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
.00000000000000000004
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
134339344986286640331347423
obase=2
6F1F767BF5E14A4DE9D5DF
110111100011111011101100111101111110101111000010100101001001101111011\
001110101011101111
```

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
```

Example:
Create an RSA key

```
$ openssl genrsa -out demo_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).

```
$ openssl req -key demo_ca.key -new -out demo_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) []:NJ
Locality Name (eg, city) [Default City]:New Brunswick
Organization Name (eg, company) [Default Company Ltd]:Knights
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:Knights Signing 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 demo_ca.csr file to make a self-signed certifcate authority

```
$ openssl req -x509 -in demo_ca.csr -key demo_ca.key -out demo_ca.crt
```

Example:
Create a website certificate and sign it with the demo_ca.crt key

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

$ openssl req -key www.knights.edu.key -new -out www.knights.edu.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) []:NJ
Locality Name (eg, city) [Default City]:New Brunswick
Organization Name (eg, company) [Default Company Ltd]:Knights
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:www.knights.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 demo_ca.key -cert demo_ca.crt -in www.knights.edu.csr -out www.knights
.edu.crt -config openssl.cnf -create_serial

Using configuration from openssl.cnf
Can't open /home/ep/certdemo/CA/index.txt.attr for reading, No such file or directory
139987170342720:error:02001002:system library:fopen:No such file or
directory:crypto/bio/bss_file.c:74:fopen('/home/ep/certdemo/CA/index.txt.attr','r')
139987170342720:error:2006D080:BIO routines:BIO_new_file:no such file:crypto/bio/bss_file.c:81:
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       = NJ
            organizationName          = Knights
            commonName                = www.knights.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 www.knights.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 = NJ, L = New Brunswick, O = Knights, CN = Knights Signing CA
        Validity
            Not Before: Oct 29 19:16:42 2018 GMT
            Not After : Oct 29 19:16:42 2019 GMT
        Subject: C = US, ST = NJ, O = Knights, CN = www.knights.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:
        generate and display a 174bit RSA key. This is an example of using both the
        “genrsa” and “rsa” commands together with a pipe on the command line.

```
$ openssl genrsa 174 | openssl rsa -text -noout
Generating RSA private key, 174 bit long modulus
..++++++++++++++++++++++++++++
...++++++++++++++++++++++++++++
e is 65537 (0x10001)
Private-Key: (174 bit)
modulus:
    2a:d3:21:08:47:24:fe:33:fd:d1:06:f7:f1:eb:fd:
    23:5b:cd:05:c6:77:33
publicExponent: 65537 (0x10001)
privateExponent:
    1f:ed:b5:ad:14:4e:12:70:f5:06:48:cd:5f:88:2f:
    ef:08:bc:ce:93:92:b1
prime1:
    6f:1f:76:7b:f5:e1:4a:4d:e9:d5:df
prime2:
    62:a8:5e:84:80:df:93:94:ea:01:2d
exponent1:
    24:22:d7:24:f8:90:8a:d9:4b:71:81
exponent2:
    01:9d:e4:be:a3:26:06:d3:1f:ea:99
coefficient:
    5f:51:65:13:e5:f2:64:5b:7f:c3:a2
```

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 "6f:1f:76:7b:f5:e1:4a:4d:e9:d5:df * 62:a8:5e:84:80:df:93:94:ea:01:2d" | \
  sed 's/://g' | \
  tr 'a-f' 'A-F' | \
  sed 's/^/obase=16;ibase=16;/'| \
  bc | \
  tr 'A-F' 'a-f' | \
  sed 's/\(..\)/\1:/g'
2a:d3:21:08:47:24:fe:33:fd:d1:06:f7:f1:eb:fd:23:5b:cd:05:c6:77:33:
```

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.

```
$ openssl aes-256-cbc -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:     ASCII text
database_credentials.php.enc: data

$ more database_credentials.php
<?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/ep/.ssh/id_rsa): /home/ep/.ssh/rutgers
...
Your identification has been saved in /home/ep/.ssh/rutgers.
Your public key has been saved in /home/ep/.ssh/rutgers.pub.
...

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