Table of Contents

Generate CA + certificate

```
openssl genrsa -out rootCAKey.pem 2048
openssl req -x509 -sha256 -new -nodes -key rootCAKey.pem -days 7300 -out rootCACert.pem
openssl x509 -in rootCACert.pem -text
openssl x509 -outform der -in rootCACert.pem -out rootCACert.crt

-- generate server cert
openssl genrsa -out ServerKey.pem 2048
openssl req -new -sha256 -nodes \
  -key ServerKey.pem -out ServerRequest.csr -reqexts san -config \
  <(echo "[req]";
    echo distinguished_name=req;
    echo "[san]";
    echo "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
    ) \
  -subj "/CN=example.com"
printf "[san]\nsubjectAltName = DNS:example.com,DNS:www.example.net,IP:10.0.0.1\n" >v3.ext
openssl x509 -req -sha256 -in ServerRequest.csr -CA rootCACert.pem -CAkey rootCAKey.pem -
CAcreateserial -out ServerCert.pem -days 3650 -extfile v3.ext -extensions san
openssl pkcs12 -export -out cert.pfx -inkey ServerKey.pem -in ServerCert.pem -certfile
rootCACert.pem
```

Show certificate request

```
openssl req -in ServerRequest.csr -noout -text
```

Check TLS

```
nmap --script ssl-enum-ciphers janforman.com -p 443
```

Prefer PolyChacha in TLS 1.3

add in /etc/ssl/openssl.conf or /etc/crypto-policies/back-ends/opensslcnf.config

```
openssl_conf = default_conf

[default_conf]
ssl_conf = ssl_sect

[ssl_sect]
system_default = system_default_sect

[system_default_sect]
Ciphersuites =
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_SHA256:TLS_AES_128_CCM_8_
SHA256:TLS_AES_256_GCM_SHA384
Options = ServerPreference
```

PuTTY CAC

PuTTY CAC is a fork of the PuTTY, a popular Secure Shell (SSH) terminal. PuTTY CAC adds the ability to use the Windows

Certificate API (CAPI) or a Public Key Cryptography Standards (PKCS) library to perform SSH public key authentication using a private key associated with a certificate that is stored on a hardware token.

Download

SSH generate key

```
ssh-keygen -t ecdsa -b 384 -m PEM -C "Comment" -f ./key.pem
```

```
openssl req -key ./key.pem -new -nodes -x509 -days 365 -out key_certificate.pem
```

```
openssl pkcs12 -export -inkey key.pem -in key_certificate.pem -out key.p12
```

PuTTY Key Generator

Cert Identity Search

https://crt.sh/

OpenSSL conf

CentOS location

```
/etc/pki/tls/openssl.conf
```

PGP Keyserver

https://keyserver.pgp.com/vkd/GetWelcomeScreen.event

Hardware Acceleration

Check if AES-NI is enabled

```
grep -m1 -o aes /proc/cpuinfo
```

Check speed

```
openssl speed aes-128-cbc
openssl speed -evp aes-128-cbc
openssl speed -evp chacha20
```

Check OpenSSL throughput

```
dd if=/dev/zero count=100 bs=1M | ssh -c aes128-cbc localhost "cat >/dev/null"
```

Performance remarks

Decrypting a 1MB file on the Galaxy Nexus (OMAP 4460 chip)	
AES-128-GCM	41.6ms
ChaCha20-Poly1305	13.2ms

AES128 vs AES256 1.38x faster
AES128 faster on desktop due to AES-NI HW Acceleration AES-NI is between 4-8x the performance of AES
ChaCha20-Poly1305 faster on mobile phones or slower HW

AWS Graviton2 performance

AES 128bit GCM	2482MB/s
AES 256bit GCM	2014MB/s
ChaCha20-Poly1305	731MB/s

GT-AX6000 Broadcom

AES 128bit GCM	783MB/s
AES 256bit GCM	673MB/s
ChaCha20-Poly1305	297MB/s

RaspberryPI 4 Broadcom

AES 128bit GCM	783MB/s
AES 256bit GCM	673MB/s
ChaCha20-Poly1305	297MB/s

OpenSSL Cipher list

```
# openssl ciphers | sed 's/\:/\n/gi'
```

OpenSSL Performance test

```
# openssl speed md5 sha1 sha256 sha512 des des-ede3 aes-128-cbc aes-192-cbc aes-256-cbc
rsa2048 dsa2048
```

My preferred string for now

Functional with HTTP/2 protocol

```
ssl_session_timeout 4h;
ssl_session_cache shared:SSL:40M;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_prefer_server_ciphers on;
ssl_ciphers !aNULL:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-
AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256;
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains;";
```

Test StartTLS

```
openssl s_client -connect ip:21 -starttls ftp -showcerts
openssl s_client -connect ip:25 -starttls smtp -showcerts
```

Encrypt tar with password

Compress and encrypt

```
tar cvfz - * | openssl enc -e -aes128 -out secured.tar.gz
```

Decrypt and decompress

```
openssl enc -d -aes128 -in secured.tar.gz | tar xvz -C test
```

Install additional CA in CentOS / Redhat

```
place CA here -> /etc/pki/tls/certs/cert.pem
yum install /usr/bin/c_rehash
c_rehash
```

Self signed certificate + altname

```
set -e

if [ -z "$1" ]; then
  hostname="$HOSTNAME"
else
  hostname="$1"
fi

local_openssl_config="
[ req ]
prompt = no
distinguished_name = req_distinguished_name
x509_extensions = san_self_signed
[ req_distinguished_name ]
CN=$hostname
[ san_self_signed ]
subjectAltName = DNS:$hostname, DNS:localhost
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = CA:true
keyUsage = nonRepudiation, digitalSignature, keyEncipherment, dataEncipherment, keyCertSign,
cRLSign
extendedKeyUsage = serverAuth, clientAuth, timeStamping
"

openssl req \
  -newkey rsa:2048 -nodes \
  -keyout "$hostname.key.pem" \
  -x509 -sha256 -days 3650 \
  -config <(echo "$local_openssl_config") \
  -out "$hostname.cert.pem"
openssl x509 -noout -text -in "$hostname.cert.pem"
```