Overview

This tutorial will go over the different flags and supported features of our client/server application. In order to test the different functionalities, make sure you have set up a working configuration of our program by either following the Build from Docker tutorial or the Build from Source tutorial.

Supported Flags

Our client and server program both support a number of different flags to enhance the functionality of the system. These flags are listed below:

Flag	Default Value	Description
-openssl-path	“../../build/bin/openssl”	The path to openssl v3.3
-openssl-cnf-path	“../../openssl/apps/openssl.cnf”	The path to openssl v3.3 config
-dst	“127.0.0.1:9080”	The address being listened to by the server
-src	“127.0.0.1:9080”	The address being dialed by the client
-sa	“DILITHIUM3”	The algorithm used to generate certificates
-ka	“Kyber512”	The algorithm used for KEM
-ca	“../qs509/etc/crt/dilithium3_CA.crt”	The certificate authority used to sign certificates
-ca-key	“../qs509/etc/keys/dilithium3_CA.key”	The certificate authority key used to verify certificates

Only the client has access to the -dst flag and only the server has access to the -src flag.

Using the Different Flags

When you set up your server and client programs, you must be careful with specifying the different flags. When you use a flag for either the server or client, you must remember to use that same flag with the other. For example, if you specify a KEM algorithm in the server, **you must** specify the same KEM algorithm in the client. Failing to do so will result in a run time error.

OpenSSL and OpenSSLCNF flags

If you followed the instructions to build the application from source or from docker, then the default value should be correct for these flags and you do not need to worry about them.

If you have installed OpenSSL 3.3 in a directory other than /root/quantumsafe or if your project code does not exist in /root/quantumsafe/ProjectCode, then you will need to adjust the path to make sure you are pointing towards OpenSSL 3.3

Dst and Src Flags

These flags are used to adjust where the client dials and where the server listens. When launching a server you can specify exactly which IP and which port you want it to listen on. This is the same IP and port you will then want the client to dial.

If you followed the Build from Source tutorial and are using Mininet to simulate different topologies, then the different hosts in Mininet will have different IP addresses. You can print these out by running:

```
<hostname> ifconfig
```

If using our custom topologies, h1 always has an IP of 10.0.0.1 and h2 always has an IP of 10.0.0.2. Therefore, to run our client and server, you can use:

```
h1 ./server -src 10.0.0.1:9080 &
h2 ./client -dst 10.0.0.1:9080
```

Sa Flag

The Signature Algorithm flag can be used to specify which signature algorithm you want to use to create your certificate as a client or a server. There are a number of supported signature algorithms listed below:

RSA
ED25519
ED448
dilithium2
p256_dilithium2
rsa3072_dilithium2
dilithium3
p384_dilithium3
dilithium5
p521_dilithium5
mldsa44
p256_mldsa44
rsa3072_mldsa44
mldsa44_pss2048
mldsa44_rsa2048
mldsa44_ed25519
mldsa44_p256
mldsa44_bp256
mldsa65
p384_mldsa65
mldsa65_pss3072
mldsa65_rsa3072
mldsa65_p256
mldsa65_bp256
mldsa65_ed25519
mldsa87
p521_mldsa87
mldsa87_p384
mldsa87_bp384
mldsa87_ed448
falcon512
p256_falcon512
rsa3072_falcon512
falconpadded512
p256_falconpadded512
rsa3072_falconpadded512
falcon1024
p521_falcon1024
falconpadded1024
p521_falconpadded1024
sphincssha2128fsimple
p256_sphincssha2128fsimple
rsa3072_sphincssha2128fsimple
sphincssha2128ssimple
p256_sphincssha2128ssimple
rsa3072_sphincssha2128ssimple
sphincssha2192fsimple
p384_sphincssha2192fsimple
sphincsshake128fsimple
p256_sphincsshake128fsimple
rsa3072_sphincsshake128fsimple

Ka Flag

The KEM algorithm flag can be used to specify which key exchange algorithms are able to be used. There are a number of supported algorithms below:

EC
BIKE-L1
BIKE-L3
BIKE-L5
Classic-McEliece-348864
Classic-McEliece-348864f
Classic-McEliece-460896
Classic-McEliece-460896f
Classic-McEliece-6688128
Classic-McEliece-6688128f
Classic-McEliece-6960119
Classic-McEliece-6960119f
Classic-McEliece-8192128
Classic-McEliece-8192128f
Kyber512
Kyber768
Kyber1024
ML-KEM-512-ipd
ML-KEM-512
ML-KEM-768-ipd
ML-KEM-768
ML-KEM-1024-ipd
ML-KEM-1024
Sntrup761
FrodoKEM-640-AES
FrodoKEM-640-SHAKE
FrodoKEM-976-AES
FrodoKEM-976-SHAKE
FrodoKEM-1344-AES
FrodoKEM-1344-SHAKE

Ca and Ca-key Flag

The Certificate Authority and Certificate Authority Key flags specify which signing authority you would like to use to sign and verify certificates in our program. Our qs509 library offers two different, local certificate authorities. One of which is a dilithium3 authority, the other an RSA authority.

If you would like to specify a different authority to use, just put the file path leading to the authority's public certificate and the key used to verify it.