Client Certificate PKI Configuration

The idea is simple, the implementation is complex and the execution simple.

The idea is that the client sends a certificate (and key) to the server. The server checks the certificate against a CA and, if it matches, approves the authentication.

The implemention means creating a CA (NOT using a commercial one - this is completely insecure) and a client certificate from the CA. This PKI could be really complicated (Root, Intermediates, signing etc.) or it could be a single Root CA and one certificate per device or user.

The complications are that the certs must contain certain attributes and have certain attribute values formatted in certain ways. Different OSes have different requirements.

The execution is easy. The client sends the cert to the server, which compares it to the stored CA. Again, this could be made more complex by having the CA stored elsewhere and the RADIUS server having to make a call to it.

The approach here is going to be simple. A Root CA with a single certificate and using eapol_test to test.

openssl.conf:

```
#
# OpenSSL configuration file.
#

# Establish working directory.

dir = .

[ ca ]
default_ca      = CA_default

[ CA_default ]
serial          = $dir/serial
database        = $dir/index.txt
new_certs_dir       = $dir/newcerts
certificate     = $dir/cacert-2021.pem
private_key     = $dir/private/cakey.pem
default_days        = 36526
default_md      = SHA256
preserve        = no
email_in_dn     = no
nameopt         = default_ca
certopt         = default_ca
policy          = policy_match
crlDistributionPoints   = URI:http://crldp.govroam.uk/crldp.crl

[ policy_match ]
```

```
countryName     = optional
stateOrProvinceName = optional
localityName        = optional
organizationName    = optional
organizationalUnitName  = optional
commonName      = supplied
emailAddress        = optional

[ req ]
default_bits        = 2048            # Size of keys
default_keyfile     = key.pem        # name of generated keys
string_mask     = default        # permitted characters
distinguished_name  = req_distinguished_name
x509_extensions     = v3_ca

[ req_distinguished_name ]
# Variable name       Prompt string
#----------------------   ----------------------------------
countryName     = Country Name (2 letter code)
countryName_min     = 2
countryName_max     = 2

stateOrProvinceName = State or Province Name (full name)

localityName        = Locality Name (city, district)

0.organizationName  = Organization Name (company)

organizationalUnitName  = Organizational Unit Name (department, division)

emailAddress        = Email Address
emailAddress_max    = 40

commonName      = Common Name (hostname, IP, or your name)
commonName_max      = 64

# Default values for the above, for consistency and less typing.
# Variable name           Value
#------------------------------   ------------------------------
countryName_default     = GB
stateOrProvinceName_default = England
localityName_default        = Manchester
0.organizationName_default  = Scarfolk
organizationalUnitName_default  = Scarfolk
emailAddress_default            = mike.richardson@jisc.ac.uk

distinguished_name  = req_distinguished_name
req_extensions      = v3_req
```

```
[ v3_ca ]
basicConstraints    = CA:TRUE
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer:always
crlDistributionPoints   = URI:http://crldp.govroam.uk/crldp.crl

[ v3_req ]
basicConstraints    = CA:FALSE
subjectKeyIdentifier    = hash

[ xpclient_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
crlDistributionPoints   = URI:http://crldp.govroam.uk/crldp.crl

[ xpserver_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
crlDistributionPoints   = URI:http://crldp.govroam.uk/crldp.crl


[ server ]
basicConstraints        = CA:FALSE
subjectKeyIdentifier    = hash
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
#extendedKeyUsage = serverAuth
extendedKeyUsage = 1.3.6.1.5.5.7.3.1, serverAuth
crlDistributionPoints   = URI:http://crldp.govroam.uk/crldp.crl

[ client ]
basicConstraints = CA:FALSE
nsCertType = client, email, server
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth, emailProtection
crlDistributionPoints   = URI:http://crldp.govroam.uk/crldp.crl
```

The defaults need changing, as do the crlDistributionPoints. The config has been adapted to many things over years so is far from optimal. There will be stuff that's unnecessary and stuff left out but it's been tested and I know it works.

xpextensions:

```
[ xpclient_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2

[ xpserver_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
```

The Root CA

First, generate the certificate:

```
openssl req -new -x509 -extensions v3_ca -keyout private/cakey.pem -out
cacert.pem -days 36500 -config ./openssl.cnf -sha256
```

Remember the password, you'll need it later for the client cert.

Then, convert it to a PCKS12 file:

```
openssl pkcs12 -export -out cacert.pfx -inkey private/cakey.pem -in
cacert.pem
```

The Client Certificate

First, the CSR:

```
openssl req -new -nodes -reqexts client -out newclients/client-req.pem -days
3650 -config ./openssl.cnf
```

It will prompt for a number of fields. The key one is the hostname. It must be set to the username@realm needed. The realm will be used in the outer ID so should be in the right format for routing. The username part is mostly irrelevant, unless used at other points for authorisation.

Then the cert is signed against the CA:

```
openssl ca -out newclients/client-cert.pem -extensions client -config
./openssl.cnf -infiles newclients/client-req.pem
```

Convert the certificate to PKCS12:

```
openssl pkcs12 -export -out newclients/client-cert.pfx -inkey key.pem -in
newclients/client-cert.pem
```

And you're done.