***Chapter 3***

This chapter focuses on the choice of parameter requirements, searches, and choices for the instantiations of SQIsign in the NIST-I, NIST-III, and NIST-V security levels. About SQIsign, parameter requirements refer to the specific set of mathematical parameters chosen to optimize efficiency and security in the signature scheme. However, about SQIsign, the specific choice of parameters within the requirements defined in this research has no impact on the security of SQIsign. The efficiency of SQIsign is on choosing a prime *p* of suitable size such that contains many smooth prime factors. There are five recommended sizes for security parameters λ. I would be giving three here and they include Prime size of ; the degree Dcom of the commitment isogeny *φcom* of size approximately 22λ ; the degree of Dchall of the challenge isogeny *φchall* of size approximately 2λ. The parameter of our choice should mainly prioritize fast verification and in doing so maintain reasonable signing performance. Following this, we are going to look at suitable methods for finding parameters. One of which includes the **Sieve-and-boost.** In this method,we search for primes of the form for a smooth number and for different values of to ensure that *φchall*  can be computed as a chain of 2-isogenies and 3-isogenies. When searching for primes of bit size smaller than 256, 384, and 512 bits respectively, the search space for *x* is rough of size Bits. The algorithm starts by identifying smooth numbers in the search interval for *x* using the sieve implementation. For each smooth *x* and suitable values for *f’, g’* we store the primes of the form then simultaneously computing the B-smooth parts of by using a product tree. The numbers are chosen to give primes suitable for the tree security levels.

***Chapter 6***

**Performance Analysis**

The SQIsign library is built into several sub-modules that are linked to libraries supporting the NIST Signature API such as EC (Elliptic Curve) and isogeny computation, implementation of the KLPT algorithm, the implementation of the ideal to isogeny algorithm also known as Id2iso, the module for quaternion computation, arbitrary precision module based on GMP (GNU Multiple Precision Arithmetic Library), implementing the signature key generation signing and verification protocols. These modules support flexibility development, allowing separate handling of different mathematical and computational tasks within the signature scheme.

The **optimized implementation** is the same as the **reference implementation**. The reference implementation is built with two configurations which are using the GMP system installation on Ubuntu 22.04LTS and using a custom-built version with disabled assembly code.

**Key and Signature Size**

The SQIsign key and signature sizes are listed below for each security level. The key and signature size of SQIsign is critical for ensuring both security and efficiency. A higher key size shows a great level of resistance to attacks. A smaller signature size reduces the amount of data transferred which is important. Minimal key and signature sizes are most preferred in SQIsign to reduce memory and storage requirements. A table with numbers and a few words

Description automatically generated

The **optimized implementation** is the same as the **reference implementation**. The reference implementation is built with two configurations which use the GMP system installation on Ubuntu 22.04LTS and a custom-built version with disabled assembly code.

The **performance evaluation** was performed on an Intel x86 64-bit CPU with results being the median of ten benchmark runs.

***Chapter 7***

This section deals with implementation details and constants that are used in various algorithms and protocols. These constants drive the efficiency and accuracy of its algorithms. These constants help the system achieve reliable outcomes example is when searching for prime numbers within a fixed bit size. The KLPT\_random\_prime\_attempts is used when looking for a random prime number of bit size *k.* We try at most *k*(KLPT\_random\_prime\_attempts) random integers until a prime is found. This comes from the fact that a random number of bit size *k* has a probability Of being prime. The KLPT\_equiv\_bound\_coeff is a constant that is bound on the absolute value of the coefficients of the linear combination of the small basis used to find the equivalent ideal. Also, KLPT\_equiv\_num\_iter is a constant which is the maximum number of trials to find an equivalent ideal of prime norm. The selection of these constants is determined as thus: taking a value of *C* roughly equal to the size of the search space, so this means . For a typical ideal, we have .

There is a constant KLPT\_repres\_num\_gamma\_trial that defines the maximum number of attempts made in the algorithm FullRepresentInteger defined as thus: KLPT\_repres\_num\_gamma\_trial = 2KLPT\_gamma\_exponent\_center\_shift.

Also, there are various constants in SigningKLPT includes KLPT\_signing\_klpt\_length this constant gives the length of the output of the signing KLPT algorithm also we have the KLPT\_signing\_number\_strong\_approx which is the number of vectors tried for the strong approximation step inside the signing KLPT algorithm and we also have two other constants. These constants are fixed so that

A close-up of a computer code

Description automatically generated

Furthermore, we have KeyGenKLPT, and the constants for these are KLPT\_keygen\_length which is the constant of the length of the alternate secret key isogeny we also have the **KLPT\_Keygen\_number\_strong\_approx** which is the constant of the number of vectors tried for the strong approximation step inside the KeyGenKLPT algorithm. This signing algorithm can be fixed the same way it was done previously (the signing KLPT algorithm).

The various constants for the SpecialEichlerNorm can be defined as follows: the **KLPT\_eichler\_num\_alternate\_order** which is the number of precomputed orders that we use can be fixed by setting it equal to 7. Also, we have the **KLPT\_eichler\_number\_strong\_approx** which is the maximum number of trials for the strong approximation step inside the Eichler norm algorithm this parameter can be fixed by setting the ceiling function equal to . We also have the **KLPT\_eichler\_norm\_equiv\_ideal** which is the number of equivalent ideals tried before aborting; this parameter can be fixed by the ceiling function of . We have the **KLPT\_eichler\_number\_mu\_norm** which is equal to the ceiling function of which can be fixed by the maximum number of target norms tried for .