X.509 certificates for user authentication in MongoDB

for Windows 10

Reference: A similar tutorial is available in the MongoDB documentation:
https://docs.mongodb.com/manual/tutorial/configure-x509-client-authentication

github.com/shauryashaurya
shauryashaurya@gmail.com

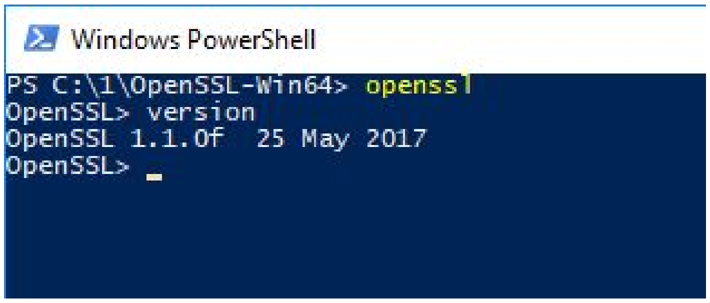
All code in this document is wrapped in single or triple backticks (\`) - ignore these marks when typing code into the terminal or your text editor. These marks are added to make the page compliant with github flavored markdown.

Prep-work

We'd need to install OpenSSL for windows first.
OpenSSL does not provide a binary for windows, however one is available from:
https://slproweb.com/products/Win32OpenSSL.html
We'll use the 64-bit version listed towards the bottom of this page. Look for the words "Win64 OpenSSL" and download the full package (not the 'lite' or 'light' package).

1. Install the downloaded package (you may need administrative privileges).
2. Once installed, open a command window and type:

```
`openssl`
```



```
PS C:\1\OpenSSL-Win64> openssl
OpenSSL> version
OpenSSL 1.1.0f  25 May 2017
OpenSSL> _
```

3. You should see the openssl prompt - this means the install worked successfully.

Overview

This tutorial uses MongoDB version 3.4

The full exercise will be done in 3 steps:

1. Creation of a self-signed x.509 certificate
2. Addition of a user (with the same credentials as the x.509 cert)
3. Logging in into mongo using the certificate

Creating a self-signed x.509 certificate

Typically a certificate would be provided by a certifying authority, however for the purposes of development, we'll create our own i.e. self-signed certificate.

Creating the x.509 certificate requires the following steps:

1. Configure OpenSSL to create mongodb friendly certificates
2. First create the identity for the Certificate Authority (the root organization that'll verify everyone's identity)
    a. Create the root private key and certificate
3. Create the identity for the individual user
    a. Create the private key and generate a certificate signing request (a user can't create it's own certificate, it must be verified by a Certificate Authority)
    b. Create the certificate
4. Merge the certificate and the private key into a single pem file

5. Validate the certificate to make sure everything is fine

Let's carry these out step-by-step:

Configure OpenSSL

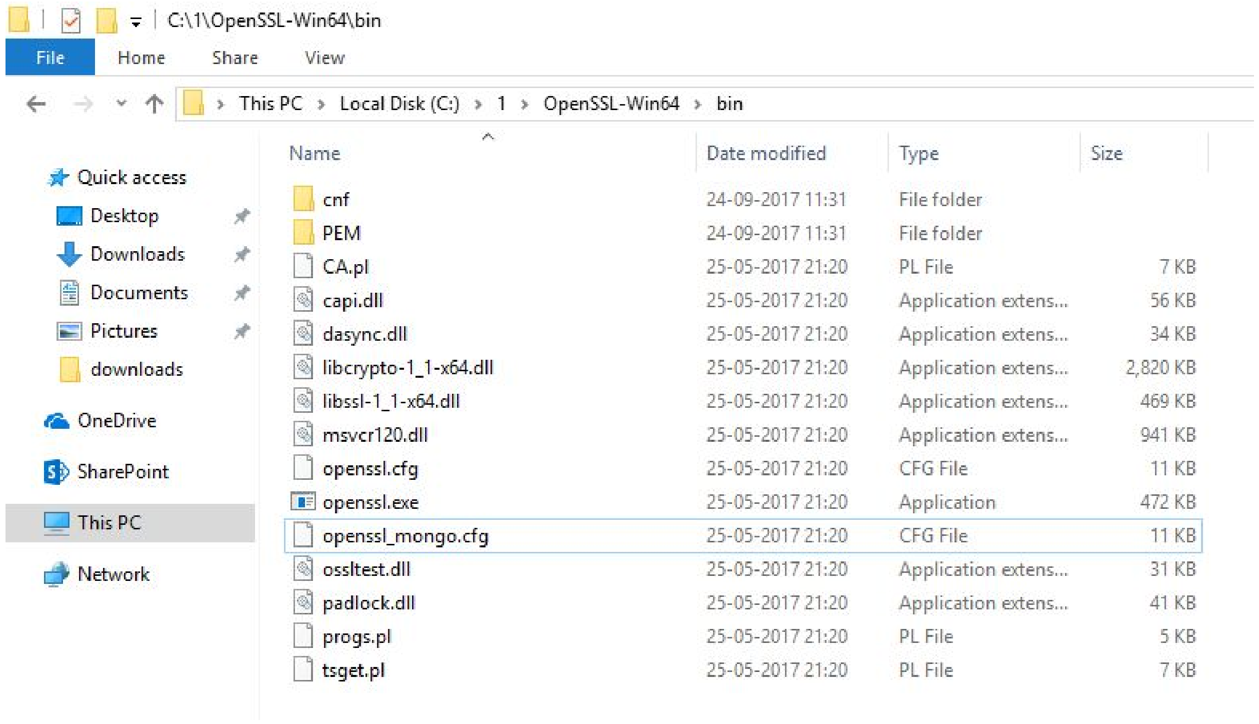
For MongoDB to work with x.509 certificates, the following conditions need to be satisfied:

- A single Certificate Authority (CA) must issue the certificates for both the client and the server.
- Client certificates must contain the following fields:
    - `keyUsage = digitalSignature`
    - `extendedKeyUsage = clientAuth`
- Each unique MongoDB user must have a unique certificate.

There's other conditions too but this is our focus for now. Let's set `keyUsage` and `extendedKeyUsage` fields.

0. Optional: You may want to add `[OpenSSL install location]\bin` to your Environment Variables' Path variable. For e.g. in my case OpenSSL is installed in `C:\1\OpenSSL-Win64` so I have added `C:\1\OpenSSL-Win64\bin` to my Path.

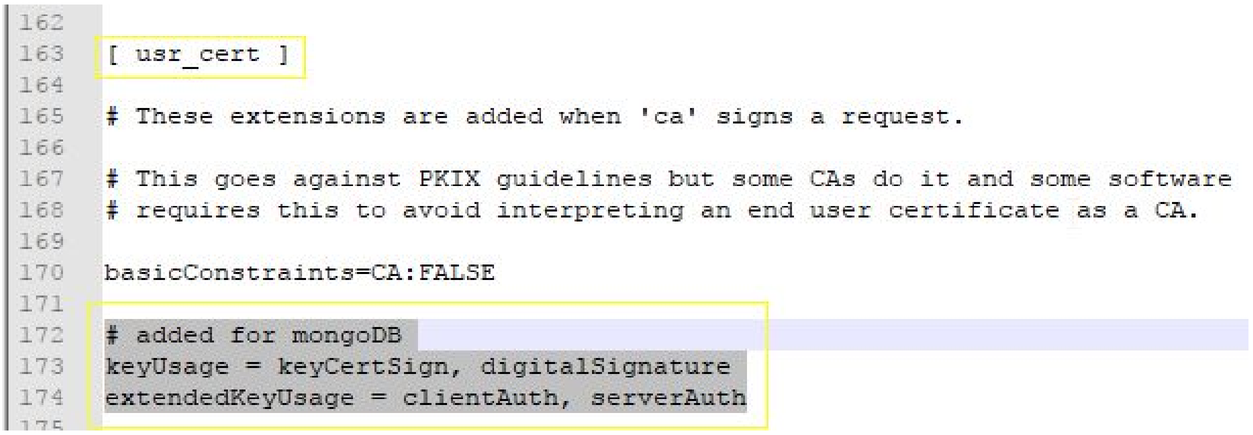
1. Navigate to the directory where OpenSSL is installed, go into the folder called ‘bin’ and make a copy of `` `openssl.cfg` ``. Call this `` `openssl_mongo.cfg` ``.



Name	Date modified	Type	Size
cnf	24-09-2017 11:31	File folder	
PEM	24-09-2017 11:31	File folder	
CA.pl	25-05-2017 21:20	PL File	7 KB
capi.dll	25-05-2017 21:20	Application extens...	56 KB
dasync.dll	25-05-2017 21:20	Application extens...	34 KB
libcrypto-1_1-x64.dll	25-05-2017 21:20	Application extens...	2,820 KB
libssl-1_1-x64.dll	25-05-2017 21:20	Application extens...	469 KB
msvcr120.dll	25-05-2017 21:20	Application extens...	941 KB
openssl.cfg	25-05-2017 21:20	CFG File	11 KB
openssl.exe	25-05-2017 21:20	Application	472 KB
openssl_mongo.cfg	25-05-2017 21:20	CFG File	11 KB
ossltest.dll	25-05-2017 21:20	Application extens...	31 KB
padlock.dll	25-05-2017 21:20	Application extens...	41 KB
progs.pl	25-05-2017 21:20	PL File	5 KB
tsget.pl	25-05-2017 21:20	PL File	7 KB

2. Edit openssl_mongo.cfg and search for the configuration section titled:
   `` `[ usr_cert ]` ``
   Add the following lines in this section:

~~~
```

# added for mongoDB
keyUsage = keyCertSign, digitalSignature
extendedKeyUsage = clientAuth, serverAuth
```
~~~



```
[ usr_cert ]

# These extensions are added when 'ca' signs a request.

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# added for mongoDB
keyUsage = keyCertSign, digitalSignature
extendedKeyUsage = clientAuth, serverAuth
```

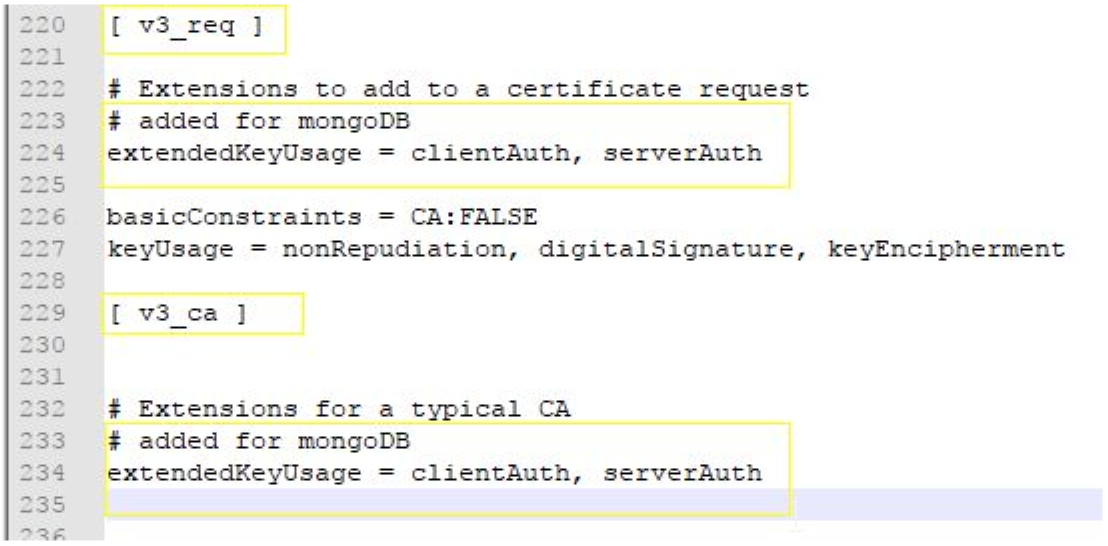
3. In the sections titled:

   `[ v3_ca ]` and

   `[ v3_req ]`

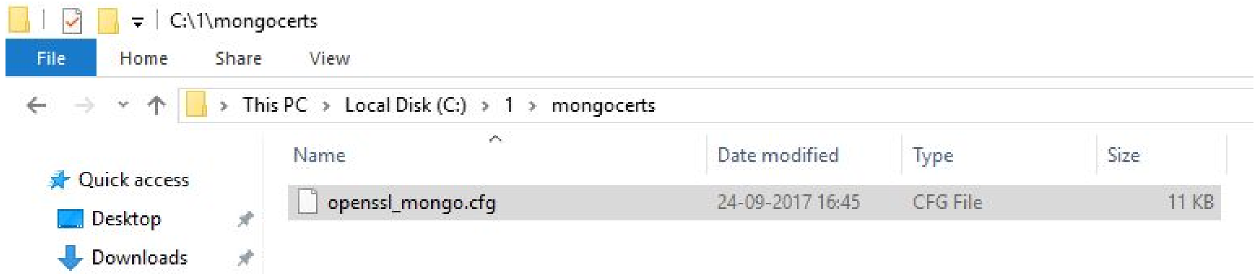
   Add the following lines:

   ```
   # added for mongoDB
   extendedKeyUsage = clientAuth, serverAuth
   ```



```
[ v3_req ]

# Extensions to add to a certificate request
# added for mongoDB
extendedKeyUsage = clientAuth, serverAuth

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]


# Extensions for a typical CA
# added for mongoDB
extendedKeyUsage = clientAuth, serverAuth

```

4. Save & close `` `openssl_mongo.cfg` `` and move it to a directory where we'll create our certificates. In my case I moved it to a folder called mongocerts



Name	Date modified	Type	Size
openssl_mongo.cfg	24-09-2017 16:45	CFG File	11 KB

You are now ready to start making the certificates.

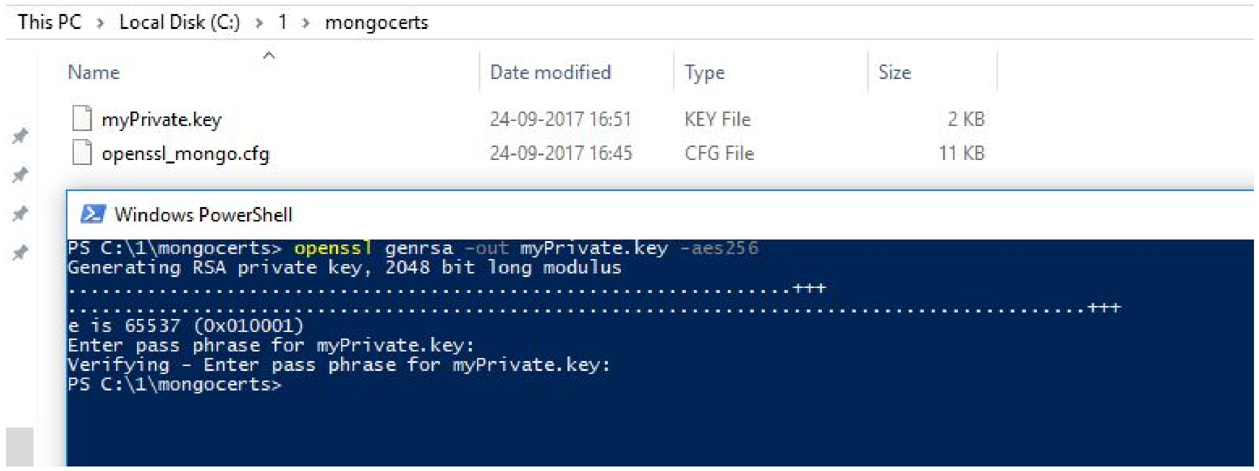
Certificate Authority's private key and certificate

Key

To create the private key, use the following command:

`` `openssl genrsa -out myPrivate.key -aes256` ``

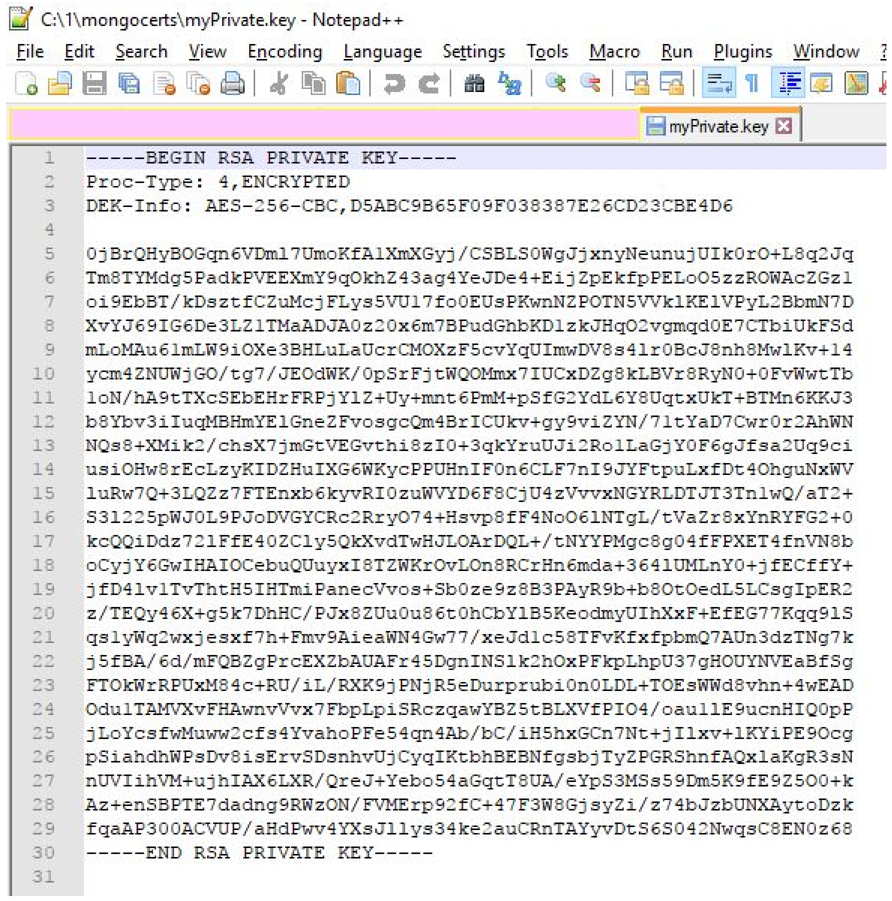
The system will ask for a passphrase, I used: `` `1234` ``



Name	Date modified	Type	Size
myPrivate.key	24-09-2017 16:51	KEY File	2 KB
openssl_mongo.cfg	24-09-2017 16:45	CFG File	11 KB

```
PS C:\1\mongocerts> openssl genrsa -out myPrivate.key -aes256
Generating RSA private key, 2048 bit long modulus
...........................................................+++
..............................................................................+++
e is 65537 (0x010001)
Enter pass phrase for myPrivate.key:
Verifying - Enter pass phrase for myPrivate.key:
PS C:\1\mongocerts>
```

Notice that the system creates a .key file. If you were to open this in a text editor, you'll realize that it's a plain text file.


C:\1\mongocerts\myPrivate.key - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
myPrivate.key

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,D5ABC9B65F09F038387E26CD23CBE4D6

0jBrQHyBOGqn6VDml7UmoKfAlXmXGyj/CSBLS0WgJjxnyNeunujUIk0rO+L8q2Jq
Tm8TYMdg5PadkPVEEXmY9qOkhZ43ag4YeJDe4+EijZpEkfpPELoO5zzROWAcZGzl
oi9EbBT/kDsztfCZuMcjFLys5VUl7fo0EUsPKwnNZPOTN5VVklKElVPyL2BbmN7D
XvYJ69IG6De3LZlTMaADJA0z20x6m7BPudGhbKDlzkJHqO2vgmqd0E7CTbiUkFSd
mLoMAu6lmLW9iOXe3BHLuLaUcrCMOXzF5cvYqUImwDV8s4lr0BcJ8nh8MwlKv+l4
ycm4ZNUWjGO/tg7/JEOdWK/0pSrFjtWQOMmx7IUCxDZg8kLBVr8RyN0+0FvWwtTb
loN/hA9tTXcSEbEHrFRPjYlZ+Uy+mnt6PmM+pSfG2YdL6Y8UqtxUkT+BTMn6KKJ3
b8Ybv3iIuqMBHmYElGneZFvosgcQm4BrICUkv+gy9viZYN/7ltYaD7Cwr0r2AhWN
NQs8+XMik2/chsX7jmGtVEGvthi8zI0+3qkYruUJi2RolLaGjY0F6gJfsa2Uq9ci
usiOHw8rEcLzyKIDZHuIXG6WKycPPUHnIF0n6CLF7nI9JYFtpuLxfDt4OhguNxWV
luRw7Q+3LQZz7FTEnxb6kyvRI0zuWVYD6F8CjU4zVvvxNGYRLDTJT3TnlwQ/aT2+
S31225pWJ0L9PJoDVGYCRc2RryO74+Hsvp8fF4NoO6lNTgL/tVaZr8xYnRYFG2+0
kcQQiDdz72lFfE40ZCly5QkXvdTwHJLOArDQL+/tNYYPMgc8g04fFPXET4fnVN8b
oCyjY6GwIHAIOCebuQUuyxI8TZWKrOvLOn8RCrHn6mda+364lUMLnY0+jfECffY+
jfD4lvlTvThtH5IHTmiPanecVvos+Sb0ze9z8B3PAyR9b+b8OtOedL5LCsgIpER2
z/TEQy46X+g5k7DhHC/PJx8ZUu0u86t0hCbYlB5KeodmyUIhXxF+EfEG77Kqq9lS
qslyWq2wxjesxf7h+Fmv9AieaWN4Gw77/xeJdlc58TFvKfxfpbmQ7AUn3dzTNg7k
j5fBA/6d/mFQBZgPrcEXZbAUAFr45DgnINSlk2hOxPFkpLhpU37gHOUYNVEaBfSg
FTOkWrRPUxM84c+RU/iL/RXK9jPNjR5eDurprubi0n0LDL+TOEsWWd8vhn+4wEAD
OdulTAMVXvFHAwnvVvx7FbpLpiSRczqawYBZ5tBLXVfPIO4/oaulIE9ucnHIQ0pP
jLoYcsfwMuww2cfs4YvahoPFe54qn4Ab/bC/iH5hxGCn7Nt+jIlxv+lKYiPE9Ocg
pSiahdhWPsDv8isErvSDsnhvUjCyqIKtbhBEBNfgsbjTyZPGRShnfAQxlaKgR3sN
nUVIihVM+ujhIAX6LXR/QreJ+Yebo54aGqtT8UA/eYpS3MSs59Dm5K9fE9Z5O0+k
Az+enSBPTE7dadng9RWzON/FVMErp92fC+47F3W8GjsyZi/z74bJzbUNXAytoDzk
fqaAP300ACVUP/aHdPwv4YXsJllys34ke2auCRnTAYyvDtS6S042NwqsC8EN0z68
-----END RSA PRIVATE KEY-----
```

Cool!

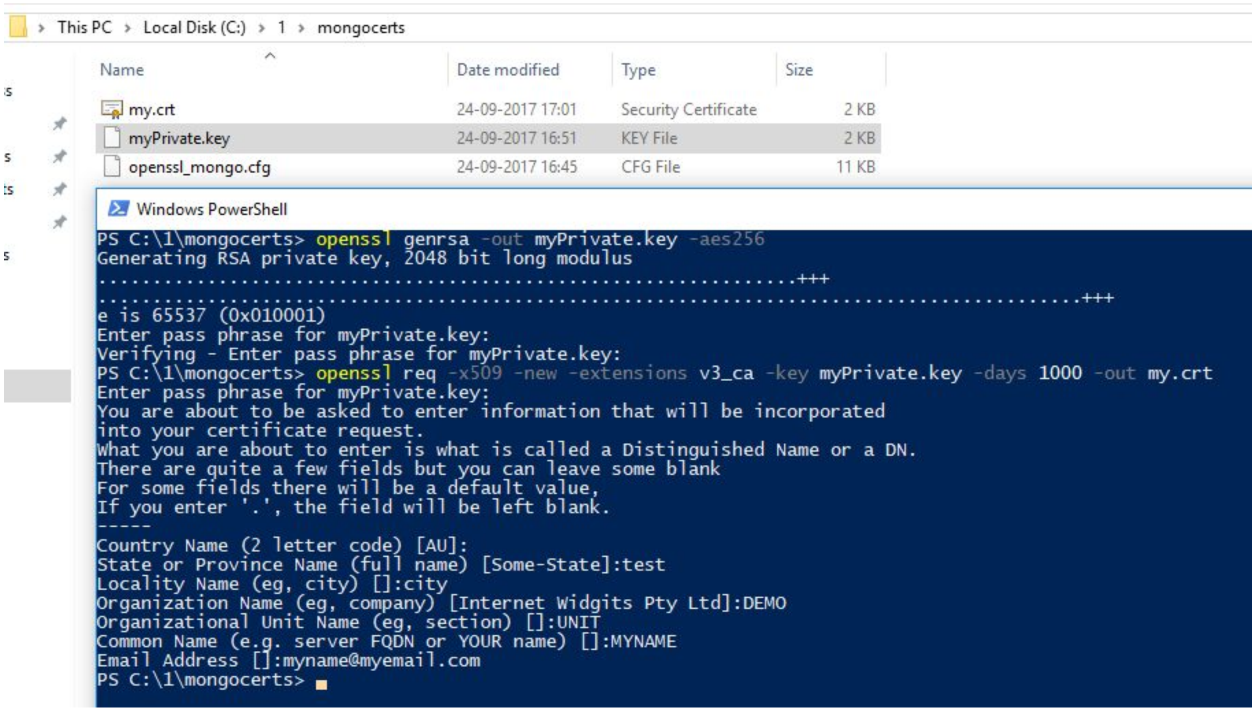
On to creating the root certificate.

Certificate

Use the following command.

`openssl req -x509 -new -extensions v3_ca -key myPrivate.key -days 1000 -out my.crt`

The system will ask you to enter the passphrase for the key (which is `1234` if you used the same one as me) and then ask a bunch of questions to make the certificate. The values that you enter here are combined to create the “Distinguished Name” of your certificate. *Remember these values.*



```
This PC > Local Disk (C:) > 1 > mongocerts

Name                 Date modified      Type                  Size
my.crt               24-09-2017 17:01   Security Certificate  2 KB
myPrivate.key        24-09-2017 16:51   KEY File              2 KB
openssl_mongo.cfg    24-09-2017 16:45   CFG File              11 KB
```

```
PS C:\1\mongocerts> openssl genrsa -out myPrivate.key -aes256
Generating RSA private key, 2048 bit long modulus
...............................................................+++
.........................................................................................+++
e is 65537 (0x010001)
Enter pass phrase for myPrivate.key:
Verifying - Enter pass phrase for myPrivate.key:
PS C:\1\mongocerts> openssl req -x509 -new -extensions v3_ca -key myPrivate.key -days 1000 -out my.crt
Enter pass phrase for myPrivate.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:test
Locality Name (eg, city) []:city
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DEMO
Organizational Unit Name (eg, section) []:UNIT
Common Name (e.g. server FQDN or YOUR name) []:MYNAME
Email Address []:myname@myemail.com
PS C:\1\mongocerts>
```

Here are the values that I used:

- Country Name - `AU`
- State or Province name - `test`
- Locality name - `city`
- Organization name - `DEMO`
- Organizational unit name - `UNIT`
- Common Name - `MYNAME`
- Email Address -`myname@myemail.com`

Feel free to use the ones you like. Again, open the certificate in a text editor - you'll realize that this is a plain text file too!

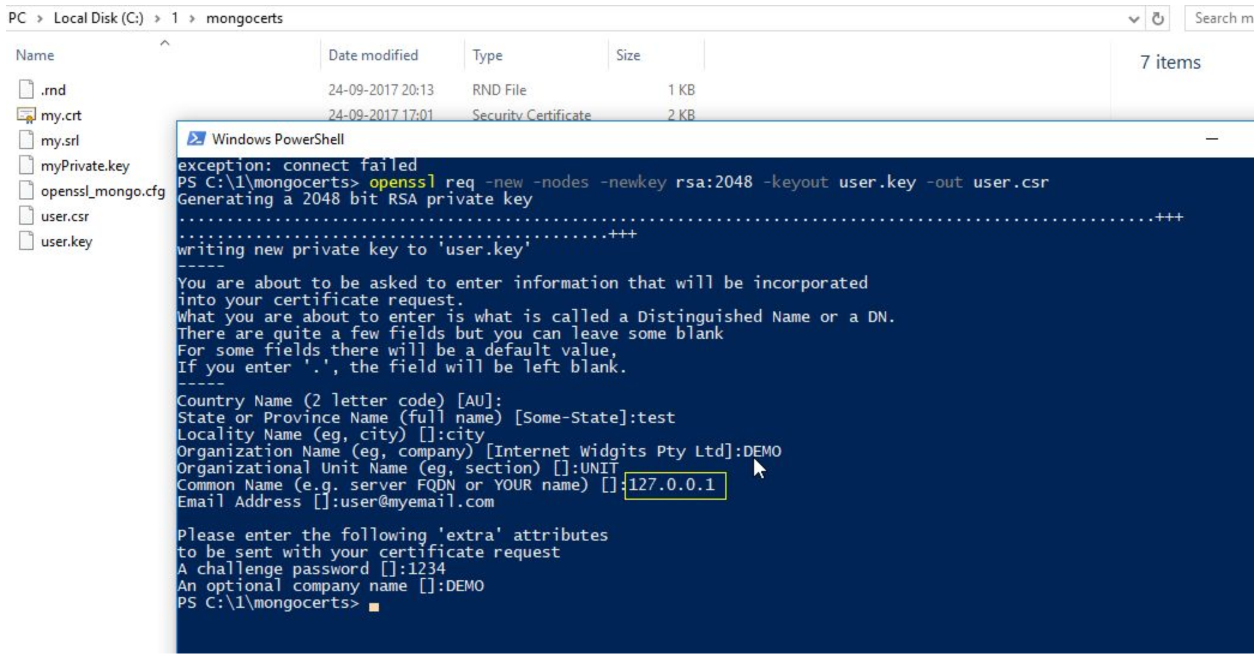
With our root certifying authority creds ready, our certifying authority is ready to sign any end-user's identity. Let's move on to creating a certificate signing request.

User's Key and Certificate Signing Request

We'll use a single command to create both the key and the CSR for a user.

Try the following in the command window next:

```
openssl req -new -nodes -newkey rsa:2048 -keyout user.key -out user.csr
```



```
.rnd              24-09-2017 20:13   RND File              1 KB
my.crt            24-09-2017 17:01   Security Certificate  2 KB
my.srl
myPrivate.key
openssl_mongo.cfg
user.csr
user.key
```

```
exception: connect failed
PS C:\1\mongocerts> openssl req -new -nodes -newkey rsa:2048 -keyout user.key -out user.csr
Generating a 2048 bit RSA private key
.........................................................................................................+++
...............................................+++
writing new private key to 'user.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:test
Locality Name (eg, city) []:city
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DEMO
Organizational Unit Name (eg, section) []:UNIT
Common Name (e.g. server FQDN or YOUR name) []:127.0.0.1
Email Address []:user@myemail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:1234
An optional company name []:DEMO
PS C:\1\mongocerts>
```

We now have the user key and CSR. Assuming you are running the whole shebang from your localhost, make sure you put common name (CN) to be `127.0.0.1` otherwise put the relevant IP of the Mongodb server here - when authenticating, the CN should match the IP of the server.

Here are the values that I used when creating these:

- Country Name: `AU`
- State or Province Name: `test`
- Locality Name: `city`
- Organization Name: `DEMO`
- Organizational Unit Name: `UNIT`
- Common Name: `127.0.0.1`
- Email Address: `user@myemail.com`
- The 'extra' attributes
    - A challenge password: `1234`

- An optional company name:`DEMO`

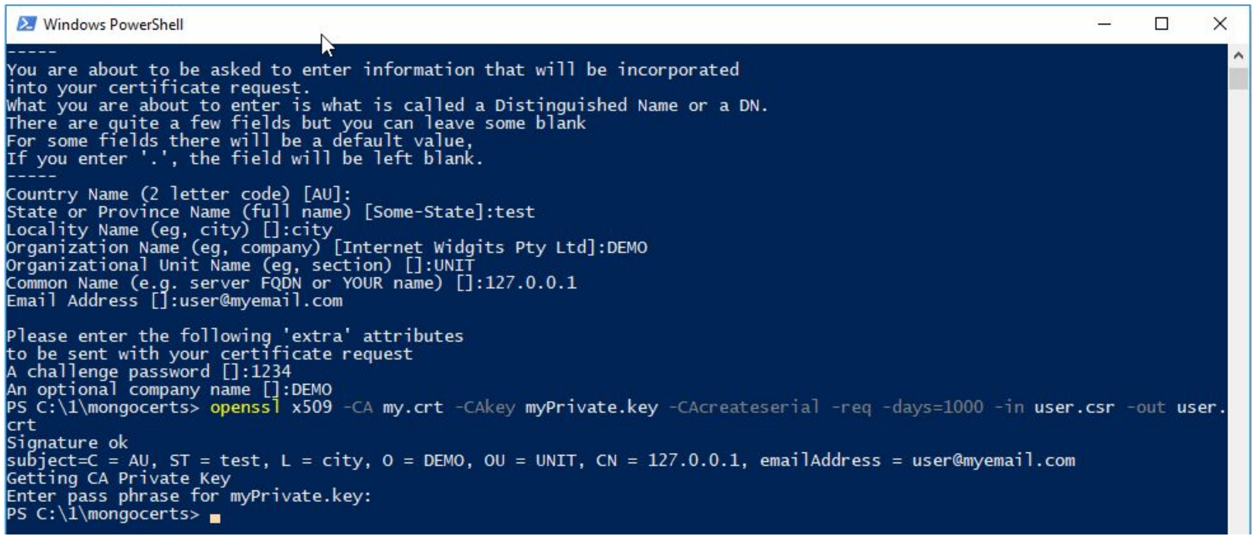
Again, *remember these values,* they form the "Distinguished Name" of your user's certificate. The key and certificate signing request are both plain text files, just like the others that were created before.

We are getting close. Let's create the x.509 certificates for our user.

User's X.509 certificate from CSR

To generate the public certificate, use the following command:

```
openssl x509 -CA my.crt -CAkey myPrivate.key -CAcreateserial
-req -days=1000 -in user.csr -out user.crt
```



```
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:test
Locality Name (eg, city) []:city
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DEMO
Organizational Unit Name (eg, section) []:UNIT
Common Name (e.g. server FQDN or YOUR name) []:127.0.0.1
Email Address []:user@myemail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:1234
An optional company name []:DEMO
PS C:\1\mongocerts> openssl x509 -CA my.crt -CAkey myPrivate.key -CAcreateserial -req -days=1000 -in user.csr -out user.
crt
Signature ok
subject=C = AU, ST = test, L = city, O = DEMO, OU = UNIT, CN = 127.0.0.1, emailAddress = user@myemail.com
Getting CA Private Key
Enter pass phrase for myPrivate.key:
PS C:\1\mongocerts>
```

Perfect. Now to make it usable, we want to bundle the certificate and the key in a single file. By the way, before we go there, what kind of file would the final certificate be? Text, plain text. :)

Merge the user's key & cert

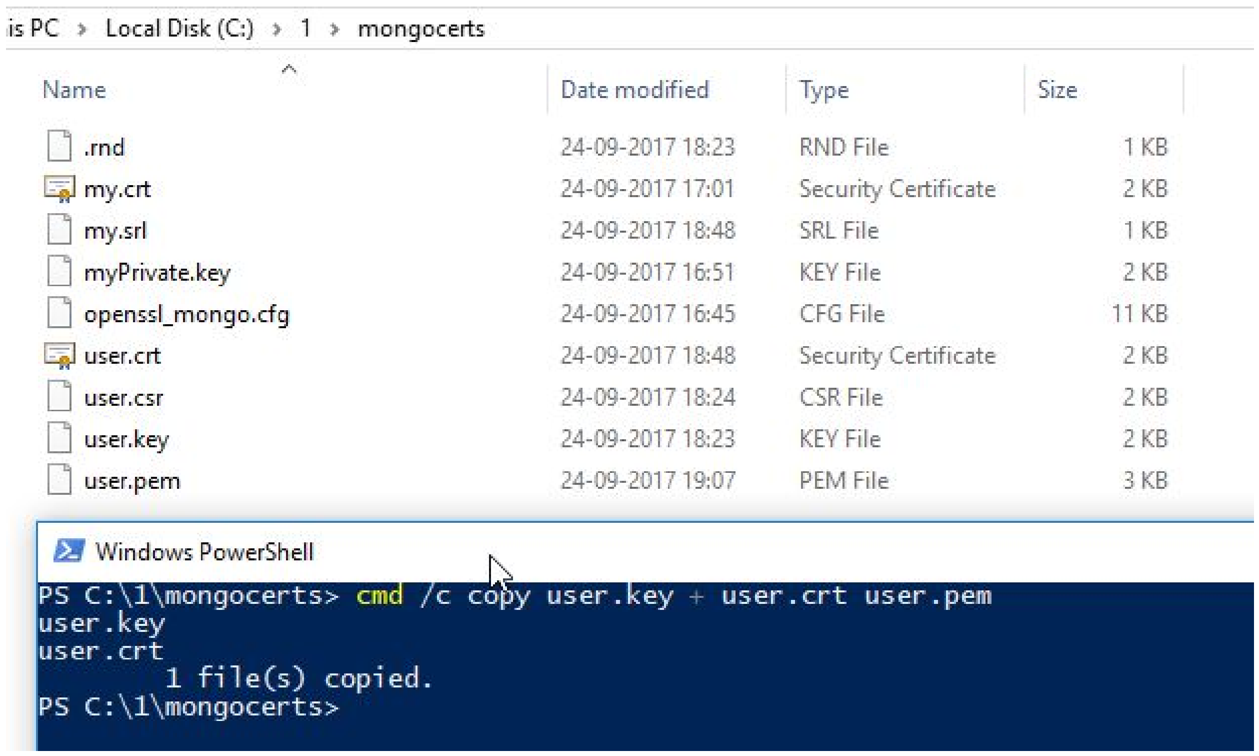
The format we'll merge this to is called PEM (Privacy Enhanced Mail). As with all other certificates and keys that we have generated till now, this is a plain text file too.

Creating this is super simple. Use concatenation, or from the windows command-line, use `copy`, like so:

```
 copy user.key + user.crt user.pem
```

However, if you are using powershell, the plain old `copy` may not work, in which case, use:

```
cmd /c copy user.key + user.crt user.pem
```


is PC › Local Disk (C:) › 1 › mongocerts

Name	Date modified	Type	Size
.rnd	24-09-2017 18:23	RND File	1 KB
my.crt	24-09-2017 17:01	Security Certificate	2 KB
my.srl	24-09-2017 18:48	SRL File	1 KB
myPrivate.key	24-09-2017 16:51	KEY File	2 KB
openssl_mongo.cfg	24-09-2017 16:45	CFG File	11 KB
user.crt	24-09-2017 18:48	Security Certificate	2 KB
user.csr	24-09-2017 18:24	CSR File	2 KB
user.key	24-09-2017 18:23	KEY File	2 KB
user.pem	24-09-2017 19:07	PEM File	3 KB

Windows PowerShell

```
PS C:\1\mongocerts> cmd /c copy user.key + user.crt user.pem
user.key
user.crt
        1 file(s) copied.
PS C:\1\mongocerts>
```

Some of you may point out that we could've used the get-content commandlet in Powershell. Sure! Feel free to use it instead.

Certificate validation

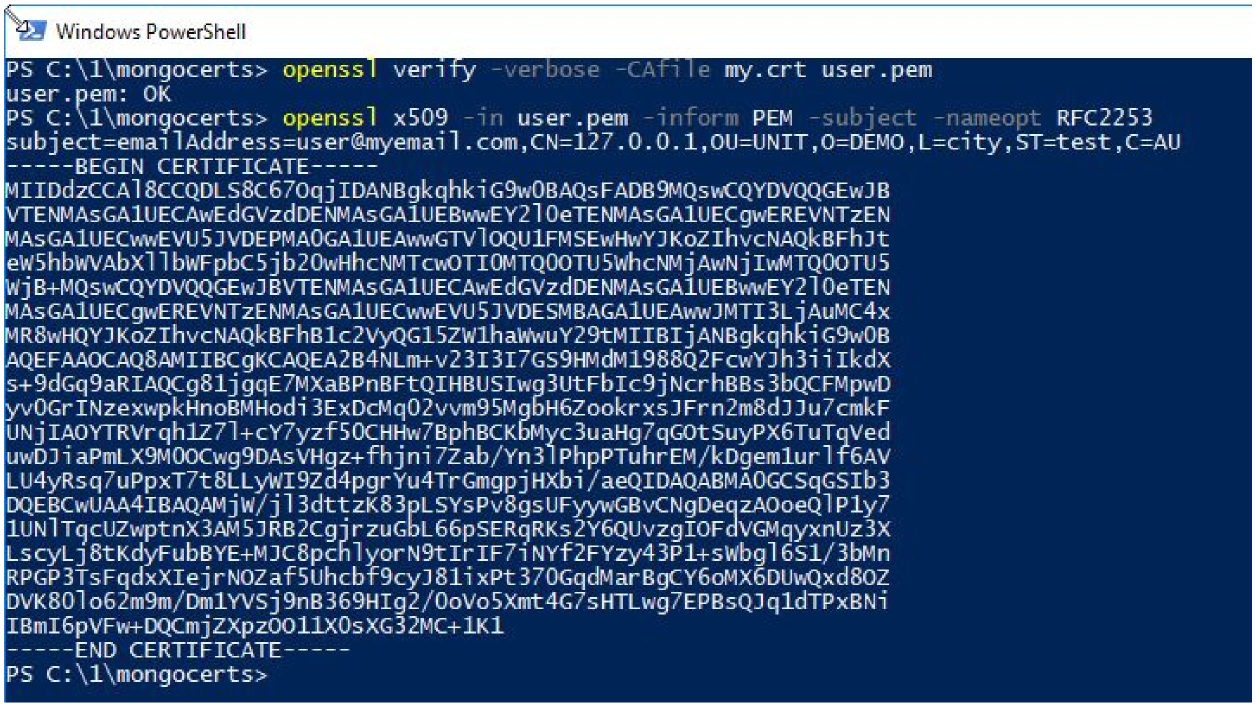
Whew! Nearly there!

Our certificate is mint fresh and just to be sure, we'd want to verify it *against the certificate authority*. Use the following command:

```
openssl verify -verbose -CAfile my.crt user.pem
```

Now, let's check out the subject line (aka the 'Distinguished Name') of our certificate, we'll use this to create a user in Mongo, so keep a note of it. Run the following command:

```
openssl x509 -in user.pem -inform PEM -subject -nameopt RFC2253
```



```
PS C:\1\mongocerts> openssl verify -verbose -CAfile my.crt user.pem
user.pem: OK
PS C:\1\mongocerts> openssl x509 -in user.pem -inform PEM -subject -nameopt RFC2253
subject=emailAddress=user@myemail.com,CN=127.0.0.1,OU=UNIT,O=DEMO,L=city,ST=test,C=AU
-----BEGIN CERTIFICATE-----
MIIDdzCCAl8CCQDLS8C67OqjIDANBgkqhkiG9w0BAQsFADB9MQswCQYDVQQGEwJB
VTENMAsGA1UECAwEdGVzdDENMAsGA1UEBwwEY2l0eTENMAsGA1UECgwEREVNTzEN
MAsGA1UECwwEVU5JVDEPMA0GA1UEAwwGTVlOQU1FMSEwHwYJKoZIhvcNAQkBFhJt
eW5hbWVAbXllbWFpbC5jb20wHhcNMTcwOTI0MTQ0OTU5WhcNMjAwNjIwMTQ0OTU5
WjB+MQswCQYDVQQGEwJBVTENMAsGA1UECAwEdGVzdDENMAsGA1UEBwwEY2l0eTEN
MAsGA1UECgwEREVNTzENMAsGA1UECwwEVU5JVDESMBAGA1UEAwwJMTI3LjAuMC4x
MR8wHQYJKoZIhvcNAQkBFhB1c2VyQG15ZW1haWwuY29tMIIBIjANBgkqhkiG9w0B
AQEFAAOCAQ8AMIIBCgKCAQEA2B4NLm+v23I3I7GS9HMdM1988Q2FcwYJh3iiIkdX
s+9dGq9aRIAQCg81jgqE7MXaBPnBFtQIHBUSIwg3UtFbIc9jNcrhBBs3bQCFMpwD
yv0GrINzexwpkHnoBMHodi3ExDcMq02vvm95MgbH6ZookrxsJFrn2m8dJJu7cmkF
UNjIAOYTRVrqh1Z7l+cY7yzf50CHHw7BphBCKbMyc3uaHg7qGOtSuyPX6TuTqVed
uwDJiaPmLX9M0OCwg9DAsVHgz+fhjni7Zab/Yn3lPhpPTuhrEM/kDgem1urlf6AV
LU4yRsq7uPpxT7t8LLyWI9Zd4pgrYu4TrGmgpjHXbi/aeQIDAQABMA0GCSqGSIb3
DQEBCwUAA4IBAQAMjW/jl3dttzK83pLSYsPv8gsUFyywGBvCNgDeqzAOoeQlP1y7
1UNlTqcUZwptnX3AM5JRB2CgjrzuGbL66pSERqRKs2Y6QUvzgIOFdVGMqyxnUz3X
LscyLj8tKdyFubBYE+MJC8pchlyorN9tIrIF7iNYf2FYzy43P1+sWbgl6S1/3bMn
RPGP3TsFqdxXIejrNOZaf5Uhcbf9cyJ81ixPt370GqdMarBgCY6oMX6DUwQxd8OZ
DVK8Olo62m9m/Dm1YVSj9nB369HIg2/OoVo5Xmt4G7sHTLwg7EPBsQJq1dTPxBNi
IBmI6pVFw+DQCmjZXpzO011X0sXG32MC+1K1
-----END CERTIFICATE-----
PS C:\1\mongocerts>
```

Brilliant! Our self-signed certificate is ready to go.

Add a user to Mongodb

Before we enable authentication, we now need to add a user to MongoDB.

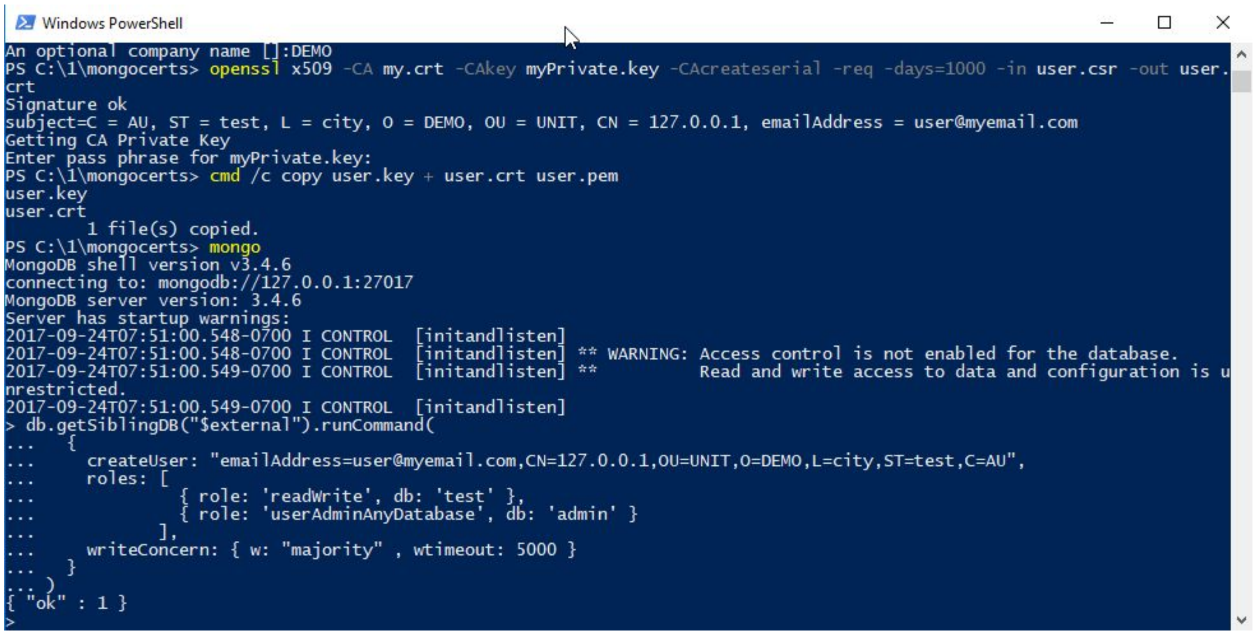
1. Open two command-line/ powershell windows, in one fire:

   `mongod`
2. This should start the MongoDB server. (Ensure that the folder `C:\data\db` exists, this is where Mongodb will store its data by default)
3. In the other window, fire:

   `mongo`
4. You should enter the mongo shell.
5. Now run the following command to add the user (notice that there are no spaces in the `subject` - it exists as it was copied from the user's certificate)

   ```
   db.getSiblingDB("$external").runCommand(
     {
       createUser:
   "emailAddress=user@myemail.com,CN=127.0.0.1,OU=UNIT,O=DEMO,L=city,ST=test,C=AU",
       roles: [
                { role: 'readWrite', db: 'test' },
                { role: 'userAdminAnyDatabase', db: 'admin' }
              ],
       writeConcern: { w: "majority" , wtimeout: 5000 }
     }
   )
   ```

6. If the user got added successfully, you should see `{ "ok" : 1 }



```
An optional company name []:DEMO
PS C:\1\mongocerts> openssl x509 -CA my.crt -CAkey myPrivate.key -CAcreateserial -req -days=1000 -in user.csr -out user.
crt
Signature ok
subject=C = AU, ST = test, L = city, O = DEMO, OU = UNIT, CN = 127.0.0.1, emailAddress = user@myemail.com
Getting CA Private Key
Enter pass phrase for myPrivate.key:
PS C:\1\mongocerts> cmd /c copy user.key + user.crt user.pem
user.key
user.crt
        1 file(s) copied.
PS C:\1\mongocerts> mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.6
Server has startup warnings:
2017-09-24T07:51:00.548-0700 I CONTROL  [initandlisten]
2017-09-24T07:51:00.548-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-09-24T07:51:00.549-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2017-09-24T07:51:00.549-0700 I CONTROL  [initandlisten]
> db.getSiblingDB("$external").runCommand(
...    {
...      createUser: "emailAddress=user@myemail.com,CN=127.0.0.1,OU=UNIT,O=DEMO,L=city,ST=test,C=AU",
...      roles: [
...               { role: 'readWrite', db: 'test' },
...               { role: 'userAdminAnyDatabase', db: 'admin' }
...             ],
...      writeConcern: { w: "majority" , wtimeout: 5000 }
...    }
... )
{ "ok" : 1 }
>
```

7. Type `exit` to exit the mongo client, move to the mongod window and hit `CTRL+C` to shut down the mongod server.

On to the last leg.

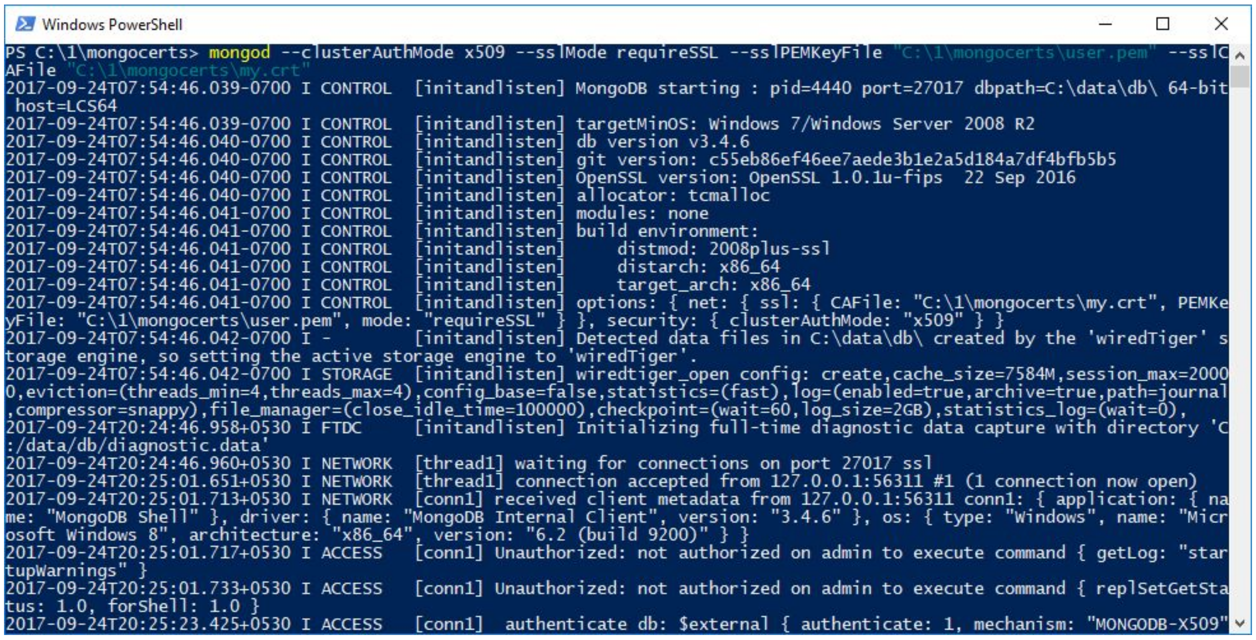
Using X.509 to login

This is the simple bit. We'll first start the mongod server with ssl enabled and then login using the mongo client. Once in, we'll authenticate our user using the subject line.

1. Start `mongod` with SSL enabled:

   ```
   mongod --clusterAuthMode x509 --sslMode requireSSL
   --sslPEMKeyFile "C:\1\mongocerts\user.pem" --sslCAFile
   "C:\1\mongocerts\my.crt"
   ```

   Notice that I am using the paths relevant to my system, replace these with the paths that you have chosen to store the certificates in on your system.



```
PS C:\1\mongocerts> mongod --clusterAuthMode x509 --sslMode requireSSL --sslPEMKeyFile "C:\1\mongocerts\user.pem" --sslC
AFile "C:\1\mongocerts\my.crt"
2017-09-24T07:54:46.039-0700 I CONTROL  [initandlisten] MongoDB starting : pid=4440 port=27017 dbpath=C:\data\db\ 64-bit
 host=LCS64
2017-09-24T07:54:46.039-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2017-09-24T07:54:46.040-0700 I CONTROL  [initandlisten] db version v3.4.6
2017-09-24T07:54:46.040-0700 I CONTROL  [initandlisten] git version: c55eb86ef46ee7aede3b1e2a5d184a7df4bfb5b5
2017-09-24T07:54:46.040-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips  22 Sep 2016
2017-09-24T07:54:46.040-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2017-09-24T07:54:46.041-0700 I CONTROL  [initandlisten] modules: none
2017-09-24T07:54:46.041-0700 I CONTROL  [initandlisten] build environment:
2017-09-24T07:54:46.041-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2017-09-24T07:54:46.041-0700 I CONTROL  [initandlisten]     distarch: x86_64
2017-09-24T07:54:46.041-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2017-09-24T07:54:46.041-0700 I CONTROL  [initandlisten] options: { net: { ssl: { CAFile: "C:\1\mongocerts\my.crt", PEMKe
yFile: "C:\1\mongocerts\user.pem", mode: "requireSSL" } }, security: { clusterAuthMode: "x509" } }
2017-09-24T07:54:46.042-0700 I -        [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' s
torage engine, so setting the active storage engine to 'wiredTiger'.
2017-09-24T07:54:46.042-0700 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7584M,session_max=2000
0,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal
,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-09-24T20:24:46.958+0530 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C
:/data/db/diagnostic.data'
2017-09-24T20:24:46.960+0530 I NETWORK  [thread1] waiting for connections on port 27017 ssl
2017-09-24T20:25:01.651+0530 I NETWORK  [thread1] connection accepted from 127.0.0.1:56311 #1 (1 connection now open)
2017-09-24T20:25:01.713+0530 I NETWORK  [conn1] received client metadata from 127.0.0.1:56311 conn1: { application: { na
me: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Windows", name: "Micr
osoft Windows 8", architecture: "x86_64", version: "6.2 (build 9200)" } }
2017-09-24T20:25:01.717+0530 I ACCESS   [conn1] Unauthorized: not authorized on admin to execute command { getLog: "star
tupWarnings" }
2017-09-24T20:25:01.733+0530 I ACCESS   [conn1] Unauthorized: not authorized on admin to execute command { replSetGetSta
tus: 1.0, forShell: 1.0 }
2017-09-24T20:25:23.425+0530 I ACCESS   [conn1]  authenticate db: $external { authenticate: 1, mechanism: "MONGODB-X509"
```

`mongod` should start.

2. In another command / powershell window, use the following command to start the `mongo` client:

   ```
   mongo --ssl --sslPEMKeyFile "C:\1\mongocerts\user.pem"
   --sslCAFile "C:\1\mongocerts\my.crt"
   ```

   Again, remember to tweak the paths to reflect the location of the certificates on your system.

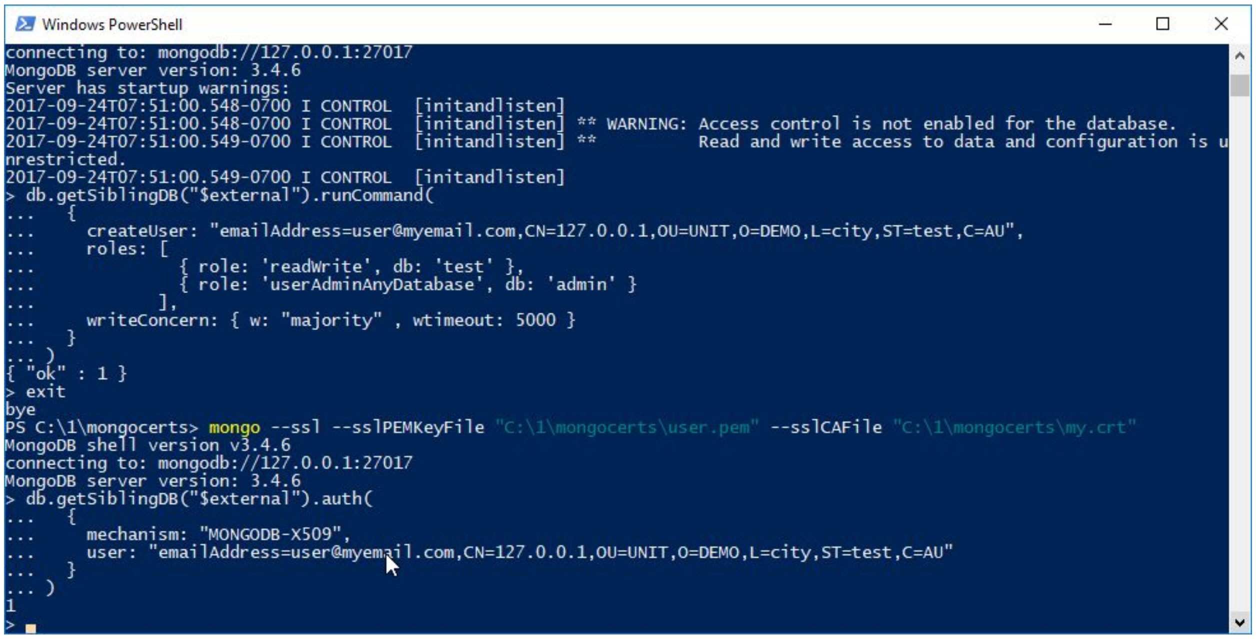
3. Once `mongo` client starts, use the following command to authenticate our user:

   ```
   db.getSiblingDB("$external").auth(
     {
       mechanism: "MONGODB-X509",
       user:
   "emailAddress=user@myemail.com,CN=127.0.0.1,OU=UNIT,O=DEMO,
   ```

```
L=city,ST=test,C=AU"
   }
)
```

Remember to mark the mechanism as `”MONGODB-X509”`

If all goes well, the user should now be authenticated and you should see `1` as the output.



Windows PowerShell

```
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.6
Server has startup warnings:
2017-09-24T07:51:00.548-0700 I CONTROL  [initandlisten]
2017-09-24T07:51:00.548-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-09-24T07:51:00.549-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2017-09-24T07:51:00.549-0700 I CONTROL  [initandlisten]
> db.getSiblingDB("$external").runCommand(
...    {
...      createUser: "emailAddress=user@myemail.com,CN=127.0.0.1,OU=UNIT,O=DEMO,L=city,ST=test,C=AU",
...      roles: [
...              { role: 'readWrite', db: 'test' },
...              { role: 'userAdminAnyDatabase', db: 'admin' }
...            ],
...      writeConcern: { w: "majority" , wtimeout: 5000 }
...    }
... )
{ "ok" : 1 }
> exit
bye
PS C:\1\mongocerts> mongo --ssl --sslPEMKeyFile "C:\1\mongocerts\user.pem" --sslCAFile "C:\1\mongocerts\my.crt"
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.6
> db.getSiblingDB("$external").auth(
...    {
...      mechanism: "MONGODB-X509",
...      user: "emailAddress=user@myemail.com,CN=127.0.0.1,OU=UNIT,O=DEMO,L=city,ST=test,C=AU"
...    }
... )
1
> 
```

There, You are now logged into Mongodb using X.509 certificate authentication.

Sweet!

Read more

- The OpenSSL Cookbook: https://www.feistyduck.com/library/openssl-cookbook/online/
- Awesome cryptography: https://github.com/sobolevn/awesome-cryptography

Who me?

github.com/shauryashaurya
shauryashaurya@gmail.com
www.linkedin.com/in/shauryashaurya/