/OU=School of Computing and Technology/CN=
Cyber Security
issuer= /C=UK/ST=Gloucestershire/L=Cheltenham/O=University of Gloucestershire/OU=School of Computing and Technology/CN=C
yber Security

D:\Dropbox\Dropbox\Cryptography_and_Security_CT5046\Week 9\Lab\PythonScripts>
```

Figure: How all of this works

Certificate Signing Request

Certificate Signing Request I

```
#!/usr/bin/python3


from __future__ import print_function, unicode_literals

from OpenSSL import crypto

# load private key
ftype = crypto.FILETYPE_PEM
with open('key.pem', 'rb') as f:
        key = f.read()
```

Certificate Signing Request II

```
key = crypto.load_privatekey(ftype, key)
req    = crypto.X509Req()

alt_name  = [ b"DNS:www.helloworld.com",
              b"DNS:doc.helloworld.com" ]
key_usage = [ b"Digital Signature",
              b"Non Repudiation",
              b"Key Encipherment" ]

# country (countryName, C)
# state or province name (stateOrProvinceName, ST)
# locality (locality, L)
```

Certificate Signing Request III

```
# organisation (organisationName, O)
# organisational unit (organisationalUnitName, OU)
# common name (commonName, CN)

req.get_subject().C  = "GB"
req.get_subject().ST = "United Kingdom"
req.get_subject().L  = "Gloucestershire"
req.get_subject().O  = "University of Gloucestershire"
req.get_subject().OU = "School of Computing and Technology'
req.get_subject().CN = "Cyber Security"
req.add_extensions([
    crypto.X509Extension( b"basicConstraints",
```

Certificate Signing Request IV

```
                           False,
                           b"CA:FALSE"),
    crypto.X509Extension( b"keyUsage",
                           False,
                           b",".join(key_usage)),
    crypto.X509Extension( b"subjectAltName",
                           False,
                           b",".join(alt_name))
])

req.set_pubkey(key)
req.sign(key, "sha256")
```

Certificate Signing Request V

```python
csr = crypto.dump_certificate_request(ftype, req)
with open("cert.csr", 'wb') as f:
        f.write(csr)
```

OpenSSL.conf configuration details

```
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEnciphermen
subjectAltName = @alt_names
[alt_names]
DNS.1 = www.helloworld.com
DNS.2 = doc.helloworld.com
```

```
# Generate a root CA

openssl genrsa -out ca-key.pem 2048

openssl req -x509 -new -nodes -key ca-key.pem -days 1000 \
-out ca.pem -subj "/CN=root-ca"

# Set up a certificate generation request

openssl genrsa -out key.pem 2048

python3 csr.py

# Sign a certification generation request
```

```
openssl x509 -req -in cert.csr -CA ca.pem -CAkey
\  ca-key.pem -CAcreateserial -out cert.pem -days 365 \
 -extensions v3_req -extfile openssl.conf

# Verify

openssl x509 -in cert.pem -text -noout
```

Digital signing

- In order to sign a file, we need to generate both a *private* and a *public* key
- We will sign the file with a private key and the recipient can then verify it using a public key
- To that end, type in:
  - `openssl genrsa -out private.key 2048`
  - `openssl rsa -in private.key -pubout -out public.key`

Digital signing I

```
from __future__ import print_function, unicode_literals

from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
from Crypto.Hash import SHA256


def signer(privkey, data):
    rsakey = RSA.importKey(privkey)
    signer = PKCS1_v1_5.new(rsakey)
    digest = SHA256.new()
    digest.update(data)
```

Digital signing II

```
    return signer.sign(digest)


with open('private.key', 'rb') as f:
        key = f.read()

with open('plaintext.txt', 'rb') as f:
        data = f.read()

sign = signer(key, data)
```

Digital signing III

```
with open('plaintext.txt.sha256', 'wb') as f:
        f.write(sign)
```

Verification

- When executed, our *Python* script will give up a SHA256 digest
- But we need to use it to verify the validity of our file
- To that end type in:
  - `openssl dgst -sha256 -verify public.key -signature plaintext.txt.sha256 plaintext.txt`

UNIVERSITY OF GLOUCESTERSHIRE

Bringing it all together

- This week we looked at *Digital signatures & Certificates*
- We also looked at how to generate a digital certificate and use it in verification process
- Next week: *SSL and VPN*

Post-sessional work

- Using the in-lab exercise at starting point, create a public key certificate of your own and use it to create a digital signature for both:
  - a *PDF* file
  - a *PNG* file of your choosing.
- **Hint:** you might want to use the `timeit.timeit` function

Q & A

UNIVERSITY OF GLOUCESTERSHIRE