Small OpenSSL guide

Command line

Printing a certificate

```
openssl x509 -in <your_certificate> -text -noout
```

Reading fields

```
openssl x509 -noout -<yourfield> -in <yourcertificate.pem>
```

A list of all the fields can be found as:

```
openssl x509 -noout --help
```

Verifying a chain (giving root and/or intermediate certificates)

```
openssl verify -CAfile Root.pem -untrusted <intermediatecertificate.pem> <yourcertificate.pem>
```

Generating a self-signed certificate

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -sha256 -days 365
```

Python library

- Don't use pyOpenSSl (it has been deprecated, especially the crypto part). It can be used for TLS.
- Use `hazmat cryptography`
- https://cryptography.io/en/latest/x509/

Generating a CSR request (taken from the documentation)

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography import x509
from cryptography.x509.oid import NameOID
from cryptography.hazmat.primitives import hashes


# Generate our key

key = rsa.generate_private_key(

    public_exponent=65537,
    key_size=2048,

)

# Write our key to disk for safe keeping

with open("path/to/store/key.pem", "wb") as f:

    f.write(key.private_bytes(

        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
        encryption_algorithm=serialization.BestAvailableEncryption(b"passphrase"), # -nodes in OpenSSL command line
```

```
    ))
# load_pem_private_key() if already generated


# Generate a CSR
csr = x509.CertificateSigningRequestBuilder().subject_name(x509.Name([
    # Provide various details about who we are.
    x509.NameAttribute(NameOID.COUNTRY_NAME, "US"),
    x509.NameAttribute(NameOID.STATE_OR_PROVINCE_NAME, "California"),
    x509.NameAttribute(NameOID.LOCALITY_NAME, "San Francisco"),
    x509.NameAttribute(NameOID.ORGANIZATION_NAME, "My Company"),
    x509.NameAttribute(NameOID.COMMON_NAME, "mysite.com"),
])).add_extension(
    x509.SubjectAlternativeName([
        # Describe what sites we want this certificate for.
        x509.DNSName("mysite.com"),
        x509.DNSName("www.mysite.com"),
        x509.DNSName("subdomain.mysite.com"),
    ]),
    critical=False,
# Sign the CSR with our private key.
).sign(key, hashes.SHA256())
# Write our CSR out to disk.
with open("path/to/csr.pem", "wb") as f:
    f.write(csr.public_bytes(serialization.Encoding.PEM))
```

Generating a self-signed certificate

Generate our key

```
key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
)
# Write our key to disk for safe keeping
with open("path/to/store/key.pem", "wb") as f:
    f.write(key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
```

```
        encryption_algorithm=serialization.BestAvailableEncryption(b"passphrase"),
    ))

subject = issuer = x509.Name([
    x509.NameAttribute(NameOID.COUNTRY_NAME, "US"),
    x509.NameAttribute(NameOID.STATE_OR_PROVINCE_NAME, "California"),
    x509.NameAttribute(NameOID.LOCALITY_NAME, "San Francisco"),
    x509.NameAttribute(NameOID.ORGANIZATION_NAME, "My Company"),
    x509.NameAttribute(NameOID.COMMON_NAME, "mysite.com"),
])
cert = x509.CertificateBuilder().subject_name(
    subject
).issuer_name(
    issuer
).public_key(
    key.public_key()
).serial_number(
    x509.random_serial_number()
).not_valid_before(
    datetime.datetime.now(datetime.timezone.utc)
).not_valid_after(
    # Our certificate will be valid for 10 days
    datetime.datetime.now(datetime.timezone.utc) + datetime.timedelta(days=10)
).add_extension(
    x509.SubjectAlternativeName([x509.DNSName("localhost")]),
    critical=False,
# Sign our certificate with our private key
).sign(key, hashes.SHA256())
# Write our certificate out to disk.
with open("path/to/certificate.pem", "wb") as f:
    f.write(cert.public_bytes(serialization.Encoding.PEM))
```