Impossibility of Superlogarithmic Lower Bounds for the Evaluation of Isogenies

Adrian Hamelink

School of Computer and Communication Sciences

Semester Project

June 2020

Responsible
Prof. Serge Vaudenay
EPFL / LASEC

Supervisor
Khashayar Barooti
EPFL / LASEC

Introduction

The main goal of this report was to find a lower bound on the number of operations required to evaluate an l-degree isogeny between elliptic curves. The following chapters mirror the path we took in learning.

In the first Chapter 1, we very briefly introduce some algebraic geometry theory in order to actually define elliptic curves and their isogenies. We studied the book *The Arithmetic of Elliptic Curves* by Silverman which provided us with a good algebraic understanding of these mathematical objects, and following up with the book *Mathematics of Public Key Cryptography* by Galbraith for other insights.

We then were able to understand "CSIDH: An Efficient Post-Quantum Commutative Group Action" by Castryck et al. as well the surface variant of CSIDH. These papers are presented in Chapter 2. They introduced us to a concrete application of isogeny based cryptography, and highlighted the importance of finding lower bounds on their evaluation.

In Chapter 3, we take a deeper dive into the structure of the so-called *kernel polynomials* of isogenies. These object appear naturally when considering the structure of formulas which describe isogenies. At the same time, Bernstein et al. published *Faster computation of isogenies of large prime degree* in which they present a new way to evaluate them, therefore introducing a lower upper bound on the number of operations required to evaluate a kernel polynomial. In parallel, we started exploring the field of algebraic complexity theory, first through papers by Strassen and Schnorr and later studying the book *Algebraic Complexity Theory* by Bürgisser, Clausen, and Shokrollahi. This turned out to be invaluable as we realized our goal could be achieved using the theory of straight-line programs and finite dimensional machines.

Chapter 4 is devoted to laying out the theory relating to this subject. We used the framework of *Complexity and Real Computation* by Blum et al. since we came across the paper "On the intractability of Hilbert's Nullstellensatz and an algebraic version of P vs. NP" by the co-authors Shub and Smale and it allowed us to maintain throughout the rest of the report.

The following Chapter 5 presents in detail the main theorem from this latter paper, which connects the problem of computing $k!$ and the algebraic version of $\mathbf{P} \overset{?}{=} \mathbf{NP}$. It proves that if the former is hard, the the latter must be false.

The final Chapter 6 contains our main contribution in which we adapt the idea from Chapter 5 to the evaluation of kernel polynomials. We succeed in showing that if evaluating the kernel polynomial is easy in $\mathbb{C}$, then it must be true that $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$, therefore, proving that a superlogarithmic lower bound is most likely impossible.

Chapter 1

Elliptic Curves & Isogenies

In this chapter we introduce some mathematical background on elliptic curves and their isogenies.

This section is heavily inspired by the works of Galbraith and Silverman in *Mathematics of Public Key Cryptography* [10] and *The Arithmetic of Elliptic Curves* [15]. While we mostly cite Galbraith, we originally studied the theory of elliptic curves from the perspective of Silverman.

1.1 Elliptic Curves

Throughout this chapter, we suppose K is a perfect field contained in a fixed algebraic closure $\bar{K}$.

We assume the reader is familiar with basic notions from algebraic geometry concerning elliptic curves and can therefore skip this chapter. We only recall the basic definitions which we reproduce from [10]. They are included here as reference, and we refer to the cited works which contain more insight.

We first start by defining affine spaces and the different objects that they induce.

Definition 1.1 (Affine n-space [10, Section 5.1]). The *affine n-space* over K is defined as $\mathbb{A}^n(K) = K^n$. The *affine line* and *affine plane* are defined as $\mathbb{A}^1(K) = K^1$ and $\mathbb{A}^1(K) = K^1$ respectively.

Definition 1.2 (Affine algebraic set [10, Definition 5.1.1])**.** Define

$$V(S) = \left\{P \in \mathbb{A}^n(\bar{K}) : f(P) = 0 \text{ for all } f \in S\right\}$$

for a set $S \subseteq K[x_1, \ldots, x_n]$.

If $S = \{f_1, \ldots, f_m\}$ then we write $V(f_1, \ldots, f_m)$ for $V(S)$. An *affine algebraic set* is a set $X = V(S) \subset \mathbb{A}^n$ where $S \subseteq K[x_1, \ldots, x_n]$.

If K'/K is an algebraic extension, the K'-rational points of $X = V(S)$ are

$$X(K') = X \cap \mathbb{A}^n(K') = \left\{P \in \mathbb{A}^n(K') : f(P) = 0 \text{ for all } f \in S\right\}$$

Definition 1.3 (Ideal [10, Definition 5.1.13])**.** The *ideal* over K of a set $X \subseteq \mathbb{A}^n(\bar{K})$ is

$$I_K(X) = \left\{f \in K[x_1, \ldots, x_n] : f(P) = 0 \text{ for all } P \in X(\bar{K})\right\}$$

Define $I(X) = I_{\bar{K}}(X)$. An algebraic set X is *defined over* K if $I(X)$ can be generated by elements of $K[x_1, \ldots, x_n]$.

Definition 1.4 (Affine coordinate ring [10, Definition 5.1.18])**.** The *affine coordinate ring* over K of an affine algebraic set $X \in \mathbb{A}^n$ defined over K is defined as

$$K[X] = K[x_1, \ldots, x_n]/I_K(X).$$

Most of the above definitions match their counterparts in projective spaces closely. We define them next.

Definition 1.5 (Projective space [10, Definition 5.2.1])**.** The *projective space* over K of dimension n is defined as

$$\mathbb{P}^n(K) = \left\{(a_0 : a_1 : \ldots : a_n) \in \mathbb{A}^{n+1} : (a_0 : a_1 : \ldots : a_n) \neq (0 : \ldots : 0)\right\}$$

where

$$(a_0 : \ldots : a_n) = \{(\lambda a_1, \ldots, \lambda a_n) : \lambda \in K^\star\}$$

is the equivalence class of $(a_0 : \ldots : a_n)$.

Definition 1.6 (Projective algebraic set [10, Definition 5.2.6])**.** Let $f \in K[x_0, \ldots, x_n]$ be a homogeneous polynomial. A point $P = (x_0, \ldots, x_n) \in \mathbb{P}^n(K)$ is a *zero* of f if $f(x_0, \ldots, x_n) = 0$. Let S be a set of polynomials and define

$$V(S) = \left\{P \in \mathbb{P}^n(\bar{K}) : P \text{ is a zero of } f \text{ for all homogeneous } f \in S\right\}.$$

A *projective algebraic set* is a set $X = V(S) \subseteq \mathbb{P}^n(\bar{K})$ for some $S \subseteq K[x_0, \ldots, x_n]$. Such a set is also called a *projective K-algebraic set.* For $X = V(S)$ and K'/K an algebraic extension of K we define

$$X(K') = \left\{P \in \mathbb{P}^n(K') : f(P) = 0 \text{ for all homogeneous } f \in S\right\}$$

Definition 1.7 (Varieties [10, Definition 5.3.1])**.** An affine K-algebraic set $X \subseteq \mathbb{A}^n$ is *K-reducible* if $X = X_1 \cup X_2$ for X_1, X_2 two K-algebraic sets such that $X_1, X_2 \neq X$. It is *K-irreducible* if there exists no such decomposition, and *geometrically irreducible* if X is $\bar{K}$-irreducible. An *affine variety* over K is a geometrically irreducible affine K-algebraic set defined over K.

A projective K-algebraic set $X \subseteq \mathbb{P}^n$ is *K-irreducible* (resp. *geometrically irreducible*) if $X \neq X_1 \cup X_2$ for X_1, X_2 two K-algebraic sets such that $X_1, X_2 \neq X$ (resp. projective $\bar{K}$-algebraic sets). A *projective variety* over K is a geometrically irreducible K-algebraic set defined over K.

Definition 1.8 (Function fields [10, Definition 5.4.1])**.** Let X be an affine variety defined over K. The *function field* $K(X)$ is the set

$$K(X) = \{f_1/f_2 : f_1, f_2 \in K[X], f_2 \notin I_K(X)\}$$

We consider this set under the equivalence class

$$f_1/f_2 \equiv f_3/f_4 \iff f_1f_4 - f_2f_3 \in I_K(X)$$

If X is a projective variety defined over K, then its *function field* is

$$K(X) = \{f_1/f_2 : f_1, f_2 \in K[X] \text{ homogeneous of same degree }, f_2 \notin I_K(X)\}$$

Elements of $K(X)$ are called *rational functions* under the same equivalence relation for projective spaces.

For the next definition, we refer to [15, Silverman] for the definition of regularity.

Definition 1.9 (Rational maps [10, Definition 5.5.1])**.** Let X be an affine or projective variety defined over K, and Y an affine variety in $\mathbb{A}^n$ over K. Let $\phi_1, \ldots, \phi_n \in K(X)$. A map $\phi : X \to \mathbb{A}^n$ of the form

$$\phi(P) = (\phi_1(P), \ldots, \phi_n(P))$$

is called *regular* at a point $P \in X(\bar{K})$ if all ϕ_i are regular at P. A *rational map* $\phi : X \to Y$ defined over K is a map of the same form such that, for all $P \in X(\bar{K})$ for which ϕ is regular at P, then $\phi(P) \in Y(\bar{K})$.

Let X be an affine or projective variety defined over K, and Y a projective variety in $\mathbb{P}^n$ over K. Let $\phi_0, \ldots, \phi_n \in K(X)$. A map $\phi : X \to \mathbb{P}^n$ of the form

$$\phi(P) = (\phi_0(P) : \ldots : \phi_n(P))$$

is called *regular* at a point $P \in X(\bar{K})$ if there is a function $g \in K(X)$ such that all $g\phi_i$ are regular at P, and for some i one have $(g\phi)(P) \neq 0$. A *rational map* $\phi : X \to Y$ defined over K is a map of the same form such that, for all $P \in X(\bar{K})$ for which ϕ is regular at P, then $\phi(P) \in Y(\bar{K})$.

Definition 1.10 (Morphism [10, Definition 5.5.12])**.** Let X, Y be varieties over K and let $U \subseteq X$ be open. A rational map $\phi : U \to Y$ over K which is regular at every point $P \in U(\bar{K})$ is called a *morphism* over K.

Definition 1.11 (Curve [10, Definition 7.1.17])**.** A *curve* is a projective non-singular variety of dimension 1. A *plane curve* is a *curve* given by an equation $F(x, y, z) = 0$ and is the set $V(F(x, y, z)) \subseteq \mathbb{P}^2$, where $V(F)$ is the variety defined by F.

Definition 1.12 (Weierstrass equation [10, Definition 7.2.1])**.** Let $a_1, \ldots, a_6 \in K$.

A *Weierstrass equation* is a projective algebraic set E over K given by

$$y^2z + a_1xyz + a_3yz^2 = x^3 + a_2x^2z + a_4xz^2 + a_6z^3.$$

The *affine Weierstrass equation* is

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6z.$$

Definition 1.13 (Point at infinity [10, Definition 7.2.5])**.** Let E be a Weierstrass equation over K. The point $(0 : 1 : 0) \in E(K)$ is denoted $\mathcal{O}_E$ and is called the *point at infinity.*

In some cases, we may denote this point as ∞.

Definition 1.14 (Elliptic curve [10, Definition 7.2.8])**.** An *elliptic curve* is a curve given by a non-singular Weierstrass equation.

Definition 1.15 (Group law [10, p. 9.1])**.** Let E be an elliptic curve over a field K given by a non-singular affine Weierstrass equation. This curve defines a group with neutral element $\mathcal{O}_E$.

Definition 1.16 (Isogeny [10, Definition 9.2.2])**.** Let E_1, E_2 be an elliptic curve over a field K given by a non-singular affine Weierstrass equation, and let $\phi : E_1 \to E_2$ be a morphism of varieties such that $\phi(\mathcal{O}_{E_1}) = \mathcal{O}_{E_2}$, Then ϕ is an *isogeny.* The *kernel* of ϕ is the set $\ker(\phi) = \{P \in E_1 : \phi(P) = \mathcal{O}_{E_2}\}$. The *degree* of ϕ is the degree of the morphism. Define the sets $\mathrm{Hom}_K(E_1, E_2)$ and $\mathrm{End}_K(E_1)$ as the sets of isogenies from E_1 to E_2 or itself respectively.

Chapter 2

CSIDH & CSURF

In this chapter we discuss the CSIDH and CSURF cryptography systems discovered by Castryck et al. in [7] These papers inspired us to look into the computational aspect of evaluating isogenies, since the speed of the cryptographic protocol depends on them.

Before introducing the constructions, we briefly mention hard homogeneous spaces.

2.1 Hard Homogeneous Spaces

A useful mathematical structure in cryptography are the so called *Hard Homogeneous Spaces (HHS)* introduced by Jean-Marc Couveignes. These spaces formalize the idea of the discrete logarithm problem, and are implemented for isogenies in CSIDH.

Definition 2.1 (Homogeneous Space [9], [7, Definition 1]). Let G be a commutative group. A *Homogeneous Space* X for G is a finite set X of same cardinality $S = \#X = \#G$ on which G acts transitively and regularly.

Example. Homogeneous Spaces

- Let G be a commutative group and $X = G$. Then G acts on itself.
- Let X be an affine space, and G its vector space.

For cryptography, we are interested in *hard homogeneous spaces*:

Definition 2.2. A *hard homogeneous spaces* is a homogeneous space in which the following operations are easy,

- Computing the group operation in G.
- Sampling randomly from G with (close to) uniform distribution.
- Decide validity and equality of a representation of elements of X.
- Computing the action of a group element $g \in G$ on some $x \in X$.

and the following must be hard

- Given $x_1, x_2 \in X$, find $g \in G$ such that $g \circ x_1 = x_2$.
- Given $x_1, x_2, y \in X$ such that $x_2 = g \circ x_1$, find $y' = g \circ y$.

The next example constitutes a hard homogenous space in some situations.

Example (Discrete Logarithm)**.** Let S be the set of generators of a cyclic group C and $G = \left(\mathbb{Z}_{|C|}\right)^{\times}$ its automorphism group. G acts on S by exponentiation. This setup constitutes in some situations a hard homogeneous space (depending on the order of C). The group action is then $g \circ c = g^c$.

Unfortunately, these kind of setups are vulnerable to an attack using Shor's algorithm on quantum computers.

These homogeneous spaces allow us to establish a Diffie-Hellman key exchange because the group action is commutative. The CSIDH construction presented next is one of them.

2.1.1 CSIDH

The idea in CSIDH is to define a group action over a set of isomorphism classes of elliptic curves. It uses a group representing isogenies acting on a set of supersingular elliptic curves given in Montgomery form. The usefulness of this construction for cryptography comes from the fact that there is no known way to efficiently obtain the structure of the group acting on the curves.

We start with some definition and notations from [7].

Definition 2.3. The Frobenius endomorphism $\pi \in \operatorname{End}_p(E)$ of an elliptic curve E over $\mathbb{F}_p$ for $p \geq 5$ satisfied a characteristic equation

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

where $t \in \mathbb{Z}$ is the trace of Frobenius. It is also defined more generally over varieties $V \subset \mathbb{P}^n$ as the map $\pi : V \to V, \pi = [X_0^p, \ldots, X_n^p]$ [15].

The curve E is supersingular if and only if $t = 0$ and we have $\pi^2 = -p$.

The ring of $\mathbb{F}_p$-rational endomorphisms $\operatorname{End}_p(E)$ of E is an order $\mathcal{O}$ in the imaginary quadratic field $K = \mathcal{O} \otimes_{\mathbb{Z}} \mathbb{Q} \cong \mathbb{Q}(\sqrt{\Delta})$ where $\Delta = t^2 - 4p$. Note that $\mathcal{O}$ always contains π so it always contains $\mathbb{Z}[\pi]$.

An invertible ideal $\mathfrak{a} \in \mathcal{O}$ defines an elliptic curve $\phi_{\mathfrak{a}} : E \to E/\mathfrak{a}$ of degree $\mathrm{N}(\mathfrak{a})$ [18]. The set of these invertible ideals is $\mathrm{cl}(\mathcal{O})$.

$\mathcal{E}_p(\mathcal{O}, \pi)$ defines the set of $\mathbb{F}_p$-isomorphism classes of elliptic curves E over $\mathbb{F}_p$ for which $\operatorname{End}_p(E) \cong \mathcal{O}$.

We can now define the group action concretely with the following theorem.

Theorem 2.1 ([18, Theorem 4.5])**.**
Let $\mathcal{O}$ be an order in an imaginary quadratic field, and $\pi \in \mathcal{O}$ such that $\mathcal{E}_p(\mathcal{O}, \pi)$ is non-empty. Then the ideal class group $\mathrm{cl}(\mathcal{O})$ acts freely and transitively on the set $\mathcal{E}_p(\mathcal{O}, \pi)$ via the map:

$$\begin{array}{ccc} \mathrm{cl}(\mathcal{O}) \times \mathcal{E}_p(\mathcal{O}, \pi) & \to & \mathcal{E}_p(\mathcal{O}, \pi) \\ ([\mathfrak{a}], E) & \mapsto & E/\mathfrak{a} \end{array}$$

in which $[\mathfrak{a}]$ is chosen as an integral representative. □

We now explore how to compute these group action. As we will see in the next chapter, we can compute an isogeny given a generator P of the kernel.

In this section, we consider a prime $p = 4 \cdot l_1 \cdots l_n - 1$ where l_i are small distinct odd primes larger than 3. We fix $E_0 : y^2 = x^3 + x$ over $\mathbf{F}_p$ which is supersingular since $p \equiv 3 \mod 4$. The Frobenius endomorphism satisfies $\pi^2 = -p$ so its $\mathbf{F}_p$-endomorphism ring is an order in the imaginary quadratic field $\mathbf{Q}(\sqrt{-p})$. In particular, $\mathrm{End}_p(E_0) = \mathbb{Z}[\pi]$.

We first note from [14] that $\mathrm{cl}(\mathcal{O})$ is a finite abelian group with size $\#\,\mathrm{cl}(\mathcal{O}) \approx \sqrt{|\Delta|}$. Since our curve is supersingular, $\#\,\mathrm{cl}(\mathcal{O}) \approx \sqrt{p}$ because $|\Delta| = \sqrt{4p}$. The best known algorithm for computing the structure of this group is sub-exponential time, and therefore impractical in the real world.

From [11, Proposition 9.5.2 and 9.5.3], any element of the class group can be represented as a product of small prime ideals.

$$\mathfrak{a} = \prod_i \mathfrak{l}_i^{e_i}$$

Since the group is abelian, we only need to find the isogeny for each individual ideal $\mathfrak{l}$ and compose them together.

We have the ideal $l_i\mathcal{O} = \mathfrak{l}_i\bar{\mathfrak{l}_i}$ where $\mathfrak{l}_i = (l_i, \pi - 1)$ and $\bar{\mathfrak{l}_i} = (l_i, \pi + 1)$ The kernel of $\phi_{\mathfrak{l}}$ is the intersection of the kernel of the multiplication by l_i map $[l_i]$ and the kernel of $\pi - 1$. In other words,

$$P \in \ker \phi_{\mathfrak{l}} \iff [l_i]P = \infty, \quad \pi(P) = P$$

The last condition implies that the point must lie in the curve defined over $\mathbb{F}_p$.

The kernel of $\bar{\mathfrak{l}_i}$ is defined as the points Q of order l_i defined over $\mathbb{F}_{p^2}$ but not $\mathbb{F}_p$ and such that $\pi(Q) = -Q$.

Therefore computing the isogeny $\phi_{\mathfrak{a}} : E \to E/\mathfrak{a}$ where $\mathfrak{a} = \prod_i \mathfrak{l}_i^{e_i}$ requires only finding points of order l_i in the base curve, and applying the Vélu formula for each factor l_i.

The following proposition shows us what the form of the images of these isogenies look like.

Proposition 2.2 ([7, Proposition 8]).
Let $p \geq 5$ be a prime such that $p \equiv 3 \mod 8$ and let $E/\mathbb{F}_p$ be a supersingular elliptic curve. Then $\mathrm{End}_p(E) = \mathbb{Z}[\pi]$ if and only if there exists $A \in \mathbb{F}_p$ such that E is $\mathbb{F}_p$-isomorphic to the curve

$$E_A : y^2 = x^3 + Ax^2 + x.$$

Moreover, if such an A exists then it is unique. □

In practice, since the ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ are known, an element $\mathfrak{a}$ would be represented only by the exponents $(e_1, \ldots, e_n)$. It is thus easy to compute the group action on an elliptic curve. Moreover, it is also easy to compute the coefficient A of the resulting Montgomery curve, and isogeny computations are easier since we can only consider the x coordinate of the isogeny. The security of this system therefore rests on the assumption that computing the isogeny between two curves is hard.

An implementation of Diffie-Hellman key exchange can be explored in more detail in [7, Section 6].

2.2 CSIDH on the surface

After the introduction of CSIDH, Castryck and Decru came up with an alternative implementation called "CSIDH on the Surface" [6] which allows for primes p such that $p \equiv 7 \mod 8$. For curves defined over a finite field $\mathbb{F}_p$ with this property, we can create a similar construction when the endomorphism ring the $\mathbb{F}_p$-isomorphism classes of the curves is $\mathbf{Z}[(1+\sqrt{p})/2]$. One difference is that these curves will have the form

$$E_A : y^2 = x^3 + Ax - x.$$

The computation of isogenies over these curves is very similar, and only requires a few sign changes. Moreover, the prime 2 splits in $\mathbb{Q}(\sqrt{-p})$ as $(2) = (2, (\sqrt{-p}-1)/2)(2, (\sqrt{-p}+1)/2)$, and therefore an element of $\mathrm{cl}(\mathcal{O})$ contains as factor a prime ideal $(2, (\sqrt{-p}-1)/2)$. This is interesting since formulas for computing degree-2 isogenies in this context are much simpler and greatly improve the performance of the construction.

We do not dive further into this paper however, as the ideas are very similar to the ones from CSIDH. Both of these construction highlight the importance of computing isogenies. This inspired us to look into more detail at the computational cost of evaluating such an isogeny. In particular, we tried to show the existence of a lower bound on the number of operations required.

Chapter 3

Computation Bounds for Isogeny Evaluation

In this chapter, we will explore in more detail how to compute isogenies. The first section develops the formula for isogenies by exploiting properties of elliptic curves. Initially, we were hoping to obtain a closed form of a special function we often call *kernel polynomial*. Unfortunately, we will see that it not possible. Instead, we present a new method for evaluating isogenies by Bernstein et al., providing an upper bound on the number of operations required to evaluate isogenies.

Theorem 3.1 ([10, Theorem 9.7.5 on p. 179])**.**
Given two curves over K:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$
$$E' : Y^2 + a_1'XY + a_3'Y = X^3 + a_2'X^2 + a_4'X + a_6'$$

Let $\phi : E \to E'$ be an isogeny of elliptic curves over K.

We can express ϕ as:

$$\phi(x, y) = (\phi_1(x), y\phi_2(x) + \phi_3(x))$$

where

$$2\phi_3(x) = -\tilde{a_1}\phi_1(x) - \tilde{a_3} + (a_1x + a_3)\phi_2(x)$$

In particular, if ϕ is separable, we have

$$\begin{aligned} \phi_2(x) &= \phi_1(x)' = d\phi_1(x)/dx \\ \phi(x, y) &= (\phi_1(x), cy\phi_1(x)' + \phi_3(x)) \\ 2\phi_3(x) &= -\tilde{a_1}\phi_1(x) - \tilde{a_3} + c(a_1x + a_3)\phi_1(x)' \end{aligned}$$

for some $c \in \overline{K}^\star$ $\square$

We notice that given the function $\phi_1(x)$, we can easily compute the full isogeny ϕ. The following theorem and corollaries encapsulates Vélu's work in [17]. It allows us to give an more succint form for ϕ_1.

Theorem 3.2 ([10, Theorem 25.1.6 on p. 547] [17])**.**
Let E be an elliptic curve over K defined by the polynomial

$$F(x,y) = x^3 + a_2x^2 + a_4x + a_6 - \left(y^2 + a_1xy + a_3y\right) = 0.$$

Let G be a finite subgroup of $E(\overline{K})$. Let G_2 be the set of points in $G - \{\mathcal{O}_E\}$ of order 2, and let G_1 be such that

$$\#G = 1 + \#G_2 + 2\#G_1, \quad G = \{\mathcal{O}_E\} \cup G_2 \cup G_1 \cup \{-Q : Q \in G_1\}.$$

Write

$$F_x = \frac{\partial F}{\partial x} = 3x^2 + 2a_2x + a_4 - a_1y, \quad F_y = \frac{\partial F}{\partial y} = -2y - a_1x - a_3$$

For a point $Q = (x_Q, y_Q) \in G_1 \cup G_2$ define the quantities

$$\begin{aligned} u(Q) &= (F_y(Q))^2 = (-2y - a_1x - a_3)^2 \\ t(Q) &= \begin{cases} F_x(Q) & \text{if } Q \in G_2 \\ 2F_x(Q) - a_1F_y(Q) & \text{if } Q \in G_1. \end{cases} \end{aligned}$$

Note that if $Q \in G_2$ then $F_y(Q) = 0$.
Define

$$t(G) = \sum_{Q \in G_1 \cup G_2} t(Q), \quad w(G) = \sum_{Q \in G_1 \cup G_2} \left(u(Q) - x_Q t(Q)\right).$$

and set

$$\begin{aligned} A_1 = a_1 \quad & A_2 = a_2 \\ A_3 = a_3 \quad & A_4 = a_4 \\ A_6 = a_6 - & \left(a_1^2 + 4a_2t(G) - 7w(G)\right). \end{aligned}$$

Then the map $\phi : (x, y) \mapsto (X, Y)$ where

$$X = x + \sum_{Q \in G_1 \cup G_2} \frac{t(Q)}{x - x_Q} + \frac{u(Q)}{(x - x_Q)^2}$$

$$Y = y - \sum_{Q \in G_1 \cup G_2} \left[u(Q) \frac{2y + a_1 x + a_3}{(x - x_Q)^3} + t(Q) \frac{a_1(x - x_Q) + y - y_Q}{(x - x_Q)^2} + \frac{a_1 u(Q) - F_x(Q) F_y(Q)}{(x - x_Q)^2} \right]$$

is a separable isogeny from E to

$$E' : Y^2 + A_1 XY + A_3 Y = X^3 + A_2 X^2 + A_4 X + A_6$$

with kernel G. □

This theorem reveals a strong like between an isogeny and its kernel. In fact, the following corollary states that we can obtain an isogeny whose kernel is G for any finite subgroup of points.

Corollary 3.2.1 ([10, Corollary 25.1.7 on p. 550])**.**
Let E be an elliptic curve defined over K and G a finite subgroup of $E(\overline{K})$ that is defined over K. Then there is an elliptic curve $\tilde{E} = E/G$ defined over K and an isogeny $\phi : E \to \tilde{E}$ defined over K with $\ker(\phi) = G$. □

Moreover, the form given in Theorem 3.2 allows us to write the isogeny more succinctly as a rational function, as well as bounding the degrees.

Corollary 3.2.2 ([10, Corollary 25.1.8 on p. 550])**.**
Let $\phi : E \to \tilde{E}$ be a separable isogeny of odd degree l between elliptic curves over K. Write $\phi(x, y) = (\phi_1(x), \phi_2(x, y))$, where $\phi_1(x), \phi_2(x, y)$ are rational functions. Then $\phi_1(x) = u(x)/v(x)^2$, where $\deg u = l$ and $\deg v = (l-1)/2$. Also, $\phi_2(x, y) = (y w_1(x) + w_2(x))/v(x)^3$, where $\deg w_1 \leq 3(l-1)/2$ and $\deg w_2 \leq (3l - 1)/2$. □

The following theorem gives a more explicit representation of the rational function defining the isogeny. It defines the polynomial ψ which we call the *kernel polynomial* since it is defined by the points in the kernel of the isogeny. Computing an isogeny depends heavily on the computation of this polynomial.

Theorem 3.3 ([10, Lemma 25.1.16 on p. 551])**.**
Let E be an elliptic curve defined over K:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Let G be a finite subgroup of $E(\bar{K})$ of odd order $2d+1$. Let $G_1 \subset G$ such that $\#G_1 = d$ and $G = \{\mathcal{O}_E\} \cup G_1 \cup \{-P | P \in G_1\}$. The kernel polynomial of G is given by:

$$\psi(X) = \prod_{P \in G_1} (X - x(P))$$

whose roots are the x coordinates of elements in G_1.

Define $b_2 = a_1^2 + 4a_2$, $b_4 = 2a_4 + a_1a_3$ and $b_6 = a_3^2 + 4a_6$. Then there exists an isogeny $\phi : E \to \tilde{E}$ with $\ker \phi = G$, of the form:

$$\phi(X, Y) \mapsto (\phi_1(X), \phi_2(X, Y)) = \left(\frac{A(X)}{\psi(X)^2}, \frac{B(X, Y)}{\psi(X)^3} \right) \tag{3.1}$$

□

By letting $p(X) := 4X^3 + b_2X^2 + 2b_4X + b_6$ and $c_1 := \prod_{P \in G_1} x(P)$, the function for the first coordinate of the isogeny in 3.1 becomes

$$\phi_1(X) = (2d + 1)X - c_1 - p(X) \left(\frac{\psi(X)'}{\psi(X)} \right)' - \frac{p(X)'}{2} \left(\frac{\psi(X)'}{\psi(X)} \right). \tag{3.2}$$

Formula 3.2 shows that the isogeny is actually completely defined by its kernel polynomial. Indeed, evaluating the isogeny requires the evaluation of the derivatives of ψ, and $c_1 = \pm\psi(0)$.

Our interest lies in finding a lower bound bound on the number of operations required to simply evaluate an isogeny. Interestingly, a Theorem by Baur and Strassen in [1], which we found in [5, 7.7 Derivative Inequality], proves that the number of operations required to evaluate the derivative of a function is at most 3 times the number needed for the evaluation of the function itself. Therefore, we can evaluate ψ, its first and second derivative, as well as c_1 in time proportional to that of just evaluating ψ.

Thus, proving a lower bound on the number of operations required to evaluate an isogeny comes down to finding one for its kernel polynomial.

As we started our research, we thought it might be possible to write the kernel polynomial as a function of the isogeny. Observing formula 3.1, it would seem like $\psi(X)^2 = A(X)\phi_1(X)$. Unfortunately, if we develop the terms appearing in formula 3.2 we observe that $A(X)$ is also implicitly defined by ψ. The kernel polynomial ψ does not actually appear explicitly in the formula for the x-coordinate of the isogeny.

$$\left(\frac{\psi(X)'}{\psi(X)}\right) = \sum_{P\in G_1} \frac{1}{(X - x([s]P))}$$
$$\left(\frac{\psi(X)'}{\psi(X)}\right)' = \sum_{P\in G_1} \frac{-1}{(X - x([s]P))^2}$$

In the next section, we present a new upper bound using a method discovered by Bernstein et al. in [2]. It can evaluate a degree-l isogeny in time $\tilde{\mathcal{O}}(\sqrt{l})$ for elliptic curves defined over $\mathbb{F}_q$.

3.1 Computing the kernel polynomial in $\tilde{\mathcal{O}}(\sqrt{l})$

This section presents the paper *Faster computation of isogenies of large prime degree* by Bernstein et al. Therein, the authors adapt a *baby-step, giant-step*-type algorithm to the context of elliptic curves to speed up the computation of the kernel polynomial ψ. Obviously, we could compute ψ in time $\tilde{\mathcal{O}}(l)$ by simply computing all $\{x(P), x([2]P), \ldots, x([(l-1)/2]P)\}$ iteratively. Looking at the latter, we wonder wether it would be possible to obtain some of the values by using the ones previously computed, rather than compute *all* the multiples of P.

We first generalize the problem by considering the evaluation of a similar polynomial in $\mathbb{F}_q$ of the form

$$h_S(X) = \prod_{s\in S}(X - f([s]P)) \in \mathbb{F}_q[X]$$

where q is a prime power, S is a finite subset of $\mathbb{Z}$, P a generator of a cyclic group $\mathcal{G}$, and $f : \mathcal{G} \to \mathbb{F}_q$ is a function.

We first consider a simpler case of h to demonstrate the idea for reducing the number of steps required for evaluation. Let $\mathcal{G}$ be a cyclic group generated by an n-th root of unity $P = \zeta$ of $\mathbb{F}_q[X]$, $f([s]P) = \zeta^s$ and $S = \{0, \ldots n-1\}$. Suppose also that $n = ab$ where $a, b \in \mathbb{N}$.

$$h_S(X) = \prod_{s\in S}(X - f([s]P)) = \prod_{k=0}^{n-1}(X - \zeta^k) = \prod_{i=0}^{a-1}\prod_{j=0}^{b-1}(X - \zeta^{i\cdot b+j})$$

The algorithm we construct relies on two polynomials B, G which can be thought of as the baby-step and giant-step polynomials respectively.

$$G(Y) = \prod_{i=0}^{a-1} (X - Y \cdot \zeta^{i \cdot b}) \in \mathbb{F}_q[X][Y]$$
$$B(Y) = \prod_{j=0}^{b-1} (Y - \zeta^{j}) \in \mathbb{F}_q[Y]$$

Recall that if p, q are univariate monic polynomials with coefficients in an integral domain, the resultant $Res(p, q)$ is $\prod_{i=1}^{d_p} q(\alpha_i)$, where d_p is the degree of p, and the α_i are the roots of p. This tool allows us to *combine* the polynomials B, G by evaluating G in the b roots of B. We therefore need only to compute the baby steps $\left\{\zeta^1, \zeta^2, \ldots, \zeta^{b-1}\right\}$ and the giant steps $\left\{\zeta^b, \zeta^{2b}, \ldots, \zeta^{(a-1)b}\right\}$, for a total of $\mathcal{O}(a + b)$ calls to f.

$$Res_Y(B, G) = \prod_{j=0}^{b-1} G(\zeta^j) = \prod_{j=0}^{b-1}\prod_{i=0}^{a-1} (X - \zeta^j \cdot \zeta^{i \cdot b}) = \prod_{j=0}^{b-1}\prod_{i=0}^{a-1} (X - \zeta^{i \cdot b + j})$$
$$= \prod_{k=0}^{n-1} (X - \zeta^k) = h(X) \in \mathbb{F}_q[X]$$

Note that $Res_Y(B, G)$ evaluates the resultant of B, G viewed as polynomials in the Y variable, so the result is another polynomial in the X variable.

We can generalize this example if we suppose that $S = (I + J) \cup K$, where I, J, K are finite subset of $\mathbb{Z}$ such that $(I + J) \cap K = \emptyset$, and such that I and J have *no common differences*, that is $i_1 - i_2 \neq j_1 - j_2$ for all $i_1, i_2 \in I, j_1, j_2 \in J$. If this is the case then the map $I \times J \to I + J, (i, j) \mapsto i + j$ is a bijection. We can compute the following:

$$h_I(Y) = \prod_{i \in I} (Y - \zeta^i) \in \mathbb{F}_q[Y]$$
$$H_J(X, Y) = \prod_{j \in J} (X - Y \cdot \zeta^j) \in \mathbb{F}_q[X][Y]$$
$$h_{I+J}(X) = Res_Y(h_I, H_J) = \prod_{i \in I} H_J(\zeta^i) = \prod_{i \in I}\prod_{j \in J} (X - \zeta^{i+j})$$
$$= \prod_{(i,j) \in I \times J} (X - \zeta^{i+j}) = \prod_{l \in I+J} (X - \zeta^l) \in \mathbb{F}_q[X]$$
$$h_K(X) = \prod_{k \in K} (X - \zeta^k) \in \mathbb{F}_q[X]$$
$$h_S(X) = h_{I+J}(X) \cdot h_K(X) = \prod_{l \in I+J} (X - \zeta^l) \cdot \prod_{k \in K} (X - \zeta^k)$$
$$= \prod_{s \in (I+J) \cup K} (X - \zeta^s) \in \mathbb{F}_q[X]$$

Evaluating the above functions in $X = \alpha$ would require the computation of $\{\zeta^i, \zeta^j, \zeta^k | i \in I, j \in J, k \in K\}$ for a total of $\tilde{\mathcal{O}}(\max\{\#I, \#J, \#K\})$ operations, where $\tilde{\mathcal{O}}$ is uniform over $\mathbb{F}_q$. Since $\#S = \#I \cdot \#J + \#K$, the minimum of $\max\{\#I, \#J, \#K\}$ is attained when these sets are of size $\tilde{\mathcal{O}}(\sqrt{S})$.

Coming back to the evaluation of the kernel polynomial, we let $f(\cdot) = x(\cdot)$ and take $S := \{1, \dots, (l-1)/2\}$. We obtain the kernel polynomial.

$$h_s(X) = \prod_{s \in S} (X - x([s]P))$$

Unfortunately, the resultant *trick* does not work here since

$$Res_Y(h_I, H_J) = \prod_{(i,j) \in I \times J} (X - x([i]P)x([j]P)) \neq \prod_{(i,j) \in I \times J} (X - x([i+j]P))$$

An important result from this paper is the discovery of the *elliptical resultant* which allows us to use the construction from above in the context of elliptic curve. It depends on the existence of biquadratic polynomials for elliptic curves.

Lemma 3.4 (Biquadratic polynomial for elliptic curves [2, Lemma 4.3])**.** Let q be a prime power and let $E/\mathbb{F}_q$ be an elliptic curve. There exists polynomials F_0, F_1 and F_2 in $\mathbb{F}_q[X_1, X_2]$ such that

$$(X - x(P+Q))(X - x(P-Q)) = X^2 + \frac{F_1(x(P), x(Q))}{F_0(x(P), x(Q))} X + \frac{F_2(x(P), x(Q))}{F_0(x(P), x(Q))}$$

for all $P, Q \in E$ such that $\mathcal{O}_E \notin \{P, Q, P+Q, P-Q\}$

Example. If a curve E is in Montgomery form $By^2 = x^3 + Ax^2 + x$ then the biquadratic polynomials are

$$\begin{aligned} F_0(X_1, X_2) =& (X_1 - X_2)^2 \\ F_1(X_1, X_2) =& -2((X_1X_2 + 1)(X_1 - X_2) + 2AX_1X_2) \\ F_2(X_1, X_2) =& (X_1X_2 - 1)^2 \end{aligned}$$

We also need to slightly change the way we decompose the set S.

Definition 3.1 (Index system [2, Definition 4.6])**.** Let I, J be finite sets of integers.

1. We say that (I, J) is an *index system* if the maps $I \times J \to \mathbb{Z}$ defined by $(i, j) \mapsto i + j$, $(i, j) \mapsto i - j$ are both injective and have disjoint images. Then $I + J$ and $I - J$ are both in bijection with $I \times J$.

2. If S is a finite subset of $\mathbb{Z}$, then we say that an index system (I, J) is an *index system for S* if $I + J$ and $I - J$ are both included in S.

We write $I \pm J$ for the union of I and J.

Now let $P \in E$ be a point of order n and let (I, J) be an index set such that $I, J, I+J, I-J$ do not contain any element of $n\mathbb{Z}$. The *elliptic resultant* $E_J(X,Z) \in \mathbb{F}_q[X][Z]$ can now be defined as

$$E_J(X,Z) = \prod_{j\in J} F_0(Z, x([j]P))X^2 + F_1(Z, x([j]P))X + F_2(Z, x([j]P))$$

We use this formula along with $\Delta_{I,J} = Res_Z(h_I(Z), D_J(Z)) \in \mathbb{F}_q[X]$ and $D_J(Z) = \prod_{j\in J} F_0(Z, x([j]P)) \in \mathbb{F}_q[Z]$, to compute $h_{I\pm J}(X) \in \mathbb{F}_q[X]$. Indeed,

$$\begin{aligned}
h_{I\pm J}(X) &= \prod_{(i,j)\in I\times J} (X - x([i+j]P))(X - x([i-j]P)) \\
&= \prod_{i\in I}\prod_{j\in J}\left(X^2 + \frac{F_1(x([i]P), x([j]P))}{F_0(x([i]P), x([j]P))}X + \frac{F_2(x([i]P), x([j]P))}{F_0(x([i]P), x([j]P))}\right) \\
&= \frac{\prod_{i\in I} E_J(X, x([i]P))}{\prod_{i\in I}\prod_{j\in J} F_0(x([i]P), x([j]P))} = \frac{\prod_{i\in I} E_J(X, x([i]P))}{\prod_{i\in I} D_J(x([i]P))} \\
&= \frac{Res_Z(h_I(Z), E_J(X,Z))}{\Delta_{I,J}}
\end{aligned}$$

Therefore, using the following sequence of computations, we can evaluate $h_{I\pm J}(X)$ in α.

$$\begin{aligned}
h_I(Z) &= \prod_{i\in I}(Z - x([i]P)) \in \mathbb{F}_q[Z] \\
D_J(Z) &= \prod_{j\in J} F_0(Z, x([j]P)) \in \mathbb{F}_q[Z] \\
\Delta_{I,J} &= Res_Z(h_I(Z), D_J(Z)) \in \mathbb{F}_q \\
E_J(Z) &= \prod_{j\in J} F_0(Z, x([j]P))\alpha^2 + F_1(Z, x([j]P))\alpha + F_2(Z, x([j]P)) \in \mathbb{F}_q[Z] \\
R &= Res_Z(h_I(Z), E_J(Z)) \in \mathbb{F}_q \\
h_{I\pm J} &= \frac{R}{\Delta_{I,J}} \in \mathbb{F}_q \\
h_{S\setminus(I\pm J)} &= \prod_{k\in S\setminus(I\pm J)} (\alpha - x([k]P)) \in \mathbb{F}_q \\
h_S &= h_{S\setminus(I\pm J)} \cdot h_{I\pm J} \in \mathbb{F}_q
\end{aligned}$$

We can observe that computing h_S only requires computing $x([l]P)$ for $l \in I \cup J \cup (S \setminus (I \pm J))$. The same complexity bound from above also applies.

Next, we show how to actually compute the kernel polynomial $\psi(X)$ using the above calculations. In order to actually compute the kernel polynomial of an l-isogeny $\phi : E \to E'$, we set P a generator of $\ker\phi$ and $S = \{1, 3, \dots, l-2\}$. With $b = \lfloor\sqrt{l-1}/2\rfloor, b' = \lfloor(l-1)/4b\rfloor$, we define the index system for S as

$$I = \{2b(2i+1) | 0 \le i < b'\}, \qquad J = \{1, 3, \dots, 2b-1\}$$

We can now compute ψ using the above.

$$\psi(X) = h_S(X) = \prod_{s\in S}(X - x([s]P))$$

Example. If our elliptic curves are given in Montgomery form $E : y^2 = x^3 + Ax^2 + x$, we can evaluate the isogeny $\phi : E \to E/\mathcal{G}$ using the above, where $\mathcal{G}$ is a finite group of E generated by a point P. Costello and Hisil showed in [8] that we can represent the curve as

$$\phi : (X, Y) \mapsto (\phi_x(X), c_0 Y \phi'(X))$$

where $c_0 = \prod_{s=0}^{l/2} x([s]P)$, and

$$\phi_x(X) = X \prod_{s=0}^{l} \frac{x([s]P)X - 1}{X - x([s]P)}$$

We see that

$$\phi_x(X) = \frac{X^l \cdot h_S(1/X)}{h_S(X)^2}, \quad \text{where } S = \{1, 3, \dots, l-2\}$$

Therefore, we only need to apply the formulas twice.

This completes the presentation of the new method for computing kernel polynomials. We conclude by mentioning that for now, the $\tilde{\mathcal{O}}(\sqrt{l})$ upper bound on the number of operation required is the lowest known. If this bound were to be optimal, it would be interesting to prove. Otherwise, even a theoretical lower bound would be useful, which is what we will show in the rest of the report.

Chapter 4

Algebraic Complexity & Computation Machines

In this chapter, we will introduce some theory concerning abstract computation machines. These machines are useful because they allow us to model circuits performing calculations in an algebraic setting. Moreover, they often exhibit interesting properties which can be exploited using theory from many fields including analysis, topology and algebraic geometry. They can also be used to prove a certain lower bounds on the number of elementary operations required to compute a formula, as we will demonstrate in the following chapters.

This chapter only presents the basic definitions of such machines. The main result we are interested in is the Canonical Path Theorem 4.2. Given a certain machine capable of solving a decision problem, in certain settings this theorem proves the existence of a polynomial vanishing exactly over the decision set. It will be indispensible to prove the results from Chapter 5 and Chapter 6.

Unless stated otherwise, all results and definitions and results in this chapter are taken from *Complexity and Real Computation*, Chapter 2: Definitions and First Properties of Computation [4].

Definition 4.1 (Finite dimensional machine [4, Definition 1 on p. 40])**.** Let K be a field. A *finite-dimensional machine* M over K consists of a finite directed graph with four types of nodes: *input, computation, branching, output.* The unique input node has no incoming edges and only one outgoing edge. All other types of nodes may have possibly several incoming edges. Computation nodes have only one edge and branching nodes have exactly two, depending on a **YES/NO** result.

The machine also has three spaces: an *input space* $\mathcal{I}_M$, an *state space* $\mathcal{S}_M$ and an *output space* $\mathcal{O}_M$. These are all subsets of K^n, K^m, K^l respectively for n, m, l positive integers.

Associated to each of these nodes of the graph are maps of these spaces and *next node* assignments.

- an input node has a linear map $I : \mathcal{I}_M \to \mathcal{S}_M$ and a unique next node β_1.
- a computation node ν has a computation map which is a rational polynomial $g_\nu : \mathcal{S}_M \to \mathcal{S}_M$ and a unique next node β_ν.
- a branch node has a branching function which is a non-zero polynomial function $h_\nu : \mathcal{S}_M \to K$. The next node along the **YES** edge, β_ν^+ is associated with the condition $h_\nu(z) = 0$ and the next node along the **NO** outgoing edge, β_ν^- with $h_\nu(z) \neq 0$.
- an output node ν has an associated linear map $O_\nu : \mathcal{S}_M \to \mathcal{O}_M$ and no next node.

Figure 4.1 is an example of such a machine. It solves the decision problem 5.5 from Chapter 5, deciding whether $z \in \{1, \dots, k\}$. In it, the input node is red, the computation nodes are blue, the output nodes are black, and the branch nodes are black. The input space $\mathcal{I}_M$ is $\mathbb{C}^3$, the output space $\mathcal{O}_M$ is $\{\texttt{Yes}, \texttt{No}\}$, and its state space is $\mathbb{C}^4$.

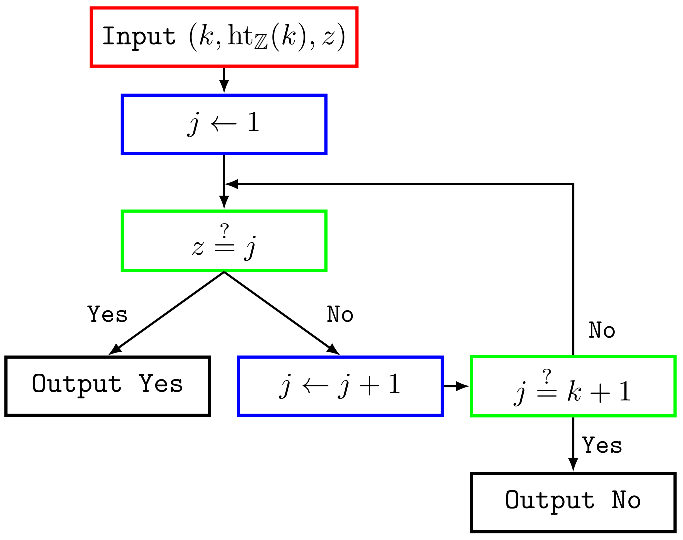

Figure 4.1: Finite-dimensional machine deciding the Twenty Questions problem

Given a machine M we can define a function which can execute it, by operating on the machine's nodes and state. We call it the *computing endomorphism*.

Definition 4.2 (Computing endomorphism [4, 2.2 on p. 44])**.** Let K be a field, and and M a machine over K. Let $\mathcal{N} = \{1, \ldots, N\}$ be the set of nodes with 1 being the input and N the output node. We call the space of node/state pairs $\mathcal{N} \times \mathcal{S}$ the *full state space* of M. Associated with M is the natural *computing endomorphism*:

$$H : \mathcal{N} \times \mathcal{S} \to \mathcal{N} \times \mathcal{S}$$

of the full state to itself. It maps a node/state pair (ν, x) to (ν', x') determined by the graph of M and its associated maps.

Described explicitly, we define *next node* assignments and *computation maps* for each node $\nu \in \mathcal{N}$. Let $\beta_\nu = N$ for $\nu = N$, and $g_\nu(x) = x$ for $\nu = 1, N$, or a branch node. Let $\mathcal{B} \subset \mathcal{N}$ be the subset of branch nodes of M and $\mathcal{C} = \mathcal{N} \setminus \mathcal{B}$ the computation nodes. Then

$$\begin{aligned} H(\nu, x) =& (\beta_\nu, g_\nu(x)) \text{ for } \nu \in \mathcal{C} \\ H(\nu, x) =& \begin{cases} (\beta_\nu^-, g_\nu(x)) \text{ if } h_\nu(x) = 0 \\ (\beta_\nu^+, g_\nu(x)) \text{ if } h_\nu(x) \neq 0 \end{cases} \quad \text{for } \nu \in \mathcal{B} \end{aligned}$$

This map can be iterated to simulate an execution of the machine, since each step of the computation is an application of this map.

The next definition pertains to the *time* required to execute the machine on certain inputs, and define the *input-output map* which is the function representing the machine.

Definition 4.3 ([4, 2.2 on p. 44])**.** Let $x \in \mathcal{I}_M$ and let $x^0 = I(x) \in \mathcal{S}_M$. The *initial point* $z^0 = (1, x^0) \in \mathcal{N} \times \mathcal{S}$ generates the *computation* $z^0, z^1, \ldots, z^k, \ldots$. It is the orbit of z^0 under iterates of H.

$$\begin{aligned} z^0 &= (1, x^0) \\ z^1 &= (\nu^1, x^1) = H(z^1) \\ &\vdots \\ z^k &= (\nu^k, x^k) = H(z^{k-1}) \end{aligned}$$

Let $\pi_{\mathcal{N}} : \mathcal{N} \times \mathcal{S} \to \mathcal{N}$ be the projection of the full state space onto $\mathcal{N}$. Then the sequence of nodes

$$\nu^0 = 1, \nu^1, \ldots, \nu^k \ldots \text{ where } \nu^k = \pi_{\mathcal{N}}(z^k) \text{ for } k = 0, 1, \ldots$$

is called the *computation path* γ_x *traversed* by input x.

The computation *halts* if there is a *time* T such such that $z^T = (N, u)$ for some $u \in \mathcal{S}$. If such T exists, the finite sequence $(z^0, z^1, \dots, z^T) \in (\mathcal{N} \times \mathcal{S})^{T+1}$ is called a *halting sequence* and the finite sequence of nodes $(\nu^0, \nu^1, \dots, \nu^T = N) \in \mathcal{N}^{T+1}$ is a *halting path traversed by* x.

The *halting-time function* $T_M(x)$ is the least T such that M *halts* on input x in *time* T. If T does not exists, we define $T_M(x) = \infty$.

The *input-output map* Φ_M is defined as

$$\Phi_M(x) = O(x^T) \in \mathcal{O}_M$$

whenever T exists.

The halting set of M is Ω_M and is the set of all inputs on which M halts.

$$\Omega_M = \{x \in M | T_M < \infty\}$$

Thus $T_M : \Omega_M \to \mathbb{Z}^+$ and $\Phi_M : \Omega_M \to \mathcal{O}_M$.

We can suppose that our machine is in the particular following form. It is actually equivalent to the above definition [4, Proposition 2, Section 2.3].

Definition 4.4 (Normal form [4, 2.3 on p. 48])**.** A machine M over K as defined above is in *normal form* if we assume:

- the set of nodes $\mathcal{N}$ is $\{1, \dots, N\}$, where 1 denotes the input node, and N denotes the output node.
- iterates of the computing endomorphism H are always defined on an initial point $z^0 = (1, I(x))$ for $x \in \mathcal{I}$.
- all branch nodes are *standard*, that is, that $h_\nu(z) = z_1$ for each branch node ν and $z \in \mathcal{S}$. This means that it outputs the first coordinate of z.

The first two conditions do not restrict our previous definition. For the third, we notice we can transform the machine M to a machine M' also over K such that $\Phi_M = \Phi_{M'}$, $\mathcal{S}_{M'} = K \times \mathcal{S}_M$. Therein, we modify each branch node ν by inserting a computation node ν' before it, with $g_{\nu'}(y_0, y) = (h_\nu(y), y)$. We then replace ν by a standard branch node.

Given this new form, we can formulate some more definitions which make branching conditions within the machine easier to manipulate algebraically.

Definition 4.5. Let M be a machine in normal form over K. Let $x \in \mathcal{I}_M$ and let $z^0, z^1, \dots, z^k, \dots$ be a computation with initial input point $z^0 = (1, I(x))$ and $z^k = (\nu^k, x^k) = H^k(1, I(x))$ for $k = 0, 1, \dots$.

Let $\gamma(=\gamma_x)$ be the computation path $\nu^0 = 1, \nu^1, \ldots, \nu^k, \ldots$ and let $\gamma(k)$ be the *initial computation path* $(\nu^0, \nu^1, \ldots, \nu^k) \in \mathcal{N}^{k+1}$ of γ of *length* k. The *initial path set* is

$$\mathcal{V}_{\gamma(k)} = \{x' \in \mathcal{I}_M | \gamma_{x'}(k) = \gamma(k)\}$$

It is the set of points in the input space whose computation paths coincide with γ for the first k steps.

At each step k in the path γ, M *evaluates* a rational map

$$G_{\gamma(k)} : \mathcal{I}_M \to \mathcal{S}$$

defined by $G_{\gamma(k)} = g_{\nu^k} \circ \ldots \circ g_{\nu^0} \circ I$ the composition of the computation maps along the initial path $\gamma(k)$ with I. For $x \in \mathcal{V}_{\gamma(k)}$ we have $G_{\gamma(k)}(x) = x^{k+1} = \pi_{\mathcal{S}} H^{k+1}(1, I(x))$ where $\pi_{\mathcal{S}} : \mathcal{N} \times \mathcal{S} \to \mathcal{S}$ is the projection onto $\mathcal{S}$.

At each branch step k in the path γ, M *evaluates* the *step-k branching function*

$$f_{\gamma(k)} : \mathcal{I} \to R$$

defined by $f_{\gamma(k)} = \pi_1 \circ G_{\gamma(k)}$ where $\pi_1 : \mathcal{S} \times \mathcal{S}$ is the projection onto the first coordinate. When K is a field, $f_{\gamma(k)}$ may be a rational function or a polynomial.

Finally, we define the *left and right branching conditions* along the path $\gamma(k)$.

$$L_{\gamma(k)} = \{f_{\gamma(k')} | k' < k, k' \text{ is a branch step in } \gamma, \nu^{k'+1} = \beta^-(\nu^{k'})\}$$
$$R_{\gamma(k)} = \{f_{\gamma(k')} | k' < k, k' \text{ is a branch step in } \gamma, \nu^{k'+1} = \beta^+(\nu^{k'})\}$$

These are the sets of branching functions which, on input x branch left on equality, and right on inequality respectively.

We can define

$$\mathcal{V}_{\gamma(k)} = \{x \in \mathcal{I}_M | f(x) \neq 0, g(x) = 0, f \in L_{\gamma(k)}, g \in R_{\gamma(k)}\}$$

which is the set of all inputs which follow the same path as x.

4.1 Canonical Path Construction

In this section we present the Canonical Path Theorem. We use in the next chapters to extract a rational polynomial from a decision machine which vanishes only on the decision set.

We must assume that K is an infinite unordered field.

Let M be a machine over K and let $\gamma = \nu^0, \nu^1, \ldots, \nu^k, \ldots$ be a computation path of M. To each branch step k in γ, there is an associated rational branching function $f_{\gamma(k)}$ evaluated by M. A branch step k is *exceptional* if $f_{\gamma(k)}$ is 0 on $\mathcal{V}_{\gamma(k)}$. These are the branches on which for an input from $\mathcal{V}_{\gamma(k)}$, $f_{\gamma(k)}$ evaluates to 0 and branches right. We have $\nu^{k+1} = \beta^+(\nu^k)$ and $\mathcal{V}_{\gamma(k+1)} = \mathcal{V}_{\gamma(k)}$. We call the other branch steps *nonexceptional.* That is all inputs following the canonical path will go left at that branch and $f_{\gamma(k)}(x) \neq 0$.

Let $\mathcal{B}_\gamma = \{k | k \textit{ is a nonexceptional branch step of } \gamma\}$. This is the set of all indices of branches at which computation goes left (**No**).

Definition 4.6 (Canonical path [4, 2.2 Definition 5 on p. 57])**.** The path γ is a *canonical path* of M if $\mathcal{B}_\gamma \neq \emptyset$ and for each $k \in \mathcal{B}_\gamma$, $\nu^{k+1} = \beta^-(\nu^k)$. This path is traversed by inputs that always take the **No** outgoing edge at nonexceptional branch steps. There is at most one of these for a machine M.

We recall that a set $X \subset K^n$ is *Zariski dense* if only the zero polynomial vanishes on X.

Lemma 4.1 ([4, 2.5 Proposition 3 on p. 57])**.**
We can define the initial path set $\mathcal{V}_\gamma$ as

$$\mathcal{V}_\gamma = \{x \in \mathcal{I}_M | f_{\gamma(k)}(x) \neq 0 \text{ for each } k \in \mathcal{B}_\gamma\} = \{x \in \mathcal{I}_M | F_\gamma(x) \neq 0\}$$

where $F_\gamma = \prod_{k \in \mathcal{B}_\gamma} f_{\gamma(k)}$. Since K is infinite, $\mathcal{V}_\gamma$ is Zariski dense in K^n. □

This lemma shows that the condition of K being an infinite field is necessary. Indeed, if K were finite, then initial path set is also finite, and therefore not dense.

Theorem 4.2 (Canonical Path Theorem [4, 2.5 Theorem 5 on p. 47])**.**
Let $S \subset K^n$. Suppose M is a decision machine for S over K and that γ is its canonical path. If S is not Zariski dense in K^n, then $S \cap \mathcal{V}_\gamma = \emptyset$. Hence

$$\Phi_{M|\mathcal{V}_\gamma} \equiv 0$$
$$F_{\gamma|S} \equiv 0$$

That is, the machine when evaluated on V_γ will always output **False**, and the function F_γ always evaluates to 0 on the decision set.

To prove this theorem, we must state an essential lemma on Zariski dense sets.

Lemma 4.3.
Let $X \subseteq K^n$ be dense with respect to the Zariski topology in K^n and suppose $\phi : X \to K$ is a rational function. If the image of ϕ is a finite set, then it consists of a single point. □

Proof of Theorem 4.2. Suppose $S \cap \mathcal{V}_\gamma \neq \emptyset$. Since $\Phi_M|_S \equiv 1$, there must be some elements $x \in \mathcal{V}_\gamma$ such that $\Phi_M(x) \equiv 1$. The image of Φ is the finite set $\{0, 1\}$, and therefore the image of $\Phi_M|_{\mathcal{V}_\gamma}$ must be the single value 1, by Lemma 4.3. The set S is the set for which M returns 1, so we must have $\mathcal{V}_\gamma \subseteq S$. This implies that S is also Zariski dense. $\square$

We are now fully equipped to tackle the next chapter in which we use these machines to evaluate decision problems. The power of Theorem 4.2 is that it proves the existence of a polynomial vanishing only on S. This does not require us the actually define the machine but only suppose it exists!

Chapter 5

Computing $k!$

In this chapter, we present the main result from Shub and Smale in "On the intractability of Hilbert's Nullstellensatz and an algebraic version of P vs. NP"[13]. Is states that which states that if computing $k!$ is *hard*, then $\mathbf{P} \neq \mathbf{NP}$ over $\mathbb{C}$.

We begin by recalling the weak form of the Hilbert Nullstellensatz as well as one of its corollaries.

Theorem 5.1 (Hilbert Nullstellensatz (weak)).
Every maximal ideal in the polynomial ring $R = \mathbb{C}[x_1, \dots, x_n]$ has the form

$$(x_1 - a_1 m \dots, x_n - a_n)$$

for some $a_1, \dots, a_n \in \mathbb{C}$.

Corollary 5.1.1.
Given complex polynomials $f_1, \dots, f_l : \mathbb{C}^n \to \mathbb{C}$ of degree d_i, for $i = 1, \dots, l$, there exists $z \in \mathbb{C}^n$ such that $f_i(z) = 0$ for all i, if and only if, there also exists polynomials $g_1, \dots, g_l : \mathbb{C}^n \to \mathbb{C}$ for $i = 1, \dots, l$, such that

$$\sum_{i=1}^{n} g_i f_i = 1$$

We are actually more interested in the latter, as we can use it to define two computation problems.

Problem 5.2 (Effective Hilbert Nullstellensatz).
Given complex polynomials $f_1, \dots, f_l : \mathbb{C}^n \to \mathbb{C}$, *find* the polynomials $g_1, \dots, g_l : \mathbb{C}^n \to \mathbb{C}$ such that

$$\sum_{i=1}^{n} g_i f_i = 1$$

Problem 5.3 (Decisional Hilbert Nullstellensatz)**.**
Given complex polynomials $f_1, \dots, f_l : \mathbb{C}^m \to \mathbb{C}$ of degree d_i, for $i = 1, \dots, l$, *decide* if there exist $z \in \mathbb{C}^m$ such that $f_i(z) = 0$ for all i.

Both these problems are believed to be intractable, that is hard to compute in polynomial time. No proof exist though so we must formulate it as a conjecture, which we believe is true.

Conjecture 5.4 (Intractability of the Decisional Hilbert Nullstellensatz)**.**
There is no polynomial time algorithm over $\mathbb{C}$ which solves the Decisional Hilbert Nullstellensatz 5.3.

It was proved in [3] that this decision problem is $\mathbf{NP}_{\mathbb{C}}$-complete, and therefore proving that it is in $\mathbf{P}_{\mathbb{C}}$ would prove that $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$. Or in a different form

$$\mathbf{P}_{\mathbb{C}} \neq \mathbf{NP}_{\mathbb{C}} \iff \text{Conjecture 5.4 is true}$$

5.1 Twenty Questions

Twenty Questions is originally a spoken parlor game popular in the 19th century. One player chooses an object or subject that the others must guess, by asking only questions with a yes/no answer. By asking 20 questions, one could theoretically decide what the subject is from a set of 2^{20} possibilities.

The following problem is given in [13] with a similar idea in mind.

Problem 5.5 (Twenty Questions ($\mathbb{C}$))**.**
The *Twenty Question over* $\mathbb{C}$ is the following decision problem.

$$\begin{aligned} \textbf{input:} &\ (k, \mathrm{ht}(k), z) \in \mathbb{N} \times \mathbb{N} \times \mathbb{C} \\ \textbf{decide:} &\ z \in \{1, 2, \dots, k\} \end{aligned}$$

where $\mathrm{ht}(k) := \lfloor \log k \rfloor$.

In Chapter 4, we gave an example of a machine solving this problem in an unordered field. It must check every possibility and therefore runs in time $\mathcal{O}(k)$. If the field is ordered though, we can use a binary search algorithm in time $\mathcal{O}(\mathrm{ht}(k))$.

We first introduce a slightly broader decision problem (Y, Y_{yes}) as follows

Define $Y := \mathbb{C}^\infty$ as our alphabet and let our language be

$$Y_{\text{yes}} := \bigcup_{k=1}^{\infty} \left\{ (k, \mathrm{ht}(k), z_1, \dots, z_{\mathrm{ht}(k)}, 0, \dots) : z_1 \in \{1, 2, \dots, k\} \right\}$$

The embedding of 20 questions into (Y, Y_{yes}) becomes:

$$(k, \mathrm{ht}(k), z) \mapsto (z, \mathrm{ht}(k), z, 1, \dots, 1, 0, \dots)$$

where the number of ones is $\mathrm{ht}(k) - 1$.

Lemma 5.6 ([4, 7.3 Lemma 6 on p. 137])**.**
The (Y, Y_{yes}) decision problem is in $\mathbf{NP}_{\mathbb{C}}$.

Proof. We first consider an input/witness pair $x, \omega \in Y \times Y$ given by:

$$x := (u_1, u_2, z_1, \ldots, z_n)\,, \qquad \omega := (w_0, \ldots, w_{n-1}, v_0, \ldots, v_{n-1}, y_0, \ldots, y_{n-1})\,.$$

Such a witness is an element of the alphabet Y of size polynomial in the size of the input. It is given to a predicate $R(x, \omega)$ which indicates whether $x \in Y_{\text{yes}}$ or not. More precisely, the witness defines three positive integers in binary form all strictly less 2^n

$$a = \sum_{i=0}^{n-1} 2^i v_i, \quad b = \sum_{i=0}^{n-1} 2^i y_i, \quad c = \sum_{i=0}^{n-1} 2^i w_i < 2^n$$

and such that

$$u_2 = c, \quad z_1 = 1 + a, \quad u_1 = z_1 + b$$

If the input is correct, then then $n = u_2 = \text{ht}(k)$ is a positive integer, and $z_1, u_1 = k$ are also positive integers satisfying $1 \le z_1 \le k$.

Algorithm 1 explicitly describes this procedure and runs in time cu_2 for some constant $c > 0$. That is, its runtime is $\mathcal{O}(\log k))$ when the input is correct.

On input $x = \big(k, \text{ht}(k), z_1, \ldots, z_{\text{ht}(k)}\big) \in Y_{\text{yes}}$, the machine answers **True**.
Therefore, the decision problem (Y, Y_{yes}) is in $\mathbf{NP}_{\mathbb{C}}$. □

Using the framework from Chapter 4, we can establish that there are machines over $\mathbb{C}$ which are able to solve Problem 5.5. We are more interested though in machines over $\mathbb{Z}$ solving Problem 5.7 which is the same as Problem 5.5 over a different ring.

Problem 5.7 (Twenty Questions ($\mathbb{Z}$))**.**

input: $(k, \text{ht}(k), z) \in \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{Z}$
decide: $z \in \{1, 2, \ldots, k\}$

where $\text{ht}(k) := \lfloor \log k \rfloor$.

The first important tool which we state here is used for the proof of Theorem 5.10. We will also use it in the next section.

Proposition 5.8 (Elimination of Constants [4, 7.4 Proposition 9 on p. 136])**.**
Let $K \subset L$ be fields where $K \subset \overline{\mathbb{Q}}$. Let (Y, Y_{yes}) be a decision problem solved by a machine M over L. Then there is a machine M' over K solving the restriction of (Y, Y_{yes}) to K and a constant $c \in \mathbb{N}$ such that

$$T_{M'}(y) \le (T_M(y))^c \qquad \text{for all } y \in Y \cap K^\infty$$

□

Algorithm 1: Verify Twenty Questions Questions

```
Input: element $x := (u_1, u_2, z_1, \ldots, z_n) \in Y$
       witness $\omega := (w_0, \ldots, w_{n-1}, v_0, \ldots, v_{n-1}, y_0, \ldots, y_{n-1}) \in Y$
Output: $x \overset{?}{\in} Y_{\text{yes}}$
1  Check $u_2$ is a positive integer by addition of 1s
2  Check size of $x$ is $u_2 + 2$ and size of $\omega$ is $4u_2$
3  $n \leftarrow u_2$
4  Check $w_{n-1} \equiv 1$                          // ht is correct
5  Check $u_1 \equiv \sum_{i=0}^{n-1} 2^i w_i$           // $u_2 \equiv \text{ht}(u_1)$
6  for $i = 0, \ldots, n-1$ do
7      Check $w_i(1 - w_i) \equiv 0$                 // all $w_i$s are 0 or 1
8      Check $v_i(1 - v_i) \equiv 0$                 // all $v_i$s are 0 or 1
9      Check $y_i(1 - y_i) \equiv 0$                 // all $y_i$s are 0 or 1
10 end
11 $a \leftarrow \sum_{i=0}^{n-1} 2^i v_i$
12 $b \leftarrow \sum_{i=0}^{n-1} 2^i y_i$            // $a, b$ are positive integers $\leq 2^{u_2}$
13 Check $z_1 \equiv 1 + a$                    // $z_1$ is a positive integer $\geq 1$
14 Check $u_1 \equiv z_1 + b$                  // $z_1$ is a positive integer $\leq u_2$
15 if all Check pass then return $\top$
16 else return $\bot$
```

It states that solving a problem in a subfield requires only polynomially more time than solving it in the original field.

The following theorem is not directly applied in this chapter, but is required for the proof of Proposition 5.8. We present it for completeness.

Definition 5.1 (Witness)**.** Let $f \in \overline{\mathbb{Q}}[t_1, \ldots, t_s]$.

A *witness* $w \in \overline{\mathbb{Q}}^s$ for f satisfies the property:

$$f(w) = 0 \implies f \equiv 0$$

The definition of a witness alone is not particularly useful without the following theorem proving the existence of one under some condition.

Theorem 5.9 (Witness Theorem [4, 7.3 Theorem 4 on p. 129])**.**
Let $F(x, t) = F(x_1, \ldots, x_r, t_1, \ldots, t_s)$ be a polynomial in $r + s = n$ variables with coefficients in $\mathbb{Z}$ and let $F_x(t) \in \overline{\mathbb{Q}}[t_1, \ldots, t_s]$ be defined by $= F(x, t)$ for each $x \in \overline{\mathbb{Q}}^r$. If $N \in \mathbb{N}$ satisfies

$$\log N \geq 4n\tau^2 + 4\tau, \qquad \tau = \tau(F)$$

Then for all $x \in \overline{\mathbb{Q}}^r$, there exists an algebraic number w_1 over the set $\{2^N, x_1^N, \ldots, x_r^N\}$ such that the point $w = (w_1, \ldots, w_s)$, with $w_i = w_{i-1}^N$ for $i = 2, \ldots, s$, is a witness for F_x. □

Finally, we can state the following theorem which bounds the time required to decide Problem 5.7 over $\mathbb{Z}$ by the time required to decide Problem 5.5 over $\mathbb{C}$.

Theorem 5.10 ([4, 7.5 Theorem 6 on p. 139]).
If $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$, then Problem 5.7 (*twenty questions* over $\mathbb{Z}$) is tractable.

Proof. By Lemma 5.6, (Y, Y_{yes}) is in $\mathbf{NP}_{\mathbb{C}}$ and by hypothesis it is in $\mathbf{P}_{\mathbb{C}}$. Therefore, there is a machine M which decides $\mathbf{NP}_{\mathbb{C}}$ in time $\mathcal{O}((\log k)^c)$. Since we embed *twenty questions* into (Y, Y_{yes}), the restriction of the machine M to *twenty questions* over $\mathbb{C}$ also decides this latter problem in time $\mathcal{O}((\log k)^c)$. Finally, by the Elimination of Constants Proposition 5.8, there is a machine M' which decides *twenty questions* over $\mathbb{Z}$ in time $\mathcal{O}((\log k)^{c'})$. □

5.2 Computing $k!$

In this section we show that a machine M which solves Twenty Questions over $\mathbb{Z}$ in time $\mathcal{O}((\log k)^c)$ actually computes the value $k!$ as an intermediary step.

We first formalize the meaning of computing a value in a defined amount of time.

Definition 5.2 (Cost of computation [4, p. 126]). Let $m \in \mathbb{Z}$ and let $x_0, x_1, \ldots, x_l$ be a computation of m of length l, where $x_0 = 1, x_l = m$. For all $k, 1 \le k \le l$ there are integers $i, j, 0 \le i, j < k$ such that $x_k = x_i \circ x_j$, where $\circ$ is an addition, subtraction or multiplication operation.

Define $\tau : \mathbb{Z} \to \mathbb{N}$ by letting $\tau(m)$ be the minimum length of a computation of m.

We can also extend this definition to polynomials in several variables.

Let $G \in \mathbb{Z}[t_1, \ldots, t_n]$ and consider the finite sequence

$$(u_0 = 1, u_1 = t_1, \ldots, u_n = t_n, u_{n+1}, \ldots, u_{n+l} = G)$$

where for $n < k \le n + l$, we have $u_k = v \circ w$ for some $v, w \in \{u_0, \ldots, u_{k-1}\}$

$\tau(G)$ is the minimum length of such a sequence.

A simple bound on τ for integers is the following.

Proposition 5.11 ([4, 7.1 Proposition 1 on p. 126]).
For all $m \in \mathbb{Z}^+$ we have

$$\tau(m) \le 2\log(m)$$

Proof. This is trivial using a double and add algorithm, since m can be written as:

$$m = \sum_{i=0}^{\lfloor \log(m) \rfloor} m_i 2^i$$

where $m = (m_0, \ldots, \lfloor \log(m) \rfloor)$.

The 2^is are all computed using $\lfloor \log(m) \rfloor + 1$ multiplications, and we then use at most $\lfloor \log(m) \rfloor + 1$ additions to obtain m. $\square$

This brings us to wonder about the following problem.

Problem 5.12 (Is $k!$ ultimately easy to compute?)**.**
Does there exists a constant $c > 0$ such that

$$\tau(k!) \leq (\log k)^c \qquad \text{for all } k \in \mathbb{Z}^+$$

This problem can be rephrased using the following definition.

Definition 5.3 ((ultimately) easy to compute sequences)**.** Given a sequence of integers a_k, we say that a_k is *easy to compute* if there is a constant c such that $\tau \leq (\log k)^c$ for all $k > 2$.

It is *hard to compute* otherwise.

A sequence a_k is *ultimately easy to compute* if there are nonzero integers m_k such that the sequence $m_k a_k$ is easy to compute.

It is *ultimately hard to compute* otherwise.

We can now state and prove the main result from [13].

Theorem 5.13 ([4, 7.1 Theorem 2 on p. 126])**.**
If the sequence $k!$ is ultimately hard to compute, then $\mathbf{P}_{\mathbb{C}} \neq \mathbf{NP}_{\mathbb{C}}$.

Since it is believed to be hard to show that $\mathbf{P}_{\mathbb{C}} \neq \mathbf{NP}_{\mathbb{C}}$, it would imply that finding a proof showing the hardness of computing $k!$ is also hard itself.

Before proving Theorem 5.13 we need another tool.

Lemma 5.14.
Let $f \in \mathbb{Z}[t_0, t_1, \ldots, t_m]$ have degree d.

If f is zero on every integer point of the cube in $\mathbb{R}^{m+1}$ centered at $(0, \ldots, 0)$, then $f \equiv 0$.

Proof. By induction on m. Let $m = 0$ and $f \in \mathbb{Z}[t]$ be of degree d. Then if $f(k) = 0, k \in \left\{-\lfloor \frac{d+1}{2} \rfloor, \ldots, 0, \ldots, \lfloor \frac{d+1}{2} \rfloor\right\}$, f would have $d+1$ zeros, which is impossible, so $f \equiv 0$.

Suppose the statement is true for $m-1$ and let $f \in \mathbb{Z}[t_0, \ldots, t_m]$ be of degree d.

We can consider f over $\mathbb{Z}[t_0][t_1, \ldots, t_m]$, and by evaluating it in $t_0 \in \left\{-\lfloor \frac{d+1}{2} \rfloor, \ldots, \lfloor \frac{d+1}{2} \rfloor\right\}$, we obtain a polynomial over $\mathbb{Z}[t_1, \ldots, t_m]$. Applying the induction hypothesis over the latter wields the result. $\square$

Finally, we can use all these tools to prove the main theorem of this chapter.

Proof of Theorem 5.13. We show the contrapositive, that is

$$\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}} \implies k! \text{ is ultimately easy to compute}$$

By Theorem 5.10, Twenty Questions over $\mathbb{Z}$ only is tractable. This implies there is a machine M over $\mathbb{Z}$ which solves the decision Problem 5.7 time $< (\log k)^c$. Using the Canonical Path Theorem 4.2, there exists a polynomial F over the input variables $k, \mathrm{ht}(k), t$, which vanishes on the decision set $\{(k, \mathrm{ht}(k), t) : k \in \mathbb{N}, t \in \{1, \dots, k\}\}$

For each k, we consider $F_k \in \mathbb{Z}[t]$ where $F_k(t) = F(k, \mathrm{ht}(k), z)$. This polynomial thus vanishes on $\{1, \dots, k\}$, and $\tau(F_k) \leq (\log k)^c$, since the machine M evaluates F in that same amount of time. The degree of F_k is bounded by $2^{\tau(F_k)}$ since that is the maximum number of operations a machine could perform in time $\tau(F_k)$. Since it is not identically zero, the contrapositive of Lemma 5.14 asserts there is an integer l such that $F_k(l) \neq 0$ and $|l| \leq 2^{\tau(F_k)}$. We take l to be minimal with this property. Using Lemma 5.11 and the above, we have

$$\tau(l) \leq 2\log(l) \leq 2\tau(F_k) \leq 2(\log k)^c$$

Computing $F_k(l)$ can be done by computing l and then computing F_k evaluated at l. Therefore,

$$\tau(F_k(l)) \leq \tau(l) + \tau(F_k) \leq 3(\log k)^c$$

We next argue that $F_k(l)$ must contain $k!$ as factor by checking two possibilities ($l \notin \{1, \dots, k\}$ by construction).

$l > k$: By minimality of l, F_k would be zero on $\{1, \dots, l-1\}$ and then

$$F_k(t) = \prod_{i=1}^{l-1}(t-i)g_1(t), \qquad \text{for some } g_1 \in \mathbb{Z}[t]$$

$l \leq 0$: If this is the case then F_k would be zero on $\{l+1, \dots, k\}$ and then

$$F_k(t) = \prod_{i=l+1}^{k}(t-i)g_2(t), \qquad \text{for some } g_2 \in \mathbb{Z}[t]$$

The evaluation of F_k in l yields a sequence $m_k = F_k(l)/k!$ for each k, and thus the sequence $m_k k! = F_k(l)$ is easy to compute since $\tau(F_k(l)) \leq 3(\log k)^c$, which implies that $k!$ is ultimately easy to compute. $\square$

Chapter 6

Kernel Polynomials & $\mathbf{P}_{\mathbb{C}}$ vs. $\mathbf{NP}_{\mathbb{C}}$

In this chapter, we will try to use a similar construction as Chapter 5 to prove that an *easy* evaluation of the kernel polynomial would imply that $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$. This link might seem far fetched, but to understand our motivation for pursuing this route we must look back to Chapter 3. Indeed, if we take the special case of the polynomial

$$h_S(X) = \prod_{s \in S} (X - s)$$

with $S = 1, \dots, k$, $P = 1$ and $f(x) = x$, then we notice that evaluating h_S in 0 gives $\pm k!$. This motivated our research in adapting the *twenty questions* construction to elliptic curves over $\mathbb{C}$.

6.1 Twenty Questions for Elliptic Curves

We start by formulating the Twenty Questions Problem 5.5 for elliptic curves over $\mathbb{C}$. Throughout, we consider E to be an elliptic curve over $\mathbb{C}$, given in Weierstrass form with coefficients $\{a_i\}_{i=1}^{6}$.

Problem 6.1 (Twenty Questions (Elliptic Curve)).

$$\begin{aligned} \textbf{input:} \ & (P, \operatorname{ht}(P), Q) \in E \times \mathbb{N} \times E \\ \textbf{decide:} \ & Q \in \langle P \rangle \setminus \mathcal{O}_E \end{aligned}$$

where $\operatorname{ht}(P) := \lfloor \log \operatorname{ord}(P) \rfloor$ and $P \neq \mathcal{O}_E$.

In this chapter, we consider machines as defined in Chapter 4, and explain how they handles elliptic curves. First of all, the input to the machine must contain $\{a_i\}_{i=1}^{6}$ which are the coefficients of the curve E we are working on. Next, a point $P \in E$ is simply a tuple of coordinates which our machine can simply store independently. Algorithm 3 in the appendix shows how the point addition $\oplus$ is computed when no point is the point at infinity. If the two points sum to infinity, then the machine will branch and continue accordingly. We notice that this function takes constant time. In our analysis, we will be concerned with the number of point additions we perform. We therefore consider the big-O notation $\mathcal{O}_E(\cdot)$ which counts the number of elementary operations, as well as the number of point additions.

Once again, we consider a broader decision problem (Y, Y_{yes}) in which we embed *twenty questions.*

Let the alphabet be $Y := \mathbb{C}^{\infty}$. For all curves E over $\mathbb{C}$ defined by coefficients $\{a_i\}_{i=1}^{6}$ and all points $P \in E \setminus \mathcal{O}_E$, and all $Q \in \langle P \rangle \setminus \mathcal{O}_E$ we define an element of Y_{yes} as

$$Y_{\text{yes}, E, P, Q} := \left(\{a_i\}_{i=1}^{6}, x(P), y(P), \text{ht}(P), x(Q), y(Q), x_1, \ldots, x_{\text{ht}(P)}, 0, \ldots\right)$$

The last variables are added in order to set the size of the input to $\mathcal{O}(\text{ht}(P))$.

The decision set is then

$$Y_{\text{yes}} := \bigcup_{E/\mathbb{C}} \bigcup_{P \in E \setminus \mathcal{O}_E} \bigcup_{Q \in \langle P \rangle \setminus \mathcal{O}_E} Y_{\text{yes}, E, P, Q}$$

and the embedding from Problem 6.1 into (Y, Y_{yes}) is

$$(P, \text{ht}(P), Q) \mapsto \left(\{a_i\}_{i=1}^{6}, x(P), y(P), \text{ht}(P), x(Q), y(Q), 1, \ldots, 1\right)$$

where there are $\text{ht}(P)$ ones.

As we did previously, we provide an algorithm verifying the decision problem (Y, Y_{yes}) in polynomial time.

Theorem 6.2.
The decision problem (Y, Y_{yes}) is in $\mathbf{NP}_{\mathbb{C}}$.

Proof. Given an input/witness pair (x, ω) given by

$$x := \left(\{a_i\}_{i=1}^{6}, x_P, y_P, u_1, x_Q, y_Q, x_1, \ldots, x_n\right), \omega := (k_1, \ldots, k_n)$$

Algorithm 2 will verify in time $\mathcal{O}_{EC}(\text{ht}(P))$ that x belongs to the decision set Y_{yes}.

The witness simply represents the discrete log of Q in base P written in binary, and it's length is clearly polynomial in the size of the input. The algorithm then performs a double-and-add subroutine to compute $[k]\,P$ and verifies it equals Q. Since the points P, Q are not the points at infinity, and $Q \in \langle P \rangle$, the algorithm will decide $\perp$ whenever it encounters the point at infinity during the computation of $[k]P$. This simplifies our algorithm since we don't need to consider them at all. However, the original double and add algorithm initializes the point R to the point at infinity. In order circumvent this and prevent the failure on line 14, we initialize R to N the first time we enter the condition $k_j = 1$.

Algorithm 2: Verify Twenty Questions Questions (Elliptic Curve)

```
   Input: element $x := \left(\{a_i\}_{i=1}^{6}, x_P, y_P, u_1, x_Q, y_Q, x_2, \ldots, x_n\right) \in Y$
          witness $\omega := (k_1, \ldots, k_n) \in Y$
   Output: $x \overset{?}{\in} Y_{\text{yes}}$
1  Check $u_1$ is an integer by addition of 1s
2  Check size of $x$ is $u_1 + 11$ and size of $\omega$ is $u_1$
3  $n \leftarrow u_2$
4  Check $y_P^2 + a_1 x_P y_P + a_3 y_P \equiv x_P^3 + a_2 x_P^2 + a_4 x_P + a_6$        // $P \in E$
5
6  Check $y_Q^2 + a_1 x_Q y_Q + a_3 y_Q \equiv x_Q^3 + a_2 x_Q^2 + a_4 x_Q + a_6$        // $Q \in E$
7  $(x_N, y_N) \leftarrow (x_P, y_P)$
8  for $i = 1, \ldots, n$ do
9      Check $k_i(1 - k_i) \equiv 0$                    // all $k_i$s are 0 or 1
       // handle all summations to the point at inifinity as
          failures and return $\perp$
10     if $k_i \equiv 1$ then
11         if $i \equiv 1 \vee k_i \prod_{j=1}^{i-1}(1 - k_j) \equiv 1$ then
               // the $k_j = 0$ for $j = 1, \ldots, i-1$ and $k_i = 1$
12             $(x_R, y_R) \leftarrow (x_N, y_N)$
13         else
14             $(x_R, y_R) \leftarrow \oplus(x_R, y_R, x_N, y_N)$
15         end
16     end
17     $(x_N, y_N) \leftarrow \oplus(x_N, y_N, x_N, y_N)$
18 end
19 Check $x_R \equiv x_Q \wedge y_R \equiv y_Q$
20 if all Check pass then return $\top$
21 else return $\perp$
```

$\square$

Our main result from this chapter will consist of showing that if it is true that $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$, then a machine M solving Twenty Questions (EC) in time $\mathcal{O}_E\left((\log \operatorname{ord} P)^c\right)$ actually produces and evaluates a polynomial divisible by

$$\prod_{i=1}^{\operatorname{ord} P-1} \left(X - x\left([i]P\right)\right).$$

This latter polynomial is very similar to the kernel polynomial, and we can adapt the result to obtain a result applicable to it, by redefining our Problem 6.1 to decide if the point Q is in $\{P, [2]P, \ldots, [(l-1)/2]P\}$.

We first adapt a few definitions from Chapter 5.

Definition 6.1 (Cost of computation in Elliptic Curves)**.** Let $P \in E$ be a point of the elliptic curve E, and $k \in \mathbb{N}$ and integer.

Let

$$\Big(u_{-7} = a_1, \ldots, u_{-2} = a_6,\\ u_{-1} = x(P), u_0 = y(P),\\ u_1 \ldots, , u_{l-1},\\ u_l = x([k]P), u_{l+1} = y([k]P)\Big)$$

be a computation of $[k]P$ of length $l + 9$.

For all $1 \leq k \leq l+1$ we have, $u_k = v \circ w$ for some $v, w \in \{u_{-7}, \ldots, u_{k-1}\}$, and some $\circ \in \{+, -, \times, /\}$.

$\tau([k]P)$ is the minimum number l of such a computation.

The same is valid for polynomials $F \in \mathbb{C}[t_1, \ldots, t_n]$. We consider the computation sequence

$$\Big(u_{-7} = a_1, \ldots, u_{-2} = a_6,\\ u_{-1} = x(P), u_0 = y(P),\\ u_1 = t_1, \ldots, u_n = t_n,\\ u_{n+1}, \ldots, u_{n+l} = F\Big)$$

where for $n < k \leq n+l$, we have $u_k = v \circ w$ for some $v, w \in \{u_{-7}, \ldots, u_{k-1}\}$ and $\circ \in \{+, -, \times, /\}$.

$\tau(F)$ is the minimum number l of such a computation.

Proposition 6.3.
For all $P \in E$ and all $k \in \{1, \ldots, \operatorname{ord} P - 1\}$, we have

$$\tau([k]P) = \mathcal{O}_E\left(\log k\right)$$

Proof. This is trivial using a double and add algorithm. □

This now allows us to formulate our final theorem.

Theorem 6.4.
Let $P \in E$. If

$$\psi(X) = \prod_{i=1}^{\operatorname{ord} P-1} (X - x([i]P))$$

is hard to evaluate for all $x \in \mathbb{C}$, then $\mathbf{P}_{\mathbb{C}} \neq \mathbf{NP}_{\mathbb{C}}$.

Proof. Suppose $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$.

Then there exists a machine M over $\mathbb{C}$ which solves the decision problem (Y, Y_{yes}) in time $\mathcal{O}_E\left((\operatorname{ht} P)^c\right)$ for some $c > 0$, since it is in $\mathbf{NP}_{\mathbb{C}}$.

Because $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$, the restriction of this machine to the variables $\{a_1, \dots, a_6, x_P, y_P, \operatorname{ht}, x_Q, y_Q\}$ then also solves *twenty questions* for elliptic curves in time $\mathcal{O}_E\left((\operatorname{ht} P)^c\right)$, by the Elimination of Constants Proposition 5.8. By the Canonical Path Theorem 4.2, there exists a polynomial F in several variables in $\mathbb{C}$ which vanishes only on the decision set.

We consider the polynomial

$$G_{\mathcal{C}}(X) = F(a_1, \dots, a_6, x(P), y(P), \operatorname{ht} P, X, y_Q)$$

where $\mathcal{C} := (a_1, \dots, a_6, x(P), y(P), \operatorname{ht} P, x, y_Q)$.

By construction, $G_{\mathcal{C}}$ vanishes on $\{x(P), x([2]P), \dots, x([\operatorname{ord} P - 1]P)\}$ whenever the input is in the decision set. We obtain the univariate polynomial

$$G_{\mathcal{C}}(X) = \psi(X)g(X), \qquad \text{for some } g \in \mathbb{Q}[\mathcal{C}][X].$$

The polynomial g is over $\mathbb{Q}[\mathcal{C}]$ since the machine performs elementary operations on the variables in $\mathcal{C}$.

We have $\tau(\psi) \leq \tau(G_{\mathcal{C}})$ since the machine must M must compute ψ during the computation of $G_{\mathcal{C}}$. This implies that $\tau(\psi) = \mathcal{O}_E\left((\operatorname{ht}(P))^c\right)$.

Therefore, for any $X \in \mathbb{C}$, $\tau(\psi(X)) = \mathcal{O}_E\left((\operatorname{ht}(P))^c\right)$, and we conclude that the computation of the kernel polynomial is easy.

This proves the contrapositive of the theorem. □

By the preceding theorem, we obtain our final result. Unless $\mathbf{P}_{\mathbb{C}} = \mathbf{NP}_{\mathbb{C}}$, we conclude that the evaluation of the kernel polynomial of an isogeny can not be done in time less that logarithmic in its degree.

6.2 Further work

Unfortunately, the main result does not translate directly to finite fields, as many arguments we use depend on the base field being of infinite characteristic. The Canonical Path Theorem 4.2 for example, requires the initial path set to be Zariski dense. This does not mean however that it cannot be done.

We could for instance look into the Néron-Tate height to try and adapt some results to elliptic curves in general. This would require more background into later chapters of [15], which we hope to do in the future.

Moreover, it seems like Problem 6.1 *twenty questions* over elliptic curves, if defined for finite fields, could be solved by an algorithm solving the discrete logarithm. Therefore, instead of connecting the computation of kernel polynomials to $\mathbf{P} = \mathbf{NP}$, we would connect it to discrete log problem which is also believed to be hard.

There are also many more interesting results from complexity theory which might be applicable. Indeed, the subject is incredibly vast, and the research we conducted using [5] and [4] revealed other avenues we could explore.

It would also be interesting to further pursue the idea of applying this *twenty questions* type of argument to a broader class of functions, such as those described in Chapter 3. These functions are a special case of *q-holonomic functions.*

Finally, the impossibility of a superlogarithmic lower bound for evaluating isogenies does not exclude the possibility of a polylogarithmic bound, so there may still be room for improvements in that regards.

Bibliography

[1] Walter Baur and Volker Strassen. “The complexity of partial derivatives”. In: *Theoretical Computer Science* 22.3 (1983), pp. 317–330. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(83)90110-X. URL: http://www.sciencedirect.com/science/article/pii/030439758390110X.

[2] Daniel J. Bernstein et al. *Faster computation of isogenies of large prime degree.* Cryptology ePrint Archive, Report 2020/341. https://eprint.iacr.org/2020/341. 2020.

[3] L. Blum, M. Shub, and S. Smale. “On a theory of computation over the real numbers NP completeness, recursive functions and universal machines”. In: *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science.* IEEE, 1988. DOI: 10.1109/sfcs.1988.21955. URL: https://doi.org/10.1109%2Fsfcs.1988.21955.

[4] Lenore Blum et al. *Complexity and Real Computation.* Springer New York, 1998. DOI: 10.1007/978-1-4612-0701-6. URL: https://doi.org/10.1007%2F978-1-4612-0701-6.

[5] Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic Complexity Theory.* Springer Berlin Heidelberg, 1997. DOI: 10.1007/978-3-662-03338-8. URL: https://doi.org/10.1007%2F978-3-662-03338-8.

[6] Wouter Castryck and Thomas Decru. “CSIDH on the Surface”. In: *Post-Quantum Cryptography.* Springer International Publishing, 2020, pp. 111–129. DOI: 10.1007/978-3-030-44223-1_7. URL: https://doi.org/10.1007%2F978-3-030-44223-1%5C%5F7.

[7] Wouter Castryck et al. “CSIDH: An Efficient Post-Quantum Commutative Group Action”. In: *Lecture Notes in Computer Science.* Springer International Publishing, 2018, pp. 395–427. DOI: 10.1007/978-3-030-03332-3_15. URL: https://doi.org/10.1007%2F978-3-030-03332-3%5C%5F15.

[8] Craig Costello and Huseyin Hisil. *A simple and compact algorithm for SIDH with arbitrary degree isogenies.* Cryptology ePrint Archive, Report 2017/504. https://eprint.iacr.org/2017/504. 2017.

[9] Jean-Marc Couveignes. *Hard Homogeneous Spaces.* Cryptology ePrint Archive, Report 2006/291. `https://eprint.iacr.org/2006/291`. 2006.

[10] Steven D. Galbraith. *Mathematics of Public Key Cryptography.* Cambridge University Press, 2009. DOI: `10.1017/cbo9781139012843`. URL: `https://doi.org/10.1017%2Fcbo9781139012843`.

[11] Ulrich Vollmer Johannes Buchmann. *Binary Quadratic Forms.* en. Vol. 20. Algorithms and Computation in Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. ISBN: 9783540463672. DOI: `10.1007/978-3-540-46368-9`. URL: `http://link.springer.com/10.1007/978-3-540-46368-9` (visited on 06/03/2020).

[12] C. P. Schnorr. "Improved Lower Bounds on the Number of Multiplications/Divisions which are Necessary of Evaluate Polynomials". In: *Theor. Comput. Sci.* 7 (1978), pp. 251–261.

[13] Michael Shub and Steve Smale. "On the intractability of Hilbert's Nullstellensatz and an algebraic version of P vs. NP". In: *Duke Mathematical Journal* 81.1 (1995), pp. 47–54. DOI: `10.1215/s0012-7094-95-08105-8`. URL: `https://doi.org/10.1215%2Fs0012-7094-95-08105-8`.

[14] Carl Siegel. "Über die Classenzahl quadratischer Zahlkörper". ger. In: *Acta Arithmetica* 1.1 (1935), pp. 83–86. URL: `http://eudml.org/doc/205054`.

[15] Joseph H. Silverman. *The Arithmetic of Elliptic Curves.* Springer New York, 2009. DOI: `10.1007/978-0-387-09494-6`. URL: `https://doi.org/10.1007%2F978-0-387-09494-6`.

[16] Volker Strassen. "Polynomials with Rational Coefficients Which are Hard to Compute". In: *SIAM Journal on Computing* 3.2 (June 1974), pp. 128–149. DOI: `10.1137/0203010`. URL: `https://doi.org/10.1137%2F0203010`.

[17] Jacques Vélu. "Isogénies entre courbes elliptiques." In: *Comptes Rendus de l'Académie des Sciences - Series A - Sciences Mathématiques* 273.4 (July 1971), pp. 238–341. URL: `https://gallica.bnf.fr/ark:/12148/bpt6k56191248/f52.image`.

[18] William C. Waterhouse. "Abelian varieties over finite fields". en. In: *Annales scientifiques de l'Ecole Normale Supérieure* Ser. 4, 2.4 (1969), pp. 521–560. DOI: `10.24033/asens.1183`. URL: `http://www.numdam.org/item/ASENS_1969_4_2_4_521_0`.

Appendix

Algorithm 3: Elliptic Curve Addition Formula

```
Function $\oplus((x_{P_1}, y_{P_1}), (x_{P_2}, y_{P_2}))$
    if $x_{P_1} \equiv x_{P_2} \wedge y_{P_1} + y_{P_2} + a_1 x_{P_2} + a_3 \equiv 0$ then
        // Handle point at infinity
    else
        if $x_{P_1} \equiv x_{P_2}$ then
            $\lambda \leftarrow \frac{3x_{P_1}^2 + 2a_2 x_{P_1} + a_4 - a_1 y_{P_1}}{2y_{P_1} + a_1 x_{P_1} + a_3}$
            $\nu \leftarrow \frac{-x_{P_1}^3 + a_4 x_{P_1} + 2a_6 - a_3 y_{P_1}}{2y_{P_1} + a_1 x_{P_1} + a_3}$
        else
            $\lambda \leftarrow \frac{y_{P_2} - y_{P_1}}{x_{P_2} - x_{P_1}}$
            $\nu \leftarrow \frac{y_{P_1} x_{P_2} - y_{P_2} x_{P_1}}{x_{P_2} - x_{P_1}}$
        end
        $x_{P_3} \leftarrow \lambda^2 + a_1\lambda - a_2 - x_{P_1} - x_{P_2}$
        $y_{P_3} \leftarrow -(\lambda + a_1)a_3 - \nu - a_3$
        return $x_{P_3}, y_{P_3}$
    end
```