**Quantum Cryptography Project**

**Introduction-**

To whoever is reading this document, hello! I’m Oliver, a summer intern who worked on this quantum encryption project. In this documentation, I’ll entail the research and progress I’ve made, and you can recreate the process through my steps. I will also be providing multiple links for resources on other tutorials and research that you can read up on.

**Purpose-**

The age of quantum is coming soon, where we’ll have computers that can compute things many times faster than usual. This also means that many of our current security systems, including RSA, are not safe. The new quantum systems will be faster at cracking our systems. In response to this new age, we must prepare beforehand algorithms and systems that can handle the new computing capability. In the coming steps, I will demonstrate a proof of concept by running a local web server with quantum safe encryption.

**The Algorithms: Intro-**

RSA encryption is estimated to be obsolete by 2030. In the Wikipedia article below, NIST attempts to standardize some early algorithms. There were originally 69 algorithms competing and after 3 rounds of testing, there was 7 algorithms left in 2021. For most of the demos in this document, we will be using CRYSTALS-KYBER. CRYSTALS-KYBER is currently being used by Cloudflare, Amazon, and is heavily advertised by IBM. NTRU and McEliece are alternative cryptosystems that have seen to be quantum safe so far and have been around before 2000’s. Due to their age, there is much more research and availability. However, due to quantum encryption being a volatile and rapidly advancing field, it is up to the user to keep up to date on which algorithms are safe.

Beyond algorithms for encryption, there is also algorithms for server and client-side key generation. We will also be using part of the CRYSTAL family, called CRYSTALS-

DILITHIUM.

Below are some links for more information, where you can read more about the algorithms and their current state.

* Post Quantum Cryptography Information- <https://en.wikipedia.org/wiki/NIST_Post-Quantum_Cryptography_Standardization>
* CRYSTALS website- <https://pq-crystals.org/kyber/>
* NTRU- <https://en.wikipedia.org/wiki/NTRU>
* McEliece- <https://en.wikipedia.org/wiki/McEliece_cryptosystem>

**Resources-**

In order to prototype these new quantum safe algorithms, we will be using the Open Quantum Safe project. This is an open-source project which will allow us to prototype the server and implement the algorithms. The website features many demos and applications that can be used to test. OQS also has the benefit of supporting many different quantum and hybrid algorithms. We will also be demoing these servers through docker, so make sure that is installed as well.

Open Quantum Safe- <https://openquantumsafe.org/>

Open Quantum Safe Demos- <https://openquantumsafe.org/applications/tls/>

Algorithms Supported- <https://github.com/open-quantum-safe/openssl#supported-algorithms>

**Creating a Basic Server-**

As an initial start, make sure you have the two folders with the docker files in them. One should be named Curl, and the other Quantum.

We will create and start our server by typing the following into the command line/terminal:

**docker network create httpd-test**

**docker build -t optumquantum2:dev <file path to quantum folder’s dockerfile>**

**docker run --network httpd-test -p 4433:4433 --name oqs-httpd optumquantum2:dev**

Next, create a new terminal and enter the following command to retrieve the information from the server by running:

**docker build -t optumcurl2:dev <file path to curl folder’s dockerfile>**

**docker run --network httpd-test -p 8080:8080 optumcurl2:dev**

The result should look something like this:

Graphical user interface, text

Description automatically generated

On the left is the server, and at the bottom you can see the GET requests coming in. On the right is our GET request, and the html page is printed out at the bottom. This is also reflected in Docker, and you should see something like this:

Graphical user interface, text, application, Word

Description automatically generated

**Features and Customization-**

Beyond creating a basic server, OQS has multiple features and commands to change the server. For example, adding a name to server is customizable by adding the name after “–name”. This can be seen in the initial **Creating a Basic Server section**.

Using --curves, we can also alter the GET command to instead use a quantum safe algorithm to retrieve the webpage. In this example, we used p256\_kyber512 which is a hybrid version of the CRYSTALS-Kyber algorithm.

A picture containing text

Description automatically generated

We can also create keys and certificates for the server. In this example, we’ll be using dilithium2which is part of CRYSTALS-Dilithium; you are free to change it to whatever is supported. We will start by creating and accessing the directory to store the keys and certificates.

**mkdir -p server-pki && cd server-pki**

Next, we can create CA key and certificate using **dilithium2:**

**docker run -v `pwd`:/opt/tmp -it openquantumsafe/curl openssl req -x509 -new -newkey dilithium2 -keyout /opt/tmp/CA.key -out /opt/tmp/CA.crt -nodes -subj "/CN=oqstest CA" -days 365**

Next is the server key:

**docker run -v `pwd`:/opt/tmp -it openquantumsafe/curl openssl req -new -newkey dilithium2 -keyout /opt/tmp/server.key -out /opt/tmp/server.csr -nodes -subj "/CN=httpd.server.my.org"**

And finally, is the server certificate:

**docker run -v `pwd`:/opt/tmp -it openquantumsafe/curl openssl x509 -req -in /opt/tmp/server.csr -out /opt/tmp/server.crt -CA /opt/tmp/CA.crt -CAkey /opt/tmp/CA.key -CAcreateserial -days 365**

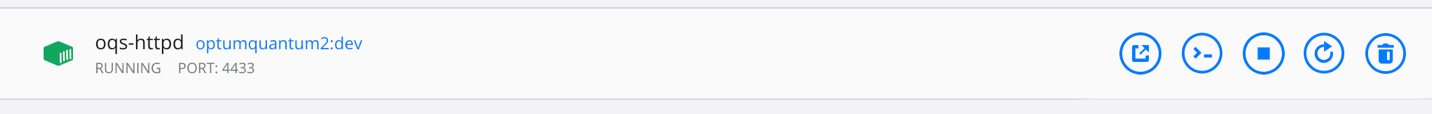
Assuming there was no errors, your final result should look something like this:

Text

Description automatically generated

**Finding Server Files-**

By default, the port number is 4433. However, this and many other features can be changed in their respective files. To access these files easily, simply just press the CLI button on the server in the docker container.



This will bring up a terminal that is currently accessed to the server. Simply ls to view the contents:

Text

Description automatically generated with medium confidence

You can change the port number in httpd.conf files, along with any other features. Below are some of the common file paths you will need to customize the server.

**httpd logfile folder: /opt/httpd/logs**

**httpd configuration folder location: /opt/httpd/httpd-conf**

* **This folder contains two files: httpd.conf for baseline httpd configuration and httpd-ssl.conf for all TLS/SSL specific configuration options.**

**httpd PKI location: /opt/httpd/pki**

**Server key: /opt/httpd/pki/server.key**

**Server certificate: /opt/httpd/pki/server.crt**

**Algorithms: In Depth Analysis-**

Because quantum security is a relatively new space, it is likely not fit for consumer use since it is still in prototyping phase. As mentioned above, NIST has yet to standardize it. However, it is a good idea to get started since quantum is on the horizon. One of the main things we can do right now is to implement hybrid cryptography. We can implement traditional key encryption such as RSA, and combine them with newer quantum-safe keys. This way, we can ensure the server is secured both ways and each part acts as a failsafe for the other.

Graphical user interface, text, application, Word

Description automatically generated As mentioned earlier, we are mainly looking at 3 algorithms. You will notice that for NTRU, there are two version, one being the prime. The main difference is that NTRU prime dealt with a potential algebraic structure exploit. This makes the algorithm overall more robust, but no one has ever exploited NTRU’s algebraic structure in the first place. For caution sake, it is probably a better idea to learn towards NTRU prime.

In terms of performance, it is hard to compare traditional encryption to quantum encryption. Because modern CPU’s have specific sections of memory portioned off to encrypt/decrypt, it provides skewed statistics that favor traditional encryption. Hardware acceleration also comes into play, and it is overall not a swell time comparing RSA to CRYSTAL-Kyber for example. That being said, it is much easier to compare quantum algorithms with other quantum algorithms. In a paper also in this GitHub, there is a document written by Andrew Nash on the benefits and risks of each algorithm we’re currently looking at. While Kyber is our main candidate, there are some underlying issues that could become problems in the future if hackers manage to crack it. You can also find a link below, that lists the benchmark speeds per algorithm tested by OQS.

<https://openquantumsafe.org/benchmarking/visualization/openssl_speed.html>

One way you can performance test the 3 different algorithms internally is through the following command:

**docker run -it openquantumsafe/curl openssl speed -seconds 2 kyber512 ntrulpr653 saber**

Text, letter

Description automatically generated

You should see a result like this, where it gives you the time for key generation, encapsulation, and decapsulation. For more information, see the following links below:

<https://github.com/open-quantum-safe/openssl#performance-testing>

<https://github.com/open-quantum-safe/oqs-demos/tree/master/curl#performance-testing>