


Lab 6: Trust and Digital Certificates

Objective: Digital certificates are used to define a trust infrastructure within PKI (Public Key Infrastructure). A certificate can hold a key pair, while a distributable certificate will only contain the public key. In this lab we will read-in digital certificates and analyse them.

 **Web link (Weekly activities):** <https://asecuritysite.com/esecurity/unit06>

A Introduction

No	Description	Result
A.1	<p>From:</p> <p> Web link (Digital Certificate): http://asecuritysite.com/encryption/digitalcert</p> <p>Open up Certificate 1 and <u>identify</u> the following:</p>	<p>Serial number:</p> <p><u>Effective date:</u></p> <p>Name:</p> <p>Issuer:</p> <p>What is CN used for:</p> <p>What is ON used for:</p> <p>What is O used for:</p> <p>What is L used for:</p>
A.2	<p>Now open-up the ZIP file for the certificate, and view the CER file.</p>	<p>What other information can you gain from the certificate:</p> <p>What is the size of the public key:</p> <p>Which hashing method has been used:</p> <p>Is the certificate trusted on your system: [Yes][No]</p>

A.3 For Example 2 to Example 6. Complete the following table:

Cert	Organisation (Issued to)	Date range when valid	Size of public key	Issuer	Root CA	Hash method	Is it trusted?
2							
3							
4							
5							
6							

A.4 Now download the DER files from:


 **Web link (Digital Certificate):** <http://asecuritysite.com/der.zip>

Now use openssl to read the certificates:

```
openssl x509 -inform der -in [certname] -noout -text
```

B Creating certificates

Now we will create our own self-signed certificates.

No	Description	Result
B.1	<p>Create your own certificate from:</p> <p> Web link (Create Certificate): http://asecuritysite.com/encryption/createcert</p> <p>Add in your own details.</p>	<p>View the certificate, and verify some of the details on the certificate.</p> <p>Can you view the DER file?</p>

We have a root certificate authority of My Global Corp, which is based in Washington, US, and the administrator is admin@myglobalcorp.com and we are going to issue a certificate to

My Little Corp, which is based in Glasgow, UK, and the administrator is admin@mylittlecorp.com.

No	Description	Result
B.2	<p>Create your RSA key pair with:</p> <pre>openssl genrsa -out ca.key 2048</pre> <p>Next create a self-signed root CA certificate ca.crt for My Little Corp:</p> <pre>openssl req -new -x509 -days 1826 -key ca.key -out ca.crt</pre>	<p>How many years will the certificate be valid for?</p> <p>Which details have you entered:</p>
B.3	<p>Next go to Places, and from your Home folder, open up ca.crt and view the details of the certificate.</p>	<p>Which Key Algorithm has been used:</p> <p>Which hashing methods have been used:</p> <p>When does the certificate expire:</p> <p>Who is it verified by:</p> <p>Who has it been issued to:</p>
B.4	<p>Next we will create a subordinate CA (My Little Corp), and which will be used for the signing of the certificate. First, generate the key:</p> <pre>openssl genrsa -out ia.key 2048</pre> <p>Next we will request a certificate for our newly created subordinate CA:</p> <pre>openssl req -new -key ia.key -out ia.csr</pre> <p>We can then create a certificate from the subordinate CA certificate and signed by the root CA.</p> <pre>openssl x509 -req -days 730 -in ia.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out ia.crt</pre>	<p>View the newly created certificate.</p> <p>When does it expire:</p> <p>Who is the subject of the certificate:</p> <p>Which is their country:</p> <p>Who signed the certificate:</p> <p>Which is their country:</p> <p>What is the serial number of the certificate:</p> <p>Check the serial number for the root certificate. What is its serial number:</p>
B.5	<p>If we want to use this certificate to digitally sign files and verify the signatures, we need to convert it to a PKCS12 file:</p>	<p>Can you view ia.p12 in a text edit?</p>

	<code>openssl pkcs12 -export -out ia.p12 -inkey ia.key -in ia.crt -chain -CAfile ca.crt</code>	
B.6	<p>The crt format is in encoded in binary. If we want to export to a Base64 format, we can use DER:</p> <pre>openssl x509 -inform pem -outform pem -in ca.crt -out ca.cer</pre> <p>and for My Little Corp:</p> <pre>openssl x509 -inform pem -outform pem -in ia.crt -out ia.cer</pre>	<p>View each of the output files in a text editor (ca.cer and then ia.cer). What can you observe from the format:</p> <p>Which are the standard headers and footers used:</p>

B.7 Enter and run the following program, and verify its operation:

```
import OpenSSL.crypto
from OpenSSL.crypto import load_certificate_request, FILETYPE_PEM

csr = '''-----BEGIN NEW CERTIFICATE REQUEST-----
MIICyTCCABECAQAwajELMAkGA1UEBhMCVUSxDTAJBgNVBAgTBTE5vbmUxEjAQBGNV
BACTCUVkaw5idXJnaDEXMBUGA1UEChMOTXkgTG10dGx1IENvcnAxDDAKBgNVBAST
A01MQZERMA8GA1UEAxMITUxdlm5vbmUwggE1MA0GCSqGSIb3DQEBAQUAA4IBDwAw
ggEKAoIBAQCUE68ggssJ210wGxfkjCX3PG/RgSb5VpAp2rzavx71M9Bhg9kuORE
OP7BQC3E6DGu+xba3NdnhrHAFNa+hH9dntZr1xb98am5q9+Tum76V1toIseOMDdu
UE9IpxXoFvD6b0inbFznbrjFj3XUuzIIqvvisw4rIOxzgbwqZ5+F7YpP8d59eww0
6ixzJKoeE/+Gw7Slsdr1+QAUax05MHTweMYbZEhr2M8f1RA4o81zEd2twCK85F
6VS/EkCzUG1cqDBQq7D2S9MWN8Zk2P7CS8/yZx7uRTmt1t3UWKLuyIN0TU3IjCeY
t53P6C+9DT6UD0fDFZRBcmPOH+qb6/YBAGMBAAGGgJAYBgkqhkiG9w0BCQcxCMJ
UXdlcnR5MTIzMA0GCSqGSIb3DQEBAQUAA4IBAQCqpXjmaQf2/o/xbNZG5ggAV8yv
d6rsabnov5zIkciT9NQXsPJEi84u7CbcRiYqY5h7XlMwjv476mAGbgAVZB2ZhI1p
qLa1+lX9xwhFbuLHNRxZCUMM0g9KQZaZtKAQd1DVU/vPZRjq+EHGoPFG7R9QKGD0
k1b4DqOvInWLOs+yuWT7YYtwdr2TNKPpcBqbzCYzrWL6UaUN7LYFpNn4BbqXRgVw
iManUh9fvLMe7oreYfTaevXT/506Sj9WvQFXTCLtRhs+m30q22/wUK0ZZ8APjpwf
rQMegvzXXEIO3xEGRBi5/wXJxsawRLCM3ZSGPu/ws950oM5Ahn8K8HBDkubQ
-----END NEW CERTIFICATE REQUEST-----'''

req = load_certificate_request(FILETYPE_PEM, csr)
key = req.get_pubkey()
key_type = 'RSA' if key.type() == OpenSSL.crypto.TYPE_RSA else 'DSA'
subject = req.get_subject()
components = dict(subject.get_components())
print "key algorithm:", key_type
print "key size:", key.bits()
print "Common name:", components['CN']
print "Organisation:", components['O']
print "Organisational unit", components['OU']
print "City/locality:", components['L']
print "State/province:", components['ST']
print "Country:", components['C']
```

 **Web link (CSR):**

<https://asecuritysite.com/encryption/csr>

D.8 Now check the signing on these certificate requests:

MIICyTCCABECAQAwajELMAKGA1UEBhMCVUSxDALBgNVBAGTBESvbmUXEjAQBGNV
BACTCUVkaw5idXJnaDEXMBUGA1UEChMOTXkgTG10dGx1IENvcnAxDDAKBgNVBAST
A01MQzERMA8GA1UEAxMITUXDLm5vbmUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
ggEKAoIBAQCUCuQE68qgssJ210WgxfkjCX3PG/RgSb5VpAp2rzavx71M9Bhg9kuORE
OP7BQC3E6DGu+xbx3NdnhrHAFNa+hH9dnTzr1xb98Am5q9+Tum76V1toIseOMDdu
UE9IpxXoFvDgb0inbfFznbrjFj3XUUXIIqvviw4rIOxzgwbqWZ457Y7pP8d59eww0
6ixZjKoeE/+gw7S1sdr1+QQAUAx05MHTwemybZEHir2M8f1RA4o81zEd2twCK85F
6VS/EkCZUGlCqDBQQ7D2S9MWN8Zk2P7CS8/yZx7urTmT1t3UWKLuyIN0TU3IjCey
t53P6C+9DT6UD0fDFZRCmPOH+qb6/YBAGMBAAGGgJAYBgkqhkiG9w0BCQCxCxMJ
Uxd1cNr5MTizMA0GCSqGSIb3DQEBBQUAA4IBAQCqpXjmaQf2/o/xbNZG5ggAV8yy
d6rsabnov5ZIkciT9NQXSPJEi84u7CbcirYqY5h7X1Mwjv476mAGbgAvfZB2Zhi1p
qla1+1x9xwhFbuLHNRXZCYMM0g9KQZaTzKAQDLVU/vPzrjq+EHGOpFg7R9QKGD0
klb4DqoFvInWLOS+yuwT7YUwtwdr2TNKPPcBqBzCqzRWL6UAUn7LYFpNn4BbbqRgVw
iManuh9fvLMe7oreYfTaevXT/506Sj9wvQFXTcLTrhs+m30q22/wUK0ZZ8APjpwf
rQMevgzXXEIO3xEGRBi5/wXJxsawRLCM3ZSGPu/ws950om5Ahn8K8HBDkubq

MIIDPZCZAqgcAQAwwZDELMAKGA1UEBHMCMQ04xCzAJBgNVBAGTAmJqMQswCQYDVQQH
EwJiajERMA8GA1UEChMibXhjei5uZXQxETAPBgNVBASTCGI4Y3oubmV0MRUwEwYD
VQQDEWx3d3cubXhjei5uZXQwgZ8wDQYJKoZIhvcNAQEBAQADgY0AMIGJAoGBAMQ7
an4v6PHRusBA0prMWXMWJCXYlA01H0x8pvzj96T5Gwg++JPCQE9guPgGwlD02U0B
NDoeABed1fWykZ+JV5UfiOesJo5swrziUpdm7hf34UApNXho6r4bLIEykw/RnmB
GKnncDq1Pkype+mLR4dp0bnHzhe3lo1ntgd6NpxbAgMBAAggggZMBogCiSGAQBB
gjcnAgMXDBYKNS4yLjm3OTAumjb7BgorBgEEAYI3AGEOMw0wazAOBgNVHQ8BAf8E
BAMCBPAwRAYJKoZIhvcNAQkPBDCwNTAOBgqqhkig9w0DAgICAIAwDgYIKoZIhvcN
AwQCAGCAMACGBSSoAwIHMAoGCCqGSIB3DQMhMBMGAlUDJQMMMAoGCCsGAQUFBwMB
MIH9BgorBgEEAYI3DQICMYHuMIHrAgEBH1oATQBpAGMACgBvAHMAbwBmAHAQAIABS
AFMAQQAgaFMAQwBoAgeAbgBuAGUABAgAEACgB5AHAadABvAGCAcgBhAHAAaAABp
AGMAIABQAHtAbwb2AGKAZAB1AHIDgYKAaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
AA
AA
AA
AA
9DK4absUyy///jlIpvsfmIdHXfAskan7w7PH87asp1wdb6lD9snvLZix1UGK7VQG6
wuFYNI1mqjh1m7ITvvzhjdnx7EzCKBXsxEm4mmbvSNVzqOKAWSDE0gvHq9aCSby
NFBQQMow94LqrG/kuIQTjwvdZA==MIIBYjCCATMCAQAwYkxCZAJBgNVBAYTA1VTMRMwEQYDVQKEwPDYwZm9ybm1h
MRYwFAYDVQQHEW1Nb3VudGFpb1BwawV3MRMwEQYDVQQKEwPhB29nbGUGSW5jMR8w
HQYDVQQLExZjbmZvcmlhdGlvbiBUZWNobm9sb2d5MRcwFQYDVQQDEW53d3cuZ29v
Z2x1LmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwGykCgYEAPztYJCHJ4VpVXHfV
I1stQT104qC03hjx+ZkPyvdYd1Q4+qbAetWxmCUKYHThVRd5aXSq1PzyIBwiemZr
wF1RQddZ1izXA1VRDWWAo60KecqeAXnnUK+5fxoTI/UgwsHre8tJ+x/TMHAQKR/J
ciWPhqaqhsJuzZbvAdGA80BLxdMCAwEAAaAAMA0GCSqGSIb3DQEBBQUAA4GBAih1
4PvFq+e7ipARGI5ZM+GZX6mpCz44DTo0JkwfRdf+BtrsaC0q68eTf2XhYosq4fkh
Q0uA0avog3f5iJxCa3Hp5gxhJQ6zV6kJO6TESuaa0hEko9sdpCoPonRBm2i/XRD2D
6iNh8f8z0ShGsFqjDgFhyF3o+1uyj+UC6H1Qw7bn
-----END CERTIFICATE REQUEST-----

C Elliptic Curve Key Creation

Elliptic curve key pairs are increasingly used within corporate Web sites.

In Openssl we can view the curves with the `ecparam` option:

```
openssl ecparam -list_curves
```

Outline some of the curve names:

By performing an Internet search, which are the most popular curves (and where are they used)?

We can create our elliptic parameter file with:

```
openssl ecparam -name secp256k1 -out secp256k1.pem
```

Now view the details with:

```
openssl ecparam -in secp256k1.pem -text -param_enc explicit -noout
```

What are the details of the key?

Now we can create our key pair:

```
openssl ecparam -in secp256k1.pem -genkey -noout -out mykey.pem
```

Now we will encrypt your key pair (and add a password), and convert it into a format which is ready to be converted into a digital certificate:

```
openssl ec -aes-128-cbc -in mykey.pem -out enckey.pem
```

Finally we will convert into a DER format, so that we can import the keys into a system:

```
openssl ec -in enckey.pem -outform DER -out enckey.der
```

Examine each of the files created and outline what they contain:

Now pick another elliptic curve type and perform the same operations as above. Which type did you use?

Outline the commands used:

If you want to create a non-encrypted version (PFX), which command would you use:

Go to www.cloudflare.com and examine the digital certificate on the site.


What is the public key method used?

What is the size of the public key?

What is the curve type used?

E PFX files

We have a root certificate authority of My Global Corp, which is based in Washington, US, and the administrator is admin@myglobalcorp.com and we are going to issue a certificate to My Little Corp, which is based in Glasgow, UK, and the administrator is admin@mylittlecorp.com.

No	Description	Result
E.1	<p>We will now view some PFX certificate files, and which are protected with a password:</p> <p> Web link (Digital Certificates): http://asecuritysite.com/encryption/digitalcert2</p>	<p>For Certificate 1, can you open it in the Web browser with an incorrect password:</p> <p>Now enter “apples” as a password, and record some of the key details of the certificate:</p> <p>Now repeat for Certificate 2:</p>

E.2	Now with the PFX files (contained in the ZIP files from the Web site), try and import them onto your computer. Try to enter an incorrect password first and observe the message.	<p>Was the import successful?</p> <p>If successful, outline some of the details of the certificates:</p>

F Cracking Certificates

Digital certificates are often protected with a simple password. With this we can use a Python program to try various passwords on the certificate, and if it does not create an exception, then we have found the required password. First download the following pfx files:

 https://asecuritysite.com/cert_crack.zip

Now for fred.pfx, crack the password with the following code:

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

str="fred.pfx"
passwords=["ankle","battery","password","bill","apple","apples","orange"]

for password in passwords:
    try:
        pfx = open(str, 'rb').read()

        p12 = OpenSSL.crypto.load_pkcs12(pfx, password)
        print "Found: ",password

        privkey=OpenSSL.crypto.dump_privatekey(OpenSSL.crypto.FILETYPE_PEM,
p12.get_privatekey())

        cert=OpenSSL.crypto.dump_certificate(OpenSSL.crypto.FILETYPE_PEM,
p12.get_certificate())

        cert = x509.load_pem_x509_certificate(cert, default_backend())

        print " Issuer: ",cert.issuer
        print " Subect: ",cert.subject
        print " Serial number: ",cert.serial_number
        print " Hash: ",cert.signature_hash_algorithm.name
        print privkey
        print certificate

    except:
        print "Not working: ",password
```

What is the password?

The files bill01.pfx, bill02.pfx ... bill18.pfx have a password which are fruits. Can you determine the fruits used:

The files country01.pfx, country02.pfx ... country06.pfx have a password which are countries. Can you determine the countries used:

What I should have learnt from this lab?

The key things learnt:

- Understand how digital certificates are generated and ported onto systems.
- Understand how we can create a key pair for RSA and Elliptic Curve.