Factoring Integers with Elliptic Curves

By

Daniel Tyebkhan

* * * * * * * * *

Submitted in partial fulfillment of the requirements for

Honors in the Department of Mathematics

Union College

March 2023

ABSTRACT

DANIEL TYEBKHAN Factoring Integers with Elliptic Curves.

Department of Mathematics, March 2023.

ADVISOR: HATLEY, JEFFREY

Elliptic curves are abelian groups known for their many applications ranging from the cryptographic standards underlying much of the modern internet, to number theory where they play a central role in Andrew Wiles' proof of Fermat's Last Theorem. In this paper, we introduce the basic theory of elliptic curves, beginning with the Weierstrass equation and a brief overview of projective space, followed by a discussion of their group law. We then shift to the realm of factoring and describe how and why Lenstra's Algorithm works by leveraging elliptic curves to illuminate prime factors of composite integers. We conclude with some discussion of our own implementation of Lenstra's Algorithm.

ACKNOWLEDGEMENT

I thank the fantastic professors of the Union College Department of Mathematics who guided me through the last four years of mathematical inquiry and discovery. Especially, I thank this project's advisor, Professor Jeffrey Hatley, for his mentorship from my first math class at Union through the completion of this thesis. His willingness to scour the internet with me to answer questions coupled with his patience, knowledge, and poignant explanations were critical in helping me grasp nuances associated with the contents of this work. Alongside our discussions about music, guitar, and rock climbing, Jeff's guidance made this thesis one of the highlights of my college experience.

Additionally, I thank my family – Mom, Dad, Sarah, and Joshua – for their love and support throughout this journey.

Contents

ABSTRACT ii

ACKNOWLEDGEMENT iii

1. Introduction 1

2. Basics of Elliptic Curves 2

2.1. Elliptic Curves and the Weierstrass Equation 2

2.2. Geometric Addition of Points on Elliptic Curves 3

2.3. Projective Space 6

2.4. Group Theoretic Properties of Elliptic Curves 8

3. Factoring 12

3.1. Classical $p-1$ Method 12

3.2. Lenstra's Algorithm 15

4. Implementation 18

4.1. Results 19

References 21

1. Introduction

The underlying questions dealt with by elliptic curves trace through the works of Fermat, Abel, and as far back as Diophantus. However, it was the last century which saw the curves' formalization by Weil and subsequent rise to the forefront of number theory with a wide variety of applications in a number of related fields. In cryptography, they form the basis for much of modern encryption on the web as they provide a more space-efficient alternative to the classical RSA model. In the early 1990s, Andrew Wiles used elliptic curves in his landmark proof of Fermat's Last Theorem, conquering a centuries old challenge in mathematics. They also have applications in primality testing and form the basis for an efficient factorization algorithm by Lenstra.

In this paper, we focus on the application of elliptic curves to factoring. Factoring plays a central role in many sectors of mathematics, and by extension, their real-world applications. It is a heavily researched area with many implications due to its high asymptotic complexity and importance to cryptography. In this paper, we build up the basic theory of elliptic curves, how they are described, and how they form additive, abelian groups over arbitrary fields. We then explain how these properties can be leveraged to efficiently factor integers via Lenstra's algorithm. We finish with a discussion of the algorithm's implementation details and performance.

2. Basics of Elliptic Curves

2.1. **Elliptic Curves and the Weierstrass Equation.**

Definition 1. Formally, an elliptic curve over a field **K** is a two-variable, non-singular, cubic curve with at least one **K**-rational point.

Although there are several representations of elliptic curves, in this paper, we will work with one known as the Weierstrass Equation.

Definition 2. The *Weierstrass Equation* for an elliptic curve E over a field **K** is

$$E\colon y^2 = x^3 + Ax^2 + B$$

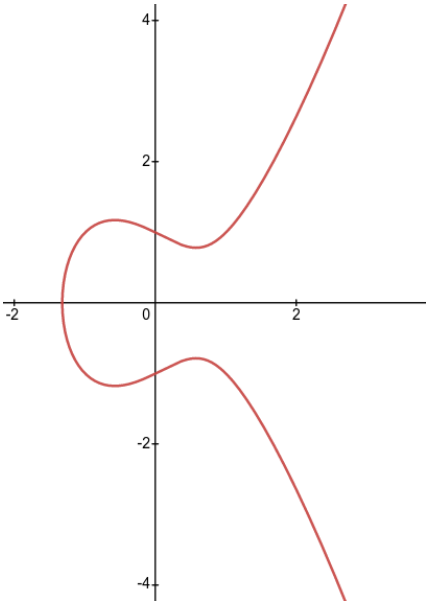
where A and B are elements of **K** and x, y are variables.

In order for a curve described by the Weierstrass equation to have the desired properties, we require it to be non-singular and we restrict **K** to be not of characteristic 2 or 3[1]. We can test that a curve is non-singular by checking that the curve's discriminant, $-(4A^3 + 27B^2)$, is non-zero.

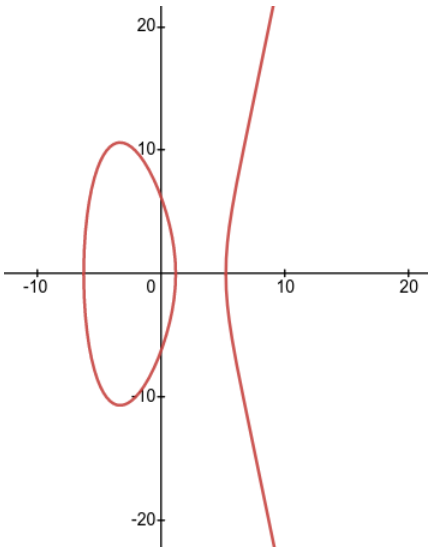
Figure 2.1 contains examples of elliptic curves over the real numbers with their equations in Weierstrass form. One interesting thing to note from both the equation and the figures is that elliptic curves are symmetrical over the x-axis. This property will be important in defining its group law which involves reflection over the x-axis.

One other property which bears mentioning is that elliptic curves all contain a special point at infinity (denoted ∞) which will be discussed in §2.3. In fact,

[1]If the characteristic of **K** is 2 or 3, a more general equation can be used to describe an elliptic curve.



(A) The elliptic curve $y^2 = x^3 - x + 1$



(B) The elliptic curve $y^2 = x^3 - 34x + 37$

FIGURE 2.1. Examples of Elliptic Curves Over $\mathbf{R}$ in Weierstrass Form

this is the required rational point in the definition. Thus we can write the K-points of an elliptic curve as a set:

$$E(\mathbf{K}) = \{\infty\} \cup \{(x, y) \in \mathbf{K}^2 \mid y^2 = x^3 + Ax + B \text{ and } A, B \in \mathbf{K}\}.$$

2.2. **Geometric Addition of Points on Elliptic Curves.** In this section, we work toward defining an additive group over the points on an elliptic curve by providing a geometric explanation of point addition. This notion will be made more rigorous in § 2.4. Consider $E(\mathbf{R})$ and points $P, Q \in E$. We select $\mathbf{R}$ as our field here because it is easy to visualize, but the algebraic equations we develop from this process hold over any field. We can determine a third point called $P + Q \in E(\mathbf{R})$ through the following process:

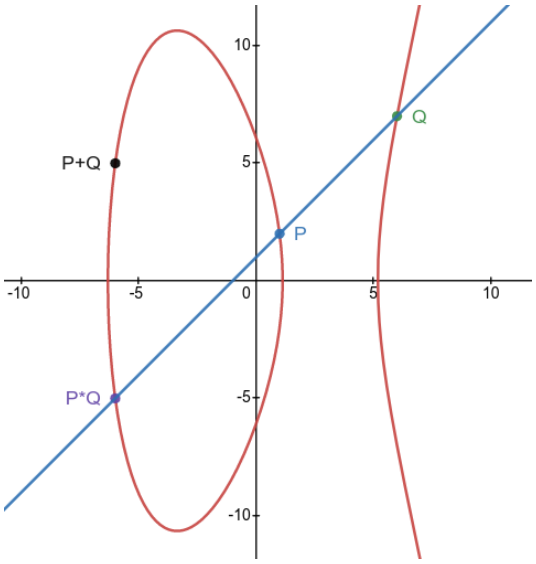


FIGURE 2.2. The addition of points $P = (1, 2)$ and $Q = (6, 7)$ on the elliptic curve $y^2 = x^3 - 34x + 37$. See Example 1.

(1) Draw the line L through P and Q.

(2) Find the third point of intersection between L and $E(\mathbf{R})$, call it $P * Q$.

(3) Reflect $P * Q$ over the x-axis to obtain $P + Q$.

From this geometric definition, we can also see that our group will be abelian since the line between P and Q is exactly the same as the line between Q and P.

Example 1. Consider the elliptic curve $E : y^2 = x^3 - 34x + 37$ over $\mathbf{R}$. Let $P = (1, 2)$ and $Q = (6, 7)$. It is easy to verify that P and Q fall on E. We will compute $P + Q$ according to the above steps.

(1) Compute L to be the line through P and Q. A simple calculation shows that L is defined by $y = x + 1$.

(2) Find $P * Q$ by substituting L into E:

$$(x + 1)^2 = x^3 - 34x + 37$$

Then solving:

$$\begin{aligned} 0 &= x^3 - x^2 - 36x + 36 \\ &= (x-1)(x+6)(x-6). \end{aligned}$$

Thus there are roots at $x = 1, 6, -6$. We already know about 1 and 6 from P and Q respectively so our third point of intersection is $P * Q = (-6, -5)$.

(3) Reflecting $P * Q$ about the x-axis gives us $P + Q = (-6, 5)$ which we can verify also lies on E.

See Figure 2.2 for a visualization of this process.

The example works out nicely, but the definition as it stands seems to fall short in a couple of places. Obviously, we need to be able to add a point to itself so we define $P + P$ by taking L to be the tangent line to E at P. Then P is a point of double intersection and $P + P$ is obtained by finding the third point of intersection and reflecting it across the x-axis. Algebraically, the point of double intersection works out as P will be 2 of the roots of the cubic obtained from setting $L = E$.

Also, note that step 2 of our addition process requires that a line through any two points on the curve will intersect the curve at exactly one other point. In our examples so far, this fact seems intuitive except in the cases where L is vertical and appears to only intersect the curve twice. To account for this case, we work in an extension of the field in which the curve is defined known as projective space, where L also intersects E at the aforementioned point at

infinity. This point lies on all elliptic curves, and all vertical lines, and will be the group's identity element.

2.3. **Projective Space.** Informally, two-dimensional projective space over a field K, denoted $\mathbb{P}^2_K$, is an extension of K to include a set of so-called points at infinity. These points at infinity are interesting to us since one of them lies on every elliptic curve over $\mathbb{P}^2_K$ and all vertical lines in K^2 when considered in $\mathbb{P}^2_K$. We denote this point ∞. One way to conceptualize ∞ is to think of it as a point sitting infinitely high directly above the y-axis. Then, imagine "curling" the plane into a sphere. The point at infinity now sits at the very top of the sphere so any vertical line will naturally "wrap around" and intersect ∞. This means that ∞ is effectively also at the bottom of the y-axis.

Definition 3. Two-Dimensional Projective Space over a field **K** is the set of equivalence classes of triples (x, y, z) where $x, y, z \in K$ and x, y, z are not all zero. Two points (x_1, y_1, z_1) and (x_2, y_2, z_2) are said to be equivalent (written $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$) if there exists a non-zero $\lambda \in K$ such that

$$(x_1, y_1, z_1) = (\lambda x_2, \lambda y_2, \lambda z_2).$$

Because the triples depend solely on the ratio of x, y, and z, we denote the equivalence class of (x, y, z) as $(x : y : z)$.

Definition 4. For a triple $(x : y : z)$ in $\mathbb{P}^2_K$, if $z \neq 0$, then $(x : y : z) = \left(\frac{x}{z} : \frac{y}{z} : 1\right)$. Such points are called the **finite points** of $\mathbb{P}^2_K$. Triples with $z = 0$ are the **points at infinity**.

We need a couple other definitions to show that each elliptic curve over a projective space contains exactly one point at infinity.

Definition 5. The two dimensional **affine plane** over K is

$$A_K^2 = \{(x, y) \in K \times K\}.$$

Note that $A_K^2 \subset \mathbb{P}_K^2$ and elements can be mapped

$$(x, y) \mapsto (x : y : 1).$$

Thus we have a way to map affine points on our elliptic curve to points in $\mathbb{P}_K^2$.

Definition 6. Recall from linear algebra that a polynomial in K is **homogeneous** of degree n if it is a sum of terms of the form $ax^i y^j z^k$ with $a \in K$ and $i + j + k = n$. Also, a polynomial F that is homogeneous of degree n satisfies $F(\lambda x, \lambda y, \lambda z) = \lambda^n F(x, y, z)$ for all $\lambda \in K$.

Thus, given $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_2, y_2, z_2)$, and $p_1 \sim p_2$, a homogeneous polynomial F has $F(p_1) = 0$ if and only if $F(p_2) = 0$. This means that the set of zeros for F in $\mathbb{P}_K^2$ is well defined. We can make a polynomial $f(x, y)$ homogeneous of degree n by inserting the appropriate powers of z:

$$F(x, y, z) = z^n f\left(\frac{x}{z}, \frac{y}{z}\right)$$

and

$$f(x, y) = F(x, y, 1).$$

Proposition 1. *An elliptic curve E given by $y^2 = x^3 + Ax + B$ over projective space contains exactly one point at infinity, namely $(0 : 1 : 0)$.*

Proof. First, write E in homogeneous form:

$$y^2 z = x^3 + Axz^2 + Bz^3.$$

Recall that each affine point (x, y) maps to $(x : y : 1)$. To obtain the points at infinity on E, set $z = 0$ and solve. This yields $x^3 = 0$, meaning that the points at infinity are where $x = 0, z = 0$, and y has any non-zero value (i.e. $(0 : y : 0)$). Since $\frac{0}{y} = 0$ for any y, this point is equivalent to the point $(0 : 1 : 0)$. Since $x = 0$ was the only solution, $(0 : 1 : 0)$ is the only point at infinity contained in E. □

Finally, for our addition definition, we need

Proposition 2. *All lines which are vertical in* $\mathbf{K}^2$ *run through* $(0 : 1 : 0)$.

Proof. A vertical line is given by the equation $x = n$ for some constant $n \in K$. In homogeneous form, this is $xz^2 = nz^3$ or $x = nz$. This means that the points on the line are of the form $(nz : y : z)$. Setting $z = 0$ then gives us the point $(0 : y : 0)$ and y must be non zero since we cannot have x, y, z all zero. Thus this point is equivalent to $(0 : 1 : 0)$. □

Because we showed that $(0 : 1 : 0)$ lies on all vertical lines and all elliptic curves, it is a point of intersection for any vertical line and elliptic curve. In this paper, when working with curves, we will generally work in affine coordinates (e.g. (x, y)), treating ∞ as a special case. From this point on, an elliptic curve over any ring or field should be thought of as implicitly being over the projective space associated with the relevant algebraic structure.

2.4. **Group Theoretic Properties of Elliptic Curves.** In this section, we will prove that the $\mathbf{K}$-points on an elliptic curve form an abelian group.

In § 2.2 we described the process by which points are added together. First we derive the relevant equations to make these calculations simpler. We begin

with two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ on an elliptic curve E given by $y^2 = x^3 + Ax + B$. Let L be the line between P_1 and P_2. We now need to consider several cases:

(1) Assume that $P_1 \neq P_2$ and neither point is ∞. Then L has slope $m = \frac{y_2 - y_2}{x_2 - x_1}$. Now assume $x_1 \neq x_2$ so L is not vertical. Then L is given by $y = m(x - x_1) + y_1$. We can then find the final point of intersection through substitution and the knowledge of two existing roots in the equation. This yields the point

$$x = m^2 - x_1 - x_2$$

$$y = m(x - x_1) + y_1$$

and reflecting across the x-axis gives

$$x_3 = m^2 - x_1 - x_2$$

$$y_3 = m(x_1 - x_3) - y_1,$$

our final point.

(2) Assume that $x_1 = x_2$, but $y_1 \neq y_2$ so the line L between P_1 and P_2 is vertical. As we saw in § 2.3, L will intersect the curve again at ∞. But, in projective space, $(0, y, 0) \sim (0, -y, 0)$ so the reflection of ∞ is itself. This also reflects and reinforces our previous geometric interpretation of ∞; because ∞ is at both the top and bottom of the y-axis, reflecting it about the x-axis returns itself. Thus our sum here is simply ∞.

(3) Next, consider the case where $P_1 = P_2 = (x_1, y_1)$. Since we are using the tangent line to E at P_1 for addition, we find the derivative of E at

P_1 to obtain the slope of L:

$$m = \frac{dy}{dx} = \frac{3x_1^2 + A}{2y_1}.$$

If $y_1 = 0$, m is undefined and L is vertical so we fall back to the previous case. Otherwise, we can assume $y_1 \neq 0$. As in the first case, we have L defined by $y = m(x - x_1) + y_1$. Solving similarly for the third point of intersection (remembering P_1 counts as 2 separate points and represents two roots of the equation), we get

$$x_3 = m^2 - 2x_1$$

$$y_3 = m(x_1 - x_3) - y_1$$

(4) Finally, consider the case in which $P_2 = \infty$. Then L must be vertical and will therefore intersect E at the reflection of P_1 over the x-axis. Rereflecting this point returns us back to P_1 so it follows that

$$P_1 + \infty = P_1.$$

To make these cases more compact we have the following:

Definition 7. Let E be an elliptic curve defined by $y^2 = x^3 + Ax + B$. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on E with P_1, P_2 not both ∞. Define $P_1 + P_2 = (x_3, y_3)$ as follows:

(1) If $x_1 \neq x_2$, then

$$x_3 = m^2 - x_1 - x_2$$

$$y_3 = m(x_1 - x_3) - y_1$$

where $m = \frac{y_2 - y_1}{x_2 - x_1}$.

(2) If $x_1 = x_2$, but $y_1 \neq y_2$, then $P_1 + P_2 = \infty$.

(3) If $P_1 = P_2$ and $y_1 \neq 0$, then

$$x_3 = m^2 - 2x_1$$

$$y_3 = m(x_1 - x_3) - y_1$$

where $m = \frac{3x_1^2 + A}{2y_1}$.

(4) If $P_1 = P_2$ and $y_1 = 0$, then $P_1 + P_2 = \infty$.

Also, define

$$P + \infty = P$$

for all points P on E.

Now we can show that an elliptic curve does in fact form a group.

Theorem 1. *The set of points on an elliptic curve E form an additive, abelian group under the operation defined in Definition 7.*

Proof.

(1) The commutativity property is obvious since, as stated earlier, the line between P_1 and P_2 is the same as the one through P_2 and P_1. This can also be worked out algebraically from the formulas in the definition.

(2) By definition, ∞ functions as the group identity element.

(3) To get an inverse, we simply need the line through the points to be vertical, so for a point P, take P^{-1} to be the reflection of P across the x-axis. Then $P + P^{-1} = \infty$ which is the same as $P^{-1} + P$ by commutativity.

(4) The proof of associativity is long and involves lots of algebra in projective space so we omit it here. It can be found in Washington [5]. It is, however, interesting to note that the reason we reflect the third point of intersection over the x-axis rather than taking it unaltered as the sum is to satisfy the property of associativity.

□

3. Factoring

Finding the prime factorization of an integer n is the task of finding prime numbers $p_1, p_2, \ldots, p_k$ such that $p_1 p_2 \cdots p_k = n$. Here we examine a classical method of factoring, then an improvement on it which utilizes Elliptic Curves.

For this section it will be helpful to recall:

Theorem 2 (Fermat's Little Theorem)**.** *For an integer a and a prime p with $p \nmid a$,*

$$a^{p-1} \equiv 1 \mod p.$$

3.1. **Classical $p-1$ Method.** Before we look at Lenstra's Algorithm which utilizes elliptic curves, we will first examine its classical analogue, the $p-1$ algorithm, discovered by Pollard [2]. Lenstra based his algorithm on this method, so this will help us gather the intuition and see the improvements made by Lenstra. First we need

Definition 8. An integer m is **B-smooth** if all of the prime factors of m are less than or equal to B.

Example 2. The number 35 is 7-smooth since $35 = 7 \cdot 5$. It would also be correct to say that 35 is B-smooth for any $B \geq 7$ since clearly all prime factors of 35 would be less than or equal to B.

Similarly, $385 = 5 \cdot 7 \cdot 11$ is B-smooth for any $B \geq 11$ and $25 = 5^2$ is 5-smooth.

The $p-1$ algorithm to factor the product of two primes $n = pq$ is as follows:

Algorithm 1 $p-1$ Factoring Algorithm

```
a ← RANDOMINTEGER() mod n
B ← LARGEINTEGER()
return FACTOR(n, a, B)

procedure FACTOR(n, a, b)
    if b = 0 or a < 2 then
        return Factor not found
    end if
    q ← gcd(a − 1, n)
    if q > 1 then
        return q
    end if
    return FACTOR(n, a^b mod n, b − 1)
end procedure
```

Example 3. Let $n = 55$ and suppose we picked $a = 7$ and chose $B = 15$. Then we calculate $\gcd(7-1, 55) = 1$ so we don't have a factor. Now compute $a_1 \equiv 7^{15} \equiv 43 \mod 55$. Since $\gcd(43-1, 55) = 1$, we recurse and compute $a_2 \equiv 43^{14} \equiv 34 \mod 55$. Since $\gcd(34-1, 55) = 11$, we have found a factor in only two iterations of the algorithm.

Why does this work? Assume that $p-1$ is B-smooth (it will be for a large enough B). Then it is likely that $B!$ is a multiple of $p-1$ since all prime factors of $p-1$ are less than B and are therefore factors of the product $B!$. If there

are multiple occurrences of a prime in the factorization of $p-1$ these will likely be picked up in other factors of $B!$. Also, assume that $p \nmid a$. We can ensure this happens by calculating $\gcd(k, n)$ for all primes $k < a$. This is practical since the selection of a is arbitrary so a can always be picked relatively small. If we find a value larger than 1, we already have a factor. Otherwise, we know $p \nmid a$. By Fermat's Little Theorem, we now have

$$a_1 \equiv a^{B!} \equiv 1 \mod p.$$

Next assume that $q-1$ is divisible by some prime $\ell > B$. Examining the units of the group $\mathbf{Z}/q\mathbf{Z}$, there are a maximum of $\frac{q-1}{\ell}$ with order not divisible by ℓ and at least $\frac{(\ell-1)(q-1)}{\ell}$ with order divisible by ℓ. This makes it highly likely that the order of a is divisible by ℓ and meaning that

$$a_1 \equiv a^{B!} \not\equiv 1 \mod q.$$

Since a_1 is a multiple of p and not q, we know

$$\gcd(a_1 - 1, pq) = p.$$

In other words, it's unlikely that you'll get $gcd = n$. Thus, we have found one prime factor of n and uncovering the other is a matter of simple division.

The trick to successfully running the $p-1$ method is a good selection of B. Selecting B too small makes it unlikely that $p-1$ will be B-smooth. On the other hand, B must be small enough that computing the program will eventually terminate if the parameters are unsuccessful. According to Washington [5], 10^8 can be a good approximate value for B given its "medium" size, but with access to more powerful computational resources, a larger number may

be better. If neither $p-1$ nor $q-1$ are B-smooth, we must resort to picking a new value of a and starting over. However, this reselection strategy can be mitigated by choosing p such that there is a prime ℓ' with $\ell' | p-1$ and $\ell' > B$. Then the algebra regarding probabilities of ℓ above shows the chance that $a_1 \equiv 1 \mod p$ is at best $\frac{1}{\ell'}$ which becomes very small as ℓ' increases making this method effectively useless. Lenstra's Algorithm overcomes this challenge by carrying out analogous operations over elliptic curves, allowing the factorer to change groups entirely if one value of a or B doesn't work.

3.2. **Lenstra's Algorithm.** Lenstra's Algorithm, which uses elliptic curves for factorization, overcomes the issue of choosing a new a in the classical method by allowing the attacker to switch groups entirely if a set of parameters fails to factor n.

Before we get into the details of the algorithm, let's look at what happens if we consider an elliptic curve over a ring which is not a field.

Example 4. Let E be an elliptic curve defined by $y^2 = x^3 - x + 1 \mod 35$. While computing $(1,1)+(26,24)$, we compute the slope $\frac{23}{25}$ which is finite mod 7, but infinite mod 5 because $\gcd(35,5) = 5 \neq 1$. This makes the resulting point "partially" at infinity so it cannot be expressed in affine coordinates as defined in § 2.3 [2]. In other words, this fails because 25 has no multiplicative inverse mod 35 so it does not make sense to talk about $\frac{23}{25} \mod 35$. However, when this failure occurs, we have a denominator which shares a common factor greater than 1 with our composite modulus and thus can compute a factor of

[2]In this situation, we could use the Chinese Remainder Theorem and work mod 5 and mod 7 separately to obtain a solution, but this is not possible in all instances. There is a way to define the group law to work over arbitrary rings which can be found in [5].

the modulus. This is the key observation which makes Lenstra's algorithm work.

The algorithm is as follows

(1) Choose k random elliptic curves $E_1, \ldots, E_k$ and a corresponding random point P_i on each curve.
(2) Choose an integer B and compute $(B!)P_i$ on E_i for all i.
(3) If step (2) fails on any curve because a slope does not exist mod n (i.e. the denominator of m from Definition 7 is non-invertible mod n), stop because a factor has been found.
(4) If step (2) succeeds, choose a larger value of B or a new set of curves and start over.

In practice, step (2) can be performed in parallel since each curve is completely independent of the others meaning that you generally want to choose k to be the maximum number of cores you have available to run the algorithm. Like in the classical method, $(B!)P$ is computed recursively.

In step (1), random curves can be created as follows:

(1) Select a random integer $A \mod n$ and a random pair of integers $P = (u, v) \mod n$.
(2) Select C such that $C \equiv v^2 - u^3 - Au \mod n$.
(3) If $y^2 = x^3 + Ax + C$ is non-singular, we have a desired curve with a point (u, v). Otherwise, pick new random integers and try again.

Now let's look at an example of Lenstra's algorithm.

Example 5. We want to factor 35. Suppose we choose E to be $y^2 = x^3 - 20x + 21 \mod 35$, and $P = (15, -4)$. Select $B = 9$. Now we compute $(9!)P$ recursively.

First $2P = (0, 14)$. Then $3(2P) = 2P + 2P + 2P$. Doing the first addition of $2P + 2P$ according to the group law we obtain a slope with denominator 7. However, 7 is not invertible mod 35 since $\gcd(7, 35) = 7 \neq 1$ so we have a factor.

In general, Lenstra's algorithm works well for numbers with a prime factor smaller than 10^{40}, but is outperformed by methods such as a quadratic sieve or number field sieve for larger factors. However, Lenstra's remains relevant as some implementations of the sieve methods use Lenstra's algorithm internally to look for smaller sized prime factors of numbers that occur in intermediate steps.

To see why Lenstra's Algorithm works, we need

Theorem 3 (Hasse)**.** *Let E be an elliptic curve over the finite field F_q. Then the order of $E(\mathbf{F}_q)$ satisfies*

$$|q + 1 - \#E(\mathbf{F}_q)| \leq 2\sqrt{q}.$$

We omit the proof here, but it can be found in Washington [5].

Consider $n = pq$ where p and q are distinct primes and take some elliptic curve $E \mod n$. We can regard E as an elliptic curve mod p and mod q. From a slight rearrangement of Hasse's theoerem, we can see that

$$p + 1 - 2\sqrt{p} < \#E(\mathbf{F}_p) < p + 1 + 2\sqrt{p}.$$

We now also need a couple of facts:

(1) It is possible, given any integer within the Hasse bound, to construct a curve with that order using Atkin-Morain's method [1]. Thus, each integer in the range actually does occur for some elliptic curve.
(2) As B increases, the density of B-smooth integers in the interval increases.
(3) The distribution of orders of random elliptic curves is uniform enough for the algorithm to work.

As a result of these statements, the selection of several random elliptic curves will likely result in at least one with B-smooth order. The more curves we pick, the higher the probability that we run into such a curve. Assuming E is such a curve, and P is the chosen point on E, we then likely have $(B!)P \equiv \infty \mod p$. Because the order of P falls into the range of Hasse's theorem for $\mathbf{F}_p$, it is unlikely that $(B!)P = \infty$ working on $E \mod q$. Thus, while calculating $(B!)P \mod n$, we will likely obtain a slope with a denominator d which is divisible by p, but not q. Then $\gcd(d, n)$ should be the factor p.

4. Implementation

As a supplement, we wrote a Python implementation of Lenstra's algorithm available at [4]. The project contains an implementation of an elliptic curve over $\mathbf{Z}/q\mathbf{Z}$ with functions to construct and add projective points on the curve as well as multiply points by integers. The general structure was inspired by the SageMath [3] software package.

One interesting detail is the implementation of integer multiplication of points on an elliptic curve. Naively, a point could be multiplied by a number n by simply adding the point to itself n times. This is certainly simple to

implement, especially if there is an existing function for point addition. However, given that the algorithm relies on repeatedly multiplying a point by a steadily increasing integer, making this process more efficient greatly reduces runtime at larger values. There exist several algorithms to do so, however for our purposes, we choose the double-and-add method. It takes as input an integer n and a point P and produces nP as output through the following:

Algorithm 2 Double-and-Add

```
bits ← intToBitString(n)        ▷ get the bits of n from least to most significant
product ← ∞
t ← P
for b ∈ bits do
    if b = 1 then
        product ← product + t
    end if
    t ← t + t
end for
return prod
```

This algorithm leverages powers of 2 by repeatedly doubling the point to be added, then only adding it on powers of 2 present in n. It reduces the run time from $\Theta(n)$ to $\Theta(\log_2(n))$ which makes a significant difference, especially as n grows extremely large in the algorithm. It is interesting to note that while double-and-add works well to speed up our application, it is not used in implementations of elliptic curves designed for encrypting information since its run time varies with n and it is thus susceptible to timing attacks.

4.1. **Results.** Our implementation factored the following numbers running with 6 threads on an Intel(R) Core(TM) i7-8750H CPU at 2.20GHz.

n	Time (hh:mm:ss)
35	00:00:00.020
2442534499	00:00:00.025
31287702260288579971	00:00:00.089
288505916380974655850792112180674104072 9	11:18:09.86

For smaller n, the time it takes to calculate factors with a brute force approach would be slighlty faster since it avoids the overhead of Lenstra's algorithm. However, as n gets large, the runtime increases dramatically. For example, the 40 digit number factored in 11:18:09.86 would take approximately 465356888944913 days or 1274950380671 years to find the smaller factor on the same processor. Given that the earth is only expected to survive another 4 billion years, factoring a number that large in the naive manner is impractical.

References

[1] A Oliver L Atkin and Francois Morain. "Finding suitable curves for the elliptic curve method of factorization". In: Mathematics of Computation 60.201 (1993), pp. 399–405.

[2] J. M. Pollard. "Theorems on factorization and primality testing". In: Mathematical Proceedings of the Cambridge Philosophical Society 76.3 (1974), pp. 521–528. DOI: 10.1017/S0305004100049252.

[3] The Sage Developers. SageMath, the Sage Mathematics Software System (Version x.y.z). https://www.sagemath.org. YYYY.

[4] Daniel Tyebkhan. Simple Lenstra's Implementation. Version 1.0.0. Mar. 2023. URL: https://github.com/DanielTyebkhan/EllipticCurveFactorization.

[5] Lawrence C. Washington. Elliptic curves: number theory and cryptography. 6000 Broken Sound Parkway NW, Suite 300. Boca Raton, FL: Chapman & Hall/CRC, 2008.