# Fuctional Spec and Project Plan

Cavan Phelan - C00249198

# TABLE OF CONTENTS

# 1. INTRODUCTION

Cryptography is evolving constantly, and it plays a major role in our world even though it is often hidden. People rely on cryptography constantly in their everyday life but do not know how it works, or the advantages some algorithms have over others. This also brings in quantum algorithms. People keep hearing how it will change the world, why it will change the world. Benchmarking is a good way to see the differences in the different eras of cryptography, and what types of algorithms provide certain advantages.

## 1.1. PURPOSE OF THE DOCUMENT

This document proves an overview of what algorithms I will be benchmarking and how I am able to run the benchmarks as required. It will also provide how a user is expected to interact with the system. A project plan will be provided to keep track of the work of the project.

## 1.2. PROJECT SCOPE

The scope of the project includes a comparison of the performance of traditional algorithms and quantum algorithms, also the different algorithm types, on a range of tasks, in order to compare the relative advantages and limitations of these algorithms.

## 1.3. PURPOSE OF THE BENCHMARKING

The purposes of benchmarking these algorithms are the following.

**[1]** To bring more awareness to Cryptography, something that everyone uses every day, but not many people know about. It should be important to know and understand something that protects us and allows us to securely communicate together.

**[2]** To compare the performance of traditional algorithms with quantum algorithms. This will help us showcase the potential advantages or disadvantages of using quantum algorithms over traditional algorithms on certain tasks. These benchmarks will also help uncover areas where more research and development are needed regarding quantum algorithms, considering they are relatively new.

## 1.4. SCOPE OF THE BENCHMARKING

The performance of these algorithms will be measured using a range of metrics. These metrics include runtime, memory usage (if applicable), CPU usage, key lengths, complexity, accuracy etc. These metrics will be collected and reported using benchmarking techniques. The performance of the algorithms will be evaluated and compared based on; the relative performance on different tasks such as encryption and decryption, the improvements in performance provided by quantum algorithms and the potential application of the algorithms.

# 2. SYSTEM ARCHITECTURE

This will provide an overview of the system and how it will function. This includes APIs and libraries, the UI and its functions and more.

## 2.1. REQUIREMENTS

This will provide the requirements from a software point of view, including the APIs used to make this all work.

### 2.1.1. BOUNCY CASTLE (BC)

This is a Java API for doing cryptography within Java. The API helps implement and perform cryptographic operations. This also allows us to implement new quantum algorithms selected by NIST.

### 2.1.2. JAVA MICRO-BENCHMARKING HARNESS (JMH)

This is a framework which is used to help write and run microbenchmarks. Microbenchmarks help benchmark small chunks of code rather than the entire algorithm. This helps us provide more in-depth benchmarks regarding specific tasks such as encryption and decryption.

### 2.1.3. JAVA UNIVERSAL NETWORK/GRAPH (JUNG)

This framework will help me graph benchmarking results into a report to make a more visual representation of the benchmarks.

## 2.2. QUANTUM ALGORITHMS TO BE BENCHMARKED

### 2.2.1. CRYSTALS-DILITHIUM

This algorithm is a lattice-based digital signature. Digital signatures are used to provide message integrity and authenticity.

### 2.2.2. CRYSTALS -KYBER

This algorithm is a key-encapsulation, which is a hybrid of symmetric and asymmetric key algorithms.

### 2.2.3. SPHINCS+

This algorithm is hash based; it's aim is to reduce the hash sizes whilst making it more secure.

### 2.2.4. SIKE

This algorithm is like CRYSTALS-Kyber, which is mentioned above, except is based on the Super-singular Isogeny Diffie-Hellman key exchange protocol.

### 2.2.5. PICNIC

This algorithm is like CRYSTALS-Dilithium, but instead its Digital Signatures work with Zero-Knowledge proofs.

## 2.3. PRE-QUANTUM ALGORITHMS TO BE BENCHMARKED

### 2.3.1. AES (CTR/CBC)

A widely used symmetric-key algorithm, meaning it only uses one key to encrypt/decrypt. Counter (CTR) mode transforms the traditional block cipher into a stream cipher.

### 2.3.2. MD5

This hashing algorithm use to be the world standard until it was broken. This should not be used to hash passwords or other data. Despite this, it is still used to verify files have not been altered in transit using hash checksums.

### 2.3.3. SHA-256 / SHA-512

These algorithms are from the SHA-2 hashing family and are commonly used worldwide.

### 2.3.4. RSA

This algorithm utilizes asymmetric cryptography, meaning it used a public key to encrypt and a private key to decrypt.

### 2.3.5. EL GAMAL

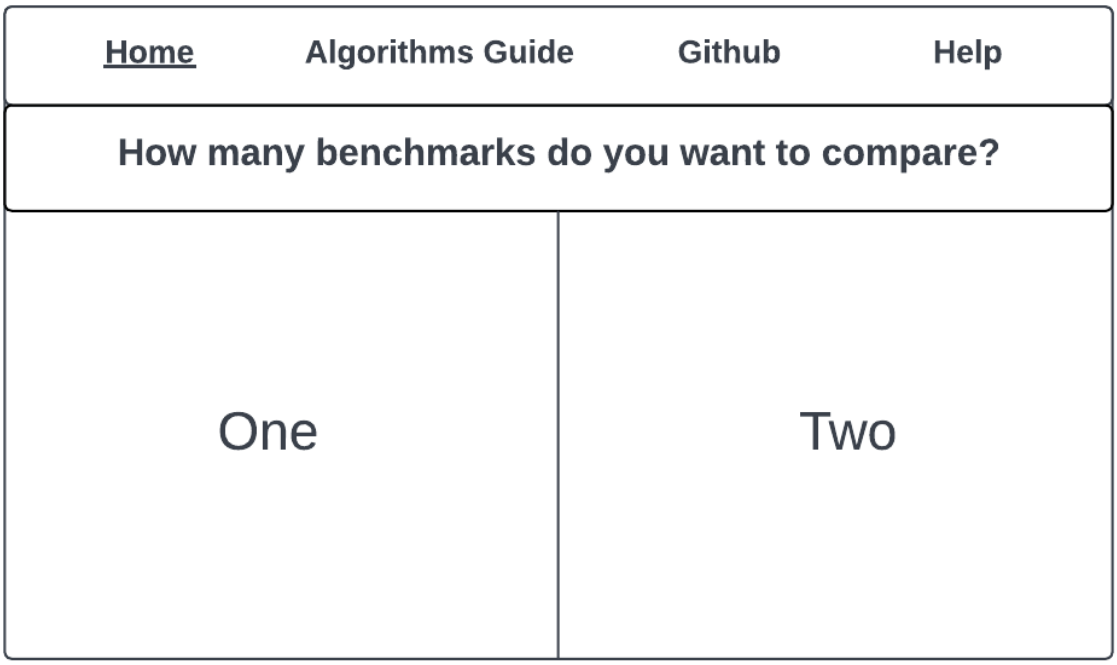
This algorithm is very similar to RSA, except it takes advantage of a different key exchange algorithm, Diffie-Hellman.

# 3. SYSTEM INTERFACES AND THEIR FUNCTIONS

This will provide an overview of the planned application interfaces. It will describe the functions of each page and what they user is able to do through the page's functions provided.

This overview is more than likely going to be changed over time, this is just a rough idea of how it could be implemented. If there is time available, it will be optimised to be as easy as possible to use.

## 3.1. HOME PAGE




### 3.1.1. HOME PAGE FUNCTION

Once the user opens the application this home page will greet the user. It will prompt them on whether they want to benchmark a single algorithm, or to benchmark two algorithms and compare them against one another.

## 3.2. ALGORITHM SELECTION PAGE

| Home | Algorithm Selection | Algorithms Guide |
|---|---|---|

| Quantum Algorithms | Pre-Quantum Algorithms |
|---|---|
| CRYSTALS-Dilithium | AES (CTR/CBC mode) |
| CRYSTALS-Kyber | MD5 |
| SPHINCS+ | SHA-256 |
| SIKE | RSA |
| Picnic | ElGamal |

### 3.2.1. ALGORITHM SELECTION PAGE FUNCTION

This page allows the user to select the number of algorithms they have chosen on the previous page. If the user chose two algorithms, they could then click to select two algorithms and then these will be later benchmarked.

## 3.3. ALGORITHMS GUIDE PAGE

| Home | Algorithms Guide | Github | Help |
|---|---|---|---|

| Quantum Algorithms | Pre-Quantum Algorithms |
|---|---|
| CRYSTALS-Dilithium | AES (CTR/CBC mode) |
| CRYSTALS-Kyber | MD5 |
| SPHINCS+ | SHA-256 |
| SIKE | RSA |
| Picnic | ElGamal |

### 3.3.1. ALGORITHM GUIDE PAGE FUNCTION

This interface provides the two eras of Cryptography and their respective algorithms which the tool benchmarks. Clicking on these algorithms will bring you to their developers' websites to get more information on them.

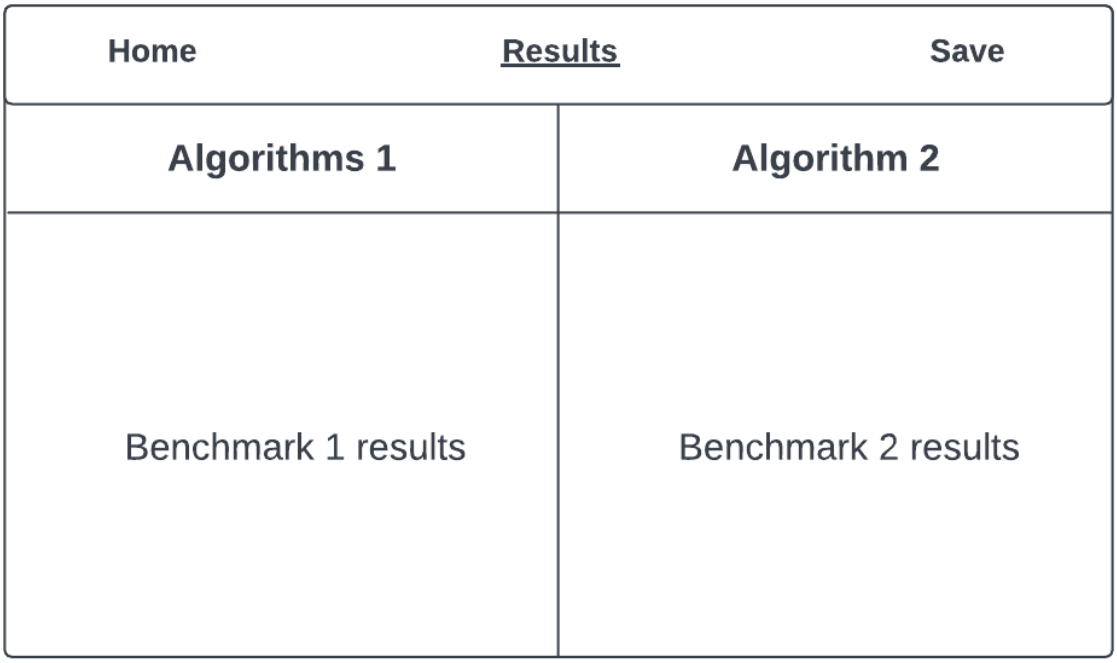
### 3.3.2. GITHUB PAGE FUNCTION

Clicking the link will redirect the user to my GitHub where all my project files will be held.

### 3.3.3. HELP PAGE FUNCTION

The help page is designed to provide users with an easy explanation on how to create their own benchmarks. A lot of the current tutorials are outdated, or not relevant to benchmarking algorithms.

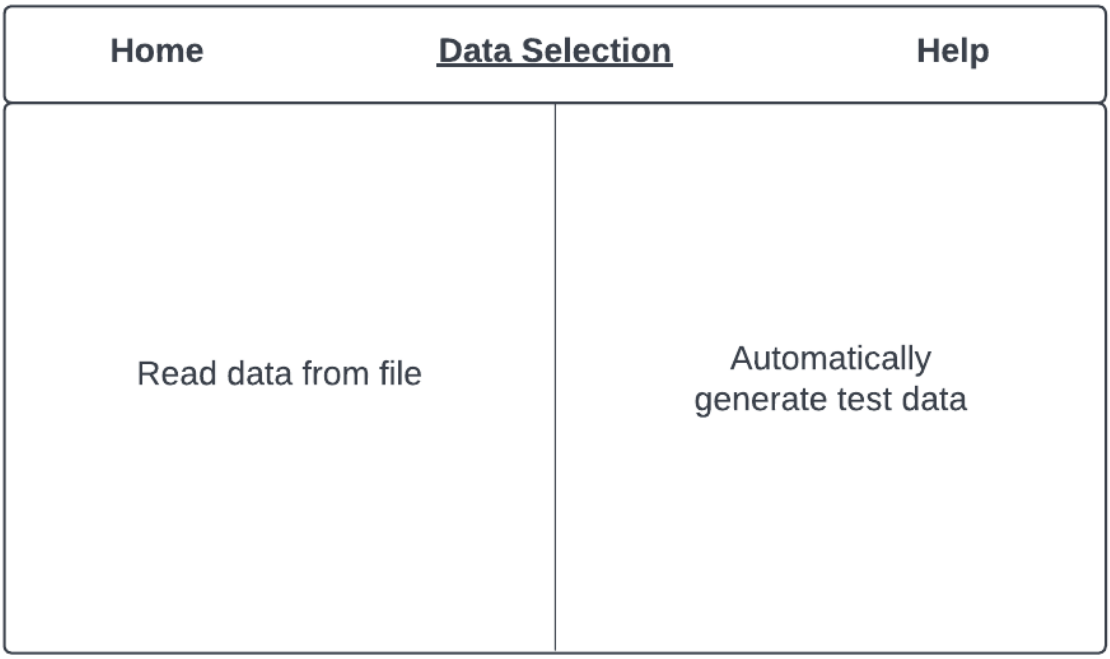
## 3.4. RESULTS PAGE




### 3.4.1. RESULTS PAGE FUNCTION

Once they benchmarks have been run, they will display on the results page, based on how many algorithms were selected. The user will have a further option to save the results as a file on their computer if they wish to do so.

## 3.5. DATA SELECTION PAGE




### 3.5.1. DATA SELECTION PAGE FUNCTION

This page will prompt the user to select whether they want to use their own test data that they can read through a file, or have the algorithms generate data on its own.

## 4. PROJECT PLAN

The project plan will help track the project progress and the order in which I hope to develop it.

| Action | Time |
|---|---|
| Create benchmarks for pre-quantum algorithms | 17th December – 31st December |
| Create benchmarks for quantum algorithms | 2nd January – 31st January |
| Create ability to read test data from a file | 1st February – 21st February |
| Test application with multiple test data | 21st February – 21st March |
| Main GUI | 21st March – 31st March |
| Link developers to algorithms list | 1st April – 3rd April |
| Final Report | 3rd April – 17th April |

### 4.1. CREATING BENCHMARKS FOR THE ALGORITHMS – SUB TASKS

1. Create manually benchmarks to confirm that the algorithms are working normally.
2. Take each step of the algorithms such as encryption, decryption, key generation, etc and microbenchmark them individually.
3. Take each individual benchmark and benchmark them together in one.
4. Test the benchmarks with varied test data and different number of warmups and iterations.

### 4.2. ABILITY TO READ TEST DATA FROM FILES

1. Learn how to read data in from files on Java.
2. Implement and test basic file reading with system prints confirming.
3. Test readable data by reading in a secret key and using it within the benchmark.
4. Slowly add more imported data to be read in and implemented.
5. Test readable data with the final benchmarks, provide user with option to read in from file or not.

### 4.3. TEST APPLICATION WITH MULTIPLE TEST DATA SETS

1. Use test data of various sizes, try to be as broad as possible with the ranges of test data.
2. Have benchmarks test different key sizes, running 5 times each or so.

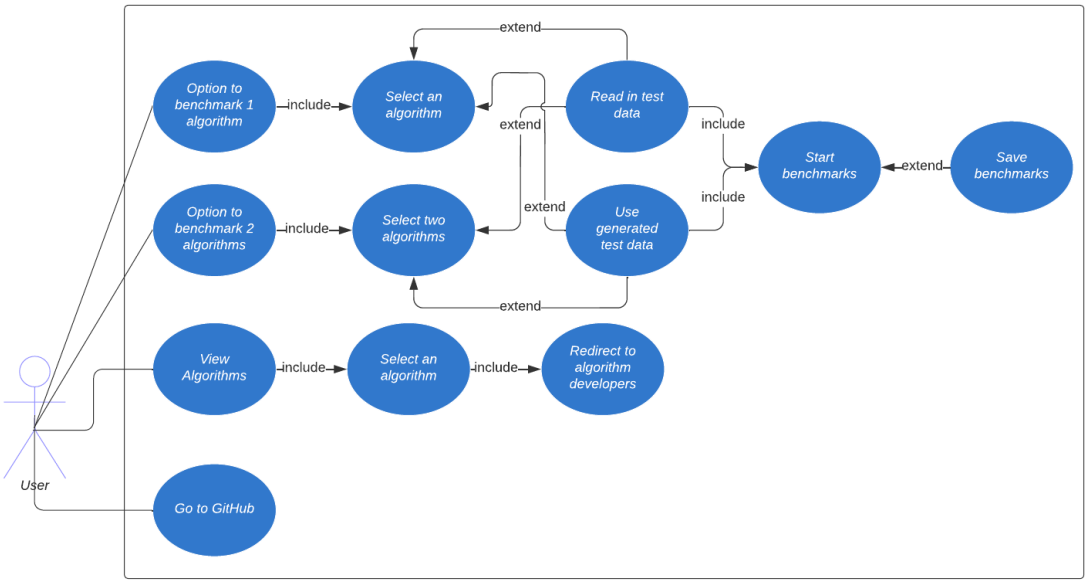
### 4.4. MAIN GUI

1. Create main pages for the hub.
2. Create benchmarking list and provide links to the developer's pages.
3. Allow users to click which algorithms to use.
4. Allow users to use their own test data or automatically generate it.

5. Provide help page to guide users how to use application. Also provide help on how to get started doing your own benchmarks.
6. Provide link to my GitHub project.

## 5. USE CASE DIAGRAM

A use case diagram is a behavioural diagram based off the system. It showcases the relationship between the user and the system and the expected functionality.



# 6. FURPS+

## 6.1. FUNCTIONALITY

1. The application allows the user select one/two algorithms to choose from to benchmark.
2. The application allows the user to decide on whether to use their own test data read in from a file or have it generated by the application.
3. The application can choose to have the benchmarks saved to a file or displayed to them.
4. The application allows user to find out more information about the algorithms and directs them to the developers' pages.

## 6.2. USABILITY

1. The application should be user friendly and easy to navigate.
2. Should be easy to read in files to the application.

## 6.3. RELIABILITY

1. Application should run smoothly and not crash during benchmarks.

## 6.4. PERFORMANCE

1. The application should be fast to navigate, and the benchmarks should run smoothly.
2. Reading in files should be smooth.

## 6.5. SUPPORTABILTY

1. The application will be run through multiple tests with various test sets.
2. The application will provide help on how to use the UI, and how to create your own benchmarks.

## 6.6. + SECURITY

1. The algorithms will be working in a secure manor, even though sensitive data should not be tested anyway.
2. The application will be secure against vulnerabilities.