Error-Corrected Quantum Cryptanalysis of CSIDH

Isabel Bromfield

Supervised by Chloe Martindale, Ryan L. Mann, and Romy Minko

Level M

40 Credit Points

May 7, 2022

Acknowledgement of Sources

For all ideas taken from other sources (books, articles, internet), the source of the ideas is mentioned in the main text and fully referenced at the end of the report.

All material which is quoted essentially word-for-word from other sources is given in quotation marks and referenced.

Pictures and diagrams copied from the internet or other sources are labelled with a reference to the web page or book, article etc.

Signed 

Date 06/05/2022

Abstract

This thesis conducts an error-corrected quantum cryptanalysis of Commutative Supersingular Isogeny Diffie-Hellmann (CSIDH). Previous quantum analyses of CSIDH have been carried out but none so far with error correction, which could provider a lot more clarity about the post-quantum security of CSIDH and its future in cybersecurity.

In order to do this, the thesis introduces the areas of cryptology, quantum computing and quantum cryptanalysis before moving into a method of breaking CSIDH using the Dihedral Hidden Subgroup Problem (DHSP). Techniques from these sections can be applied to CSIDH to develop an error corrected summary of its security, focusing on the implementation of surface codes. Estimations of the time and space complexity it takes to break CSIDH-512 are given, including quantum time complexity and time with quantumly accessible classical memory. With large amounts of computing power on unspecialised surface code, CSIDH-512 can be broken in 1.27 days.

Contents

List of Figures

Chapter 1

Introduction

Security makes up a more significant part of our online experience than most people realise. Almost all of our activities online need to be protected against attackers who might want to steal personal data, impersonate organisations or force us to download ransomware on our computers. With more and more of our lives being affected by the internet, being secure online has never been more important.

The advent of quantum computers has brought with it a shift in how we view security. Mathematical principles that formed the basis of the security of online transactions, like the discrete logarithm problem, have been shown to be insecure against algorithms employable by quantum computers. Shor's algorithm [1] revealed the first significant evidence that quantum computers could make intractable problems tractable, if not easy, to break and created a new area of research that is growing every day.

To break the current algorithms used to transfer our information securely, quantum computers will need millions of qubits. It is estimated that the size of quantum computers could double as little as every two years. In as little as 10 years, quantum computers could exist that render nearly all of the protocols that are widely used today useless. For this reason, finding schemes and mathematical problems that are resistant to quantum computers is of paramount importance.

One such scheme is CSIDH, standing for Commutative Supersingular Isogeny Diffie-Hellman, an alternative key exchange protocol similar to the widely-used Diffie-Hellman. Though a mouthful, this particular scheme rests on the security of a deceptively simple problem known as the Dihedral Hidden Subgroup Problem. This problem has been conjectured to be resistant to quantum computers - though more recent literature suggests otherwise - and is an area of much interest. Elliptic curves which form the basis of the theory of isogenies are particularly interesting and presently widely used on the internet. There is much scope in this area for research.

CSIDH was first published in November 2018 by Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny and Joost Renes [2] after a research retreat in Tenerife, leading to CSIDH to be pronounced as 'sea-side'. Since its publication, CSIDH and its offshoot signature protocol SeaSign have been the subject of several publications providing speedups [3, 4] and assessing its possible use as a new standard in quantum resistant protocols.

CSIDH narrowly missed out on applying for the NIST Post-Quantum Cryptography Standardization entrance [5]. NIST, the National Institute of Standards and Technology in the U.S.A. are holding a competition to decide what conjectured quantum-resistant protocol will be standardised and used by many systems all over the world. CSIDH's security can be measured against NIST's various 'levels' of security [6], and the original paper proposed security meeting levels 1,2 and 3 [2].

Since the paper's publication, various responses have focused on the protocol's security against quantum attacks; mainly on the two parts of evaluating the 'group action' and a sieve that combines states to give information on a secret key. These papers theorise that CSIDH does not have necessary properties to reach event NIST level 1 security [7, 8], and demonstrate areas of the protocol that could be solved in quantum subexponential time. However, these papers do not consider the errors that arise implementing code in quantum computers, something that will greatly increase the 'cost' of an attack on CSIDH.

1.1 Contributions of this Paper

Quantum computers are much more vulnerable to random errors occurring - like bit flips or phase flips of the 'qubits' in the system. Error correction, detailed in Chapter 3, is an area of much interest in how it prevents or detects these errors from occurring and helps quantum computers to function correctly. Implementations of quantum computers that utilise error correction tend to use a lot of space and time detecting and correcting these bit and phase flips. This thesis considers the extra overheads that need to be considered when attacking CSIDH, and gives an estimate for the time and space complexity of an attack on CSIDH-512 using several metrics. This is considered for the error-correcting system surface code, one of the most prominent and widely researched areas in quantum computation. The Surface code is favoured as the leading method for implementing large-scale quantum computers due to its scalability and built-in error correction.

1.2 Preliminaries

This thesis assumes an understanding of group theory and Big-O notation. Several terms referred to in this paper can be found defined in the appendix A.

This thesis gives a detailed background into cryptology and elliptic curve cryptography in chapter 2, CSIDH in Chapter 2, quantum information theory, quantum computing and surface code in chapter 3 in order to properly explain the significance of the results obtained in chapter 5. Readers with a background in isogenies and the surface code can read from chapter 4.

1.3 Acknowledgements

My sincere thanks go to Dr. Chloe Martindale, Dr. Ryan Mann and Dr. Romy Minko, all of whom were instrumental in guiding the direction and content of this thesis. I

would like to thank them for their support throughout the course of the creation of this paper, for their endless patience with me, and their sharing my enthusiasm for this exciting subject matter.

Chapter 2

Background: Cryptography and CSIDH

2.1 Cryptography: An Introduction to Subterfuge

Alice and Bob: Your Two Best Friends

Alice and Bob are two people that want to send messages to each other. They want to do this without fear of having their messages intercepted, altered, or removed by other people. 'other people' tends to mean Eve, an eavesdropper or adversary who wants to sabotage or overhear their communication in some way. Eve can breach communication by:

- confidentiality (interception)
- integrity (compromising the message received)
- availability (stopping the message from being received)

Alice, Bob and Eve (and occasionally Charlie might make an appearance) are extremely common figures in the science of cryptology; they illustrate abstract concepts of security in a clear an understandable way by acting as two parties in a conversation. Often, Alice will be sending a message and Bob receiving it, but there is no set format.

We will see these parties repeated again and again in different schemes and protocols, so make sure to say hi!

History

Cryptology refers to the use of code and ciphers to send messages securely (cryptography), as well as ways of breaking or weakening these codes (cryptanalysis).

The word cryptography itself comes from the greek *kryptos* and *graphein*, or 'hidden writing' [9]. It's in ancient Greece, Rome and Egypt that the concept of cryptography is thought to have been invented separately. In Greece, winding what appeared to be nonsensical letters around a rod of specific diameter revealed the message vertically down the rod [10]. In Rome, the famous Caesar Cipher was used to convey messages in the military. The Caesar Cipher is what's known as a *monoalphabetic shift cipher*. This cipher takes one letter at a time (hence *monoalphabetic*) and 'shifts' the letter a certain number along the alphabet (hence *shift*). For example, if we chose 3 to be our shift, the letter a would shift to d, b to e and so on. A message of the form

H E L L O W O R L D

with shift 5 would become

M J Q Q T B T W Q I

.

If Alice sends this to Bob, to decrypt the message all he needs is to 'shift' the encrypted message back by 5 letters.

Message	H	E	L	L	O
Encrypted	M	J	Q	Q	T
Decrypted	H	E	L	L	O

We call this shift the *key* of the cipher - if anyone knows the key, they can then encrypt and possibly decrypt that cipher.

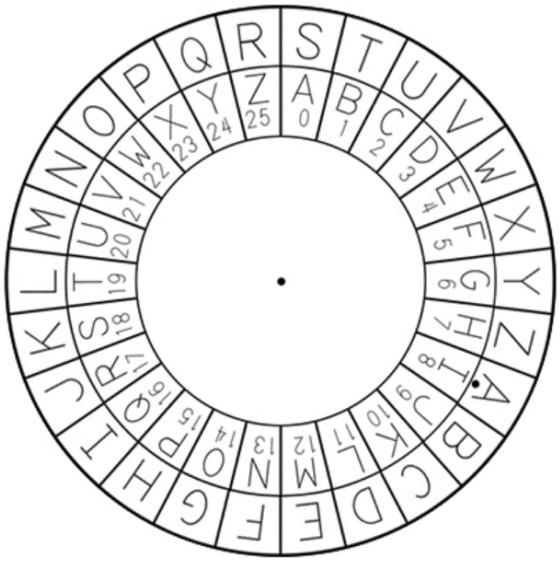


Figure 2.1: A visualisation of the caesar cipher, called the Caesar Cipher Wheel. With this tool it is possible to encrypt and decrypt from one wheel to another if the shift is known [11].

An adversary can try to learn the message by finding the key and using that to decrypt it, or by encrypting something they think might be similar to the original message.

Today cryptography has far advanced past monoalphabetic shift ciphers, as has cryptanalysis. Cryptography pervades every part of our online lives, from sending and receiving encrypted messages to entering bank details on an online shop. Governments and businesses are continuously researching new methods of cryptanalysis and vulnerabilities are being discovered every day.

Security

To give concrete notions of how secure a cryptographic protocol is, we can use Big-O notation to measure how long it takes to 'break' that problem (Appendix A). It's also useful to think about under what types of attack a protocol remains unbreakable, or what resources an adversary might have access to. For example, if an adversary has access to an oracle (Appendix A) that takes messages into ciphertexts without the

adversary themselves knowing the secret key, how easy is it for the adversary to find the secret key? One useful definition is that of perfect security [12]. Let $\mathcal{M}$ be the space of all possible messages and C be the space of all possible ciphertexts.

Definition 2.1 (Perfectly Secure). An encryption/decryption scheme is perfectly secure if for all $m \in \mathcal{M}$ and $c \in C$, we have that for an adversary guessing M given C, that:

$$\text{Prob}[M = m|C = c] = \text{Prob}[M = m]$$

i.e. that knowing the ciphertext of m gives no useful information about what m might be.

Another good measure is Kerckhoffs' Principle that a cryptosystem should be secure even if everything about the system, except the key, is public knowledge [13]. This originated with Kerckhoffs' writings in *La Cryptographie Militaire*, which put forward some axioms for security of a cryptosystem [13] (translated by Fabien Petitcolas):

(1) The system must be substantially, if not mathematically, undecipherable;

(2) The system must not require secrecy and can be stolen by the enemy without causing trouble;

(3) It must be easy to communicate and retain the key without the aid of written notes, it must also be easy to change or modify the key at the discretion of the correspondents;

(4) The system ought to be compatible with telegraph communication;

(5) The system must be portable, and its use must not require more than one person;

(6) Finally, given the circumstances in which such system is applied, it must be easy to use and must neither stress the mind or require the knowledge of a long series of rules.

Symmetric and Asymmetric Cryptography

Symmetric cryptography refers to a scheme in which Alice and Bob share the same secret key, used to both encrypt and decrypt data together. However, deciding upon this key can be risky if they don't have a secure channel of communication to start with. A good example is the one-time pad, which is perfectly secure provided Eve doesn't have access to the key. If Eve does access the key, the cipher is immediately compromised.

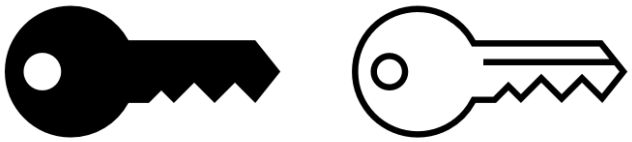


Figure 2.2: Now both Alice and Bob each have two keys: a public key and a private key.

Asymmetric cryptography or public-key cryptography refers to a scheme in which there is a pair of keys - one public key which could be known to anyone, and one private/secret key which is known only to one person - say Alice. The public key is used to encrypt data which can only be decrypted with the secret key. For example, Alice might generate both keys, send the public key to Bob (or anyone else that can see it). Bob can encrypt a message and send that back publicly to Alice, but without the secret key, no-one can decrypt the message. An excellent example is RSA [14].

The security of asymmetric cryptography often relies on the strength of a central trapdoor one-way function. A one-way function is one that is easy to compute in one direction, but very difficult to do in the other direction. A good example is multiplying primes together; it is easy to compute $3 \times 3 \times 3 \times 2 \times 5 \times 7 \times 11 \times 19 = 395010$ but it is computationally difficult to see what 395010 factors into just by looking at it. A trapdoor one-way function is a one way function where the 'difficult' direction can

Algorithm 1 RSA key generation

Require: primes $p \neq q$ of bit length λ
Ensure: a public-private key pair, pk and sk
1: Compute $n = p \times q$.
2: Compute $\varphi(n) = (p-1)(q-1)$.
3: Choose e coprime to $\varphi(n)$.
4: Compute $d = e^{-1}(mod\varphi(n))$
5: Our key pair is $pk, sk = (e, n), (d, n)$.

Algorithm 2 RSA Encryption

Require: public key $pk = (e, n)$, a message $m \in \mathbb{Z}_{n-1}$
Ensure: an encrypted message c.
1: Compute $c = m^e \pmod{n}$.
2: Send c.

Algorithm 3 RSA Decryption

Require: secret key sk, encrypted message c
Ensure: a decrypted message m.
1: Compute $c^d = m \pmod{n}$.

become easy if given a vital piece of information. In RSA, this is the exponentiation of $m = c^d \pmod{n}$. This is known as the Discrete Logarithm Problem (DLP):

Definition 2.2 (Discrete Logarithm Problem). The Discrete Logarithm Problem is that for a group $\mathcal{G}$ with element $g \in \mathcal{G}$ and a,$n \in \mathbb{Z}$, given g, n and $g^a \pmod{n}$ find a.

There are several advantages of using asymmetric cryptography to establish secure lines of communication [15]. In symmetric key encryption, key distribution must be done securely - if someone knows the key shared between Alice and Bob, they can eavesdrop on all communications and even falsify their own. Further, the size of a key in symmetric cryptography tends to be much longer than one in asymmetric cryptography; for example, in the One-Time-Pad, the key needs to be at least as long as the message in order to successfully encrypt it. However in asymmetric cryptography, the key size is much smaller and can be shared more easily. Asymmetric cryptography also allows for schemes that focus on identity authentication, i.e., verifying that the sender of a message is who they claim to be. This is especially useful for modern

cryptography online, where websites that ask users to input private information need to be able to verify their identity.

However, asymmetric cryptography tends to have much slower speeds of encryption and decryption than symmetric cryptography [16], making it unsuitable for sending large messages in bulk.

A compromise between symmetric and asymmetric cryptography is using hybrid encryption, or combined encryption, which uses a mix of two cryptographic systems to create a more secure scheme. An example could be deciding on a key and sending it using asymmetric cryptography, but then using a symmetric encryption scheme to send messages with the key decided upon. A good example of this is RSA-AES, where RSA is used to share a key and a symmetric encryption scheme called AES is used to encrypt the actual message [17, 18].

Key Distribution Schemes

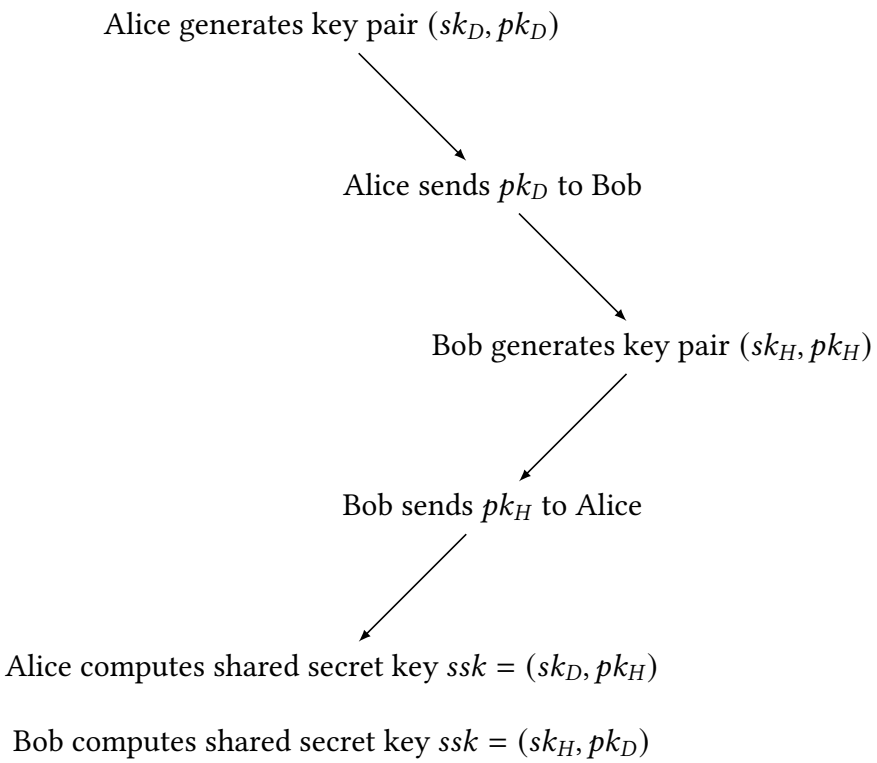
A key distribution or exchange scheme is used in symmetric cryptography. When two parties need to decide upon a key to use together to send messages, they want this key to be known by them and only them. In order to do this, they need to decide upon the key in a secure way, often using asymmetric cryptography

A typical example of a key distribution scheme is the Diffie-Hellman key exchange protocol [19]. Diffie-Hellman is an example of a non-interactive key exchange [20], "a method whereby parties who do not share any secret information can generate a shared, secret key by communicating over a public channel."

The Diffie-Hellman key exchange is, as we will see, instrumental to CSIDH and is detailed in Algorithms 4 and 5 as well as being pictured in Figure 2.3.

Both Alice and Bob would carry out Algorithm 4 with the same p and g separately, only deciding on their secret number d and h respectively.

Figure 2.3: The Diffie-Hellman key exchange.



Algorithm 4 Diffie-Hellman Key Generation

Require: a prime p and generator g of $\mathbb{Z}_1$ coprime to p.
Ensure: A public-private key pair
1: Choose a random $d \in \mathbb{Z}$
2: Compute the public key $g^d \pmod p$
3: Send the public key out

Algorithm 5 Diffie-Hellman Computation of Shared Secret Key

Require: a secret key d, a public key $g^h \pmod p$
Ensure: A shared secret key
1: Compute the secret key $ssk = (g^h)^d \pmod p$

At the end of Algorithm 5, if Alice and Bob both send each other their public keys and exponentiate those by their own secret key, they will have a shared secret key:

$$ssk = (g^h)^d \pmod p = g^{hd} \pmod p = g^{dh} \pmod p = (g^d)^h \pmod p$$

Note the commutativity of the group exponents are vital to the shared secret key being the same. Alice and Bob have shared $g^h \pmod p$ and $g^d \pmod p$ with not just each other but also with the public, but can do so without fear of their private keys d and h being found due to the Discrete Logarithm Problem.

CSIDH is a key exchange system much like Diffie-Hellman; however, instead of using group exponentiation, it utilises the group action of isogenies, or special maps, between elliptic curves. In order to introduce CSIDH properly, we need to introduce the notion of elliptic curves.

2.2 Elliptic Curves

Elliptic curves are a group of curves in 2 dimensional space that obey particular rules. These curves have particularly interesting properties and many cryptographic applications. See Figure 2.4 for an illustration.

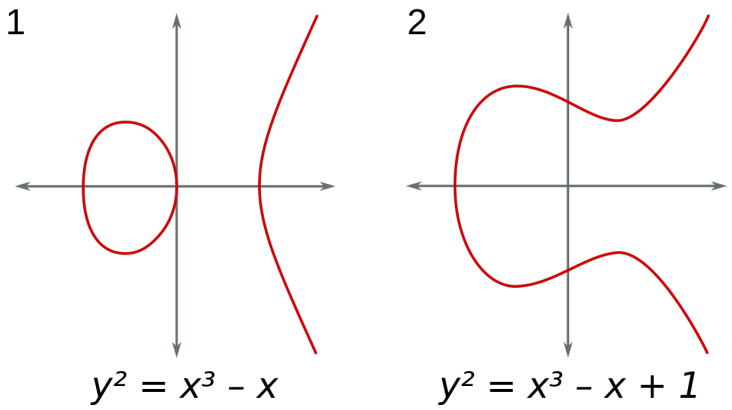


Figure 2.4: Two elliptic curves - image by GYassineMrabetTalk [21].

Elliptic curves not only contain points defined within the equations below, but also a uniquely defined *point at infinity*, O. Together with this point, we can define an elliptic curve as an abelian group with operations moving from one point to another within one well-defined curve.

Weierstrauss Curves

There are several ways to define an elliptic curve. One such widely used form is Weierstrauss form:

$$y^2 = x^3 + ax + b.$$

All elliptic curves over a field of characteristic not in $\{2, 3\}$ can be written in this form for some $a, b \in k$ where k is a field, under the condition that $4a^3 + 27b^2 \neq 0$.

Montgomery Curves

Though less widely used than Weierstrauss form, the original CSIDH paper utilises Montgomery curves for their concise representation:

$$By^2 = x^3 + Ax^2 + x.$$

Definition 2.3 (k-rational). When a point has co-ordinates in k, we call it *k-rational*. The set of all k-rational points on E is called $E(k)$.

For $B, A \in k$ we say an elliptic curve E defined over k is written E/k.

An elliptic curve has points that exist in a variety of fields. The constants of an elliptic curve can be chosen from any field k, and points on the resulting curve E are pairs of values (ϵ, η) in K^2 that solve the curve equation, where K is an extension field of k. The exception is the point of infinity.

For example, take the elliptic curve in Weierstrauss form:

$$y^2 = x^3 + x - 1$$

We have that the point $(1, -1)$ exists on the curve and is defined on any field. However, the point $(-1, \sqrt{3})$ is only defined on the curve for fields containing $\sqrt{3}$.

Groups and the J-Invariant

Notably, often the equation for a Weierstrauss curve is not unique. Two curves are known as isomorphic when they can be written as the same curve in different co-

ordinate systems. For example, the curves

$$E/Q : y^2 = x^3 + x$$

$$E^7/Q : y^2 = x^3 + 49x$$

Are isomorphic over $Q(\sqrt{7})$.

Definition 2.4 (j-invariant). We call an isomorphism class of curves a *j-invariant.* A j-invariant is defined by

$$j = \frac{1728 \times 4a^3}{4a^3 + 27b^2}.$$

All curves that have the same j-invariant are isomorphic over $\mathbb{C}$. All curves that are isomorphic over $\mathbb{C}$ have the same j-invariant. Thus there exists an equivalence relation: Two curves isomorphic over$\mathbb{C} \Leftrightarrow$ have the same j-invariant[22].

The Group Law

As stated, the set of points on an elliptic curve over some field can act as a group, and the sum of two points on a curve can be written using rational functions (fractions of polynomials) of co-ordinates.

The group law [23] states that any straight line intersects a curve at three points when accounting for multiplicity and the point at infinity. We can prove that these three points form a group, and that the sum of these three points is equal to the point at infinity.

Take two points on an elliptic curve, P and Q. Drawing a line between them will generally intersect the curve at another point, R. We take the two points P, Q and $-R$ (R reflected in the x-axis to obtain $-R$) to form 3 elements of a group, writing that $P + Q = -R$. We take the point at infinity to be the identity. The group law states that for P, Q, R all distinct and not equal to the point at infinity that:

- The identity element of the group is 0, the point at infinity
- $P + Q + R = 0$

If the line only crosses two points P and Q and the point P is at the tangent with the curve, then $P + P + Q = 0$. If the line only crosses one point P and$P$ is at the tangent with the curve, then $P + P + P = 0$.

We can write the group elements of E in a way known as projective co-ordinates. When adding rational points, rather than continuously dividing by the denominator, it makes calculations faster to store divisions until the end of the calculation, and instead write the numerator and denominator as (a, b) instead of a/b. Instead of writing a point as $(x/z, y/z)$ it will instead be written as $[x : y : z]$. The point at infinity can be written $[0 : 1 : 0]$.

One specific type of elliptic curve that CSIDH utilises is a *supersingular* elliptic curve:

Definition 2.5 (Supersingular). Let an elliptic curve E be defined over the finite field $\mathbb{K}_q$ with characteristic p. E is known as *supersingular* $\Leftrightarrow$ the kernel of the multiplication-by-p map, $E[p] = \{0\}$

A non-supersingular elliptic curve is called an *ordinary elliptic curve*. CSIDH is defined using supersingular curves over $\mathbb{F}_p$ for p prime. Not only does this increase the speed at which calculations for CSIDH can be done, it also allows a close-to-minimal choice for p for a secure key exchange, meaning that calculations made by Alice and Bob are faster, see 2.3.

Supersingular Montgomery curves are especially interesting when it comes to j-invariants: For a supersingular Montgomery curve with $B = 1$, i.e:

$$E : y^2 = x^3 + Ax^2 + x$$

The j-invariant is exactly the constant A. A uniquely defines all such elliptic curves in a given field - informally referred to as an A-invariant. We can also note with relevancy later to CSIDH that for a curve of this form, its $\mathbb{F}_p$-isomorphism class (i.e. the collection of curves isomorphic to a curve with the j-invariant A) is uniquely determined by A [2].

The Elliptic Curve Discrete Logarithm Problem (ECDLP)

The ECDLP is a variation on the Discrete Logarithm Problem, see 2.1, that showcases the uses of elliptic curves in cryptography.

Take any $n \in \mathbb{Z}$ and a curve E. We can define an automorphism P in E, where:

$$[n]P = P + P + ... + P \text{ (n times)}$$

This is the elliptic curve analogue of exponentiation in the finite field setting.

The ECDLP is the problem of computing the inverse map, $n \in Z$ from $[n]P$. For most curves if P has prime order q then the best known algorithm to get n from P has time complexity $O(\sqrt{q})$. This time complexity is achievable for any E, i.e. there are not any known ways of making use of the specific structure of an elliptic curve to speed up finding n.

The n-torsion subgroup of E is the kernel of $[n]$, meaning the set of points mapped to the point at infinity $\mathcal{O}$ under multiplication by n.

The ECDLP is widely used to ensure security in protocols today: ECDSA and EdDSA, see 2.2, both rely on the ECDLP, and Whatsapp uses a combination of several protocols in its system including Elliptic Curve Diffie-Hellman (ECDH).

Isogenies

The basic definition of an isogeny is that it is a map with particularly nice properties. We have defined elliptic curves and noted an algebraic group structure. An isogeny is a non-zero map between two elliptic curves E_1 and E_2 that preserves the properties of these curves - importantly, the group structure and the k-rationality of functions. It is usually described as a map:

$$\phi : (x, y) \longrightarrow (f(x, y), g(x, y))$$

where $f(x, y)$ and $g(x, y)$ are rational maps.

Definition 2.6 (Isogeny-Based Definition of Isomorphism). Two elliptic curves E_1 and E_2 defined over a field k are *isomorphic* [24] if there exist isogenies $\phi_1 : E_1 \longrightarrow E_2$ and $\phi_2 : E_2 \longrightarrow E_1$ whose composition is the identity.

Definition 2.7 (Degree). The *degree* of an isogeny is the smallest possible degree of a rational function that expresses the isogeny. The degree is a good measure for the computational complexity of an isogeny.

Definition 2.8 (Isogenous). Two elliptic curves are *isogenous* if there exists an isogeny between them.

Elliptic Curves in Cryptography

Schemes that rely on the ECDLP are prevalent, as mentioned. One such widely used scheme is the Elliptic Curve Diffie-Hellman key exchange (ECDH)

Elliptic Curve Diffie-Hellman

ECDH is an agreement scheme that allows two parties, each with an elliptic-curve private key and public key, to establish a shared secret key in much the same way as ordinary Diffie-Hellman [25].

Regular elliptic curve cryptography uses a property of elliptic curves similar to modular exponentiation: that is, for points on a curve we have that for $*$ point multiplication that:

$$(a * G) * b = (b * G) * a$$

for a generator G of a group, a Alice's private key and b Bob's private key.

Alice and Bob both publicly decide on a curve E and generator point on that curve G, and both secretly decide on a point a and b respectively. They then both share $G * a$ and $G * b$ respectively, publicly over the insecure channel they are using. Once Alice has received Bob's public key - $G * b$, she can multiply it by her own secret key a to get $(G * b) * a$. Bob can do the same with Alice's public key to get $(G * a) * b$. Since

$(a * G) * b = (b * G) * a$, Alice and Bob have the same shared secret key they can now use in symmetric cryptography.

Eve's problem is to find $(a * G) * b$ given E, G, $G * b$ and $G * a$. The difficulty of this attack *reduces*, see Appendix A, to the difficulty of the Elliptic Curve Discrete Logarithm Problem, a variation on the classic DLP.

It should be noted that ECDH is not the same as CSIDH; the former key exchange uses operations between points on a single elliptic curve, and the latter uses isogenies between different elliptic curves as we shall see.

2.3 Isogeny-Based Cryptography

In this section we give background information on SIDH, SIKE [26], CSIDH and give some introduction to cryptanalysis on CSIDH. TO motivate this section we refer to the points made in the initial CSIDH paper in 2018 [2]:

> Isogeny-based cryptography is a relatively new kind of elliptic-curve cryptography, whose security relies on (various incarnations of) the problem of finding an explicit isogeny between two given isogenous elliptic curves over a finite field F_q. One of the main selling points is that quantum computers do not seem to make the isogeny-finding problem substantially easier. This contrasts with regular elliptic-curve cryptography, which is based on the discrete-logarithm problem in a group and therefore falls prey to a polynomial-time quantum algorithm designed by Shor in 1994.

Isogeny-based cryptosystems do not rely on the Discrete Logarithm Problem and so are not vulnerable to the same pitfalls the DLP encounters when pitted against quantum computers, see Chapter 3.

Couveignes-Rostovtsev-Stolbunov (CRS)

CSIDH has its genesis in an isogeny-based cryptosystem first discovered in 1997 by Couveignes [27] and independently rediscovered by Rostovstev and Stolbunov [28] in 2004. Both papers were published in 2006. The protocol described is a non-interactive key exchange that relies upon ideal class groups.

In CRS, the space of public keys is the set of $\mathbb{F}_q$-isomorphism classes of ordinary elliptic curves over $\mathbb{F}_q$, where the endomorphism ring L of $\mathbb{F}_q$ is a given order $\mathcal{O}$ in an imaginary quadratic field and has a defined trace of Frobenius (in CSIDH, the trace of Frobenius becomes zero).

The ideal-class group $Cl(\mathcal{O})$ of the set of isomorphism classes of elliptic curves over $\mathbb{F}_{\text{ll}}$ is the key to constructing a key exchange.This ideal-class group (see Appendix A) $Cl(\mathcal{O})$ acts via isogenies on the set of elliptic curves with an $\mathbb{F}_q$ rational endomorphism ring $\mathcal{O}$.

In this way, the isogenies of $Cl(\mathcal{O})$ act much like elements of a finite field do in Diffie-Hellman; the order in which the elements are composed together does not matter. In this way, two parties can each compute a public-private key pair composed of: (S]E, S), a mutual curve decided on publicly together acted upon by a secret isogeny and the secret isogeny itself.

SIDH and SIKE are a Diffie-Hellman key exchange and key encapsulation method respectively that utilise isogenies on elliptic curve. While CSIDH and SIDH share much of the same terminology, CSIDH is based on CRS while SIDH is completely separate from the two.

CSIDH

CSIDH is a variation on the CRS method for key exchange that uses *supersingular* elliptic curves defined over a prime field $\mathbb{F}_p$. We now consider the set only of $\mathbb{F}_p$-rational endomorphisms, which is still an order $\mathcal{O}$ in an imaginary quadratic field. $Cl(\mathcal{O})$ acts commutatively via isogenies on the set of $\mathbb{F}_p$-isomorphism classes of elliptic curves

with the $\mathbb{F}_p$-rational endomorphism ring isomorphic to $\mathcal{O}$ (written as $\mathcal{E}_p(\mathcal{O})$).

$$Cl(\mathcal{O}) \times \mathcal{E}_p(\mathcal{O}) \to \mathcal{E}_p(\mathcal{O})$$
$$([\mathfrak{A}], E) \mapsto [\mathfrak{A}]E$$

Here the isogeny $[\mathfrak{A}]$ acts upon the curve E.

The paper was first published in 2018 [2], missing out on submission to the National Institute of Standards and Technology's Post Quantum Standardisation Process. This process hopes to reveal and decide on which supposedly 'quantum-proof' protocols will be used for key exchanges and signatures in the future, preparing for the genesis of quantum computers, see Chapter 3.

CSIDH's advantage over CRS resides for the most part in its greater efficiency.The original paper [2] estimates that at conjectured 64-bit post-quantum security, CSIDH is over 2000 times faster than CRS. At 80 milliseconds, this is still 10 times slower than the optimised SIKE implementation submitted to NIST's Post-Quantum Cryptography Standardization process [26, 29]. However, this speed is still very practical for use as a replacement for Diffie-Hellman key exchange. Other advantages include the possibilty of reusing keys; CSIDH can easily validate public keys, meaning that one party can securely use a key more than once when their identity has been verified. CSIDH's advantages over other schemes submitted to NIST's Post-Quantum Cryptography Standardization process include its reliance on only one mathematical problem, namely the problem of finding an isogeny between two given isogenous curves.

CSIDH's background with ideal class groups is similar to CRS. However, unlike CRS, the trace of Frobenius is always zero since curves used in CSIDH are supersingular. This gives the size of the ideal class group acting on a curve E as equal to approximately $2\sqrt{p}$, where p is the constant in $\mathbb{F}_p$.

In order to create our isogenies for CSIDH, it is necessary to define the Frobenius endomorphism in terms of supersingular curves.

Definition 2.9. The Frobenius endomorphism π of a supersingular curve E defined over a field $\mathbb{F}_p$ obeys the following characteristic equation in $End_p(E)$, where $End_p(E)$

is the subring of the endomorphism ring $End(E)$ consisting of endomorphisms defined over $\mathbb{F}_p$:

$$\pi^2 + p = 0$$

We write $\mathcal{O} = Z[\pi]$ is the $\mathbb{F}_p$-rational endomorphism ring, since $\mathcal{O}$ always contains the Frobenius endomorphism π. The isogeny is defined between the curve E and $E(A)$, where A is an invertible ideal of the order $\mathcal{O}$.

CSIDH Keys

The speed of CSIDH's key exchange relies on the small size of its public keys.

When written in Montgomery form:

$$y^2 = x^3 + Ax^2 + x$$

all supersingular elliptic curve isomorphism classes in $\mathbb{F}_p$ are uniquely identified by one number, A. Thus in order to receive a public key, all that is needed is A which can then be plugged into $y^2 = x^3 + Ax^2 + x$ to check for supersingularity. Since A is small, the key exchange is considerably faster than what would be otherwise used with a different elliptic curve form.

CSIDH Algorithm

Using the paper the key exchange is split into algorithm 6 and algorithm 7. Note the identical structure to Diffie-Hellman:

2.4 Classical Security of CSIDH

While developed as a post-quantum scheme, it is necessary for CSIDH to be secure under classical attacks as well. There are a few well-known attacks to find the secret key of CSIDH.

Algorithm 6 Key generation

Require: **(1)** large prime $p = 4 \cdot l_1 \cdots l_n - 1$. Here the l_i are small distinct primes greater than 2.

(2) supersingular elliptic curve $E_0 : y^2 = x^3 + x$ defined over $mathbbF_p$

(3) endomorphism ring $\mathcal{O} = \mathbb{Z}[\pi]$, created from $\mathbb{F}_p$

Ensure: a public-private key pair $(A, \mathfrak{A})$

1: Randomly sample n integers, $(e_1, ..., e_n)$ from a range $-m, ..., m$.

2: Use these integers to represent the ideal class:

$$[\mathfrak{A}]] = [\mathfrak{l}_1^{e_1}, ..., \mathfrak{l}_1^{e_n}] \in Cl(\mathcal{O})$$

where $\mathfrak{l}_i = (l_i, \pi - 1)$. $\mathfrak{A}$ is the private key.

3: Calculate $[\mathfrak{A}]E_0 : y^2 = x^3 + Ax + x$ by applying the action of $\mathfrak{A}$ to E.

4: Find the Montgomery coefficient $A \in \mathbb{F}_p$ of the elliptic curve $[\mathfrak{A}]E_0$. This is the public key.

Algorithm 7 Key exchange

Require: **(1)** Alice's public-private key pair $(A, \mathfrak{A})$

(2) Bob's public-private key pair $(B, \mathfrak{B}$

Ensure: a shared secret key, the curve $E_S = [\mathfrak{A}][\mathfrak{B}]E_0$

1: Alice generates sends her public key A to Bob over a public channel.

2: Bob also sends *his* public key B to Alice over a public channel.

3: Using Bob's key B, Alice verifies the Elliptic curve $E_B : y^2 = x^3 + Bx + x$ is in the isogeny class $Ell_p(\mathcal{O}, \pi)$ using an algorithm to verify supersingularity.

4: Alice then computes the shared secret key by applying the action of her secret key to the curve E_B:

$$[\mathfrak{A}]E_B = [\mathfrak{A}][\mathfrak{B}]E_0$$

5: Bob repeats the previous two steps (obtaining E_A) with his own secret key and Alice's public key and returns:

$$[\mathfrak{B}]E_A = [\mathfrak{B}][\mathfrak{A}]E_0$$

6: The shared secret is the Montgomery coefficient S of the common secret curve

$$[\mathfrak{A}][\mathfrak{B}]E_0 = [\mathfrak{B}][\mathfrak{A}]E_0$$

due to the commutativity of $\mathrm{cl}(\mathcal{O})$, with the curve written as

$$E_S : y^2 = x^3 + Sx^2 + x$$

.

Brute Force/Exhaustive Key Search

One method of finding the secret curve E_S in CSIDH implementation is to search through the group of all possible curves.Another would be to search through all possible isogenies. These methods could both be conducted through a simple brute force attack or marginally better with a meet-in-the-middle key search.

Pohlig-Hellman

The collection of Pohlig-Hellman style attacks would rely on the set of elliptic curves acting as a group; the curves used in CSIDH are not isomorphic to a group with 'efficiently computable operations'. There is no group linking the curves that we can compare with the action of the ideal class group. Hence there is no way of using Pohlig-Hellman style attacks on CSIDH.

Chapter 3

Background: an Introduction to Quantum Computing

Quantum computing began in 1980 [30] with Paul Benioff. The mysterious and often contradictory area of quantum mechanics could now be applied to computer science by replacing a typical 'bit' with a quantum 'qubit', a bit existing in a superposition of 0 and 1.

This idea of a superposition of bits allowed for new areas of mathematics to be born - quantum information theory and quantum computing. While large-scale quantum computers are still a long way off, the theory has been robustly studied for decades [31].Quantum effects, including superposition, the ability to 'entangle' and to 'teleport' qubits leads quantum computers to possess many interesting abilities. One of those is speeding up the solving of certain problems. The Discrete Logarithm Problem mentioned earlier in Section 2.1 is the backbone of the security of RSA/Diffie Hellman/ECDSA and has a subexponential classical time complexity of RTIME($O(2^{(\sqrt{\log q \log\log q})})$) (RTIME refers to algorithms which may use random numbers in their processing. using the index calculus method [32]). Shor's algorithm, published in 1995 [1] details a way of using quantum computation to greatly reduce the time taken to find the secret keys used in asymmetric cryptosystems that rely on

the DLP. It can break the DLP in polynomial time, see Appendix A, by using a quantum computer.

Quantum computers are still a way off from being able to break RSA and AES used widely today [33]; the largest verified quantum computer is 127 qubits [34], a far cry from the 20 million needed to break 2048-bit RSA [35]. However, it is certain that within the next two decades that cybersecurity will need to adapt to this new challenge. It is expected that quantum computers will render much of existing cryptography useless, such as RSA, Diffie-Hellman and typical elliptic curve cryptography. This means that much online traffic would become vulnerable to attack [36].

In response, there is a race to discover new methods of protecting existing algorithms against quantum computers, but more importantly, switching to algorithms that are known to be quantum safe, i.e., an algorithm where there is no known efficient (polynomial time) method a quantum computer can use to break it. Luckily, quantum computers are not useful for everything; lattice-based cryptography and supersingular elliptic curve isogeny cryptography, which this paper looks at in detail (see Chapter 2), are quantum resistant and cannot be broken quickly using Shor's Algorithm [37].

3.1 Quantum Information Theory

Before introducing how breaking CSIDH might work, it is necessary to define the relevant terms used in quantum information theory and computation. The notation used for quantum information theory largely follows the Dirac notation [38] widely used elsewhere in quantum mechanics. Dirac notation writes Hilbert space vectors as 'kets', $|\rangle$. The vector ϕ is written as $|\phi\rangle$

An n-dimensional vector space defined over a field $\mathbb{K}$ is a span of basis vectors $|v_1\rangle, ..., |v_n\rangle$ with coefficients in $\mathbb{K}$. Any vector in this vector space can be written:

$$|v\rangle = \sum_{j=0}^{n} a_j \left|v_j\right\rangle$$

where $a_j \in \mathbb{K} \text{and} \sum_{j=0}^{n} |a_j|^2 = 1$.

In general, the set of all possible states of a system is called the *state space* of a system. In quantum mechanics, the state space is the two-dimensional complex vector space where all vectors are unit length. The vectors used to represent $|0\rangle$ and $|1\rangle$ must be linearly independent. Often $|0\rangle$ and $|1\rangle$ are written in the computational basis form of:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

and

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

If using the computational basis we could write:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$$

.

Definition 3.1 (qubit). A *qubit* or 'quantum bit' is a bit that exists in a superposition of two states: $|0\rangle$ and $|1\rangle$. Any qubit takes the form

$$\alpha |0\rangle + \beta |1\rangle$$

such that $\alpha, \beta \in \mathbb{C}$ are components of $|v\rangle$ in the basis $|0\rangle, |1\rangle$ and $|a|^2 + |b|^2 = 1$.

Bras and Kets

In writing vectors this way, Dirac notation allows many other structures to be written neatly. The complex conjugate of a Hilbert space vector $|\Psi\rangle$ is written as $|\Psi\rangle^\dagger = \langle\Psi|$. In vector form, this is the complex conjugate transpose.

Inner Product

The inner product of two vectors $|\Psi\rangle$ and $|\Phi\rangle$ is written $\langle\Psi|\Phi\rangle$. The inner product defined on pairs of vectors $(|\Psi\rangle, |\Phi\rangle)$,in a complex vector space $\mathbb{C}^2$ satisfies the properties:

- $\langle\Psi|\Psi\rangle$ is real.
- $\langle\Psi|\Phi\rangle = \overline{\langle\Phi|\Psi\rangle}$.
- $\langle\Psi| (a|\Phi\rangle + b|\Omega\rangle) = a\langle\Psi|\Phi\rangle + b\langle\Psi|\Omega\rangle$.

Two vectors $|\Psi\rangle$ and $|\Phi\rangle$ are orthogonal if $\langle\Psi|\Phi\rangle = 0$. Quantum states are also normalised, i.e., $\langle\Psi|\Psi\rangle = 1$. Since $|0\rangle = (1, 0)$ and $|1\rangle = (0, 1)$, we have that $\langle 0|1\rangle = 0 = \langle 1|0\rangle$ and $\langle 0|0\rangle = 1 = \langle 1|1\rangle$.

Operators

The *outer product* $|\Psi\rangle\langle\Phi|$, represents matrix multiplication in vector form. An operator is written as a sum of outer products $\sum_{i,j=0}^{n} a|\Psi_i\rangle\langle\Psi_j| = 1$ where $a \in \mathbb{C}$ and $|\Psi_i\rangle$ is a basis vector. Operators act upon vectors to return other vectors.

For example, take the operator X.It can be written as $X = |1\rangle\langle 0| + |0\rangle\langle 1|$. In this way, X acts on $|0\rangle$ and $|1\rangle$ as follows:

$$X|0\rangle = (|1\rangle\langle 0| + |0\rangle\langle 1|)|0\rangle$$
$$X|1\rangle = (|1\rangle\langle 0| + |0\rangle\langle 1|)|1\rangle$$

Which can be written:

$$X|0\rangle = |1\rangle\langle 0|0\rangle + |0\rangle\langle 1|0\rangle$$
$$X|1\rangle = |1\rangle\langle 0|1\rangle + |0\rangle\langle 1|1\rangle$$

This simplifies to:

$$X|0\rangle = |1\rangle\langle 0|0\rangle = |1\rangle$$
$$X|1\rangle = |0\rangle\langle 1|1\rangle = |0\rangle$$

X can also be written in matrix form as a 2×2 matrix in the basis $|0\rangle$ and $|1\rangle$. We obtain:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Such that

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

and

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$$

.

It is important to note that scalars act commutatively and associatively on these bras and kets.

Measurement

Measuring a qubit changes the state of a qubit. It forces the superposition to collapse down into one state, either $|0\rangle$ or $|1\rangle$ for a measurement in the computational basis. The state $|v\rangle = \alpha|0\rangle + \beta|1\rangle$ measured using $\{|0\rangle, |1\rangle\}$ returns $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$.

This behaviour of collapse is crucial to quantum mechanics. If $|v\rangle$ is measured and returns $|0\rangle$, it becomes $|0\rangle$. If measured again it will return $|0\rangle$ every time with probability 1, even if α was not equal to one. Any 'memory' of the previous superposition of states is lost, making it impossible to know the exact state before measurement unless the original state was itself $|0\rangle$. In this case, $\alpha = 1$ and so $\beta = 0$.

Measurement of a state is made with respect to a basis. In this way, the notion of superposition of two states is basis dependent. For example, take the bases

$$B_1 = \{|0\rangle, |1\rangle\}$$
$$B_2 = \{\alpha|0\rangle + \beta|1\rangle, \overline{\beta}|0\rangle - \overline{\alpha}|1\rangle\}$$

It can be confirmed that both of these bases are orthonormal and therefore valid bases for a qubit. However, $\alpha|0\rangle + \beta|1\rangle$ is not a superposition of states with respect to B_2 - it is simply the first state in the basis.

When measuring $|v\rangle = \alpha|0\rangle + \beta|1\rangle$ using these two bases, it is clear to see that when using B_1, the measurement will return $|0\rangle$ or $|1\rangle$ probabilistically depending on the values of α and β. However, when using B_2, the outcome will always be $\alpha|0\rangle + \beta|1\rangle$. Throughout this paper, references to measurement should be taken as using the computational basis $\{|0\rangle, |1\rangle\}$.

The inherent 'probability' of returning $|0\rangle$ or $|1\rangle$ from $|v\rangle = \alpha|0\rangle + \beta|1\rangle$ is an axiom of quantum mechanics [39]):

> It is not derivable from other physical principles; rather, it is derived from empirical observation of experiments with measuring devices.

In other words, it is not possible to simply 'look' at a qubit and exactly determine α and β. It is only possible to measure several qubits that are known to have the same state $|v\rangle$ and from each measurement, roughly determine the values for α and β by how often the state collapses into $|0\rangle$ and $|1\rangle$.

What is interesting to note is that though a qubit could take one of an infinite number of states through the values of α and β, it is not possible to access this and convert it into a huge amount of classical information; any measurement of the qubit results in either $|0\rangle$ or $|1\rangle$, and since the state has now changed the original qubit cannot be measured again:

> A single measurement yields at most 1 bit of information [39].

Multi-Party States

One can consider systems with multiple quantum particles, known as multi-party systems. The state space of a quantum system grows exponentially with the number of particles. However its difference to classical computers lies in the way in which

quantum particles interact with one another, in a phenomenon known as *entangling*. Before we define entanglement, we need to define how quantum states can be combined.

The tensor product $\otimes$ is used to combine two vector spaces. For example, the vector space $\mathbb{C}^2 \otimes \mathbb{C}^2$ is spanned by the tensor product of all combinations of $|0\rangle$ and $|1\rangle$:

$$|0\rangle \otimes |0\rangle = |00\rangle$$
$$|0\rangle \otimes |1\rangle = |01\rangle$$
$$|1\rangle \otimes |0\rangle = |10\rangle$$
$$|1\rangle \otimes |1\rangle = |11\rangle .$$

Definition 3.2 (Entangled state). An entangled state consists of two or more qubits that cannot be written as a tensor product of its constituent single qubit states.

For an example of an entangled state, take:

$$|\Psi_1\rangle = \frac{|00\rangle - |11\rangle)}{2}$$

If $|\Psi_1\rangle$ is not entangled, then it can be written as a tensor product $\otimes$ of two single qubit states, for example:

$$[a_0 |0\rangle + a_1 |1\rangle] \otimes [b_0 |0\rangle + b_1 |1\rangle]$$

This is equal to

$$a_0 b_0 |00\rangle + a_0 b_1 |01\rangle + a_1 b_0 |10\rangle + a_1 b_1 |11\rangle$$

.

For this to equal $|\Psi_1\rangle$, it must be that since $|01\rangle$ is not in $|\Psi_1\rangle$, then $a_0 b_1 |01\rangle = 0$, i.e. that $a_0 = 0$ or $b_1 = 0$ or both. However, $|00\rangle \neq 0$ in $|\Psi_1\rangle$, so $a_0 \neq 0$. Also, $|11\rangle \neq 0$ in $|\Psi_1\rangle$, so $b_1 \neq 0$. This is a contradiction and so $|\Psi_1\rangle$ cannot be written in product form, meaning it is entangled.

No Cloning Theorem

The no cloning theorem describes an important feature of quantum mechanics. It states that for any unknown state $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, it is not possible to copy or 'clone' this state onto another system.

Quantum Teleportation

It is possible to send unknown states elsewhere without copying or knowing them. Say two parties - returning to Alice and Bob introduced in chapter 2 - share a maximally entangled state (meaning the probability for any measurement is the same) and Alice wants to teleport $|\Psi\rangle$ to Bob, meaning Alice wants to transmit her particle to Bob (losing it in the process). Alice can measure an operator on her state space. Depending on her outcome, she knows what Bob's state will become. She can then send a classical message to Bob to perform an operator on his state to get $|\Psi\rangle$ returned.

3.2 Quantum Computation

In order to introduce quantum computation, it is necessary to move the use of qubits into a more computational environment. Consider a classical logic 'gate'.

Definition 3.3 (Logic Gate). A classical *logic gate* is a device or function that computes an elementary logic function. Examples include AND, OR, NAND. It takes from 2 to 8 inputs and often returns 1 or 2 (for example, see Table 3.1). The two logic states, true and false, are represented by 1 and 0 in binary respectively [40].

Table 3.1: The NOT gate

Input	Output
0	1
1	0

Definition 3.4 (Quantum Logic Gate). A *quantum logic gate* is a logic gate that acts on a small number of qubits rather than classical bits. Not only can quantum gates act on multi-party states, they can also take unentangled multi-party states to entangled states.

Quantum logic gates are required to be reversible, since quantum mechanics uses unitary transformations which are always invertible.

For example, the AND gate is not reversible and has no quantum analogue due to the loss of information. Given the output 0, it is impossible to know with certainty which of the inputs was provided, see Table 3.2.

Table 3.2: The AND gate

Input	Output
00	0
01	0
10	0
11	1

Quantum gates are found in the form of operators or transformations, meaning they can often be viewed as matrices. For example, take a single-qubit gate and the quantum operation Y, applied to the qubits $|0\rangle$ and $|1\rangle$. Let $Y = i\,|1\rangle\langle 0| - i\,|0\rangle\langle 1|$. Then we have the quantum gate, see Table 3.3

Table 3.3: The Y gate

Input	Equation	Output
$\|0\rangle$	$[i\,\|1\rangle\langle 0\| - i\,\|0\rangle\langle 1\|]\,\|0\rangle$	$i\,\|1\rangle$
$\|1\rangle$	$[i\,\|1\rangle\langle 0\| - i\,\|0\rangle\langle 1\|]\,\|1\rangle$	$-i\,\|0\rangle$

Useful Gates

We describe some gates that are referred to later in the text or have some notable properties.

Pauli Gates

The following three gates are mentioned in various relevant papers, either to help in computing the CSIDH group action or to help in solving the Dihedral Hidden Subgroup Problem (DHSP) using error correction, see Section 3.3.

The Pauli Gates are a group of gates that act on one qubit. They are the gates I, X, Y and Z. I is simply the identity matrix, and X, Y and Z are defined as

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \text{ and } Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

These matrices form a basis for the vector space of Hermitian matrices with real coefficients; these matrices can be used in a linear combination to form all 2-D observables in quantum mechanics.

Clifford Gates

Clifford gates are types of gates that obey a certain property - they take Pauli operators to other Pauli operators. Together, these elements form a group called the Clifford group which is useful in quantum error correction.

The quantum Toffoli Gate

The quantum Toffoli gate, or the CCNOT gate is a reversible logic gate that acts upon three qubits as follows:

Take three qubits, $|xyz\rangle = |x\rangle\,|y\rangle\,|z\rangle$. We have that:

$$TOF\,|x\rangle\,|y\rangle\,|z\rangle = |x\rangle\,|y\rangle\,|z \oplus (x \wedge y)\rangle$$

Interestingly, the Toffoli gate is invertible despite using the classical AND operation on qubits ($\wedge$).

Table 3.4: The Toffoli gate

Input	Output
$\|000\rangle$	$\|00\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$\|111\rangle$
$\|111\rangle$	$\|110\rangle$

T-gates

The T-gate, or $\frac{\pi}{8}$ gate is another interesting gate that is instrumental in a lot of metrics used in quantum error correction. The T-gate has the matrix form:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

The T-gate is not a member of the Clifford group, but when put in a set with two elements of the Clifford group, H and $CNOT$, and the S-gate

$$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix}.$$

it forms a universal set.

Definition 3.5 (Universal Set). A *universal set* is a set of operators that can approximate any unitary matrix to an arbitrary accuracy ϵ>0 with a finite sequence of operators.

The T-gate is very useful in quantum computing. However, as it is not a Clifford group operator it is more difficult to represent on a quantum computer and so 'costs' more to create in both time and space. For this reason, the number of T-gates in any quantum circuit can be taken as the leading term for the complexity of that circuit.

Two useful metrics are born: *T-count*, which returns the total number of T-gates in the circuit; and *T-depth*, which returns the number of T-gates used in sequence for a circuit or algorithm.

The Quantum Fourier Transform

The quantum Fourier transform (QFT) is useful in a number of applications, especially in solving the Dihedral Hidden Subgroup Problem in subexponential time which we will see in chapter 4.

The QFT transform looks like this for a state space with $N = 2^n$ basis states:

$$F_{2^n} := |s\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k \in \mathbb{Z}_{2^n}} e^{\frac{\pi i k s}{2^{n-1}}} |k\rangle$$

Circuits

We use quantum circuits as a convenient way to describe a quantum algorithm.

In quantum computers, a quantum state can be initialised to $|A\rangle$ for a *bitstring* (a string of bits) A input into a circuit and then output as a bitstring in order to give information that is useful to a classical computer.

A circuit can look something like:

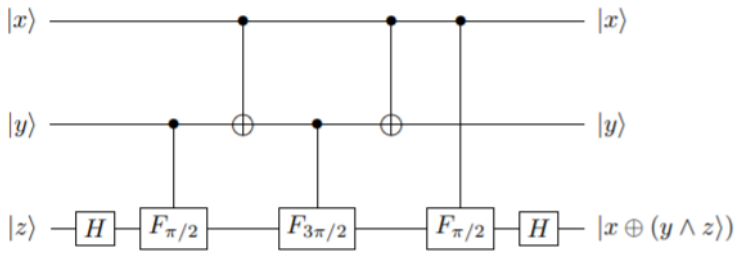


Figure 3.1: A decomposed Toffoli gate [41].

Where the horizontal lines are individual qubits which make up a multiqubit state, and various symbols represent different quantum logic gates operating on each qubit. The horizontal axis represents time, going from left to right. When a circuit finishes, a

measurement in the computational basis gives the output as a bitstring. Most gates are squares with a letter marking the name of the gate. The CNOT gate however has the symbol:

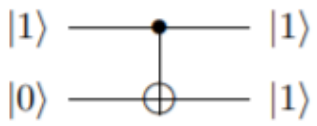


Figure 3.2: A CNOT gate [41].

3.3 Error-Correction and Surface Code

Classical and quantum computers are both subject to noise. Noise describes anything that could cause interference in a computer, usually in the form of uncontrolled electromagnetic energy. In classical computers, this noise is often of no consequence except for sensitive machinery like telescopes. However, in quantum computers noise can pose a much larger problem, as qubits are sensitive and can be degraded by many electromagnetic sources - from Wi-Fi to the earth's magnetic field.

A multiqubit system can go through *decoherence* [42], where the qubits are degraded and produce erroneous output in an algorithm. They are also prone to *dephasing* - collapsing into one of the computational basis states. This is an important problem for running quantum algorithms, as qubits can become useless extremely quickly. In order to protect against noise, quantum computers require error-correction to mitigate the effect of noise on a system.

Since errors are so rare on classical computers, error correction tends not to be of as much importance. Classical error correction can involve copying a bit several times and simply taking the majority of outcomes - *majority rule* [43]. However, this is not possible in quantum error correction; the *no-cloning theorem* means that states cannot be copied. Instead, the state is often taken to several higher dimensional entangled states, known as adding ancilla qubits.

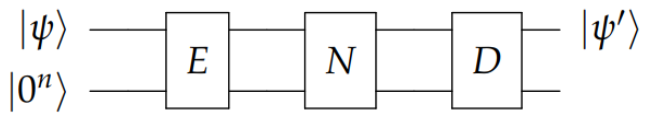


Figure 3.3: an exemplary circuit [44].

We can think of an error correcting code as a three step process:

(1) Introduce a unitary encoding operation E to our qubit $|\Phi\rangle$.

(2) An unknown operator N acts as a set of correctable 'noise' operations on the particle, randomly altering qubits in an unpredictable way.

(3) We use a unitary decoding operator D to return our corrected code.

The intention is that after a qubit is encoded, undergoes noise and is decoded, that E and D act in such a way that the input to this circuit is the same as or close to the output.

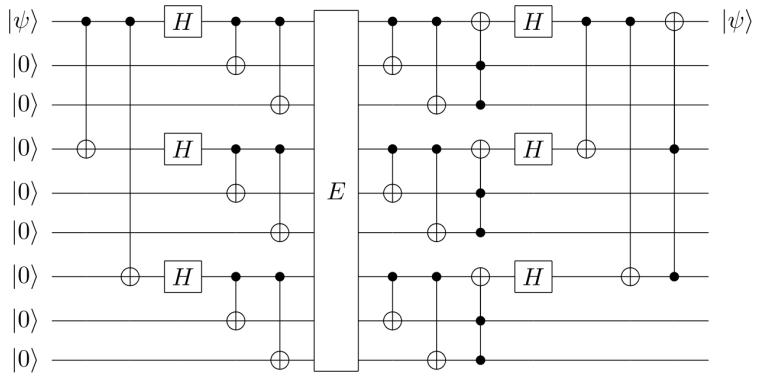


Figure 3.4: Example of ancilla qubits: the Shor code circuit [45].

A successful method of quantum error correction involves taking many physical qubits into logical qubits (groups of several qubits that are treated as one qubit) in order to stabilise the circuit against noise.

There are two possible types of quantum error correction; *passive* and *active*. Passive error correction occurs only within operations of quantum computers, whereas active

error correction produces a result from a quantum measurement and relies upon a classical computer to decide what to do next.

The Surface Code

Some of the largest areas of interest in quantum error correction are those which allow for constrained qubit movement in real situations, like superconducting qubits. An example of this is the *surface code*. The surface codes works by splitting errors into different categories: a *bit flip*, *phase flip* or a combination of the two.

The surface code is, essentially, a 2-dimensional lattice of qubits that works as an error-correcting code [46] [47]. In the surface code, it takes a minimum of thirteen physical qubits to implement a single logical qubit, a notion of a qubit composed of several physical qubits that is used to better execute a purpose. A logical qubit that is usably resistant to errors is more likely to need 1000 or more physical qubits [46].

The 2-D limitation of the surface codes means that computations that are trivial in one circuit can take many more steps in a sequence of 2-D local operations, as shown above with the number of physical qubits that would make up one decent logical qubit. However, the structure of the surface code makes them very scaleable and tolerant to local errors, much more so than other methods like Bacon-Shor codes [46].

Visualise a 2-dimensional grid as in Figure 3.5. Half are *data qubits* used for storing the information we want to protect against errors and half are *measurement qubits* used to check for errors; in Figure 3.5, open circles represent data qubits whereas closed ones represent measurement qubits. The qubits are then grouped into sections of four neighboring qubits - *nearest neighbours* in the lattice. The measurement of the 'measurement qubits' are those used to stabilise and manipulate the data qubits. The surface codes helps build these into logical qubits with increased fault tolerance.

These neighbouring qubits must be able to undergo initialization, single-qubit rotations, *CNOT* and *SWAP* operations between neighbours.

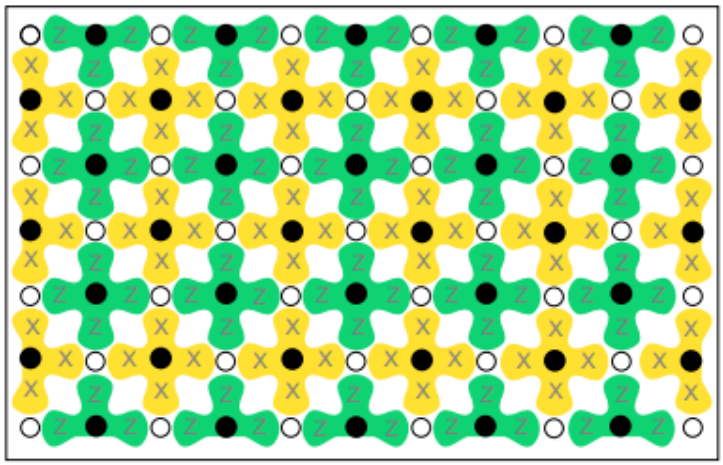


Figure 3.5: An example of a typical surface code, [46].

Each group of nearest neighbour qubits is associated with stabiliser measurement qubits, either a *Z-stabiliser* or an *X stabiliser*. These stabiliser measurement qubits are qubits that 'stabilise' the code; they detect and actively correct errors if necessary

A *measure-Z qubit* measures a Z-stabiliser, since it forces its nearby data qubits into Z-eigenstates - similarly, a *measure-X qubit* measuring an X-stabiliser. Each data qubit is coupled with 2 measure-X and 2 measure-Z qubits, and each measurement qubit is coupled with 4 data qubits.

Errors in the surface code scheme can be modelled by introducing random X bit flip and Z phase flip operators in qubit states. A bit flip is modelled using the X operator as the X operator swaps the computational basis over:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Likewise a phase flip is modelled with a Z operator as it changes the phase (sign) of elements in the computational basis:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Just these two operators can describe quite a wide range of single-qubit errors. Since clearly a Z phase flip. can be undone by applying Z once more (and similarly with

X), it appears that errors can be undone by quantum correction gates. If an erroneous bit flip is detected, it can be combatted by applying X or Z depending on the type. However, doing so without surety that the operator will not make a mistake might introduce more errors in the surface code. Instead, the error location is passed to a classical computer that simply remembers to flip the sign of the result for all subsequent computations. If a phase flip is measured, the classical computer will change the sign of every measurement of that data qubit's two adjacent measure-X qubits. Likewise if a bit flip is measured, subsequent measurements of the data qubit's two adjacent measure-Z qubits will be altered.

Something interesting to note is that for the surface code, not all errors detected need to be corrected. Active correction with a classical system only occurs when an error would affect measurement outcomes.

A Z error that is detected immediately can be corrected by changing the sign of any subsequent X measurements, whereas an X error will have no effect on the same X measurement but would have an effect on a later Z measurement. Thus if errors are located quickly, active error correction can occur using a classical system. For this reason, a lot of interest in the surface code is in error detection rather than error correction.

The difficulty in detecting errors lies in the measurement-collapse feature of quantum mechanics. When detecting errors, it is necessary to measure both X and Z on the qubit. Measuring any state in a sequence of these would destroy the state. Measuring a state that is not already in a Z-eigenstate with Z would return 1 or -1 and lose all previous information about the state; following it with an X measurement would then randomly return 1 or -1.

However, this issue can be avoided by measuring more than one qubit at a time, introducing the possibility of non-destructive error correction.

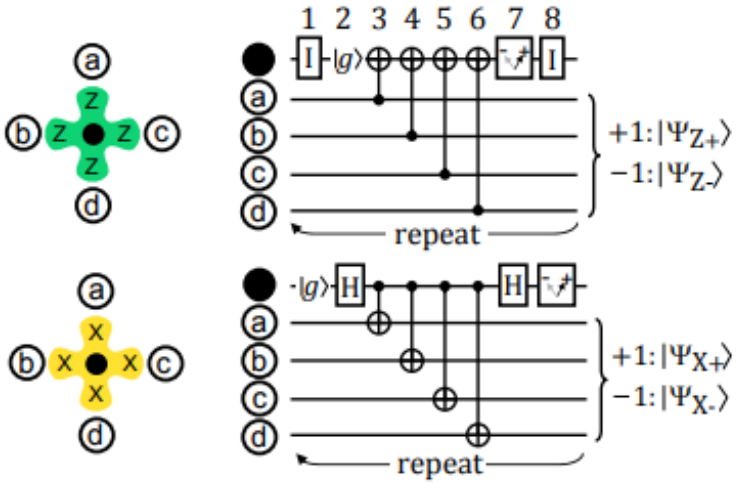


Figure 3.6: The cycle for a measure-Z and measure-X qubit stabilisation respectively, also known as *Stabiliser Codes* [46].

How To Detect Errors

Take a measure-Z qubit as in Figure 3.6. It can be used to force its neighbouring data qubits a,b,c,d into eigenstates of Z, making all of them together into an eigenstate of the operator product $Z_aZ_bZ_cZ_d$. Likewise, X forces its neighbouring data qubits into an eigenstate of the operator product $X_aX_bX_cX_d$.

In order to find an error it is important to note that stabiliser codes operate not based on the ground state (e.g. all qubits are equal $|0\rangle$) to detect errors, but the state that results from measuring all the stabilisers at the same time. These are known as *quiescent states* and there are many possibilities for which forms these can take.

These are often chosen by one iteration of the cycle in Figure 3.6 with all qubits in their ground state. Once chosen, this quiescent state is not disturbed by the cycles of Z and X stabilisers unless an error occurs. For example, if we obtain a quiescent state $|\psi\rangle$, then it becomes an eigenvector of each stabiliser with eigenvalue 1:

$$X_aX_bX_cX_d\,|\psi\rangle = \pm\,|\psi\rangle$$

$$Z_eZ_fZ_gZ_h\,|\psi\rangle = \pm\,|\psi\rangle$$

Consider a single qubit phase error on the data qubit a. We would represent this with the operator $I_a + \epsilon Z_a$, where ϵ is a number less than one representing the probability for a Z-phase flip:

$$I_a + \epsilon Z_a = (1+\epsilon)\,|0\rangle\langle 0| + (1-\epsilon)\,|1\rangle\langle 1|\,.$$

This would cause the wavefunction of the 4 qubits to become

$$|\psi\rangle \rightarrow |\psi'\rangle = (I_a + \epsilon Z_a)\,|\psi\rangle$$

What happens when we pass this through the cycle in Figure 3.6? There are two possibilities; either the error gets corrected automatically, or the error remains and is discoverable. $|\psi'\rangle$ is projected to an eigenstate of all $Z_aZ_bZ_cZ_d$ and $X_aX_bX_cX_d$ operator products.

In one case, the I_a term leads (is measured) and $|\psi'\rangle$ returns to $|\psi\rangle$ with probability $1-|\epsilon|^2$ at the end of the cycle in Figure 3.6. The error has been automatically corrected by the system.

In another case, the ϵZ_a term leads and we get the result $Z_a\,|\psi\rangle$ with probability $|\epsilon|^2$. This result is easily discoverable in the next cycle of stabilisers thanks to the fact that the erroneous data qubit is measured by both measure-Z and measure-X qubits.

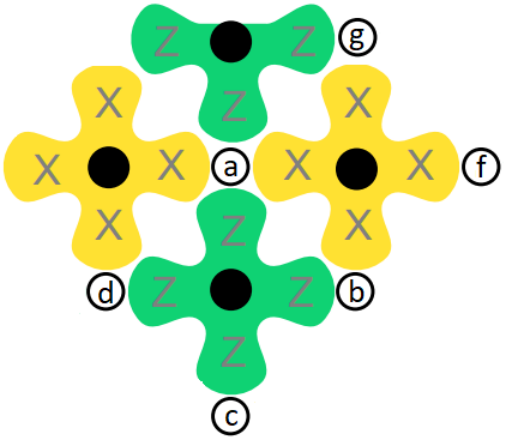


Figure 3.7: Zooming in on data qubit a [46], edited for this paper.

In Figure 3.7 the signs of the 2 measure-X qubits next to our data qubit a will flip. When commuting, X and Z operators result in a negative sign. Thus for the right hand measure-X qubits in Figure 3.7 coupled with our erroneous qubit a (with the stabiliser of the value $X_aX_bX_fX_g$) we have that:

$$\begin{aligned} X_aX_bX_fX_g(Z_a\,|\psi\rangle) &= -Z_a(X_aX_bX_fX_g\,|\psi\rangle \\ X_aX_bX_fX_g(Z_a\,|\psi\rangle) &= -(X_{abfg})Z_a\,|\psi\rangle \end{aligned} \tag{3.1}$$

Where X_{abfg} would be the returned operators from measuring the measure-X qubit. The sign flip means that $Z_a\,|\psi\rangle$ is an eigenstate of the X-stabiliser, but has a flipped eigenvalue than $|\psi\rangle$.

This flipped sign is detectable as it causes sign changes in the two measurement qubits near the data qubit a. Likewise, a bit flip will change the sign of the measure-Z outcomes adjacent to the qubit a. If the error is both a bit and phase flip (also known as a Y error since $Y = XZ$), then all 4 measurement qubits surrounding the erroneous bit will give a different result. The locations of these sign changes in measurement outcomes of the measurement qubits will reveal the location of the error. The error can be corrected using classical computer software to flip the sign of all subsequent measurements.

The Surface Code For Computation

Although it may appear that the use of the surface code makes any change to a circuit impossible, logical operators can be built on the surface code by utilising an incomplete set of stabilisers, which in turn can be used to create logical qubits.

Note Figure 3.8; the left and right incomplete X-stabilisers are called X boundaries or smooth boundaries, and the incomplete Z-stabilisers at the top and bottom of the array are called Z boundaries or rough boundaries.

An incomplete set of stabilisers increases the degrees of freedom of an array - there are more two degrees of freedom of qubits now than there are degrees of constraint. These two degrees of freedom could be used to define the array as a logical qubit.

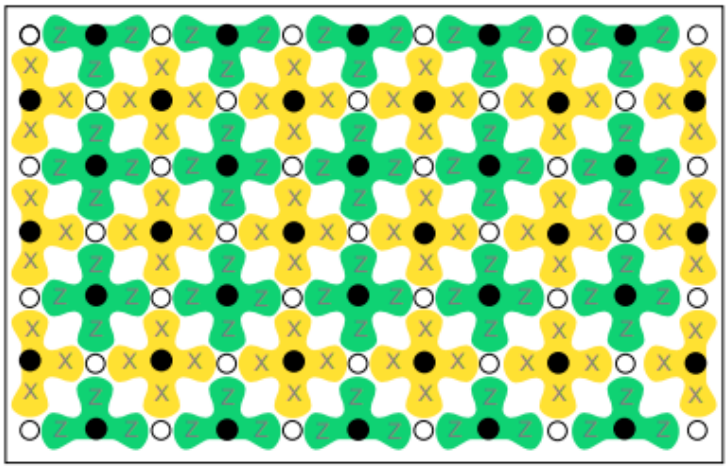


Figure 3.8: Note the incomplete stabilisers at the edges of the array [46].

It is important to note that when defining a logical operator on the array that manipulates these degrees of freedom, it cannot affect the complete stabilisers as that would negate the effects of error-correction.

We can move around this by making operations on pairs of qubits at the same time. While we know from error correction that one bit flip on a data qubit will show errors in 2 measure-Z qubits around it, we can instead consider the effect of two X operations (bit flips) on data qubits that neighbour one measure-Z qubit. Take the measure-Z qubit from Figure 3.7 surrounded by the data qubits a, b, c and d.

$$\begin{aligned} Z_a Z_b Z_c Z_d (X_a X_b |\psi\rangle) &= (-1) X_a Z_a Z_b Z_c Z_d (X_b |\psi\rangle) \\ &= (-1)^2 X_a X_b Z_a Z_b Z_c Z_d |\psi\rangle \\ &= Z_{abcd} X_a X_b |\psi\rangle \end{aligned} \tag{3.2}$$

We can see that $Z_a Z_b Z_c Z_d$ commutes with a pair $X_a X_b$. This would make the bit flips invisible to that specific measure-Z qubit, but not others due to the qubit positions.

Now consider a bit flip on the left side of the array in Figure 3.9, on the data qubit marked in red. In order for that not to be detected by its neighbouring measure-Z qubit, we must also perform a bit flip on the blue data qubit on the diagram. However, now the data qubit on the other side of other measure-Z qubit needs to be changed,

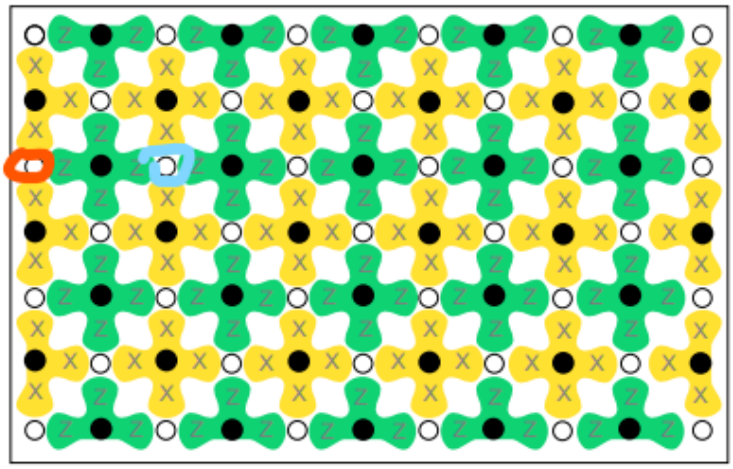


Figure 3.9: surface code with highlighted qubits. Image from [46], edited for this paper.

and so on and so on until we reach the boundary on the right side of the array, creating the line as in Figure 3.10.

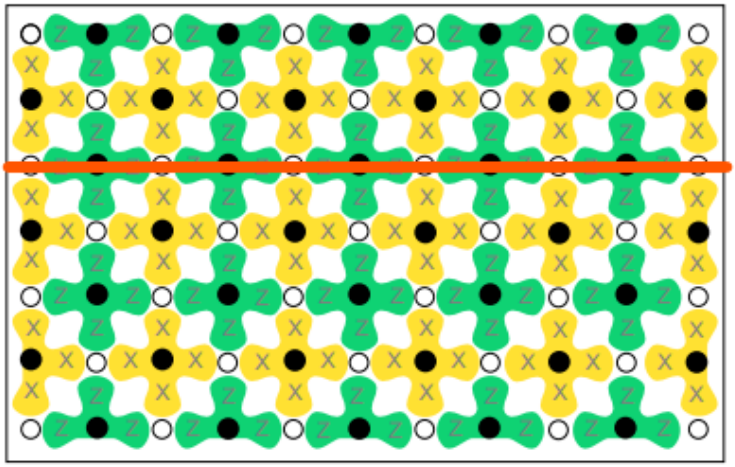


Figure 3.10: A line of qubits crossing the surface code. Image from [46], edited for this paper.

We could write this series of bit flips as a logical X operator, $X_L = X_1X_2X_3X_4X_5X_6$. Now if X_L is applied to the quiescent state $|\psi\rangle$, while the measurement outcomes of $X_L|\psi\rangle$ will be the same since no stabilisers are affected, the state itself is not equal to $|\psi\rangle$.

We could similarly create a chain of phase flips going from the top to bottom boundaries to be our Z_L logical qubit.

However, creating only one qubit no matter the size of the array is cumbersome. Instead, holes or defects are introduced as in Figure 3.11 - spaces where a measure-Z or measure-X qubit is 'turned off' and does not perform its cycle in Figure 3.6. Doing so allows us to once more utilise new degrees of freedom around this new boundary and create logical qubits in a much smaller space.

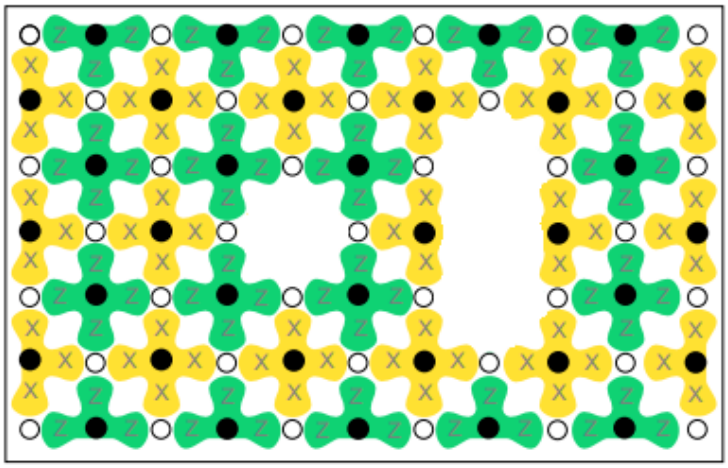


Figure 3.11: Surface code with introduced holes. From [46], edited for this paper.

We can use these and terms called *braids* to define a logical universal set and allow us to create whatever gates we like.

Magic States

In order for quantum computers to be at all useful, it is necessary that they make efficient use of magic states. Magic states are states outside of Clifford states that, when combined with Clifford states, give a universal set of operators.

A common universal set is effectively Clifford+T, the set of Clifford operators plus the T gate, where the Clifford operators are fairly easy and the T gate more difficult.

This set usually consists of the Hadamard gate H,S gate, controlled-NOT $CNOT$ gate, and the T gate, but it is easier to write these out as products of X and Z instead.

Clifford operations are relatively easy to implement in fault tolerant systems. However, non-Clifford operations are much more difficult. A magic state needs to be created in order to be used with Clifford operations to create a universal set [46]. However, this gets harder as the states get noisier and introduce errors. Magic state distillation can be used to move from several noisy states to a few purer, more fault-tolerant states [48]. This is very useful in error correction and makes up a large part of the surface code [49].

The difficulty in creating a fault-tolerant T gate (or any non-Clifford gate) is what makes surface code take up so much space; fault-tolerant Clifford gates are created in much less space and time than T gates. In this way, often when talking about the surface code, Clifford gates are treated as free and only the number of T gates is counted. Magic state distillation is wholly concerned with the goal of creating these states, and the speed of a quantum computer is led by a combination of how fast magic states can be distilled, and how fast they can be used by the computer.

In most surface code systems, the majority of the error-corrected code is split into one section for distilling the magic states and one section of qubits, the *data block*, that consumes the magic states and does the intended computation [48]. The method of parallelising these two sections to run as quickly as possible depends entirely on the algorithm being carried out.

Complexity

The error correction properties of the surface code can vary depending on the error generation rate and the types of time and space-saving features used in the surface code.

The surface codes takes up much more space than systems without this error correction, due to the space overhead involved in using logical instead of physical

qubits. Time overheads increase significantly due to the limited number of accessible logical operations. Therefore, much importance is placed upon schemes which move quantum systems into the surface code with a low space-time overhead. In fact, optimising the surface code model using the method of lattice surgery is NP-hard [50].

Some such schemes are defect based, twist based and patch based. Patch based schemes not only can be created with little surface code knowledge, they also are very simple topologically.

In order to discuss the complexity of the surface code overheads, it is necessary to discuss the design of these two blocks of magic state distillation and data block for consumption. Time-space tradeoffs of these two blocks include increasing the number of distillation blocks or implementing gates simultaneously [50].

Data blocks store the data qubits of the computation and consume magic states. We can measure the cost of using these in terms of space as *tiles* and time as *time steps*. If we want to save space, compact blocks use $1.5n + 3$ tiles for n qubits and require up to 9 time steps to consume a magic state. If we want to save time, fast blocks use $2n + \sqrt{8n} + 1$ tiles for n qubits and take only one time step per magic state [50].

However, we must also consider the cost of running magic state distillation. Daniel Litinski's 2019 paper, *Magic State Distillation: Not as Costly as You Think*, [50] states that:

> The class of magic state distillation protocols that are based on an n-qubit error-correcting code with mx X-stabilisers and k logical qubits can be implemented using $1.5(mx + k) + 4$ tiles and $n - mx$ time steps. Such protocols output k magic states.

One data block and one distillation block in reality would have T gates created and consumed one by one by the distillation and data blocks respectively. In order to speed this up, we could consider increasing the number of distillation blocks, which would in essence decrease the time taken to distill one magic state and create a T-gate. Alternatively, we can consider the utilisation of a feature of T-gates: layers of T-gates

can be arranged in such a way that gates can be executed simultaneously - down to one *T-layer* per unit time of qubit measurement. For a surface code of distance-d qubits, Joe O'Gorman and Earl T. Campbell's 2017 paper on *Quantum Computation With Realistic Magic State Factories* [48] states that:

> This results in a space-time cost for the T-gate that is only a constant multiple of a surface code overhead, namely a $O(d^2)$ spatial cost and a $O(d)$ temporal cost ... the space-time cost of a T-gate realized in a distance-d surface code is $CT(d^3)$ with $CT \approx 160 - 310$ when employing Bravyi-Haah codes.

Since Clifford gates are relatively 'cheap' compared to T-gates, they have been omitted. Since the Clifford+T gate set is universal, every circuit an be written as a combination of Clifford and T-gates. Counting the T-gates will allow us to find the overall complexity of the algorithm on surface code

It is also important to note that each gate will be comprised of logical qubits, not physical qubits. Campbell's paper [48] states that in surface code, one logical qubit would take d^2 physical qubits to create, not accounting for the extras needed to check parity for code distance d - the number of physical qubits spanning the length of the surface code.

An Evaluation of the Surface Code

As mentioned earlier, the 2D array of the surface code means that a large number of qubits are needed to make up one logical qubit - think thousands of qubits [46] when quantum computers have only reached tens of qubits in size. These qubits can be known as *overhead qubits* as they create a significant problem in the overhead for space. Another paper suggests that planar codes, a variant on the origins of the surface code, need fewer qubits for the same security as the surface code, but can only encode one qubit of information [51]. Additionally, experimental errors can occur at too frequent a rate for the surface code to be useful as a method of error correction.

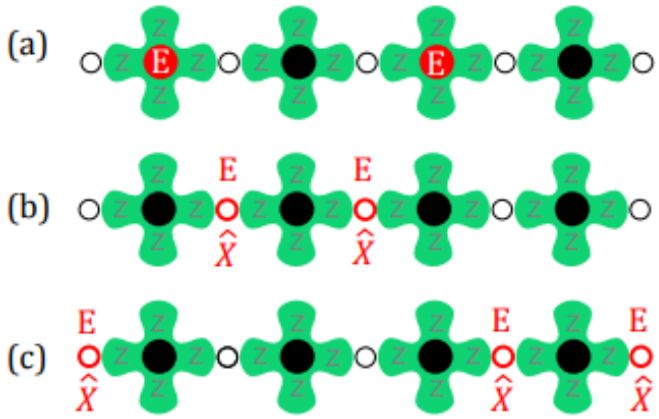


Figure 3.12: An example of an error (a) with two possible causes (b) and (c) from [46].

If errors occur frequently enough, one error result might have more than one possible cause, see Figure 3.12. If the wrong conclusions are made about which qubit cause the error, this will result in errors at the overall computation and render the entire computer useless. This expensive fault tolerance reduces how useful the surface code is for current quantum computers [52].

Some alternatives to the surface code have been suggested and are in development. One such is subsystem codes [52], where a new family called subsystem hyperbolic codes:

> ...used 4.3 times fewer physical qubits than the surface code and 5.1 times fewer qubits than current optimal subsystem codes to achieve the same physical error rate.

This subject area is only due to grow in the coming years along with quantum computers; achieving more and more qubits in one quantum computer will be useless without a robust fault-tolerant method of computation.

Chapter 4

Mathematical Basics of Security of CSIDH

The security of CSIDH relies on the difficult of finding an isogeny. This directly reduces to the Dihedral Hidden Subgroup Problem (DHSP). There are several quantum algorithms for solving variations on this problem [53, 54]. In this section we create our own, slightly modifying the algorithm used in Childs, Jao and Soukharev's 2014 paper [53] for use with CSIDH.

4.1 Kuperberg's Algorithm and Alternatives

Kuperberg's algorithm [54]gives a subexponential time quantum algorithm for the DHSP. The DHSP is as follows:

Definition 4.1 (Dihedral Hidden Subgroup Problem (DHSP)). Take G, the dihedral group D_N. We write D_N as

$$D_N = \langle x, y | x^N = y^2 = 1, yxy = 1 \rangle$$

. H is a subgroup of D_N generated by a reflection yx^s, where s is an integer in the range $\{1, N-1\}$. The problem is: for a given set S and an oracle $f : G \to S$,where

$f(a) = f(b)$ if and only if a and b are in the same right coset of H, find a generating set of H.

This subgroup H does exist by promise, but the difficulty is in defining it. When H is generated by a rotation, the hidden subgroup is easier to find; it is either 1 or has a non-trivial intersection with $C_N = \langle x \rangle$. In this case, finding $J = H \cap C_N$ is easy if given the factors of N, which can be found in subexponential time with Shor's algorithm [1]. If H is generated by a reflection, then H/J is a reflection in the quotient group G/J. Thus the problem reduces as follows:

Proposition 4.2 (Hidden slope for the DHSP). *The Dihedral Hidden Subgroup Problem of finding H in D_N reduces to finding the slope of a hidden reflection in D_N. [54]*

Since $H = \langle yx^s \rangle$, this further simplifies to finding the unknown s. Kuperberg uses this fact to create a subexponential time algorithm to find H. Kuperberg's more general algorithm from his original paper [54] is seen modified in Algorithm 8.

This method requires only $O(8\sqrt{\log_2 N})$ queries and can be conducted in quasilinear time - close to linear time.

Alternatively, Kuperberg states [54] that a conversation with Peter Hoyer in 2003 revealed an alternative method, where it is possible to recover s directly by a Quantum Fourier Transform [53]. The measured Fourier number t of these qubits reveals s by the relation:

$$\frac{t}{2^k} \sim \frac{s}{N}$$

meaning that about $O(\log N)$ computation time is saved.

Other Methods of Solving the Hidden Shift Problem

Bonnetain and Schrottenloher's 2018 paper [8] summarises two alternative methods to find hidden shifts that build upon the more general method presented. Rather than comparing pairs of qubits, one method combines k qubits in a tensor product, applying and measuring an ancilla register and projecting the result to a pair of qubits that are

Algorithm 8 Kuperberg's algorithm (edited)

Require: $f : D_N \to S$ with a hidden subgroup H that is generated by a reflection (i.e. $H = \langle yx^s \rangle$)

Ensure: H

1: Make a list L_0 of copies of the state $\rho_{D_N/H}$:

$$\rho_{D_N/H} = \frac{1}{|D_N|} \sum_a |Ha\rangle \langle Ha|$$

which forms a mixture of right cosets.

2: Extract a qubit state $|\phi_k\rangle$ from each $\rho_{D_N/H}$ using a Quantum Fourier Transform on $\mathbb{Z}_N$ and a measurement.

3: For each $0 \leq j \leq \lceil \sqrt{(log_2 N) - 2} \rceil$, we assume a list L_j of qubit states $|\phi_k\rangle$ such that $0 \leq k \leq 2^{m^2 - m(j+1)+1}$.

4: Randomly divide L_j into pairs of qubits $|\phi_k\rangle$ and $|\phi_l\rangle$ such that

$$|k - l| \leq 2^{m^2 - m(j+1)+1}$$

5: Define L_{j+1} as consisting of those qubit states of the form $\left|\phi_{|k-l|}\right\rangle$.

6: The final list L_m consists of states $|\phi_0\rangle$ and $|\phi_1\rangle$. Perform a special measurement (Ettinger-Hoyer) with different values to learn $s \in \mathbb{Z}_N$ to within $N/4$.

7: Write $N = 2^a M$ where M is odd. Then we have that:

$$C_N \cong C_{2^a} \times C_M$$

8: For each $1 \leq j \leq \lceil \sqrt{\log_2 N - 2} \rceil$, we can apply an algorithm for groups of size 2^N to obtain many $|\phi_k\rangle$ with $2^{min(a,j)} | k$, splitting down the C_{2^a} group.

9: For the C_M group, repeat steps 1-7 after applying the group automorphism $x \to x^{2-j}$ to the C_M group. This produces copies of $|\phi_{2^j}\rangle$.

10: All these observations together can be used to determine the value of s and hence determine $H = \langle yx^s \rangle$.

then mapped to $|0\rangle$ or $|1\rangle$. This returns 2^k possible sums, which are combined to obtain enough 'labels' to use in the Quantum Fourier Transform to find s.

Overall, the total number of queries here is $8\log_2(N)^2$. This method limits the number of quantum queries to be polynomial in N, and so a classical time and memory cost of $O(2^{0.291log_2(N)})$ is obtained. According to [8], CSIDH-512 with $N = 2^{256}$ can have its hidden shift s found in 2^{19} quantum queries and 2^{86} bits of classical time and memory when taking into account the probability that the Quantum Fourier Transform is successful (it fails about $1 - \frac{4}{\pi^2}$ of the time).

4.2 Applying the Hidden Subgroup Problem to CSIDH

The 2018 paper by Childs, Jao and Soukharev [53] entitled *Constructing elliptic curve isogenies in quantum subexponential time* applies Kuperberg's algorithm for the hidden shift problem to construct isogenies between two elliptic curves. This application implies that there is a subexponential-time algorithm for 'breaking' CSIDH. The algorithm used in the paper is made up of two parts: the first is solving the hidden shift problem, and the second is computing a "superposition of all isogenies originating from a given curve, which the algorithm calls as a black box" [8].

Though the paper focuses on ordinary elliptic curves, this can be applied to supersingular elliptic curve, with the only change in calculations being that the trace of Frobenius does not need to be calculated.

4.3 Construction of a Reduction to CSIDH

We create a slightly modified version of the DHSP algorithm for CSIDH and show that CSIDH can indeed be broken in quantum subexponential time.

First, the problem for an attacker with CSIDH is to find the common secret curve that Alice and Bob share:

$$[\mathfrak{A}][\mathfrak{B}]E_0$$

Recall that Alice and Bob have publicly agreed upon a curve E_0 together and then each generated their own secret isogenies, $[\mathfrak{A}]$ and $[\mathfrak{B}]$ respectively. They each then generate a new curve they share with one another publicly: Alice shares $E_A = [\mathfrak{A}]E_0$ and Bob shares $E_B = [\mathfrak{B}]E_0$. They then apply their secret isogeny to their partner's curve to obtain the shared curve $[\mathfrak{A}][\mathfrak{B}]E_0$.

The attacker has knowledge of Alice's curve E_A, Bob's curve E_B and the curve E_0. One way for an attacker to find $[\mathfrak{A}][\mathfrak{B}]E_0$ is to find an isogeny between E_0 and E_A, that is, find $[\mathfrak{A}]$. Then they could also find $[\mathfrak{B}]$ by finding the isogeny between E_0 and E_B. In this way, with knowledge of $[\mathfrak{A}]$, $[\mathfrak{B}]$ and E_0, the attacker can construct the common secret curve $[\mathfrak{A}][\mathfrak{B}]E_0$.

However, something to note is that since E_A and E_B are shared publicly, the attacker in fact only needs to compute one isogeny. If the attacker takes $E_B = [\mathfrak{B}]E_0$ and knows E_A and E_0, they only need to find one isogeny $[\mathfrak{A}]$ to compose with $[\mathfrak{B}]E_0$ to get

$$[\mathfrak{A}][\mathfrak{B}]E_0$$

as required.

In order to find an isogeny, we use the modified Childs-Jao-Soukharev Algorithm 3 [53], seen here as Algorithm 9 .

If we use the isogeny $[\mathfrak{A}]$ produced by Algorithm 9 with E_B, we get $E_S = [\mathfrak{A}][\mathfrak{B}]E_0$.

The reason this works is through the use of further algorithms listed in the paper by Childs, Jao and Soukharev [53]. Step 2 uses an algorithm by Cheung and Mosca [55] to decompose finite abelian groups. Step 3 uses the fact that $C(\mathcal{O})$ has been decomposed into a direct sum of cyclic groups to solve the hidden shift problem (equivalent to the Dihedral Hidden Subgroup Problem) for each. This uses two subexponential algorithms in [53], to find E_A such that $E_A = [\mathfrak{A}]E_0$, which in turn uses an algorithm computing a relation vector for a factor base.

Algorithm 9 Finding an isogeny

Require: A finite field $\mathbb{F}_p$, and endomorphism ring $\mathcal{O} = \mathbb{Z}[\pi]$, created from $\mathbb{F}_p$, and the isogenous supersingular elliptic curves E_0 and E_A and the set E.

Ensure: $[\mathfrak{A}]$ such that $E_A = [\mathfrak{A}]E_0$

1: Check that E_0, E_A are indeed supersingular.
2: Find Cl($\mathcal{O}$) and decompose Cl($\mathcal{O}$)= $\langle[\mathfrak{A}_1]\rangle \oplus \cdots \oplus \langle[\mathfrak{A}_k]\rangle$ where $|\langle[\mathfrak{A}_k]\rangle| = n_j$
3: Solve the hidden shift problem defined by functions $f_0, f_1 : \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k} \to \{E\}$, satisfying that $f_i(x_1, ..., x_k) = ([\mathfrak{A}_k]^{x_1} \cdots [\mathfrak{A}_k]^{x_k})E_i$ for $i \in \{0, 1\}$. This will return some $(s_1, ..., s_k) \in \mathbb{Z}_{n_1} \times \cdots \times \mathbb{Z}_{n_k}$.
4: output $[\mathfrak{A}] = [\mathfrak{A}_1]^{x_1} \cdots [\mathfrak{A}_k]^{x_k}$

Assuming the Generalised Riemann Hypothesis, the algorithm runs in time $L_q(\frac{1}{2}, \frac{\sqrt{3}}{2})$ (see Appendix A), using the fact that Kuperberg's algorithm runs with time complexity $2^{O(\sqrt{n})}$ on the Abelian Hidden Shift Problem.

Chapter 5

Quantum Cryptanalysis of CSIDH

This section will focus on the main publications describing quantum cryptanalysis of CSIDH, as well as attempting to estimate the error-corrected cost of breaking CSIDH implemented with the surface code. All use the reduction created in Chapter 4, and variants using newly discovered speed-ups for certain parts of the isogeny-finding problem.

The main method of attacking CSIDH with access to quantum computers is through Kuperberg's algorithm for the DHSP or a variant of it such as the C-sieve [56]. This makes key recovery of CSIDH into a two-step task:

(1) A quantum oracle that evaluates the group action when called on a random 'labeled' quantum state.

(2) A sieve that combines labeled states to get a more favourable state to measure and give information on the secret key.

5.1 Literature Review

Castryck, Lange, Martindale, Panny, Renes 2018

The original paper for CSIDH gave several points on its quantum cryptanalysis [2]. These all focused on the problem of finding an isogeny between the two curves E_A and E_0. Some quantum algorithms to solve the problem are ones that can be applied directly from quantum algorithms to break SIKE and SIDH.

Grover's Algorithm and Claw Finding

The original CSIDH paper briefly mentions the possibility of applying Grover's algorithm to the problem of finding the secret isogeny. Grover's algorithm is also known as the quantum search algorithm: given a set S of n-bit strings that are uniquely marked apart from the rest of the set of n-bit strings, the algorithm returns an element of S with high probability. This algorithm can be applied to the problem of finding an isogeny beween two elliptic curves; Biasse, Jao and Sankar's 2014 paper [57] conjectures a complexity of $O(p^{1/4})$ where p is the characteristic of a finite field $\mathbb{F}_q$, and quantum time complexity $L_p[1/2, \sqrt{3}/2]$ when both elliptic curves are defined over $\mathbb{F}_p$, notably subexponential.

The original paper also gives some concrete estimates using implementations of CSIDH, see Figure 5.1:

It also includes responses to the wave of cryptanalysis published after the paper's first release in a later addendum.

Bonnetain and Schrottenloher, 2018

Bonnetain and Schrottenloher [8] use a variant on Kuperberg's algorithm for arbitrary finite cyclic groups in order to give some more concrete estimates of the time complexity of finding the secret isogeny, assuming a very large amount of quantum memory is available. Their estimations return the necessary number of bits in a quantum computer

CSIDH-log p	classical $\log \sqrt[4]{p}$	Regev [53] $\log L_N[1/2, \sqrt{2}]$	Kuperberg [44] $3\sqrt{\log N}$	Kuperberg [44] $1.8\sqrt{\log N}$	Table 7 in [7]	[13]-Regev $\log L_p[1/2, 3/\sqrt{2}]$	[13]-Kuperberg $\log L_p[1/2, 1/\sqrt{2}]$	Table 8 in [7]
CSIDH-512	128	62	48	29	32.5	139	47	71
CSIDH-1024	256	94	68	41	44.5	209	70	88
CSIDH-1792	448	129	90	54	57.5	288	96	104

Figure 5.1: Estimated attack complexity ignoring limits on depth from CSIDH's original publication [2]. The leftmost columns give classical and quantum query complexities whereas the rightmost three columns give overall attack costs. We take $N = \#Cl(\mathcal{O}) = O(\sqrt{p})$ and any $O(1)$ complexities to be zero.

for CSIDH at $2^{1.8\sqrt{\log N}+2.3}$ bits and around $(5\pi^2/4)2^{1.8\sqrt{\log N}}$ for $N = \#Cl(\mathcal{O}) = O(\sqrt{p})$. These numbers increase greatly for larger implementations of CSIDH and focus more on the smallest time result rather than considering depth/space. They return a time complexity lower than the first paper's estimate but a depth that may be impractical for real implementation, see 5.2. Bonnetain and Schrottenloher also notably do not include estimates for costly operations in finding the secret isogeny, according to *Quantum Circuits for the CSIDH: optimizing quantum evaluation of isogenies* by Bernstein, Lange, Martindale and Panny [58].

According to a forum post updating their paper, their most recent version "contains a 2^{19} quantum query cost using a variant of the Regev/Childs-Jao-Soukharev algorithm, with a gate cost of $2^{48} - 2^{50}$" per oracle call (the qubits used are only those of the oracle), depending on circuit optimization. This leads to an overall gate cost of $2^{67} - 2^{69}$ which is still far below the original paper's conjectured 2^{128} security, see Figure 5.2.

Bernstein, Lange, Martindale, Panny, 2018

Published in 2018, the paper *Quantum Circuits for the CSIDH: optimizing quantum evaluation of isogenies* paper [58] analyses algorithms for computing the group action

Bit-size n of p	Number M of isogenies	T_M	s	$B(n,s)$	Toffoli gates	T-gates	Ancilla qubits
512	1300	2^{20}	15	3553	$2^{49.6}$	$2^{52.4}$	$< \mathbf{40\ 000}$
1024	4000	2^{22}	10	27231	$2^{56.2}$	$2^{59.0}$	$< 60\ 000$
1024	4000	2^{22}	15	10465	$2^{54.8}$	$2^{57.6}$	$< \mathbf{80\ 000}$
1792	10 000	$2^{23.6}$	11	51953	$2^{60.1}$	$2^{62.9}$	$< \mathbf{110\ 000}$
1792	10 000	$2^{23.6}$	15	22753	$2^{58.9}$	$2^{61.7}$	$< 140\ 000$

Figure 5.2: From Bonnetain and Shrottenloher's paper [8], showing the quantum time and qubits for the class group action for the original CSIDH parameters.

via the DHSP and their cost, detailing several speedups which help give a good idea of the cost of breaking CSIDH.

Finding the attack costs for any new curve takes a significant amount of work, a large part of which is in finding the cost of each query for queries in various DHSP algorithms.

One of the paper's main results is that given a particular implementation of CSIDH, the number of nonlinear bit operations in the group action can be estimated. Further, if it takes B nonlinear bit operations to carry out the group operation,that implies at most $2B$ Toffoli gates or $14B$ T-gates could be used with probability of failure at most ϵ >0.

For any given CSIDH parameters, the paper makes it possible to find (B, e) and so realise the cost of a specific implementation in terms of nonlinear bit operations, Toffoli gates and T-gates.

The paper also builds upon the work of Bonnetain and Schrottenloher [8] who gave an estimate for breaking CSIDH with only 2^{37} quantum gates per query and a total of only 2^{71} quantum gates. The paper notes several areas where costly parts of computing the group action are overlooked. The paper also includes a simulator open to anyone to conduct their own cost calculations on [3]

Peikert, 2020

In February 2020, Peikert published a paper refuting the claimed NIST level 1 security of CSIDH with his paper *He Gives C-Sieves on the CSIDH* [7]

This paper generalised the collimation sieve seen in Figure 5.3, or *c-sieve* method created by Kuperberg that improved upon the previous hidden shift algorithm with exponentially less quantum memory. The generalisation of the sieve now works for any finite cyclic group.

The paper splits up attacking CSIDH into two parts: the first evaluating the group action with a quantum oracle, and the second a sieving procedure that combines states to generate more favourable ones. The sieve works on states generated by the oracle to create some highly favourable states which can be used to reveal useful information about the hidden shift (and so learn the secret isogenies).

In 2011 Kuperberg's new paper on the collimation sieve reduced the quantum space needed for the original Kuperberg's algorithm to linear $O(N)$ space, but also maintained exponential $O(\exp\left(\sqrt{N}\right))$ quantum time and classical space. The paper also introduced possible trade-offs for *quantumly accessible classical memory* and quantum time.

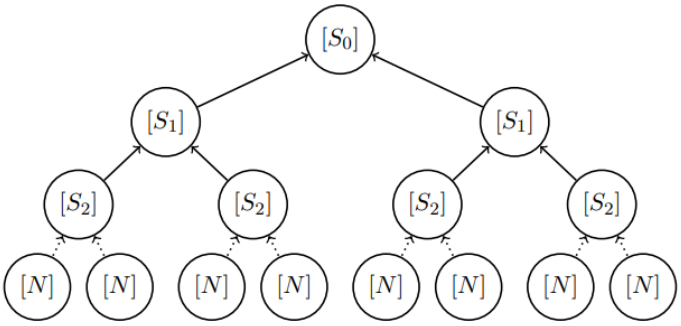


Figure 5.3: Structure of a collimation sieve [7].

Definition 5.1 (Quantumly Accessible Classical Memory). Quantumly accessible classical memory (also called QROM, Quantum Read-Only Memory [7]) is classical memory that is readable but not writeable in superposition

Peikert's contributions to the paper mostly involves the generalisation of the c-sieve to work for most CSIDH group parameters, and concretely estimates the number of queries and amount of QROM the sieve will use for different implementations. The results give both a security estimate in terms of oracle queries and bits of QROM, but also in terms of T-gates and nonlinear bit operations.

The oracle query complexity for CSIDH-512 is given to be 2^{19} oracle queries and 2^{32} bits of QROM, or alternatively for a trade-off 2^{16} oracle queries and 2^{40} bits of QROM. This improves upon Bonnetain and Schrottenloher's estimates [8] as it used QROM rather than quantum memory. It is important to note that these give 'almost' all of the secret key and so could be improved upon with slightly different methods to reveal even more of the key. In experiments on the implementation CSIDH-512, Peikert found that:

> "...the required classical resources are cryptanalytically insignificant: at most a few core-days on a commodity server with 128GB or 512GB of RAM, using only four CPU cores and less than 100GB RAM per experiment."

The T-gate complexity for an evaulation of the CSIDH-512 oracle is given as 2^{40} nonlinear bit operations, which is equivalent to $2^{40} - 2^{44}$ T-gates. When combining this with the sieve and finding the overall complexity for key recovery, the paper finds between $2^{56} - 2^{60}$ T-gates are needed, plus 2^{40} bits of QROM.

The paper notes that the sieve can close to perfectly parallelise the oracle calls and steps of the collimation sieve, meaning that the depth of the full attack is reduced to almost the depth of only the oracle. An example for one set of sieve parameters for CSIDH-512 gives T-gate complexity of the sieve as between $2^{38} - 2^{47}$, but needs at least 2^{14} oracle calls and 2^{39} T-gates; this means the complexity of the overall oracle and sieve method is not significantly when using the c-sieve than beyond just the complexity of using the oracle to get group evaluations.

> "The attack's quantum gate complexity of about $2^{56} \sim 2^{60}$ is far below the required 2^{130} for the low end of the MAXDEPTH range, and even significantly below the required 2^{74} for the high end."

However Peikert's paper has been criticised for a number of reasons. One such is the author's equivalence of 'nonlinear bit operations' and 'nonlinear qubit operations' which are not the same in the original paper. Some have criticised the assumption of low-cost access to RAM with a quantum computer, stating that the assumption that this cost is dwarfed by the oracle calls to break CSIDH is implausible and not well explained. This is refuted by the paper whose method Peikert uses to access QROM [59] and interestingly gives some information on the cost of error-correcting access to memory too.

5.2 Error Corrected Quantum Cryptanalysis of CSIDH

When considering how we can go about breaking CSIDH in 'real life' thinking about fault-tolerant algorithms and error correction is extremely important; without error correction, cryptanalysis of CSIDH using the above methods could easily give incorrect keys or return no key at all.

To think about error-corrected quantum cryptanalysis of CSIDH, we need to consider what the structure will look like if we encode a circuit for cryptanalysis of CSIDH into the surface code.

We assume that we can implement this in the surface code without too much difficulty - while the surface code might not be the most effective method of efficiently correcting errors, its scalability makes it an ideal starting point for testing real error correction of CSIDH.

Overheads

The costs of error correction on surface codes are split between that for Clifford gates and that for T-gates. Clifford gates are easy to error-correct [46] but T-gates are more difficult. We could then take the number of T-gates as a 'leading term'. Therefore we could count the overhead of computing the CSIDH group action as how many T-gates there are $\times$ how long or how difficult it is to error correct each T-gate.

We could also consider the space and time complexity of surface code error correction algorithms. Several papers have been published that attempt to reduce the time complexity of error correction on a large quantum computer. Notably, Austin Fowler's 2013 paper *Optimal Complexity Correction of Correlated Errors in the Surface Code*[60] gives an algorithm that efficiently corrects bit and phase flips in $O(1)$ time complexity given a sufficiently large quantum computer. This could make the oracle queries not much more difficult to compute with error correction included. However the resources used will greatly increase for any method of error correction in CSIDH due to the need for magic state distillation to create error-corrected T-gates.

Samuel Jaques created an interesting calculator [61] for use with his papers, using the description of the surface code from Fowler's 2012 paper [46]. This calculator can take a generic or specialised quantum algorithm (for example Shor's algorithm) to give an estimated error-corrected cost of running that algorithm, in terms of physical qubits, metres, surface code cycles,T-depth,surface code distance and number of magic state factories (sections of code used to create magic states). This extensive list of measurements gives a variety of possible metrics we can use to think about how to measure the overall cost of running cryptanalysis on CSIDH.

Overall Complexity

In the *Quantum circuits for the CSIDH* paper [58], the authors provide an example of their code run on an implementation of CSIDH, CSIDH-512. This is broken in 2^{40} non-linear bit operations on a quantum computer, or alternatively 14×2^{40} T-gates using

the paper's conjecture on the weight of a T-gate compared to an ordinary non-linear bit operation, or Peikert's estimation of between 2^{40} and 2^{44} quantum T-gates.

Peikert's 2020 paper [7] gives estimations on the security of CSIDH assuming that the user has low-cost use of QROM. The paper Peikert references for QROM [59] states that the T-gate complexity of a single collimation step can be bounded using classical memory plus just $4D$ T-gates. Further, the paper Peikert references [59]:

> "Compiling to the surface code fault tolerant gates and assuming per gate error rates of one part in a thousand reveals that one can error correct ... using only about a million superconducting qubits in a matter of hours for a realistic quantum computer."

This suggests that error correction for QROM access might be trivial and dwarfed by oracle query error correction. Further, some fault-tolerant algorithms for QROM may not require any error correction. Therefore the cost of error-corrected cryptanalysis of CSIDH using Peikert's c-sieves method [7] can be considered without the additional cost of using QROM.

Jaques's calculator [62] can provide some excellent metrics for the overall cost of breaking CSIDH-512. However, the closest method of breaking CSIDH included in the calculator is only 'generic'. This will not take into account communication across qubits and error correcting idle qubits, of which there are many in breaking CSIDH. This would waste a large amount of time, so any fault-tolerant algorithm for breaking CSIDH would need to be adapted to each implementation and avoid unnecessary error correction.

Error-corrected T-gate Complexity

As mentioned, the oracle query complexity for CSIDH-512 is 2^{16} oracle queries and 2^{40} bits of QROM. Given an optimistic cost of $2^{40} - 2^{44}$ T-gates per oracle, we have overall $2^{56} - 2^{60}$ =T-gate complexity for CSIDH-512, potentially slightly higher depending on implementation. This is the *T-count*. If we assume (plausibly) that using QROM is

perfectly parallelisable, then we can find the error corrected complexity of CSIDH-512 is simply the T-count multiplied by the cost per T-gate. We could take this as the time taken for one T-gate to operate (which takes only nanoseconds so is difficult to measure). However, since a T-gate operates on only one (logical) qubit we could find the total number of physical qubits needed for all the T-gate operations by multiplying the number of T-gates by the number of physical qubits needed for one logical qubit. This can range from 1,000 to 10,000 at the moment [46].

So we could give one estimate for the number of qubits it takes to break CSIDH as $2^{60} \times 1000 = 1.1529 \times 10^{21}$, or 2^{36} qubits. However, this does not take magic state distillation into account, which undergoes several rounds of distillation to create a fault-tolerant T-gate, using both time and qubits.

Campbell's 2017 paper [48] gives the space-time cost of a T-gate realized in a distance-d surface code as "$CT \times d^3$" where $CT = 160 - 310$. It is currently unclear how this could be applied to CSIDH to give a concrete T-gate cost, as the distance for the surface code could vary massively. However, this is an area for future research.

Error-corrected Oracle Call Complexity

As mentioned, the oracle query complexity for CSIDH-512 is 2^{16} oracle queries. The use of error-correction does not increase the number of oracle calls - however it may increase the time/resources taken for one oracle call to succeed.

Jaques's calculator

The calculator returns, for 2^{40} logical qubits, a circuit depth of 2^{40} (using NIST's standards) and 2^{64} T-gates made with perfectly parallelised magic state factories that the overall cost of breaking CSIDH-512 on a generic algorithm is as follows:

> "The circuit to compute this will have a sequential depth of $1.10e + 12$ surface code cycles (1.27 days) and will require $4.52e + 16$ physical qubits

> (an area of at least 452,398,896.21 square meters). The distance of the code is 57.
>
> It will use 14,428,405,755 magic state factories. To distill a magic state will require 2 distillation(s), with surface code distance(s) 57 and 29."

We have allowed for an error rate of 0.1 percent, a threshold of 0.316 percent, 11 cycles of the surface code for a fully parallel distillation as in Fowler's 2012 paper [46]. We have also completely prioritised time over memory costs to the maximum extent that the calculator allows. While the time taken for the algorithm seems plausible, the number of square metres of space taken up certainly does not. In reality, breaking CSIDH would cost far fewer physical qubits than this for several reasons:

- An efficient algorithm would not need error correction for idle qubits (though this takes no extra time, it does take extra space).
- An efficient algorithm would make use of the 2^{40} bits of QROM rather than extra unnecessary qubits
- Advances in error correction mean that instead of 10000 physical qubits per logical qubit, we could instead need only 1000 or maybe even 100.
- We have sacrificed space for speed on this particular calculation, meaning the same algorithm could run with far fewer qubits.

Chapter 6

Conclusion

Summary

We have conducted a review of error-corrected quantum cryptanalysis of CSIDH, taking into account the necessary background to understand this fluid and changing topic. We introduced the notions of cryptology and elliptic curves to describe CSIDH before moving into quantum computation using the surface code and error correction. These two areas have been joined to introduce the notion of an algorithm to break CSIDH-512 implemented on the surface code.

The difficulty of choosing how to measure the size of the problem of error-corrected cryptanalysis of CSIDH has been noted; while many studies give the size of quantum algorithms in T-gates, quantifying how to error-correct those T-gates is difficult due to the number of variables. Physical qubits, magic gates and 'the amount of' parallelisation all need to be taken into account. We can consider space-time trade offs in an abstract sense, but without moving more deeply into the structure of an algorithm for CSIDH run using the surface code, it is difficult to provide precise results. However, the tool created by Jaques gave an extremely interesting and comprehensive insight into the structure of such a cryptanalysis on the surface code and provided us with many ideas as to how this thesis could be developed if given enough time.

Recommendations

Overall, CSIDH is still a fairly interesting possible scheme for use as a drop-in replacement for Diffie-Hellman. From the literature review, its subexponential security limits its usefulness in all but the largest implementations, and had it been entered into the NIST competition, may have been overlooked in favour of schemes that (currently) look more robust [63]. However, CSIDH appears to have more significant overheads when accounting for error correction than some other schemes not based on isogenies [4] so its real-world application might not rule it out altogether. It is also important to note that schemes that have subexponential classical time complexity to break are still regarded as fairly secure, like RSA [64]. More research should be done into this particular field.

More generally, isogeny-based schemes are still motivating a lot of research; not only for applications in key exchanges but also signature and encryption schemes. The main idea of CSIDH has generated a large amount of interest since its inception in CRS [28, 27] and will continue to do so.

Further research

Given the opportunity, there are a number of pathways where the ideas in this thesis could be developed.

Time and Space Limitations

In this paper we have only considered a few varieties of time/space trade-offs in the literature review. Bonnetain and Schrottenloher [8] introduce a number, as well as Fowler's 2012 paper [46] giving several variations of surface code implementation with different requirements on space and time. Peikert's c-sieves paper [7] proposes alternatives to Bonnetain and Schrottenloher's work, but in place of large space and time complexity requires a large amount of QROM.

It would be interesting to consider various combinations of these trade-offs, considering the two separate goals of speed and space: how can we conduct error-corrected cryptanalysis of CSIDH in as little time as possible, or in as little space as possible? It would also be interesting to consider different or limited access to QROM. Notably, this is a very popular area for research in CSIDH at the moment owing to the variety of trade-offs and different types of error-correction that can be implemented.

Schemes Based on CSIDH

SeaSign is a signature scheme based on CSIDH's group action and the Fiat-Shamir protocol. While the security of the problem still relies on the same group action evaluation as CSIDH, it would be beneficial to research into this area to see if any other parts of the scheme or its cryptanalysis are significantly more costly than CSIDH.

CSURF, or CSIDH on the surface is an implementation of CSIDH that uses the endomorphism ring $Z[(1+\sqrt{-p})/2]$ [65]. On average, it runs 5.68 percent faster than CSIDH-512 to the same level of security. It would be very interesting to look into what makes this key exchange faster, and if that speed could affect the cryptanalysis of the system.

Error-Correcting Other Schemes

Since CSIDH was not entered into NIST's competition, it could be very beneficial to look into the realistic cryptanalysis of NIST finalists and alternates. A passing comment in the *New Quantum Cryptanalysis of CSIDH* forum [4] stated that:

> "There's a huge additional cost for fault tolerance, and presumably this will be an even bigger issue for isogeny computations than it is for Grover attacks against AES, SHA-2, ... Presumably the gap will be larger than the gap for, e.g., the AES attack, which has far fewer idle qubits and much less communication overhead."

Though a brief comment, this was very interesting and generated a number of possible roads to explore. It would be interesting to look into how the 'difficult to error-correct' parts of CSIDH compare with the most costly parts of other schemes and why that might be.

More Methods of Analysis

There are several tools we can apply to look at the quantum resource estimation of CSIDH in other ways [66]. Jaques's calculator gives invaluable insight into the structure of an algorithm run on the surface code, and how that would need to be adjusted for different algorithms so they can run as quickly as possible. It would be interesting to look into parts of breaking CSIDH that could be made more efficient on the surface code. Some sources mention the amount of qubits in breaking CSIDH that are idle for the majority of the time [4]. How could we change an algorithm to not correct errors that occur on these qubits every cycle? How could we more efficiently use these qubits in different areas. We could also talk about how much we can parallelise magic state distillation for CSIDH. How would we allocate space on a quantum computer to do this as efficiently as possible?

Moving Away From the Surface Code

We could also consider using another method of error correction. Though the surface code is simple, it has been noted as not the most efficient method for error correction in some cases. Further research could be done to look into what other candidates exist that could be used to conduct error corrected cryptanalysis of CSIDH, especially Toric code and subsystem code. It would be interesting to see if there are any parts of breaking CSIDH that are more suited to another type of quantum error-correction.

There are so many new avenues to explore that following them all would turn a master's thesis into a PhD. Such is the case in cryptology at the moment; the possibilities of quantum computers in cryptology can feel endless. Though this thesis has only

given a brief insight into one specific area, we hope it has piqued the reader's interest in the subject and shown them a glimpse into the future of security.

Appendix A

Appendix

We assume a knowledge of a few things in this thesis, namely group theory (with a small amount of number theory), and some areas of computer science linked to asymptotic complexity and oracles.

Group Theory and Number Theory

To describe CSIDH, it is necessary to have a rudimentary understanding of ideal-class groups in relation to elliptic curves [67]. Defining an ideal-class group takes several preliminary definitions. First we define order, a term well known in group theory but redefined in ring theory:

Definition A.1 (Order). The *order* $\mathcal{O}$ of a ring A is a subring of A such that A is finite dimensional over $\mathbb{Q}$ and $\mathcal{O}$ is an abelian group with a basis generated by a basis for A over $\mathbb{Q}$.

Definition A.2 ($\mathcal{O}$-Ideal). The ideal of a subring $\mathcal{O}$ is known as an $\mathcal{O}$-ideal.

Definition A.3 (Fractional Ideal). A *fractional ideal* of $\mathcal{O}$ is an $\mathcal{O}$-submodule of a field $\mathbb{F}$ of the form $\alpha\mathfrak{A}$, where $\alpha \in \mathbb{F}^*$ and $\mathfrak{A}$ is an $\mathcal{O}$-ideal.

Definition A.4 (Invertible Fractional Ideal). An *invertible fractional ideal* of $\mathcal{O}$ is a fractional ideal of $\mathcal{O}$, $\mathfrak{A}$, where there exists another fractional ideal $\mathfrak{B}$ such that $\mathfrak{A}\mathfrak{B} = \mathcal{O}$. We write that $\mathfrak{A}^{-1} = \mathfrak{B}$. Invertible fractional ideals are also written as $I(\mathcal{O}$.

Definition A.5 (Principle Fractional Ideal). A *principle fractional ideal* of $\mathcal{O}$ is a fractional ideal of $\mathcal{O}$, $\mathfrak{A}$, where $\mathfrak{A}$ is generated by a single element $a \in \mathcal{O}$ through multiplication by every element of $\mathcal{O}$. Principle Fractional Ideals are also written as $P(\mathcal{O})$.

Definition A.6 (Ideal-Class Group). The *ideal-class group* $\mathrm{cl}(\mathcal{O})$ is defined as the quotient:

$$\mathrm{cl}(\mathcal{O}) = I(\mathcal{O})/P(\mathcal{O})$$

This group will allow us to perform actions on elliptic curves in Chapter 2.3 and Chapter 4 to create CSIDH.

Computer science definitions

We provide some definitions and background for elements of the project that refer to complexity/cost, and algorithms with queries and oracles.

Asymptotic complexity

When talking about how difficult it is to 'break' or solve a problem in cryptography using an algorithm, it is very useful to describe it in terms of asymptotic complexity. Asymptotic complexity focuses on how the difficulty of solving a problem grows with the size of the problem. We can measure this growth in several ways: time, space/memory on a computer and number of nonlinear bit operations to name a few.

For example, solving problem A with input size 1 might take 5 seconds, but solving problem A with input size 2 might take 20 seconds. How can we describe the complexity of problem A in a more general case?

Big-O Notation

Big-O notation is a widely used method of measuring the complexity of a problem. It describes the growth rate of a problem as a function - for example $O(n^2)$ or $O(e^n)$. For an input size n, the size of the problem grows roughly proportional to the function inside the brackets in terms of n. We generally take this function to be the upper bound of the size of the problem, and generally ignore any constant terms or non-leading terms.

Definition A.7. $f(n) = O(g(n))$ if there exist real $c > 0$ and integer $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \leq cg(n)$.

For example, say problem B takes an input of size n and returns an output in $3n^2 + 2n + 1$ seconds. We say that Problem B has a time complexity of $O(n^2)$. Big-O notation can be used to describe most measures of complexity including time and space. This can be applied both to classical and quantum computers. For example, the Discrete Logarithm Problem (DLP) has classical time complexity $O(\sqrt{N})$ when solved with the Baby-Step Giant-Step algorithm, but has quantum time complexity $O(\log N^3)$ using Shor's algorithm, where N is the order of a cyclic group being used in the DLP.

We say an algorithm is efficient if it has no more than polynomial time complexity. A polynomial time complexity means that the Big-O function is a polynomial, e.g. n^2 and n^4.

L-notation

In parts of this thesis, L notation is instead used. L notation is an alternative to Big-O notation that is useful for expressing subexponential terms. It is written as follows:

$$L_n[\alpha, c] = e^{(c+o(1))(\log n)^{\alpha}(\log\log n)^{1-\alpha}}$$

Where c is a positive constant. When α is between 0 and 1, the complexity of the function is subexponential and superpolynomial.

Oracles and black box functions

An oracle is essentially a black box function, a function that takes an input and returns an output without the user having any knowledge of how the function works [68]. In computer science, oracles are asked 'queries' and return 'answers'. On a classical computer, an oracle is a function of the form $f : \{0,1\}^n \rightarrow \{0,1\}^m$

Often counting the number of queries made to an oracle can be a measure of how difficult a problem is to solve. Many problems exist that can be solved with fewer queries to an oracle on a quantum computer than a classical computer.

Reductions

Definition A.8. A *reduction* is a proof relating the difficulty of solving one problem to another [69]. For example say that for Y a successful adversary against a problem B, there is a successful adversary against a problem A that uses Y. Then we can say that there exists a reduction to solve A. Essentially, reductions prove that one problem is at most as difficult to solve as another.

Two problems might use slightly modified versions of one solution method to show this.

Bibliography

[1] Peter Shor. Algorithms for quantum computation: Discrete logarithms and factoring. https://klein.mit.edu/~shor/papers/algsfqc-dlf.pdf, 1995. (Accessed on 04/13/2022).

[2] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. https://csidh.isogeny.org/csidh-20181118.pdf, 2018. (Accessed on 04/13/2022).

[3] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum isogenies: Software. https://quantum.isogeny.org/software.html. (Accessed on 04/19/2022).

[4] Chris Peikert, Daniel Apon, and Daniel Bernstein. New quantum cryptanalysis of CSIDH - forum post. https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/svm1kDy6c54?pli=1. (Accessed on 04/19/2022).

[5] Post-Quantum Cryptography. Post-quantum cryptography standardisation | CSRC. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization, 2022. (Accessed on 04/13/2022).

[6] Post-Quantum Cryptography. Post-quantum cryptography standardisation evaluation criteria | CSRC. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardiza

tion/evaluation-criteria/security-(evaluation-criteria), 2022. (Accessed on 04/13/2022).

[7] Chris Peikert. He gives c-sieves on the CSIDH. Cryptology ePrint Archive, Report 2019/725, 2019. https://ia.cr/2019/725.

[8] Xavier Bonnetain and Andre Schrottenloher. Quantum security analysis of CSIDH. Cryptology ePrint Archive, Report 2018/537, 2018. https://ia.cr/2018/537.

[9] Tony Damico. A brief history of cryptography - inquiries journal. http://www.inquiriesjournal.com/articles/1698/a-brief-history-of-cryptography, 2009. (Accessed on 04/13/2022).

[10] Martine Diepenbroek. Myths and histories of the spartan scytale — university of bristol. https://research-information.bris.ac.uk/en/studentTheses/myths-and-histories-of-the-spartan-scytale, 2021. (Accessed on 04/13/2022).

[11] Sweigart. Making paper cryptography tools. https://inventwithpython.com/cracking/chapter1.html. (Accessed on 04/15/2022).

[12] Hemanta Maji. Perfect security definition. https://www.cs.purdue.edu/homes/hmaji/teaching/Fall%202016/lectures/03.pdf. (Accessed on 04/15/2022).

[13] Auguste Kerckhoffs. La cryptographie militaire (première partie). https://www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf, 1883. (Accessed on 04/15/2022).

[14] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[15] Yogesh Kumar, Rajiv Munjal, and Harsh Bardhan Sharma. Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures. In *Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures*, 2011.

[16] V.K. PACHGHARE. *Cryptography and Information Security, Third Edition.* PHI Learning Pvt. Ltd., 2019.

[17] E. S. I. Harba. Secure data encryption through a combination of aes, rsa and hmac. *Engineering, Technology and Applied Science Research*, 7(4):1781–1785, Aug. 2017.

[18] Dr. Asha Ambhaikar. Aes and rsa-based hybrid algorithms for message encryption and decryption. *INFORMATION TECHNOLOGY IN INDUSTRY*, 9:273–279, 03 2021.

[19] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[20] Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography – PKC 2013*, pages 254–271, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[21] Yassine Mrabet. Ecclines-3.svg - wikimedia commons file. https://commons.wikimedia.org/w/index.php?curid=3112726. (Accessed on 04/16/2022).

[22] Dyland Pentland. The j-invariant of an elliptic curve. https://math.mit.edu/research/highschool/primes/materials/2018/conf/, 2018. (Accessed on 04/16/2022).

[23] Rich Schwartz. The elliptic curve group law. https://www.math.brown.edu/reschwar/M1540B/elliptic.pdf. (Accessed on 04/16/2022).

[24] Andrew Sutherland. Lecture notes 5. https://math.mit.edu/classes/18.783/2015/LectureNotes5.pdf, 2015. (Accessed on 04/16/2022).

[25] Daniel Brown. Standards for efficient cryptography: Elliptic curve cryptography. http://www.secg.org/sec1-v2.pdf, 2009. (Accessed on 04/16/2022).

[26] David et al Jao. SIKE – supersingular isogeny key encapsulation. https://sike.org/, 2011. (Accessed on 04/16/2022).

[27] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. https://ia.cr/2006/291.

[28] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. https://ia.cr/2006/145.

[29] Post-Quantum Cryptography. Post-quantum cryptography standardisation | CSRC. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization. (Accessed on 04/16/2022).

[30] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22:563–591, 05 1980.

[31] IBM. What is quantum computing? | IBM. https://www.ibm.com/topics/quantum-computing. (Accessed on 04/17/2022).

[32] Leonard Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *20th Annual Symposium on Foundations of Computer Science*, pages 55–60, 1979.

[33] Mark Lapedus. The great quantum computing race. https://semiengineering.com/the-great-quantum-computing-race/, 2021. (Accessed on 04/17/2022).

[34] IBM's 127-qubit eagle is the biggest quantum computer yet. https://singularityhub.com/2021/11/22/ibms-127-qubit-eagle-is-the-biggest-quantum-computer-yet/#:~:text=Progress%20in%20quantum%20computing%20is,of%20a%20127%2Dqubit%20processor. (Accessed on 04/27/2022).

[35] Jessica Hamezlou, Tate Ryan-Mosely, Niall Firth, and Patrick Howell O'Neil. How a quantum computer could break 2048-bit rsa encryption in 8 hours | MIT technology review. https://www.technologyreview.com/2019/05/30/65724/how-a-quantum-computer-could-break-2048-bit-rsa-encryption-in-8-hours/, 2019. (Accessed on 04/17/2022).

[36] Daniel Bernstein. Introduction to post-quantum cryptography. http://www.pqcrypto.org/www.springer.com/cda/content/document/cda_downloaddocument/9783540887010-c1.pdf, 2009. (Accessed on 04/17/2022).

[37] Renato Renner. Security of quantum key distribution, 2005.

[38] P. A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(3):416–418, 1939.

[39] E.G. Rieffel and W.H. Polak. *Quantum Computing: A Gentle Introduction*. Scientific and Engineering Computation. MIT Press, 2011.

[40] Oxford Reference. Logic gate. https://www.oxfordreference.com/view/10.1093/oi/authority.20110810105307777. (Accessed on 04/17/2022).

[41] Michael Charemza. Quantum circuit diagrams. https://warwick.ac.uk/fac/sci/physics/research/cfsa/people/pastmembers/charemzam/pastprojects/mcharemza_quant_circ.pdf, 2006. (Accessed on 04/17/2022).

[42] Abdullah Ash-Saki, Mahabubul Alam, and Swaroop Ghosh. Study of decoherence in quantum computers: A circuit-design perspective. *CoRR*, abs/1904.04323, 2019.

[43] Todd Brun. Lecture notes on error correction. https://viterbi-web.usc.edu/~tbrun/Course/lecture20.pdf, 2017. (Accessed on 04/17/2022).

[44] Noah Linden and Ryan Mann. *Quantum Computation MATHM0023 Lecture Notes*. University of Bristol School of Mathematics, 2021.

[45] Quantum Computing UK. Quantum error correction: Shor code in qiskit — quantum computing UK. https://quantumcomputinguk.org/tutorials/quantum-error-correction-shor-code-in-qiskit, 2020. (Accessed on 04/22/2022).

[46] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), sep 2012.

[47] Shota Nagayama. Surface code error correction on a defective lattice. https://iopscience.iop.org/article/10.1088/1367-2630/aa5918/pdf, 2017. (Accessed on 04/17/2022).

[48] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179, sep 2017.

[49] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Phys. Rev. A*, 71:022316, Feb 2005.

[50] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, mar 2019.

[51] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, dec 2012.

[52] Oscar Higgott and Nikolas P. Breuckmann. Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead. *Phys. Rev. X*, 11:031039, Aug 2021.

[53] Andrew Childs, David Jao, and Vladimir Soukharev. Constructing elliptic curve isogenies in quantum subexponential time. *Journal of Mathematical Cryptology*, 8(1):1–29, jan 2014.

[54] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput. 35 (2005), 170-188*, 2003.

[55] Kevin Cheung and Michele Mosca. Decomposing finite abelian groups. *Quantum Information and Computation*, 1, 11 2001.

[56] Greg Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *8th Conference on the Theory of Quantum Computation, Communication and Cryptography 22 (2013), 20-34*, 2011.

[57] Biasse, Jao, and Sankar. A quantum algorithm for computing isogenies between supersingular elliptic curves (INDOCRYPT 2014). https://link.springer.com/content/pdf/10.1007/978-3-319-13039-2.pdf, 2014. (Accessed on 04/13/2022).

[58] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: Optimizing quantum evaluation of isogenies. Cryptology ePrint Archive, Report 2018/1059, 2018. https://ia.cr/2018/1059.

[59] Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Physical Review X*, 8(4), oct 2018.

[60] Austin G. Fowler. Optimal complexity correction of correlated errors in the surface code, 2013.

[61] Samuel Jaques. Sam jaques surface code project. https://sam-jaques.appspot.com/projects/surface_codes. (Accessed on 04/19/2022).

[62] Samuel Jaques. Quantum cost models for cryptanalysis of isogenies. https://uwspace.uwaterloo.ca/handle/10012/14612, 2019. (Accessed on 04/17/2022).

[63] NIST. PQC third round candidate announcement | CSRC. https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement. (Accessed on 04/19/2022).

[64] Daniel J. Bernstein and A. K. Lenstra. A general number field sieve implementation. In Arjen K. Lenstra and Hendrik W. Lenstra, editors, *The development of the number field sieve*, pages 103–126, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[65] Wouter Castryck and Thomas Decru. CSIDH on the surface. Cryptology ePrint Archive, Report 2019/1404, 2019. https://ia.cr/2019/1404.

[66] Martin Suchara, John Kubiatowicz, Arvin Faruque, Frederic T. Chong, Ching-Yi Lai, and Gerardo Paz. QuRE: The quantum resource estimator toolbox. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 419–426, 2013.

[67] Priestley. Introduction to groups, rings and fields. https://people.maths.ox.ac.uk/flynn/genus2/sheets0405/grfnotes1011.pdf, 2011. (Accessed on 04/19/2022).

[68] Mohammad Mahmoody and Avi Wigderson. Black boxes, incorporated. https://www.cs.virginia.edu/~mohammad/files/papers/BlackBoxes.pdf, 2012. (Accessed on 04/22/2022).

[69] Margus Niitsoo. Cryptographic reductions. https://courses.cs.ut.ee/2008/crypto-seminar-spring/papers/niitsoo1.pdf, 2008. (Accessed on 04/22/2022).