bc - an arbitrary precision calculator language

Windows: http://gnuwin32.sourceforge.net/packages/bc.htm
macOS: should come built in
Linux: should come built in

```
Flags:
       -l : uses mathlib libraries and makes more functions available
       -q : quiet, doesn't show headers when starting

Commands:
       scale : changes how many decimal places to use (for integer math, set to 0)
       ibase : this is the base numbering system for input
       obase : this is the base numbering system for output
       last : this returns the last outputted number

Example:
       basic usage

$ bc -q -l
/* -l command loads mathlib and sets scale=20 */
scale
20
3+4
7
4*5
20
/* a(x) is the arctan(x), we can use it to define pi */
pi=4*a(1)
radius=7
circumference=2*pi*radius
area=pi*radius^2
pi
3.14159265358979323844
radius
7
circumference
43.98229715025710533816
area
153.93804002589986868356
/* if we have three circles, how much total area is it */
last*3
461.81412007769960605068
/* if you use modulus, watch your scale */
scale
20
10%6
.0000000000000000004
scale=0
10%6
4
```

Example:
convert a hexadecimal number into decimal and binary.
Note, hexadecimal characters in bc have to be capitalized

```
$ bc -q
ibase=16
6F1F767BF5E14A4DE9D5DF
1343393449862866403313474 23
obase=2
6F1F767BF5E14A4DE9D5DF
110111100011111011101100111101111110101111000010100101001001101111 01\
0011101010111011111
```

Example:
Run our RSA algorithm and encrypt/decrypt the number 17. Our modulus is 55, our public exponent is 7 and our private exponent is 23. The modulo operator is "%", the "^" operator is used for exponentiation.

```
$ bc -q
17^7%55
8
8^23%55
17
```

openssl – certificate swiss army knife

Windows: http://gnuwin32.sourceforge.net/packages/openssl.htm
macOS: http://macappstore.org/openssl/ or google for other instructions
Linux: should be built in

```
openssl command [ command_opts ] [ command_args ]

Commands:
       x509           : give us information about a certificate file
       rsa            : give us information about a key file
       genrsa         : generate an RSA key
       s_client       : connect to a host port and talk TLS/SSL.  Supports plain SSL,
                        or TLS for smtp, pop3, imap, ftp and ldap (requires patch)
       dgst           : run a cryptographic digest like SHA256 or MD5
       aes-256-cbc    : encrypt using AES
       ecparam        : EC parameter manipulation and generation
       ec             : EC key processing
```

Example:
Create an RSA key

```
$ openssl genrsa -out ${USER}_ca.key 2048
Generating RSA private key, 2048 bit long modulus
............+++++
.....................................+++++
e is 65537 (0x010001)
```

Example:
Create a Certificate for this key (aka, certificate signing request, CSR). Make sure to use your assigned username instead of "User50"

```
$ openssl req -key ${USER}_ca.key -new -out ${USER}_ca.csr
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) [XX]:US
State or Province Name (full name) []:IL
Locality Name (eg, city) [Default City]:Chicago
Organization Name (eg, company) [Default Company Ltd]:Fun with Certificates
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:FWC User50 CA
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Example:
Self sign the ${USER}_ca.csr file to make a self-signed certifcate authority

```
$ openssl req -x509 -in ${USER}_ca.csr -key ${USER}_ca.key -out ${USER}_ca.crt
```

Example:
Create a website certificate and sign it with the ${USER}_ca.crt key. We need to give it the -reqexts and -config options to configure the Subject Alternate Name (SAN) extension that modern browsers want to see.

```
$ openssl genrsa -out fwc.ias.edu.key 2048
Generating RSA private key, 2048 bit long modulus
.......................+++++
.............................................................+++++
e is 65537 (0x010001)

$ openssl req -key fwc.ias.edu.key -new -out fwc.ias.edu.csr -reqexts SAN -config openssl.cnf
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) [XX]:US
State or Province Name (full name) []:IL
Locality Name (eg, city) [Default City]:Chicago
Organization Name (eg, company) [Default Company Ltd]:Fun with Certificates
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:fwc.ias.edu
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

$ openssl ca -keyfile ${USER}_ca.key -cert ${USER}_ca.crt -in fwc.ias.edu.csr -out fwc.ias.edu.crt -
config openssl.cnf -create_serial

Using configuration from openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            fa:90:b3:65:74:54:4f:7e
        Validity
            Not Before: Oct 29 19:16:42 2018 GMT
            Not After : Oct 29 19:16:42 2019 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = IL
            organizationName          = Fun with Certificates
            commonName                = fwc.ias.edu
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                C0:54:7D:00:02:72:EA:7B:B5:47:07:5E:BD:DE:27:DA:B9:92:5D:1D
            X509v3 Authority Key Identifier:
                keyid:4C:9A:2E:55:12:B3:BE:AA:04:AA:7F:B4:5F:63:BF:CC:58:5B:3D:9A

Certificate is to be certified until Oct 29 19:16:42 2019 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
```

```
Data Base Updated
```

Example:
open an x509 certificate and list it's contents. If a file contains multiple certificates, only the first is shown

```
$ openssl x509 -in fwc.ias.edu.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            fa:90:b3:65:74:54:4f:7e
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = IL, L = Chicago, O = Fun with Certificates, CN = FWC User50 CA
        Validity
            Not Before: Oct 29 19:16:42 2018 GMT
            Not After : Oct 29 19:16:42 2019 GMT
        Subject: C = US, ST = IL, O = Fun with Certificates, CN = fwc.ias.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:98:b1:7d:7d:c6:7a:ca:b9:e9:9c:b2:58:d5:c9:
                    d7:d7:72:17:ce:19:0d:77:1c:76:93:a4:bc:72:59:
                    04:e2:dc:42:c4:ff:93:c1:f7:3d:e9:2b:99:d8:9f:
                    e5:94:f2:59:04:65:3b:3b:1f:f9:1b:fd:56:7c:7e:
                    68:31:e1:e4:cf:41:15:30:da:cd:2a:e1:c6:b7:f6:
                    8a:1c:85:83:85:46:1d:92:00:0f:87:f4:0e:1b:7f:
                    a4:2b:cd:b0:92:84:67:e5:14:33:b8:d4:b2:c6:94:
                    f1:ef:56:a8:27:1d:1d:9d:c3:90:50:4c:4d:44:88:
                    dc:bc:3d:66:1e:14:3b:f3:42:56:eb:4a:25:3b:92:
                    26:c0:95:9e:a9:58:1c:39:6e:c6:86:53:d4:a6:ba:
                    a3:3f:85:db:46:7d:fa:e1:8d:e7:a9:de:6f:f8:05:
                    34:69:d8:89:c2:89:5f:34:5b:56:ea:f4:16:c4:30:
                    dc:22:32:99:2f:96:bc:3e:31:65:1b:ec:f7:a6:f1:
                    26:73:73:94:89:a1:98:86:2b:2e:d4:f4:01:79:ce:
                    1f:c1:9b:84:6f:5b:47:72:95:23:91:e5:31:43:88:
                    de:51:62:9d:af:67:d3:a0:cf:28:8b:22:41:c7:be:
                    8f:f1:44:63:5a:c3:89:bc:73:87:fc:8c:8a:3f:ee:
                    83:6f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                C0:54:7D:00:02:72:EA:7B:B5:47:07:5E:BD:DE:27:DA:B9:92:5D:1D
            X509v3 Authority Key Identifier:
                keyid:4C:9A:2E:55:12:B3:BE:AA:04:AA:7F:B4:5F:63:BF:CC:58:5B:3D:9A

    Signature Algorithm: sha256WithRSAEncryption
         c5:4c:62:9d:9b:23:29:b0:fc:26:08:7f:88:3a:e1:8f:62:f1:
         7a:26:65:15:9e:fe:c4:99:2b:0a:e8:2b:98:53:23:ab:06:bd:
         89:97:bb:72:ad:de:a2:5b:0c:3f:01:ab:f7:3e:0d:5b:c2:86:
         71:c1:cb:dc:75:8a:c6:39:4c:77:fe:bc:f8:76:9d:03:52:bc:
         66:89:e4:69:82:1e:66:ea:d4:1d:02:f1:4d:be:33:3e:1e:cd:
         f4:fc:c8:82:32:ed:ea:1b:15:54:12:45:94:5b:2a:79:9c:96:
         bd:cf:3e:71:ad:5c:b0:4a:91:85:79:56:e4:ee:d6:6f:f0:3c:
         03:39:20:34:c2:10:e9:91:36:ff:68:3d:b2:4d:d8:25:ec:ae:
         77:a7:21:94:fe:1d:d0:9a:41:8e:23:fe:9e:59:01:19:87:04:
         e5:e9:52:09:02:70:c4:30:f6:6a:a0:5f:2a:8b:8e:57:26:73:
         84:dd:f6:f9:1e:a2:bb:76:1d:69:66:a5:85:d0:2f:84:e0:d7:
         c9:62:a5:6f:f7:60:bf:a1:32:01:dd:e0:08:a5:0d:6d:a5:fd:
         67:23:88:90:86:d8:f0:83:59:19:89:85:73:53:64:6c:cb:a8:
         22:53:00:f4:51:26:92:1b:8d:e9:74:90:a9:69:49:20:f8:af:
         66:07:bb:03
```

Example:
Use your new certificate to host a webpage.
Use the number 80 followed by your lab number for your **--port** parameter below. For this example, we are user50, so our port is 8050.
Now, use your browser to open https://fwc.ias.edu:**port** (use the same port as before)

```
$ openssl s_server -port 8050 -cert fwc.ias.edu.crt -key fwc.ias.edu.key -www

s_server -port 8050 -cert fwc.ias.edu.crt -key fwc.ias.edu.key -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1.2    :ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2    :ECDHE-RSA-AES256-GCM-SHA384
TLSv1.2    :ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2    :ECDHE-RSA-CHACHA20-POLY1305
TLSv1.2    :ECDHE-ECDSA-AES256-CCM8   TLSv1.2    :ECDHE-ECDSA-AES256-CCM
TLSv1.2    :ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2    :ECDHE-RSA-AES128-GCM-SHA256
TLSv1.2    :ECDHE-ECDSA-AES128-CCM8   TLSv1.2    :ECDHE-ECDSA-AES128-CCM
TLSv1.2    :ECDHE-ECDSA-AES256-SHA384 TLSv1.2    :ECDHE-RSA-AES256-SHA384
TLSv1.2    :ECDHE-ECDSA-CAMELLIA256-SHA384 TLSv1.2    :ECDHE-RSA-CAMELLIA256-SHA384
TLSv1.2    :ECDHE-ECDSA-AES128-SHA256 TLSv1.2    :ECDHE-RSA-AES128-SHA256
TLSv1.2    :ECDHE-ECDSA-CAMELLIA128-SHA256 TLSv1.2    :ECDHE-RSA-CAMELLIA128-SHA256
TLSv1.0    :ECDHE-ECDSA-AES256-SHA    TLSv1.0    :ECDHE-RSA-AES256-SHA
TLSv1.0    :ECDHE-ECDSA-AES128-SHA    TLSv1.0    :ECDHE-RSA-AES128-SHA
TLSv1.2    :AES256-GCM-SHA384         TLSv1.2    :AES256-CCM8
TLSv1.2    :AES256-CCM                TLSv1.2    :AES128-GCM-SHA256
TLSv1.2    :AES128-CCM8               TLSv1.2    :AES128-CCM
TLSv1.2    :AES256-SHA256             TLSv1.2    :CAMELLIA256-SHA256
TLSv1.2    :AES128-SHA256             TLSv1.2    :CAMELLIA128-SHA256
SSLv3      :AES256-SHA                SSLv3      :CAMELLIA256-SHA
SSLv3      :AES128-SHA                SSLv3      :CAMELLIA128-SHA
TLSv1.2    :DHE-RSA-AES256-GCM-SHA384 TLSv1.2    :DHE-RSA-CHACHA20-POLY1305
TLSv1.2    :DHE-RSA-AES256-CCM8       TLSv1.2    :DHE-RSA-AES256-CCM
TLSv1.2    :DHE-RSA-AES128-GCM-SHA256 TLSv1.2    :DHE-RSA-AES128-CCM8
TLSv1.2    :DHE-RSA-AES128-CCM        TLSv1.2    :DHE-RSA-AES256-SHA256
TLSv1.2    :DHE-RSA-CAMELLIA256-SHA256 TLSv1.2    :DHE-RSA-AES128-SHA256
TLSv1.2    :DHE-RSA-CAMELLIA128-SHA256 SSLv3      :DHE-RSA-AES256-SHA
SSLv3      :DHE-RSA-CAMELLIA256-SHA   SSLv3      :DHE-RSA-AES128-SHA
SSLv3      :DHE-RSA-CAMELLIA128-SHA
---
Ciphers common between both SSL end points:
ECDHE-ECDSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-RSA-AES256-GCM-SHA384 ECDHE-ECDSA-CHACHA20-POLY1305 ECDHE-RSA-CHACHA20-POLY1305
ECDHE-RSA-AES128-SHA       ECDHE-RSA-AES256-SHA       AES128-GCM-SHA256
AES256-GCM-SHA384          AES128-SHA                 AES256-SHA
DES-CBC3-SHA
Signature Algorithms:
ECDSA+SHA256:0x04+0x08:RSA+SHA256:ECDSA+SHA384:0x05+0x08:RSA+SHA384:0x06+0x08:RSA+SHA512:RSA+SHA1
Shared Signature Algorithms: ECDSA+SHA256:RSA+SHA256:ECDSA+SHA384:RSA+SHA384:RSA+SHA512:RSA+SHA1
Supported Elliptic Curves: 0xFAFA:X25519:P-256:P-384
Shared Elliptic curves: X25519:P-256:P-384
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES128-GCM-SHA256
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID:
    Session-ID-ctx: 01000000
    Master-Key:
DA89A3ECE49C814FA7A5BECF555A427D6042AD0A7E5B1155C090958A1DA5C292186D3D99F352A639E73C4227ED7ECA39
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1557634275
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: yes
```

```
---
   0 items in the session cache
   0 client connects (SSL_connect())
   0 client renegotiates (SSL_connect())
   0 client connects that finished
   1 server accepts (SSL_accept())
   0 server renegotiates (SSL_accept())
   1 server accepts that finished
   0 session cache hits
   1 session cache misses
   0 session cache timeouts
   0 callback cache hits
   0 cache full overflows (128 allowed)
---
no client certificate available
```

You should receive a warning about connecting to a non-secure website. This is because we haven't installed our CA into our browser. We'll do that step next.

Example:

Import your CA into your browser and try browsing to https://fwc.ias.edu:**port** again. Do you notice the difference?

Start by copy and pasting your CA into a text file on your computer.

```
$ cat user50_ca.crt
-----BEGIN CERTIFICATE-----
MIIDnjCCAoagAwIBAgIJAJ0/m05WW+E/MA0GCSqGSIb3DQEBCwUAMGQxCzAJBgNV
BAYTAlVTMQswCQYDVQQIDAJJTDEQMA4GA1UEBwwHQ2hpY2FnbzEeMBwGA1UECgwV
RnVuIHdpdGggQ2VydGlmaWNhdGVzMRYwFAYDVQQDDA1GV0MgVXNlcjUwIENBMB4X
DTE5MDUxMjAzMTQ1OFoXDTE5MDYxMTAzMTQ1OFowZDELMAkGA1UEBhMCVVMxCzAJ
BgNVBAgMAklMMRAwDgYDVQQHDAdDaGljYWdvMR4wHAYDVQQKDBVGdW4gd2l0aCBD
ZXJ0aWZpY2F0ZXMxFjAUBgNVBAMMDUZXQyBVc2VyNTAgQ0EwggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQDEpUsjr02EGigL7gAfqvaPJOKHjvIUMQn2sTzD
NkYKZMoSxC9BcJ0ee+1K7Txjk62kxn/zy95Y5lcxmk4kW1AsPKtqst+Oj7q8q3vD
tMobLM7khvm97pTLJZpZewxrj1XRCi66MJt3BKhDAgoyPsJXbXddD0Rwi5s67WnQ
64zqXHVsoOMuono9W7I+9RL5nNg7nU2NAZ3RWpV45R7RX+2oJtCk7xMDj7z/xGiW
z5UXIcVpx9tHfTH0ErVVQhGvyybkj9JBf1+1eKSQksmeXzEtBwlrsdYNYYgK9Kj+
mrfEriuktwtxYmv41UlYcJVSv8rNTm3smhtKmwuhfk5eL4JnAgMBAAGjUzBRMB0G
A1UdDgQWBBQQpG5X3+RpTQkbw+FHfkVw7TAZkTAfBgNVHSMEGDAWgBQQpG5X3+Rp
TQkbw+FHfkVw7TAZkTAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IB
AQCSRuzNzwb4jibe/fadabA/6IMZcRe/wFOEdzxoQbXPRXpt9tE1kLcvprg2yMhX
nLaBO7HJ9O91bdoCuNWu2UCM6kuEIN07cEU+6ko0R4NFr6o+w/S/vLWDPvXp/yYu
djWQIC3je3Sf29RREpAlqCqwvLfKrRxzdvI2k3lUOl2y6gmwrUk7UyYsN+dpyNUD
kMJ+XsaYswOhShOWq3Ojv4Vs9nNdOMFDwkgkfL0EYsz6R2Dyz4Rm11b+0kaVWFFE
ZEiZ1jpj+QQs3C4i0oBSF0TRmGUu+whMJxiQZeGdlsyKePnVuDIV61oYL6j+cllf
2EKcNvRBnqLI3lNILfrVw5PG
-----END CERTIFICATE-----
```

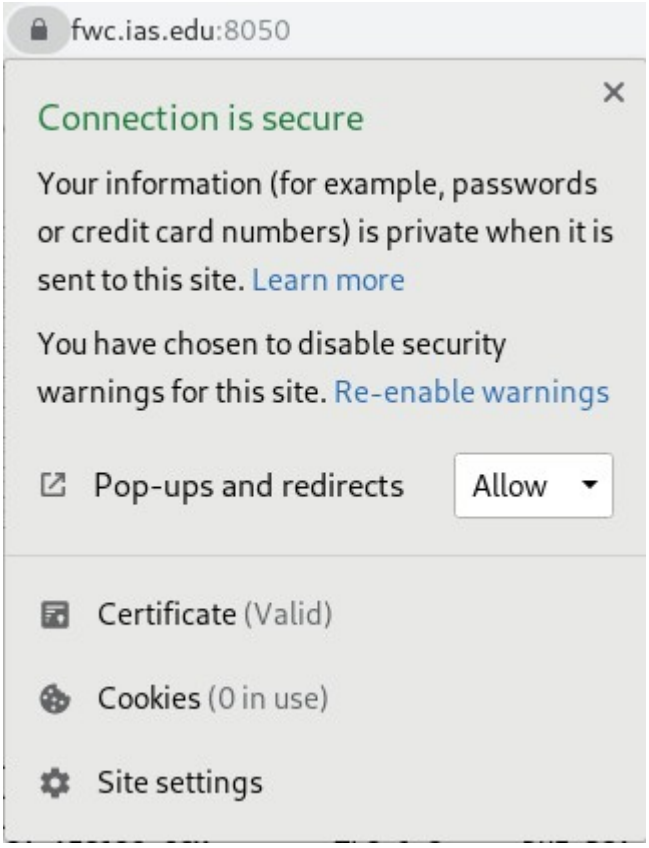
Once you have that in a file on your local computer, open your browser's preferences.

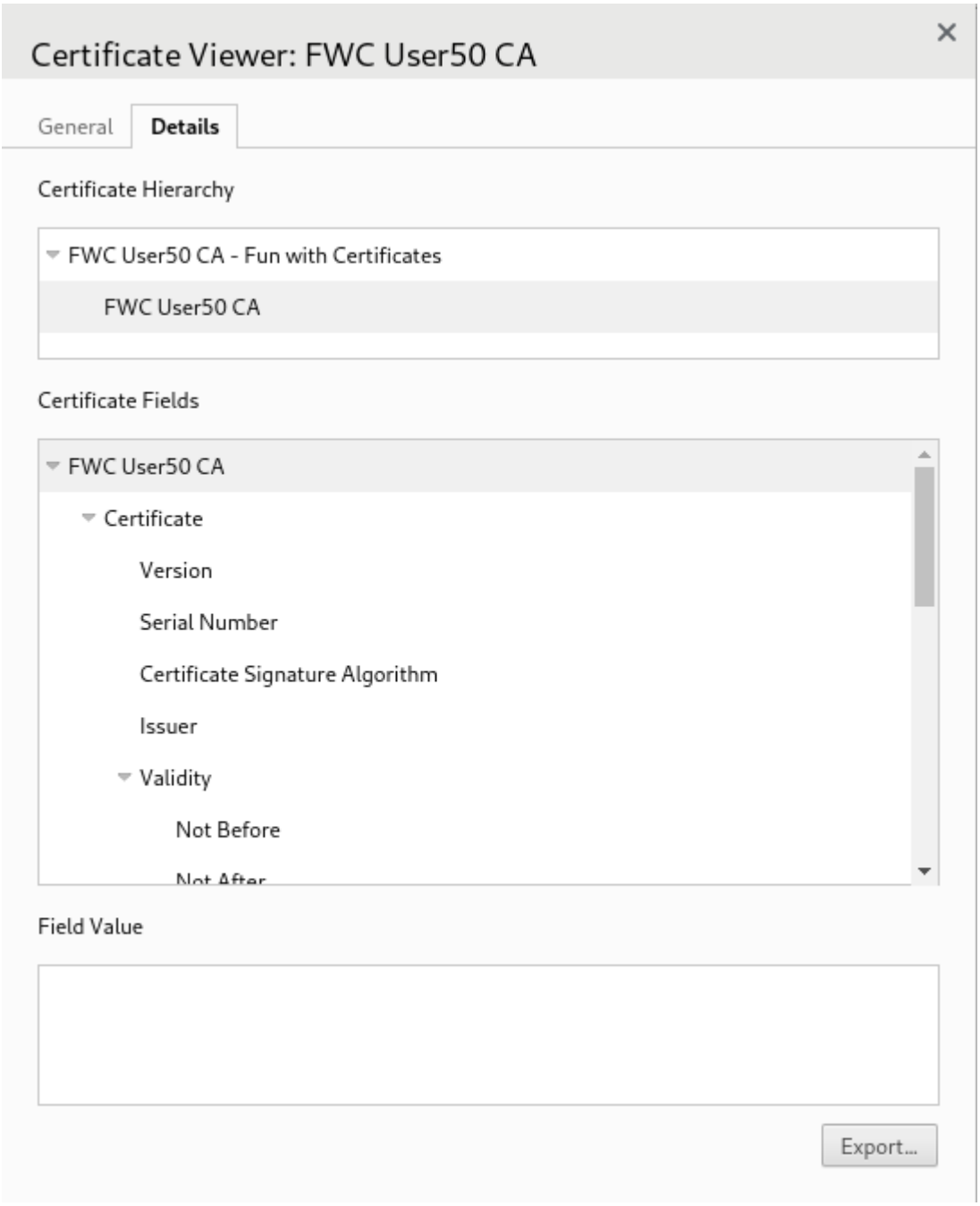
For Firefox, type in **about:preferences** into your location bar. From there, search for **certificates**. Then, click on **View Certificates**. Open the **Authorities** tab and click on the **Import** button and import your file that you created on the local computer. Trust the certificate for identifying websites. You should now see your site under **Fun with Certificates**.

For Chrome, type in **chrome://settings/** and search for **certificates**. Then, click on **Manage certificates**. Open the **Authorities** tab and click on the **Import** button and import your file that you created on the local computer. Trust the certificate for identifying websites. You should now see your site under **org-Fun with Certificates**.

Now browse to https://fwc.ias.edu:**port**

Click on the lock icon in your location bar and look at your certificate details. You should see that the certificate is valid and that the chain is your CA and your certificate that you just created. (Chrome example below)


fwc.ias.edu:8050
Connection is secure
Your information (for example, passwords or credit card numbers) is private when it is sent to this site. Learn more
You have chosen to disable security warnings for this site. Re-enable warnings
Pop-ups and redirects
Allow
Certificate (Valid)
Cookies (0 in use)
Site settings


Certificate Viewer: FWC User50 CA
General
Details
Certificate Hierarchy
FWC User50 CA - Fun with Certificates
FWC User50 CA
Certificate Fields
FWC User50 CA
Certificate
Version
Serial Number
Certificate Signature Algorithm
Issuer
Validity
Not Before
Not After
Field Value
Export...

Example:
Create an EC key and self signed EC CA Certificate in one step using prime256v1 curve.

```
$ openssl req -newkey ec -pkeyopt ec_paramgen_curve:prime256v1 -x509 -keyout ${USER}_ec_ca.key -out ${USER}_ec_ca.crt -nodes
Generating an EC private key
writing new private key to 'user50_ec_ca.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) [XX]:US
State or Province Name (full name) []:IL
Locality Name (eg, city) [Default City]:Chicago
Organization Name (eg, company) [Default Company Ltd]:Fun with Certificates
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:FWC User50 EC CA
Email Address []:
```

Example:

Create an EC key and Certificate Signing Request in one step using prime256v1 curve. Notice the only difference between this and the last request is the -x509 parameter which tells openssl to make a self signed cert. We want to sign this cert with the CA we created in the last section to make the x509 certificate.

```
$ openssl req -newkey ec -pkeyopt ec_paramgen_curve:prime256v1 -keyout fwc.ias.edu_ec.key -out
fwc.ias.edu_ec.csr -nodes -reqexts SAN -config openssl.cnf

Generating an EC private key
writing new private key to 'fwc.ias.edu_ec.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) [XX]:US
State or Province Name (full name) []:IL
Locality Name (eg, city) [Default City]:Chicago
Organization Name (eg, company) [Default Company Ltd]:Fun with Certificates
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:fwc.ias.edu
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Example:

Now we want to sign our new EC CSR with our new EC CA that we created.

```
$ openssl ca -keyfile ${USER}_ec_ca.key -cert ${USER}_ec_ca.crt -in fwc.ias.edu_ec.csr -out
fwc.ias.edu_ec.crt -config openssl.cnf -create_serial
Using configuration from openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            a6:03:88:b0:7c:86:72:40
        Validity
            Not Before: May 12 13:10:45 2019 GMT
            Not After : May 11 13:10:45 2020 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = IL
            localityName              = Chicago
            organizationName          = Fun with Certificates
            commonName                = fwc.ias.edu
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                21:A0:09:4E:4F:1A:E6:7E:98:B2:2B:22:90:35:AE:9E:09:E7:BA:4C
            X509v3 Authority Key Identifier:
                keyid:34:FA:A1:83:80:F0:7B:C5:05:18:24:45:FF:03:FC:5C:7E:37:21:AB

            X509v3 Subject Alternative Name:
                DNS:fwc.ias.edu
Certificate is to be certified until May 11 13:10:45 2020 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Example:
Take a look at the new certificate you just created.

```
$ openssl x509 -text -noout -in fwc.ias.edu_ec.crt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            a6:03:88:b0:7c:86:72:40
    Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = US, ST = IL, L = Chicago, O = Fun with Certificates, CN = FWC User50 EC CA
        Validity
            Not Before: May 12 13:10:45 2019 GMT
            Not After : May 11 13:10:45 2020 GMT
        Subject: C = US, ST = IL, L = Chicago, O = Fun with Certificates, CN = fwc.ias.edu
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:a2:52:c4:84:0b:b1:11:3c:e2:4c:ef:38:b2:56:
                    7c:b1:97:4f:a3:52:b6:75:ce:ca:b6:29:ae:4b:97:
                    fe:c5:d0:07:51:14:80:91:f3:dc:08:55:fb:0f:2f:
                    7a:30:6b:84:0f:9d:1d:83:01:1d:44:b5:b0:e8:6b:
                    9f:06:db:0c:44
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                21:A0:09:4E:4F:1A:E6:7E:98:B2:2B:22:90:35:AE:9E:09:E7:BA:4C
            X509v3 Authority Key Identifier:
                keyid:34:FA:A1:83:80:F0:7B:C5:05:18:24:45:FF:03:FC:5C:7E:37:21:AB

            X509v3 Subject Alternative Name:
                DNS:fwc.ias.edu
    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:20:7f:4c:0c:8f:b5:e7:b1:34:28:8b:89:a3:d2:67:
         8f:c3:cd:05:ed:79:95:0c:f2:54:7f:1e:46:d1:e1:87:f6:e0:
         02:21:00:ee:33:76:ed:ae:22:eb:6a:0b:91:e5:f6:e9:ec:ad:
         1e:15:40:68:f4:68:6e:de:7f:50:79:a7:15:16:c0:a9:e8
```

Example:

Using the same technique as with the RSA cert, see if you can create an openssl webserver using your new EC cert. Don't forget to test before and after installing the CA into your web browser. Do you notice any differences?

Example:
        generate and display a 512bit RSA key. This is an example of using both the
        "genrsa" and "rsa" commands together with a pipe on the command line.

```
$ openssl genrsa 512 | openssl rsa -text -noout
Generating RSA private key, 512 bit long modulus (2 primes)
....++++++++++++++++++++++++++++
.....................++++++++++++++++++++++++++++
e is 65537 (0x010001)
RSA Private-Key: (512 bit, 2 primes)
modulus:
    00:df:b1:33:f2:5e:04:25:7e:95:85:28:58:88:a2:
    92:22:79:9d:ac:be:83:29:2f:83:d4:d8:38:33:7d:
    7d:d7:5b:af:88:a6:1a:37:37:bb:70:a4:06:52:7f:
    6d:3f:77:90:be:6c:41:33:cb:12:de:31:a6:b9:3c:
    a1:97:9b:82:5b
publicExponent: 65537 (0x10001)
privateExponent:
    20:95:25:25:e6:7f:e0:1e:42:15:35:3d:40:19:be:
    03:7b:87:76:04:02:02:db:c2:ee:1b:d8:83:d7:81:
    9e:fe:9c:04:fb:08:0c:60:94:00:f7:42:90:f4:15:
    fd:d1:3d:26:65:7e:43:c6:53:60:35:1b:66:76:30:
    50:de:9e:d1
prime1:
    00:ff:44:d8:ac:a5:bc:a4:0f:a6:0f:4c:10:72:59:
    b7:25:f2:db:b6:ae:5f:7f:1b:30:b0:7c:13:7e:33:
    95:5a:87
prime2:
    00:e0:55:34:a5:d8:e7:e8:15:c3:08:57:f0:f7:4e:
    77:c4:f7:27:13:af:82:04:f8:92:7b:f6:6d:57:55:
    ad:aa:8d
exponent1:
    3a:e3:98:14:e4:3b:4b:a1:ec:8e:54:de:ea:72:76:
    05:04:2c:d9:cd:0a:6c:d9:49:f7:3c:f8:86:41:c2:
    2f:db
exponent2:
    00:df:f5:b0:bd:f2:32:74:0d:65:cc:aa:9f:33:06:
    ef:a3:80:6b:4e:c7:e5:32:39:47:64:e2:bc:7e:78:
    57:0c:f1
coefficient:
    00:f0:52:a4:ca:a2:03:30:69:fe:2a:8a:0d:e7:4d:
    ef:fa:47:1f:eb:d7:31:46:7e:43:f0:89:fa:57:00:
    4d:b3:82
```

Example:
multiple the above prime numbers together to verify the modulus. Notice the order of the "obase" and "ibase". If you switch them, you are setting your output base to 0x16, or 22 in decimal. This also should be on one line.

```
$ echo
"00:ff:44:d8:ac:a5:bc:a4:0f:a6:0f:4c:10:72:59:b7:25:f2:db:b6:ae:5f:7f:1b:30:b0:7c:13:7e:33:95:5a:87 *
00:e0:55:34:a5:d8:e7:e8:15:c3:08:57:f0:f7:4e:77:c4:f7:27:13:af:82:04:f8:92:7b:f6:6d:57:55:ad:aa:8d" | \
sed 's/://g' | \
tr 'a-f' 'A-F' | \
sed 's/^/obase=16;ibase=16;/' | \
bc | \
tr 'A-F' 'a-f' | \
sed 's/\(..\)/\1:/g'
df:b1:33:f2:5e:04:25:7e:95:85:28:58:88:a2:92:22:79:9d:ac:be:83:29:2f:83:d4:d8:38:33:7d:7d:d7:5b:af:88:a
6:1a:37:37:bb:70:a4:06:52:7f:6d:3f:77:90:be:6c:41:33:cb:12:de:31:a6:b9:3c:a1:97:9b:82:5b:
```

Example:

connect to a remote host and look at the certificates it has to offer. The "showcerts" option will display all the certs offered by the server. These can be examined with the x509 command. Also notice, once connected, we can interact with this host as if we just used telnet to port 80. This is extremely useful for troubleshooting what certificates a host offers, and how to interact over an encrypted SSL tunnel.

```
$ openssl s_client -connect security.ias.edu:443 -showcerts
CONNECTED(00000003)
depth=2 C = US, ST = New Jersey, L = Jersey City, O = The USERTRUST Network, CN = USERTrust RSA
Certification Authority
verify return:1
depth=1 C = US, ST = MI, L = Ann Arbor, O = Internet2, OU = InCommon, CN = InCommon RSA Server CA
verify return:1
depth=0 C = US, postalCode = 08540, ST = New Jersey, L = Princeton, street = 1 Einstein Drive, O =
Institute for Advanced Study, CN = www.ias.edu
verify return:1
---
Certificate chain
 0 s:/C=US/postalCode=08540/ST=New Jersey/L=Princeton/street=1 Einstein Drive/O=Institute for Advanced
Study/CN=www.ias.edu
   i:/C=US/ST=MI/L=Ann Arbor/O=Internet2/OU=InCommon/CN=InCommon RSA Server CA
-----BEGIN CERTIFICATE-----
MIIGnDCCBYSgAwIBAgIQKF1wLbhTHrcuwzkeNd5X+jANBgkqhkiG9w0BAQsFADB2
MQswCQYDVQQGEwJVUzELMAkGA1UECBMCTUkxEjAQBgNVBAcTCUFubiBBcmJvcjES
MBAGA1UEChMJSW50ZXJuZXQyMREwDwYDVQQLEwhJbkNvbW1vbjEfMB0GA1UEAxMW
SW5Db21tb24gUlNBIFNlcnZlciBDQTAeFw0xNzEyMjIwMDAwMDBaFw0xODEyMjIy
MzU5NTlaMIGeMQswCQYDVQQGEwJVUzEOMAwGA1UEERMFMDg1NDAxEzARBgNVBAgT
Ck5ldyBKZXJzZXkxEjAQBgNVBAcTCVByaW5jZXRvbjEZMBcGA1UECRMQMSBFaW5z
dGVpbiBEcml2ZTElMCMGA1UEChMcSW5zdGl0dXRlIGZvciBBZHZhbmNlZCBTdHVk
eTEUMBIGA1UEAxMLd3d3Lmlhcy5lZHUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
ggEKAoIBAQDC206rKE133BvZ5TDUtLSVmSlr2oCb6lREHM8Nta/V3VaIl7LoLmms
29xLHdSsEtgroTB7N/YyyrpUaVF0m4q+jzqT0TKa5aVB4HlFsoenPiQVl2sD+BlY
Xo1Gs2xwdHxtm8l689UeARf4OmLMOrFlr+1Lurt3o2LhBd2BsXsRYBqUnaHcH0R3
cH9X6icV1vZMWcuiEgHfLZiTzwjEzAUtbBDrrLNlvcqgSSUEveLzh5d1Uus6d+9z
MIDCn8uI8Rg3rN/n79KGSFIo7KwaSYTyc7G3BZuYrYOv4l3tulamwUqktoy6CPLE
QMZcNCZP0NTc9ThnxPM5+ZeZhfofpGfpAgMBAAGjggL7MIIC9zAfBgNVHSMEGDAW
gBQeBaN3j2yW4luHS6a0hqxxAAznODAdBgNVHQ4EFgQUMA4Z7FMCUSc1rQoRQkVP
a1i6IgEwDgYDVR0PAQH/BAQDAgWgMAwGA1UdEwEB/wQCMAAwHQYDVR0lBBYwFAYI
KwYBBQUHAwEGCCsGAQUFBwMCMGcGA1UdIARgMF4wUgYMKwYBBAGuIwEEAwEBMEIw
QAYIKwYBBQUHAgEWNGh0dHBzOi8vd3d3LmluY29tbW9uLm9yZy9jZXJ0L3JlcG9z
aXRvcnkvY3BzX3NzbC5wZGYwCAYGZ4EMAQICMEQGA1UdHwQ9MDswOaA3oDWGM2h0
dHA6Ly9jcmwuaW5jb21tb24tcnNhLm9yZy9JbkNvbW1vblJTQVNlcnZlckNBLmNy
bDB1BggrBgEFBQcBAQRpMGcwPgYIKwYBBQUHMAKGMmh0dHA6Ly9jcnQudXNlcnRy
dXN0LmNvbS9JbkNvbW1vblJTQVNlcnZlckNBXzIuY3J0MCUGCCsGAQUFBzABhhlo
dHRwOi8vb2NzcC51c2VydHJ1c3QuY29tMIIBUAYDVR0RBIIBRzCCAUOCC3d3dy5p
YXMuZWR1ghJjcm9zc3JvYWRzLmlhcy5lZHWCDmRpbmluZy5pYXMuZWR1ghZmYWN1
bHR5aG91c2luZy5pYXMuZWR1ghZnZW5ldGljaGlzdG9yeS5pYXMuZWR1gg9saWJy
YXJ5Lmlhcy5lZHWCDm1vcmFscy5pYXMuZWR1ggxwY21pLmlhcy5lZHWCDHBpdHAu
aWFzLmVkdYIQc2VjdXJpdHkuaWFzLmVkdYILc2lnLmlhcy5lZHWCDXZpZGVvLmlh
cy5lZHWCDnd3dy5ocy5pYXMuZWR1gg93d3cuaWRzLmlhcy5lZHWCD3d3dy5pdGcu
aWFzLmVkdYIQd3d3Lm1hdGguaWFzLmVkdYIPd3d3Lm5ldC5pYXMuZWR1gg93d3cu
c25zLmlhcy5lZHWCD3d3dy5zc3MuaWFzLmVkdTANBgkqhkiG9w0BAQsFAAOCAQEA
N5v8zHwEQ4uUssbYjFoTba5KJUApu2xlPK/qgh2svKOmai5FTJ/qI3CCUFdqLM4+
y6P5MIaKb48yqTSOcNvy5Upn7eio/zTTo0+XHTPUe1PnMODAnry5qby7LcPvCxCK
mtrQPM90uk0VLSVLdUMX6kFyrkiD+Y5MttANO6N8l4XVNLBZAbTJ9o+/73mNUHW2
8XZ+S7pUPV0NXrzQBQAtdFouVJrT5cG7pGfb99H4jLT6KN12LoFT1a3qgoRFQtKp
M5YlsVYKZQIz8DFGDsatdxUhce40ncHE66Em51OefCxEkdk28QHRKVOmHRONE24L
LX1q3fTRpOqxLFxg89/eSA==
-----END CERTIFICATE-----
 1 s:/C=US/ST=MI/L=Ann Arbor/O=Internet2/OU=InCommon/CN=InCommon RSA Server CA
   i:/C=US/ST=New Jersey/L=Jersey City/O=The USERTRUST Network/CN=USERTrust RSA Certification Authority
-----BEGIN CERTIFICATE-----
MIIF+TCCA+GgAwIBAgIQRyDQ+oVGGn4XoWQCkYRjdDANBgkqhkiG9w0BAQwFADCB
```

```
iDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCk5ldyBKZXJzZXkxFDASBgNVBAcTC0pl
cnNleSBDaXR5MR4wHAYDVQQKExVUaGUgVVNFUlRSVVNUIE5ldHdvcmsxLjAsBgNV
BAMTJVVTRVJUcnVzdCBSU0EgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkwHhcNMTQx
MDA2MDAwMDAwWhcNMjQxMDA1MjM1OTU5WjB2MQswCQYDVQQGEwJVUzELMAkGA1UE
CBMCTUkxEjAQBgNVBAcTCUFubiBBcmJvcjESMBAGA1UEChMJSW50ZXJuZXQyMREw
DwYDVQQLEwhJbkNvbW1vbjEfMB0GA1UEAxMWSW5Db21tb24gUlNBIFNlcnZlciBD
QTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJwb8bsvf2MYFVFRVA+e
xU5NEFj6MJsXKZDmMwysE1N8VJG06thum4ltuzM+j9INpun5uukNDBqeso7JcC7v
HgV9lestjaKpTbOc5/MZNrun8XzmCB5hJ0R6lvSoNNviQsil2zfVtefkQnI/tBPP
iwckRR6MkYNGuQmm/BijBgLsNI0yZpUn6uGX6Ns1oytW61fo8BBZ321wDGZq0GTl
qKOYMa0dYtX6kuOaQ80tNfvZnjNbRX3EhigsZhLI2w8ZMA0/6fDqSl5AB8f2IHpT
eIFken5FahZv9JNYyWL7KSd9oX8hzudPR9aKVuDjZvjs3YncJowZaDuNi+L7RyML
fzcCAwEAAaOCAW4wggFqMB8GA1UdIwQYMBaAFFN5v1qqK0rPVIDh2JvAnfKyA2bL
MB0GA1UdDgQWBBQeBaN3j2yW4luHS6a0hqxxAAznODAOBgNVHQ8BAf8EBAMCAYYw
EgYDVR0TAQH/BAgwBgEB/wIBADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUH
AwIwGwYDVR0gBBQwEjAGBgRVHSAAMAgGBmeBDAECAjBQBgNVHR8ESTBHMEWgQ6BB
hj9odHRwOi8vY3JsLnVzZXJ0cnVzdC5jb20vVVNFUlRydXN0UlNBQ2VydGlmaWNh
dGlvbkF1dGhvcml0eS5jcmwwdgYIKwYBBQUHAQEEajBoMD8GCCsGAQUFBzAChjNo
dHRwOi8vY3J0LnVzZXJ0cnVzdC5jb20vVVNFUlRydXN0UlNBQWRkVHJ1c3RDQS5j
cnQwJQYIKwYBBQUHMAGGGWh0dHA6Ly9vY3NwLnVzZXJ0cnVzdC5jb20wDQYJKoZI
hvcNAQEMBQADggIBAC0RBjjW29dYaK+qOGcXjeIT16MUJNkGE+vrkS/fT2ctyNMU
11ZlUp5uH5gIjppIG8GLWZqjV5vbhvhZQPwZsHURKsISNrqOcooGTie3jVgU0W+0
+Wj8mN2knCVANt69F2YrA394gbGAdJ5fOrQmL2pIhDY0jqco74fzYefbZ/VS29fR
5jBxu4uj1P+5ZImem4Gbj1e4ZEzVBhmO55GFfBjRidj26h1oFBHZ7heDH1Bjzw72
hipu47Gkyfr2NEx3KoCGMLCj3Btx7ASn5Ji8FoU+hCazwOU1VX55mKPU1I2250Lo
RCASN18JyfsD5PVldJbtyrmz9gn/TKbRXTr80U2q5JhyvjhLf4lOJo/UzL5WCXED
Smyj4jWG3R7Z8TED9xNNCxGBMXnMete+3Pvzdhssvb0RDwBZByogQ9xL2LUZFI/i
eoQp0UM/L8zfP527vWjEzuDN5xwxMnhi+vCToh7J159o5ah29mP+aJnvujbXEnGa
nrNxHzu+AGOePV8hwrGGG7hOIcPDQwkuYwzN/xT29iLp/cqf9ZhEtkGcQcIImH3b
oJ8ifsCnSbu0GB9L06Yqh7lcyvKDTEADslIaeSEINxhO2Y1fmcYFX/Fqrrp1WnhH
OjplXuXE0OPa0utaKC25Aplgom88L2Z8mEWcyfoB7zKOfD759AN7JKZWCYwk
-----END CERTIFICATE-----
 2 s:/C=US/ST=New Jersey/L=Jersey City/O=The USERTRUST Network/CN=USERTrust RSA Certification Authority
   i:/C=SE/O=AddTrust AB/OU=AddTrust External TTP Network/CN=AddTrust External CA Root
-----BEGIN CERTIFICATE-----
MIIFdzCCBF+gAwIBAgIQE+oocFv07O0MNmMJgGFDNjANBgkqhkiG9w0BAQwFADBv
MQswCQYDVQQGEwJTRTEUMBIGA1UEChMLQWRkVHJ1c3QgQUIxJjAkBgNVBAsTHUFk
ZFRydXN0IEV4dGVybmFsIFRUUCBOZXR3b3JrMSIwIAYDVQQDExlBZGRUcnVzdCBF
eHRlcm5hbCBDQSBSb290MB4XDTAwMDUzMDEwNDgzOFoXDTIwMDUzMDEwNDgzOFow
gYgxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpOZXcgSmVyc2V5MRQwEgYDVQQHEwtK
ZXJzZXkgQ2l0eTEeMBwGA1UEChMVVGhlIFVTRVJUUlVTVCBOZXR3b3JrMS4wLAYD
VQQDEyVVU0VSVHJ1c3QgUlNBIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MIICIjAN
BgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAgBJlFzYOw9sIs9CsVw127c0n00yt
UINh4qogTQktZAnczomfzD2p7PbPwdzx07HWezcoEStH2jnGvDoZtF+mvX2do2NC
tnbyqTsrkfjib9DsFiCQCT7i6HTJGLSR1GJk23+jBvGIGGqQIjy8/hPwhxR79uQf
jtTkUcYRZ0YIUcuGFFQ/vDP+fmyc/xadGL1RjjWmp2bIcmfbIWax1Jt4A8BQOujM
8Ny8nkz+rwWWNR9XWrf/zvk9tyy29lTdyOcSOk2uTIq3XJq0tyA9yn8iNK5+O2hm
AUTnAU5GU5szYPeUvlM3kHND8zLDU+/bqv50TmnHa4xgk97Exwzf4TKuzJM7UXiV
Z4vuPVb+DNBpDxsP8yUmazNt925H+nND5X4OpWaxKXwyhGNVicQNwZNUMBkTrNN9
N6frXTpsNVzbQdcS2qlJC9/YgIoJk2KOtWbPJYjNhLixP6Q5D9kCnusSTJV882sF
qV4Wg8y4Z+LoE53MW4LTTLPtW//e5XOsIzstAL81VXQJSdhJWBp/kjbmUZIO8yZ9
HE0XvMnsQybQv0FfQKlERPSZ51eHnlAfV1SoPv10Yy+xUGUJ5lhCLkMaTLTwJUdZ
+gQek9QmRkpQgbLevni3/GcV4clXhB4PY9bpYrrWX1Uu6lzGKAgEJTm4Diup8kyX
HAc/DVL17e8vgg8CAwEAAaOB9DCB8TAfBgNVHSMEGDAWgBStvZh6NLQm9/rEJlTv
A73gJMtUGjAdBgNVHQ4EFgQUU3m/WqorSs9UgOHYm8Cd8rIDZsswDgYDVR0PAQH/
BAQDAgGGMA8GA1UdEwEB/wQFMAMBAf8wEQYDVR0gBAowCDAGBgRVHSAAMEQGA1Ud
HwQ9MDswOaA3oDWGM2h0dHA6Ly9jcmwudXNlcnRydXN0LmNvbS9BZGRUcnVzdEV4
dGVybmFsQ0FSb290LmNybDA1BggrBgEFBQcBAQQpMCcwJQYIKwYBBQUHMAGGGWh0
dHA6Ly9vY3NwLnVzZXJ0cnVzdC5jb20wDQYJKoZIhvcNAQEMBQADggEBAJNl9jeD
lQ9ew4IcH9Z35zyKwKoJ8OkLJvHgwmp1ocd5yblSYMgpEg7wrQPWCcR23+WmgZWn
RtqCV6mVksW2jwMibDN3wXsyF24HzloUQToFJBv2FAY7qCUkDrvMKnXduXBBP3zQ
YzYhBx9G/2CkkeFnvN4ffhkUyWNnkepnB2u0j4vAbkN9w6GAbLIevFOFfdyQoaS8
Le9Gclc1Bb+7RrtubTeZtv8jkpHGbkD4jylW6l/VXxRTrPBPYer3IsynVgviuDQf
Jtl7GQVoP7o81DgGotPmjw7jtHFtQELFhLRAlSv0ZaBIefYdgWOWnU914Ph85I6p
0fKtirOMxyHNwu8=
```

```
-----END CERTIFICATE-----
---
Server certificate
subject=/C=US/postalCode=08540/ST=New Jersey/L=Princeton/street=1 Einstein Drive/O=Institute for
Advanced Study/CN=www.ias.edu
issuer=/C=US/ST=MI/L=Ann Arbor/O=Internet2/OU=InCommon/CN=InCommon RSA Server CA
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 5145 bytes and written 380 bytes
Verification: OK
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID: C11FD3CF1391A650E920226192E5CDAC66E7BFE480680C4472363FCD951D37E2
    Session-ID-ctx:
    Master-Key:
C79EF1A7D02ED2CC4D2E89F948159D08AF873F9D0BD5FCFD8A601CE3A9EFE9D9703111646503FDFDBFE7E9038F729A91
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1540846225
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: no
---
```

Example:
Calculate the SHA256 sum on a host that only has openssl and not the sha256sum tool.

```
$ sha256sum /etc/hosts
bash: sha256sum: command not found...
$ openssl dgst -sha256 /etc/hosts
SHA256(/etc/hosts)= 42c60aee9ac2254ea721673592386164914480669c06c2fad31123344fe71a7f
```

Example:
Quickly encrypt a config file to send over email to a vendor for troubleshooting purposes. I do this all the time, it isn't too difficult to explain over the phone how to decrypt, and it gives you the option of protecting sensitive data over an insecure medium. You still have to tell them the password over the phone, though, which is better than sending cleartext.

```
$ cat << EOF >> database_credentials.php
<?php

$dbuser = "dummy";
$dbpass = "drowssap";
$dbhost = "hopeyoudonthackme.com";
$dbname = "please";

?>
EOF

$ openssl aes-256-cbc -e -in database_credentials.php -out database_credentials.php.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:

$ file database_credentials.php*
database_credentials.php:     PHP script, ASCII text
database_credentials.php.enc: openssl enc'd data with salted password

$ openssl aes-256-cbc -d -in database_credentials.php.enc
<?php

$dbuser = "dummy";
$dbpass = "drowssap";
$dbhost = "hopeyoudonthackme.com";
$dbname = "please";

?>

$ xxd database_credentials.php.enc
0000000: 5361 6c74 6564 5f5f 70d2 c0dc c413 bfbf  Salted__p.......
0000010: a7e0 124e 6477 42e7 b553 17ff ee6c edb4  ...NdwB..S...l..
0000020: ab5b 15b2 ab0a 455c d2ef 0cb2 e87a 8350  .[....E\.....z.P
0000030: 2fe4 9b6a 8910 5e8a 2b56 56ce 8f5c c727  /..j..^.+VV..\.'
0000040: ebae 66b2 1218 f4fc 2c18 e375 f45d 4915  ..f.....,..u.]I.
0000050: a756 1d1d bdb2 a4ab 0b1f 844b 4fe9 7752  .V.........KO.wR
0000060: bdba c4a7 27f1 f965 0b19 3370 74cc beb3  ....'..e..3pt...
0000070: 25ad 5c94 bbb9 8581 b36e ffd1 6301 2b59  %.\......n..c.+Y

$ base64 < database_credentials.php.enc | \
mail -s "Database credentials you asked for" support@example.com
```

Example:
Inspect an ssh RSA key for it's components

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/${USER}/.ssh/id_rsa): /home/${USER}/.ssh/fwc
...
Your identification has been saved in /home/${USER}/.ssh/fwc.
Your public key has been saved in /home/${USER}/.ssh/fwc.pub.
...

$ openssl rsa -in .ssh/fwc -text -noout
Private-Key: (2048 bit)
modulus:
    00:e7:4e:c9:dc:0e:6a:22:f0:ca:48:c7:ea:6b:a9:
...
```