Examining the Probability that the Number of Points on an Elliptic Curve over $\mathbb{F}_p$ is Prime

Master's Project
MATH 898

Émilie Wheeler

Supervised by Prof. David Wehlau

Department of Mathematics & Statistics
Queen's University
Kingston, Ontario, Canada
May 13, 2015

Examining the Probability that the Number of Points on an Elliptic Curve over $\mathbb{F}_p$ is Prime

Master's Project

Émilie Wheeler

Abstract

This project examines the work in the article *The Probability that the Number of Points on an Elliptic Curve over a Finite Field is Prime*, in which authors Galbraith and McKee ask the question 'What is the probability that a randomly chosen elliptic curve over $\mathbb{F}_p$ has kq points, where k is small and q is prime?' I performed my own computations and will compare them to their results.

Contents

1 Introduction

In 2000, Steven D. Galbraith and James McKee published a [revised] paper entitled *The Probability that the Number of Points on an Elliptic Curve over a Finite Field is Prime* [1]. This paper gives two conjectures which describe the probability that an elliptic curve over $\mathbb{F}_p$ has k times a prime for its number of points, with Conjecture 1 studying this probability for the case $k = 1$. The paper also provides statistical evidence supporting these two conjectures, as well as the method they used to formulate these equations.

However, the authors did not include their methods of calculating their evidence, nor the programming software they used. I decided to use the free online open-source mathematical software system Sage, which uses the Python language. Through trial and error, and realizing the limitations of my computer's memory, we succeeded in recreating the authors' experiment using the Hurwitz class number and an algorithm from *A Course in Computational Algebraic Number Theory* by Henri Cohen [3]. I have included my computations, a sample of my Sage programs, and my data tables in this report, as well as a numerical comparison of our results.

Also included are sections on the basic notions of elliptic curves, as well as the importance of this article to modern day cryptography.

2 Background Theory on Elliptic Curves

Definition 2.1: An *Elliptic Curve* over a field K is defined by the equation

$$E : y^2 = x^3 + ax + b \tag{1}$$

where $a, b \in K$, and such that $-16(4a^3 + 27b^2) \neq 0$. Elliptic curves include the *point at infinity*, denoted O, which serves as the identity element.

Addition in Elliptic Curves: In order to add two points P and Q on an elliptic curve E to produce a third point R', we start by drawing a line L through points P and Q. L intersects the elliptic curve E at 3 points, P, Q and R (possibly O). We draw a line from R to O which intersects the curve at O, R and the point R'. R' is the sum of P and Q, denoted $R' = P \oplus Q$. (See Figure 1.) To add a point P to itself, we take L as the tangent line to E

at point P. (See Figure 2.) If the line L is a vertical line between P and Q, then $Q = -P$ and $P \oplus (-P) = O$, since the third point at which L intersects E is O.

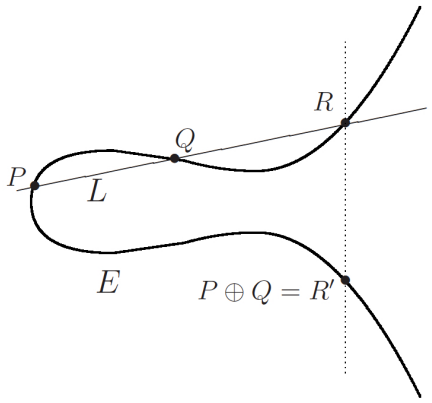

Figure 1: Addition of $P \oplus Q$, $P \neq Q$

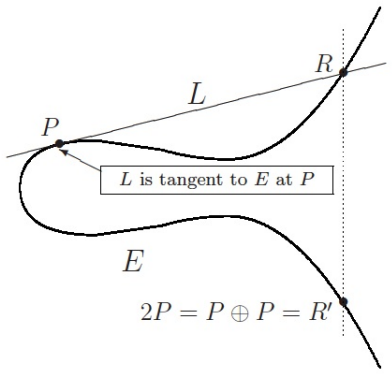

Figure 2: Addition of $P \oplus P$

Theorem 2.2: Let E be an elliptic curve. Then the addition on E has the following properties:

(a)	$P \oplus O = O \oplus P = P$	for all $P \in E$.	[Identity]
(b)	$P \oplus (-P) = O$	for all $P \in E$.	[Inverse]
(c)	$(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$	for all $P, Q, R \in E$.	[Associativity]
(d)	$P \oplus Q = Q \oplus P$	for all $P, Q \in E$.	[Commutativity]

The proof is given in [5].

By this theorem, over a finite field $\mathbb{F}_q$, the points on an elliptic curve E form a *finite abelian group*, and we denote its order by $|E(\mathbb{F}_q)|$. For fields $\mathbb{F}_p$, with p prime, we have that

$$|E(\mathbb{F}_p)| = 1 + p + \sum_{x \in \mathbb{F}_p} \left(\frac{x^3 + ax + b}{p} \right) \tag{2}$$

where $\left(\frac{\cdot}{p}\right)$ denotes the Legendre symbol, (see [2]).

For small primes, eg. $p < 200$, this formula is useful for computing $|E(\mathbb{F}_p)|$, however it proves to be impractical for larger primes. There exist more efficient algorithms for these, such as the baby-step-giant-step method (see [3] for general algorithm and below for algorithm adapted to elliptic curves), as well as the Schoof-Elkies-Atkin algorithm (see [4]), for which there are built-in algorithms in Sage.

The following is an important theorem in the field of elliptic curves, originally conjectured by Artin in 1921, and proven by Hasse in 1933:

Theorem 2.3: *Hasse's bound:* For an elliptic curve E over $\mathbb{F}_q$,

$$||E(\mathbb{F}_q)| - (q+1)| \leq 2\sqrt{q}. \tag{3}$$

We denote the *Hasse interval* by $\mathcal{H}(q) = [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$.

Multiplication in Elliptic Curves: Elliptic curve point multiplication is the operation of successively adding a point P along an elliptic curve E to itself. For a scalar n and $P = (x, y) \in E$:

$$nP = \underbrace{P \oplus P \oplus P \oplus \cdots \oplus P}_{\text{n times}}$$

Algorithm 2.4: Daniel Shank's *Baby-Step-Giant-Step algorithm:* [9]
Input: $P \in E(\mathbb{F}_p)$, $\mathcal{H}(p) = [a, b]$, the Hasse interval.
Output: An element $m \in \mathcal{H}(p)$ such that $mP = 0$.
1. Pick integers r and s such that $rs \geq b - a$.
2. Compute the set $S_{\text{baby}} = \{0, P, 2P, ..., (r-1)P\}$ of *baby steps*.
3. Compute the set $S_{\text{giant}} = \{aP, (a+r)P, (a+2r)P, ..., (a+(s-1)r)P\}$ of *giant steps*.
4. For each giant step $P_{\text{giant}} = (a + ir)P \in S_{\text{giant}}$, for $0 \leq i \leq s - 1$,

check whether $P_{\text{giant}} + P_{\text{baby}} = 0$ for some baby step $P_{\text{baby}} = jP \in S_{\text{baby}}$, for $0 \leq j \leq r-1$. If so, output $m = a + ir + j$.

This m satisfies the condition $mP = 0$, so m is a multiple of the order of P. The algorithm is guaranteed to find such an m since every integer in $\mathcal{H}(p)$ can be written in the form $a + ir + j$, for $0 \leq i \leq s-1$, $0 \leq j \leq r-1$. If we choose $r \approx s \approx 2q^{1/4}$, then the running time of Algorithm 2.4 is $O(q^{1/4})$.

If m is the only element which satisfies $mP = 0$ in $\mathcal{H}(p)$, then $m = |E(\mathbb{F}_p)|$. And if it isn't, Mestre described an algorithm which also uses the quadratic twist, with the above algorithm, and computes $|E(\mathbb{F}_p)|$ more efficiently, but still in exponential time. (See [9] and [10].)

The Schoof-Elkies-Atkin algorithm is described in [4]. The expected running time is $O(\log^4 q)$.

Theorem 2.5: *Prime Number Theorem:* The number of primes not exceeding x is asymptotic to $\frac{x}{\log x}$.

The following are the conjectures and remarks as given in paper [1]:

Note that the P_1 and P_k defined below are functions of p. We will use the notation $P_1 := P_1(p)$ and $P_k := P_k(p)$ for convenience.

Conjecture 1: Let P_1 be the probability that a number within $2\sqrt{p}$ of $p+1$ is prime. Then the probability that an elliptic curve over $\mathbb{F}_p$ has a prime number of points is asymptotic to $c_p P_1$ as $p \to \infty$, where

$$c_p = \frac{2}{3} \prod_{l>2} \left(1 - \frac{1}{(l-1)^2}\right) \prod_{l|p-1, l>2} \left(1 + \frac{1}{(l+1)(l-2)}\right). \tag{4}$$

Here, the products are over all primes l satisfying the stated conditions.

Remark 1: P_1 can be approximated by

$$\frac{1}{4\sqrt{p}} \int_{p+1-2\sqrt{p}}^{p+1+2\sqrt{p}} \frac{\mathrm{d}t}{\log t} \approx \frac{1}{\log p}. \tag{5}$$

The right hand side of this approximation is due to Theorem 2.5 as above.

Conjecture 2: For $k = 1, 2, 3, ...$, let P_k be the probability that a number within $2\sqrt{p}$ of $p+1$ is of the form kq, with q a prime. Then the probability that an elliptic curve over $\mathbb{F}_p$ has k times a prime for its number of points is asymptotic to $c_p f_k P_k$ as $p \to \infty$, where c_p is as in Conjecture 1, and f_k is a rational number, defined below, depending only on k and $\gcd(k, p-1)$. For fixed p, f_k is a multiplicative function of k. For a prime power l^t $(t \geq 1)$, we have

$$f_{l^t} = \frac{l^t(r(l^t) - r(l^{t+1}))}{1 - r(l)}, \tag{6}$$

where

$$r(l^t) = \begin{cases} \frac{1}{l^{t-1}(l-1)} & \text{if } q \not\equiv 1 \pmod{l^u} \\ \frac{l^{v+1}+l^v-1}{l^{t+v-1}(l^2-1)} & \text{if } q \equiv 1 \pmod{l^u} \end{cases} \tag{7}$$

where $u = \lceil t/2 \rceil$, $v = \lfloor t/2 \rfloor$.

Remark 2: (1) P_k can be approximated by

$$\frac{1}{4k\sqrt{p}} \int_{(p+1-2\sqrt{p})/k}^{(p+1+2\sqrt{p})/k} \frac{\mathrm{d}t}{\log t} \approx \frac{1}{k(\log p - \log k)} \tag{8}$$

(2) Table 1 gives values of f_k for some small values of k, and for prime p.
(3) Let Q_k denote the probability that for an elliptic curve over $\mathbb{F}_p$, $\frac{|E(\mathbb{F}_p)|}{k}$ is prime. Then Conjecture 1 reads

$$Q_1 \sim c_p P_1, \text{ as } p \to \infty \tag{9}$$

and Conjecture 2 reads

$$Q_k \sim c_p f_k P_k, \text{ as } p \to \infty \tag{10}$$

3 Computation Methods for Conjectures 1 and 2

3.1 Computing $c_p \cdot f_k \cdot P_k$

To compute P_1, I used the given equation in the paper, equation (5) stated above. Figure 3 shows my program in Sage which computes P_1 using the left

hand side of the given equation, and a short program which computes the right hand side as well. The function $N()$ returns a decimal approximation of the value, instead of a fraction, square root symbol, etc. It returns by default 53 bits of precision.

However, using this formula, we obtained different results. We found that the authors explicitly calculated P_1 using the basic probability equation:

$$\text{Probability} = \frac{\#\text{ of favourable events}}{\#\text{ of total possible events}} \tag{11}$$

and thus

$$P_1 \quad = \frac{\#\text{ prime numbers in } \mathcal{H}(p)}{\#\text{ numbers in } \mathcal{H}(p)}$$

To compute the numerator, we use the Sage program prime_range(m, n) which returns the list of all primes p in the range $m \leq p < n$. See Figure 4 for the Sage program.

To compute c_p, I used the equation given in Conjecture 1, equation (4) above. However, the first product in the equation for c_p is an infinite product, given that it is over all primes $l > 2$. Therefore, first, I had to approximate this product, by taking

$$\prod_{2<l<R,\ l \text{ prime}} \left(1 - \frac{1}{(l-1)^2}\right) \tag{12}$$

for larger and larger values of R. I repeated the process until I obtained the approximated product to 10 decimal places. See Figure 5 for the Sage program. The constant I obtained is

$$\prod_{l>2,\ l \text{ prime}} \left(1 - \frac{1}{(l-1)^2}\right) \approx 0.6601618159 = C \tag{13}$$

Now that I have this constant, and knowing that the second product is not an infinite one, I can now use (4) to compute c_p (see Figure 6) and $c_p \cdot P_1$.

In the paper, the table for Example 2 indicates the prime $p = 1000000019$, and using this number, I found that my value for c_p was considerably different from the authors'. However, p isn't actually a prime number, so we discovered that this was a typo, and should have read $p' = 10000000019$ (i.e., one zero was omitted in the paper.) With the corrected value, I found that

all of my results are identical to those of the authors. See Table 2 for the full results of this section.

Similar to the computation of P_1, I computed P_k using formula (8) given in [1], as well as the method defined above. For c_p, we follow the exact same computations, using the same constant C found above for the infinite product. For f_k, I also followed the formula (6) given in Conjecture 2, obtaining the same results as the paper's Table 1. Figure 7 illustrates the Sage program I used to compute the product $c_p \cdot f_k \cdot P_k$ and Figure 8 gives the Sage program for the P_k computation using the probability equation. See Tables 3-10 for these results.

3.2 Computing Q_k

The paper by Galbraith and McKee gave no indication on how to compute Q_k, only the following description, as mentioned above: Let Q_k denote the probability that for an elliptic curve over $\mathbb{F}_p$, $\frac{|E(\mathbb{F}_p)|}{k}$ is prime. Therefore, to compute Q_k, using the basic probability equation as above, we get

$$Q_k = \frac{\#\text{ of elliptic curves over } \mathbb{F}_p \text{ with } \frac{|E(\mathbb{F}_p)|}{k} \text{ prime}}{\#\text{ of elliptic curves over } \mathbb{F}_p} \tag{14}$$

Method 1: (Brute force) To compute Q_k, we first compute the number of possible elliptic curves over $\mathbb{F}_p$, for given p, by running through all $(a, b) \in \mathbb{F}_p^2$ (where $E : y^2 = x^3 + ax + b$) and verifying the discriminant rule. This gives us the denominator for Q_k. Then, we take each of those (a, b) pairs and compute the number of points of the corresponding elliptic curves using the *cardinality()* function in Sage. For fixed k, we take note of whether or not $\frac{|E(\mathbb{F}_p)|}{k}$ is prime. We take the total count of these elliptic curves to get the numerator for Q_k. See Figure 9 for computations and Table 11 for the (brief) results of this section.

This brute force method worked for primes up to $p = 10009$, however with my MacBook Pro, Memory: 8 GB 1600 MHz DDR3, it took approximately 19 hours to compute the numerator for $p = 10009$, and for the next prime in the list, $p = 100003$, my computer froze during the computation.

Therefore, we had to find another method. We use the following theorem, from Deuring, see [11], which uses the Hurwitz class number $H(D)$:

Theorem 3.1: Let $p > 3$, and let $N = p + 1 - a$ be an integer, where $-2\sqrt{p} \leq a \leq 2\sqrt{p}$. Then the number of elliptic curves E over $\mathbb{F}_p$ which have $|E(\mathbb{F}_p)| = N = p + 1 - a$ is

$$\frac{p-1}{2} H(a^2 - 4p) \tag{15}$$

where $H(D)$ is the Hurwitz class number (see §4 for definitions.)

With this theorem, we have method 2:

Method 2: (Using Deuring's Theorem) Sage has a program $qfbhclassno()$ which computes the Hurwitz class number. We apply Theorem 3.1 using this algorithm in Sage, which returns the values of Q_k, for all k, in minutes. See Figure 10 for the Sage program. There is also an algorithm in [3] § 5.3 which computes the Hurwitz Class Number, counting reduced quadratic forms:

Algorithm 3.2: Given a negative discriminant D:

1. [Initialize b] Set $H \leftarrow 1$, $b \leftarrow D \bmod 2$, $B \leftarrow \lfloor \sqrt{|D|/3} \rfloor$.
2. [Initialize a] Set $q \leftarrow (b^2 - D)/4$, $a \leftarrow b$ and if $a \leq 1$ set $a \leftarrow 1$ and go to step 4.

3'. [Test] If $a \not| q$ go to step 4. Now if $a = b$ then if $ab = q$ set $H \leftarrow H + 1/3$ otherwise set $H \leftarrow H + 1$ and go to step 4. If $a^2 = q$, then if $b = 0$ set $H \leftarrow H + 1/2$, otherwise set $H \leftarrow H + 1$. In all other cases (i.e. if $a \neq b$ and $a^2 \neq q$) set $H \leftarrow H + 2$.

4. [Loop on a] Set $a \leftarrow a + 1$. If $a^2 \leq q$ go to step 3.
5. [Loop on b] Set $b \leftarrow b + 2$. If $b \leq B$ go to step 2, otherwise output H and terminate the algorithm.

This can be translated to Pseudo-code 4.12 in the next section. Using this method to calculate the Hurwitz class number takes quite a bit longer than the built-in Sage program for large primes p. See Figure 11 for the Sage program. We will now discuss the theory behind this algorithm.

4 The Class Group and Class Numbers

The definitions, theorems, lemmas and propositions in this section (as well as their proofs) can be found in [3].

Definition 4.1: The *ring of integers* $\mathbb{Z}_K$ of an algebraic number field K is the ring of all integral elements contained in K. An *integral element* is a root of a monic polynomial with rational integer coefficients $x^n + c_{n-1}x^{n-1} + ... + c_0$, for some n.

Definition 4.2: J is a *fractional ideal* of $\mathbb{Z}_K$ if J is a $\mathbb{Z}_K$-submodule of K and there exists $r \in K$ such that $rJ \subseteq \mathbb{Z}_K$.

For example, $J = \{\frac{m}{6} \mid m \in \mathbb{Z}\}$, $r = 6$, where $K = \mathbb{Q}$.

All fractional ideals in $\mathbb{Q}$ are of this form, and are principal.

Fractional ideals form a monoid, in fact, a group. A fractional ideal I is called *invertible* if there is another fractional ideal J such that $IJ = \mathbb{Z}_K$, where $IJ = \{a_1b_1 + a_2b_2 + ... + a_nb_n \mid a_i \in I, b_i \in J, n \in \mathbb{N}\}$.

The set of principal ideals $P = \{r\mathbb{Z}_K \mid r \in K\}$, is normal in the group of fractional ideals G, i.e.,

$$P = \{r\mathbb{Z}_K \mid r \in K\} \triangleleft \{\text{Fractional ideals}\} = G$$

Definition 4.3: The *Class Group* of K is equal to the quotient group G/P and is denoted by $Cl(K)$.

$$G/P = G/_{I \sim J} \text{ if } I = rJ \text{ for some } r \in K.$$

Theorem 4.4: $Cl(K)$ is finite.

Definition 4.5: $|Cl(K)| =: h(K)$ is the *class number of K*.

Definition 4.6: A *binary quadratic form* is a function $f(x, y) = ax^2 + bxy + cy^2$, $a, b, c \in \mathbb{Z}$, denoted (a, b, c). We can represent them as $\begin{bmatrix} 2a & b \\ b & 2c \end{bmatrix}$ where $D = b^2 - 4ac$ is the *discriminant*. A quadratic form $f(\vec{x})$ is said to be *positive definite* if $f(\vec{x}) > 0$ for $\vec{x} \neq \vec{0}$. We say that f is *primitive* if $\gcd(a, b, c) = 1$. If f and g are two quadratic forms, we say that f and g are *equivalent* if there exists a matrix $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \in SL_2(\mathbb{Z})$, such that $g(x, y) = f(\alpha x + \beta y, \gamma x + \delta y)$.

Definition 4.7: A positive definite quadratic form (a, b, c) of discriminant D is said to be *reduced* if $|b| \leq a \leq c$ and if, in addition, when one of the two inequalities is an equality (i.e., either $|b| = a$ or $a = c$), then $b \geq 0$.

Definition 4.8: An integer D is called a *fundamental discriminant* if D is the discriminant of a quadratic field K. In other words, $D \neq 1$ and either $D \equiv 1 \pmod 4$ and is squarefree, or $D \equiv 0 \pmod 4$, $D/4$ is squarefree and $D/4 \equiv 2$ or $3 \pmod 4$.

We denote by $Cl(D) = h(D)$ the class number of primitive positive definite quadratic forms of discriminant D.

Proposition 4.9: In every class of quadratic forms of discriminant $D < 0$ there exists exactly one reduced form. In particular, $h(D)$ is equal to the number of primitive reduced forms of discriminant D.

Thus, instead of calculating $h(D) = |G/P|$ (which is hard), we can count the number of reduced quadratic forms.

Lemma 4.10: Let $f = (a, b, c)$ be a positive definite binary quadratic form of discriminant $D = b^2 - 4ac < 0$.
(1) If f is reduced, we have the inequality $a \leq \sqrt{|D|/3}$.
(2) Conversely, if

$$a < \sqrt{|D|/4} \text{ and } -a < b \leq a$$

then f is reduced.

We're looking in particular at fundamental discriminants, because otherwise, D is not the discriminant of any class field.

Definition 4.11: Let N be a non-negative integer. The *Hurwitz Class Number* $H(N)$ is defined as follows:
(1) If $N \equiv 1$ or $2 \pmod 4$ then $H(N) = 0$.
(2) If $N = 0$ then $H(0) = -1/12$.
(3) Otherwise (i.e. if $N \equiv 0$ or $3 \pmod 4$ and $N > 0$) we define $H(N)$ as the class number of not necessarily primitive (positive definite) quadratic forms of discriminant $-N$, except that forms equivalent to $a(x^2 + y^2)$ should be counted with coefficient $1/2$, and those equivalent to $a(x^2 + xy + y^2)$ with

coefficient 1/3.

Algorithm 3.2 counts the reduced quadratic forms (with the appropriate weights as defined in Definition 4.11) to find $H(D)$ in the following way:
Pseudo-code 4.12:

Fix $D < 0$:

Count the number of reduced forms (a, b, c) such that

- $b^2 - 4ac = D$
- $a \geq |b|$
- $c \geq a$
- $B = \sqrt{|D|/3}$, and so $a \leq B$ by Lemma 4.10, which also bounds b since $|b| \leq a$.
- $q = ac = \frac{b^2 - D}{4}$, and we want to find (a, c) such that $ac = q$, for $0 \leq a \leq c$.

If $a|q$, the algorithm runs from $a = 1$ to $\lfloor \sqrt{q} \rfloor$.
($\star$) If $a = |b|$ or $a = c$, from Definition 4.7, we get $b \geq 0$. (Add one reduced form $(a, b, q/a)$)
If ($\star$) fails, ($a \neq |b|$ and $a \neq c$) and if $b \neq 0$, then $(a, b, q/a)$ and $(a, -b, q/a)$ are two solutions. (Add two reduced forms)

The algorithm doesn't work for the cases $N = 3$ or 4. In fact, because of the following lemma, there are 2 roots of unity for $D < -4$.

Lemma 4.13: Let $\omega(D)$ be the number of roots of unity in the quadratic order of discriminant D, hence $\omega(-3) = 6$, $\omega(-4) = 4$ and $\omega(D) = 2$ for $D < -4$ and set $h'(D) = h(D)/(\omega(D)/2)$ (hence $h'(D) = h(D)$ for $D < -4$). Then for $N > 0$ we have

$$H(N) = \sum_{d^2|N} h'(-N/d^2)$$

and in particular if $-N$ is a fundamental discriminant, we have $H(N) = h(-N)$ except in the special cases $N = 3$ ($H(3) = 1/3$ and $h(-3) = 1$) and $N = 4$ ($H(4) = 1/2$ and $h(-4) = 1$).

Now one may be wondering: why does the algorithm start with $H = 1$? The answer is that the algorithm discounts the case where $a = 1$, for the following reason:

- if $D \equiv 0 \pmod 4$ and $a = 1$, then $b = 0$, $a = 1$ and $c = -D/4$, and $(1, 0, -D/4)$ is the only solution.

• if $D \equiv 1 \pmod 4$ and $a = 1$, then $b = a = 1$, $c = \frac{1-D}{4}$, and $(1, 1, \frac{1-D}{4})$ is the only solution.

While trying to write a Sage program for this algorithm, an error in Cohen's algorithm was discovered. Algorithm 3.2 described above includes Step 3' in [3], which replaces Step 3 in the original algorithm used to compute the class number $h(D)$. In this Step 3', the author, Cohen, forgets to specify the case where $b = 0$. Therefore, when we come across $b = 0$ in the algorithm, he is counting two reduced forms instead of one. Upon correcting this error, we find that the algorithm correctly computes the Hurwitz class number in our Theorem 3.1, however, as mentioned above, it does take a considerable amount of time for large primes.

5 Tables and Figures

Table 1: f_k Values

k	f_k	
1	1	
2	$\frac{3}{2}$	
3	$\frac{16}{15}$	if $p \equiv 1 \pmod 3$
	2	if $p \equiv 2 \pmod 3$
4	$\frac{5}{2}$	if $p \equiv 1 \pmod 4$
	2	if $p \equiv 3 \pmod 4$
5	$\frac{96}{95}$	if $p \equiv 1 \pmod 5$
	$\frac{4}{3}$	if $p \not\equiv 1 \pmod 5$
6	$\frac{8}{5}$	if $p \equiv 1 \pmod 3$
	3	if $p \equiv 2 \pmod 3$
7	$\frac{288}{287}$	if $p \equiv 1 \pmod 7$
	$\frac{6}{5}$	if $p \not\equiv 1 \pmod 7$
8	$\frac{9}{4}$	if $p \equiv 1 \pmod 4$
	3	if $p \equiv 3 \pmod 4$
9	$\frac{22}{15}$	if $p \equiv 1 \pmod 9$
	$\frac{7}{5}$	if $p \equiv 4, 7 \pmod 9$
	2	if $p \equiv 2 \pmod 3$
10	$\frac{144}{95}$	if $p \equiv 1 \pmod 5$
	2	if $p \not\equiv 1 \pmod 5$

```
sage: p=1009;
sage: x=var('x');
sage: a=p+1-(2*(N(sqrt(p))));
sage: b=p+1+(2*(N(sqrt(p))));
sage: P1=(1/(4*(N(sqrt(p)))))*N(integral((1/log(x)),x,a,b));
sage: P1
0.144574390251185

sage: N(1/log(p))
0.144577302528343
```

Figure 3: E.g. P_1 Computation using formula (5)

```
sage: p=10223473261;
sage: P=prime_range((p+1-2*sqrt(p)),(p+1+2*sqrt(p)+1));
sage: C=len(P);
sage: D=N(4*sqrt(p));
sage: print C/D
0.0428117787000808
```

Figure 4: E.g.: P_1 Computation using the definition (11)

```
sage: P=prime_range(3,31);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.665138396434486

sage: P=prime_range(3,100);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.661377084547185

sage: P=prime_range(3,1000);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.660245743970801

sage: P=prime_range(3,20000);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.660164861243491

sage: P=prime_range(3,1500000);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.660161844862224

sage: P=prime_range(3,64000000);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.660161816391855

sage: P=prime_range(3,400000000);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.660161815926356

sage: P=prime_range(3,450000000);
sage: N(prod((1-(1/((L-1)^2))) for L in P))
0.660161815917125
```

Figure 5: Sample: Infinite product calculations

```
sage: p=1009;
sage: x=var('x');
sage: L=var('L');
sage: a=p+1-(2*(N(sqrt(p))));
sage: b=p+1+(2*(N(sqrt(p))));
sage: P1=N(1/(4*(sqrt(p)))*integral(1/log(x),x,a,b));
sage: C=0.6601618159;
sage: R=prod((1+(1/((L+1)*(L-2)))) for L in prime_divisors(p-1) if
L<>2);
sage: Cp=(2/3)*C*R;
sage: p,P1,Cp,Cp*P1
(1009, 0.144574390251185, 0.563888217747917, 0.0815237952507325)
```

Figure 6: E.g. c_p and $c_p \cdot P_1$ Computation using P_1 formula (5)

```
sage: p=500000071;
sage: k=1;
sage: x=var('x');
sage: L=var('L');
sage: a=(p+1-(2*(N(sqrt(p)))))/k;
sage: b=(p+1+(2*(N(sqrt(p)))))/k;
sage: Pk=(1/(4*k*(N(sqrt(p)))))*N(integral((1/log(x)),x,a,b));
sage: C=0.6601618159;
sage: R=prod((1+(1/((L+1)*(L-2)))) for L in prime_divisors(p-1) if
L<>2);
sage: Cp=(2/3)*C*R;
sage: Cp
0.580697893615743

sage: p=500000071;
sage: k=10; sage: x=var('x');
sage: a=(p+1-(2*(N(sqrt(p)))))/k;
sage: b=(p+1+(2*(N(sqrt(p)))))/k;
sage: Pk=(1/(4*k*(N(sqrt(p)))))*N(integral((1/log(x)),x,a,b));
sage: Pk
0.000564094259581330
```

Figure 7: E.g. c_p, P_k Computation using P_k formula (8)

Table 2: My $c_p \cdot P_1$ Results and Difference with Galbraith and McKee's

p	P_1 (formula)	P_1 (definition)	c_p	$c_p \cdot P_1$ (using P_1 def.)	Difference (G & M - Me)
1009	0.14457	0.15741	0.56389	0.08876	0.00004
1019	0.14437	0.14097	0.44011	0.06204	0.00034
10007	0.10857	0.10746	0.44011	0.04730	-0.00011
10009	0.10856	0.10745	0.55016	0.05912	-0.00012
100003	0.08686	0.08617	0.56389	0.04859	-0.00045
100043	0.08686	0.08378	0.44011	0.03687	0.00001
199999	0.08193	0.07714	0.55048	0.04247	-0.00001
10000019	0.06204	0.06238	0.45121	0.02814	0.00001
10000079	0.06204	0.06222	0.44506	0.02769	0.00000
10000537	0.06204	0.06174	0.56923	0.03515	0.00000
500000003	0.04992	0.05042	0.44038	0.02221	0.00000
500000009	0.04992	0.05042	0.44011	0.02219	0.00000
500000041	0.04992	0.05045	0.58070	0.02929	0.00000
500000069	0.04992	0.05046	0.44019	0.02221	0.00000
500000071	0.04992	0.05046	0.58070	0.02930	0.00000
10000000019	0.04343	0.04330	0.44014	0.01906	0.00000
10000000033	0.04343	0.04330	0.55729	0.02413	0.00000
10000000061	0.04343	0.04330	0.46456	0.02011	0.00000
10000000069	0.04343	0.04330	0.55014	0.02382	0.00000
10000000097	0.04343	0.04329	0.44012	0.01905	0.00000
10000000121	0.04343	0.04330	0.46915	0.02031	0.00000
10000000147	0.04343	0.04330	0.55013	0.02382	0.00000
10000000259	0.04343	0.04331	0.44011	0.01906	0.00001
10000000469	0.04343	0.04333	0.44011	0.01907	0.00000
10000001251	0.04343	0.04334	0.58083	0.02517	0.00000
10000001551	0.04343	0.04333	0.60073	0.02603	0.00000
10000050061	0.04343	0.04335	0.60568	0.02626	0.00000
10223473261	0.04339	0.04281	0.60931	0.02609	0.00000

```
sage: p=10000000019; b=0;
sage: for a in [floor(p+1-2*sqrt(p))..ceil(p+1+2*sqrt(p))]:
...       if (a/10) in Primes():
...           b+=1;
sage: Pk=(b/(4*sqrt(p)));
sage: print N(Pk)
0.00478999999544950
```

Figure 8: P_k Computation using the definition (11) (e.g. $k = 10$ above)

Table 3: My $c_p \cdot f_k \cdot P_k$ Results: **Example 1:** $p = 500000071$, $c_p = 0.58069789$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04992	0.05046	1	0.02930	0.00000
2	0.01293	0.02547	3/2	0.02218	0.00000
3	0.00587	0.01747	16/15	0.01082	0.00000
4	0.00335	0.01336	2	0.01552	0.00000
5	0.00217	0.01093	96/95	0.00642	0.00000
6	0.00152	0.00928	8/5	0.00862	0.00000
7	0.00113	0.00814	6/5	0.00567	0.00000
8	0.00087	0.00708	3	0.01233	0.00000
9	0.00069	0.00621	7/5	0.00504	0.00000
10	0.00056	0.00560	144/95	0.00493	0.00000

Table 4: **Example 2:** $p = 1000000019$, $c_p = 0.44013535$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04825	0.04330	1	0.01906	0.00000
2	0.01248	0.02244	3/2	0.01482	0.00000
3	0.00566	0.01516	2	0.01334	0.00000
4	0.00323	0.01155	2	0.01016	0.00000
5	0.00209	0.00929	4/3	0.00545	0.00000
6	0.00147	0.00784	3	0.01036	0.00000
7	0.00109	0.00674	6/5	0.00356	0.00000
8	0.00084	0.00607	3	0.00802	0.00000
9	0.00067	0.00524	2	0.00461	0.00000
10	0.00054	0.00479	2	0.00422	0.00000

Table 5: **Example 3:** $p = 10000000033$, $c_p = 0.55728509$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04343	0.04330	1	0.02413	0.00000
2	0.01119	0.02244	3/2	0.01876	0.00000
3	0.00507	0.01516	16/15	0.00901	0.00000
4	0.00289	0.01155	5/2	0.01609	0.00000
5	0.00187	0.00929	4/3	0.00691	0.00000
6	0.00131	0.00784	8/5	0.00700	0.00000
7	0.00097	0.00674	6/5	0.00451	0.00000
8	0.00075	0.00607	9/4	0.00761	0.00000
9	0.00059	0.00524	7/5	0.00409	0.00000
10	0.00048	0.00479	2	0.00534	0.00000

Table 6: **Example 4:** $p = 10000000061$, $c_p = 0.46455831$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04343	0.04330	1	0.02011	0.00000
2	0.01119	0.02244	3/2	0.01564	0.00000
3	0.00507	0.01516	2	0.01409	0.00000
4	0.00289	0.01155	5/2	0.01341	0.00000
5	0.00187	0.00929	96/95	0.00436	0.00000
6	0.00131	0.00784	3	0.01093	0.00000
7	0.00097	0.00674	6/5	0.00376	0.00000
8	0.00075	0.00607	9/4	0.00635	0.00000
9	0.00059	0.00524	2	0.00487	0.00000
10	0.00048	0.00479	144/95	0.00337	0.00000

Table 7: **Example 5:** $p = 10000000069$, $c_p = 0.55013942$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04343	0.04330	1	0.02382	0.00000
2	0.01119	0.02244	3/2	0.01852	0.00000
3	0.00507	0.01516	16/15	0.00890	0.00000
4	0.00289	0.01155	5/2	0.01588	0.00000
5	0.00187	0.00929	4/3	0.00682	0.00000
6	0.00131	0.00784	8/5	0.00691	0.00000
7	0.00097	0.00674	6/5	0.00445	0.00000
8	0.00075	0.00607	9/4	0.00752	0.00000
9	0.00059	0.00524	7/5	0.00404	0.00000
10	0.00048	0.00479	2	0.00527	0.00000

Table 8: **Example 6:** $p = 10000000097$, $c_p = 0.44011975$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04343	0.04329	1	0.01905	0.00000
2	0.01119	0.02244	3/2	0.01481	0.00000
3	0.00507	0.01516	2	0.01334	0.00000
4	0.00289	0.01155	5/2	0.01271	0.00000
5	0.00187	0.00929	4/3	0.00545	0.00000
6	0.00131	0.00784	3	0.01035	0.00000
7	0.00097	0.00674	6/5	0.00356	0.00000
8	0.00075	0.00607	9/4	0.00601	0.00000
9	0.00059	0.00524	2	0.00461	0.00000
10	0.00048	0.00479	2	0.00422	0.00000

Table 9: **Example 7:** $p = 10000000121$, $c_p = 0.46915300$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04343	0.04330	1	0.02031	0.00000
2	0.01119	0.02243	3/2	0.01579	0.00000
3	0.00507	0.01516	2	0.01422	0.00000
4	0.00289	0.01155	5/2	0.01354	0.00000
5	0.00187	0.00929	96/95	0.00441	0.00000
6	0.00131	0.00784	3	0.01103	0.00000
7	0.00097	0.00674	6/5	0.00379	0.00000
8	0.00075	0.00607	9/4	0.00641	0.00000
9	0.00059	0.00524	2	0.00492	0.00000
10	0.00048	0.00479	144/95	0.00341	0.00000

Table 10: **Example 8:** $p = 10000000147$, $c_p = 0.55013485$

k	P_k (formula)	P_k (definition)	f_k	$c_p \cdot f_k \cdot P_k$ (using P_k def.)	Difference (G & M - Me)
1	0.04343	0.04330	1	0.02382	0.00000
2	0.01119	0.02243	3/2	0.01851	0.00000
3	0.00507	0.01516	16/15	0.00890	0.00000
4	0.00289	0.01155	2	0.01271	0.00000
5	0.00187	0.00929	4/3	0.00682	0.00000
6	0.00131	0.00784	8/5	0.00690	0.00000
7	0.00097	0.00674	6/5	0.00445	0.00000
8	0.00075	0.00607	3	0.01003	0.00000
9	0.00059	0.00524	7/5	0.00404	0.00000
10	0.00048	0.00479	2	0.00527	0.00000

```
sage: p=10009;
sage: F=Integers(p);
sage: R=[0..p-1];
sage: lis1=[];
sage: a=var('a');
sage: b=var('b');
sage: for a in R:
...       for b in R:
...           if (F(-16*(4*(a^3)+27*(b^2)))!=0):
...               lis1+=[(a,b)];
...
sage: length=len(lis1);
sage: print length
100170072

sage: lis2=[];
sage: for i in [0..(length-1)]:
...       a=lis1[i][0];
...       b=lis1[i][1];
...       E=EllipticCurve(F,[a,b]);
...       C=E.cardinality();
...       if (is_prime(C)==True):
...           lis2+=[C];
... sage: longlength=len(lis2);
sage: print longlength
6034824
```

Figure 9: Method 1: Q_k Computation

Table 11: Method 1: Q_k Results and Difference with the Text

p	# Total E.C. over $\mathbb{F}_p$	# E.C. over $\mathbb{F}_p$ with q points	Q_k	Difference (G & M - Me)
1009	1017072	94584	0.092996366	0.00066363
1019	1037342	75332	0.072620216	-0.00000022
10007	100130042	4767859	0.047616668	0.00000333
10009	100170072	6034824	0.060245779	0.00000422

```
sage: p = 10223473261; T=0;
sage: R = [round(-2*sqrt(p)+1/2)..round(2*sqrt(p)-1/2)];
sage: for a in R:
       N=p+1-a;
       if (is_prime(N)==True):
           HE = pari(-1*(a^2)+4*p);
           H = HE.qfbhclassno();
           M = ((p-1)/2)*H;
           T+=M;
sage: print T
2725122367141904730
```

Figure 10: Method 2: Q_k Computation using the built-in Hurwitz Class Number formula in Sage

```
sage: def three_prime(h,a,b,q):
       if ((q/a) not in ZZ): return h
       if a==b:
           if a*b==q: return h + 1/3
           return h+1
       if a*a==q:
           if b==0: return h + 1/2
           return h+1
       if b==0: return h+1
       return h+2


sage: def HCN(N):
       if N==0: return -1/12;
       R=Integers(4);
       if (R(N)==1 or R(N)==2): return 0;
       D = -N; H=1;
       if is_even(D):
              bb=0;
       else:
              bb=1;
       B=floor(sqrt(abs(D)/3));
       for b in [bb..B+1,step=2]:
             q=(b^2-D)/4; a=b;
             for a in [b..floor(sqrt(q))]:
                   if a>1: H = three_prime(H,a,b,q);
       return H
sage: p = 1009; T=0;
sage: R = [round(-2*sqrt(p)+1/2)..round(2*sqrt(p)-1/2)];
sage: for c in R:
          N=p+1-c;
          if (is_prime(N)==True):
              HE = (-1*(c^2)+4*p);
              HH = HCN(HE);
              M = ((p-1)/2)*HH;
              T+=M;
sage: print T
94584
sage: p = 1009; T=0; k=2
sage: R = [round(-2*sqrt(p)+1/2)..round(2*sqrt(p)-1/2)];
sage: for a in R:
       N=(p+1-a)/k;
       if (N in ZZ):
           if (N in Primes()):
               HE = pari(-1*(a^2)+4*p);
               HH = HCN(HE);
               M = ((p-1)/2)*HH;
               T+=M;
sage: print T
49392
```

Figure 11: Method 2: Q_k Computation using Algorithm 3.2 from [3]

Table 12: My results: $c_p \cdot P_1 \sim Q_1$ for large primes

p	$c_p \cdot P_1$	Q_1	Difference
10000000469	0.01907	0.01908	-0.00001
10000001251	0.02517	0.02513	0.00004
10000001551	0.02603	0.02609	-0.00005
10000050061	0.02626	0.02627	-0.00001
10223473261	0.02609	0.02607	0.00001

Table 13: My results: $c_p \cdot f_k \cdot P_k \sim Q_k$ for large primes and small values of k

p	k	$c_p \cdot f_k \cdot P_k$	Q_k	Difference
10000000469	1	0.01907	0.01908	-0.00001
	2	0.01481	0.01481	0.00000
	3	0.01336	0.01337	-0.00002
10000001251	1	0.02517	0.02513	0.00004
	2	0.01956	0.01958	-0.00002
	3	0.00940	0.00946	-0.00006
10000001551	1	0.02603	0.02609	-0.00005
	2	0.02023	0.02030	-0.00007
	3	0.00973	0.00970	0.00003
10000050061	1	0.02626	0.02627	-0.00001
	2	0.02027	0.02037	-0.00009
	3	0.00982	0.00976	0.00006
10223473261	1	0.02609	0.02607	0.00001
	2	0.02039	0.02055	-0.00016
	3	0.00997	0.00993	0.00004

6 Relevance to Modern Cryptography

In 1985, Neal Koblitz and Victor S. Miller independently suggested using the algebraic structure of elliptic curves over finite fields in public-key cryptosystems. Similar to protocols based on elementary number theory (e.g. Diffie-Hellman key exchange and ElGamal cryptosystem), elliptic curve cryptosystems are based on the infeasibility of solving the discrete logarithm problem of a random point on the elliptic curve with respect to a publicly-known base point.

Definition 6.1: The following problem is known as the *Discrete Logarithm Problem* (DLP): Given g, A, and a prime p such that $A \equiv g^a \pmod{p}$, find a.

Definition 6.2: Let E be an elliptic curve over the finite field $\mathbb{F}_p$ and let P and Q be points in $E(\mathbb{F}_p)$. The *Elliptic Curve Discrete Logarithm Problem* (ECDLP) is the problem of finding an integer n such that $Q = nP$.

The DLP and ECDLP are "one-way functions", i.e., functions that are easy to compute, but whose inverses are difficult to compute. The fastest known algorithm to solve the ECDLP in $E(\mathbb{F}_p)$ takes approximately $\sqrt{p}$ steps (see [5]), which makes the ECDLP appear much more difficult than the DLP, whose fastest known algorithm (a number field sieve algorithm) takes approximately $e^{(\log p)^{1/3}(\log(\log p))^{2/3}}$ time (see [6]).

The following is an excerpt from the 1999 article *Recommended Elliptic Curves for Federal Government Use*, section 1.1 Choice of Key Lengths, published by the Computer Security Division of the National Institute of Standards and Technology (NIST) ([8]):

> The principal parameters for elliptic curve cryptography are the elliptic curve E and a designated point G on E called the *base point*. The base point has order r, a large prime. The number of points on the curve is $n = fr$ for some integer f (the *cofactor*) not divisible by r. For efficiency reasons, it is desirable to take the cofactor to be as small as possible.

In that article, the authors use cofactors 1, 2 and 4 as examples. By using these cofactors, the public key (data used for encryption that everybody has access to) and the private key (secret data for decryption only one party

has access to) are approximately the same length, and the private key is approximately twice the symmetric cryptographic security length. The symmetric cryptographic security is a logarithmic measure of the fastest known computational attack on the algorithm, for example $\sqrt{p}$ for the ECDLP, as seen above. It is currently believed that having a symmetric cryptographic security approximately half the length of the private key gives an effectively secure cryptosystem in elliptic curve cryptography. Therefore, according to the above publication by the U.S. government, using cofactors $f = 1$, 2 and 4, and a sufficiently large prime r, an elliptic curve having fr points will be effectively secure against currently known attacks. As an example, the National Security Agency (NSA) recommends using curve P-256, of the form $E : y^2 \equiv x^3 - 3x + b \pmod{p}$, $f = 1$, $p = 115792089210356248762697446949407573530086143415290314195533631308867097853951$ and $r = 115792089210356248762697446949407573529996955224135760342422259061068512044369$.

7 Conclusion

In conclusion, I performed my own calculations of Conjectures 1 and 2 from Galbraith and McKee's paper [1] with almost identical results, possibly using a different computation method. Thus, my work provides numerical evidence for these conjectures and indicates that they give a very good estimate of the probability that an elliptic curve over $\mathbb{F}_p$ has k times a prime for its number of points. (See Tables 12 and 13.)

During this process, I became very familiar with a new mathematical software system, having no previous experience with Sage or Python. I found Sage to be a very easy program to use and to be very convenient, in that it is free and simply runs on the internet. I believe my acquired proficiency in Sage will prove to be very useful in my future endeavours.

Lastly, I learned about the Hurwitz class number and its application to computing the cardinality of an elliptic curve, as well as the notion of reduced quadratic forms and their one-to-one correspondence with the class number.

The application of Conjectures 1 and 2 to Elliptic Curve Cryptography is clear, from the quote from [8] given in §6, which indicates that the number of points on an elliptic curve to be used for cryptographic purposes should

be $|E(\mathbb{F}_p)| = kq$ with k small and q prime, i.e., $\frac{|E(\mathbb{F}_p)|}{k}$ should be prime.

Therefore, the question addressed in [1], "what is the probability that a randomly chosen elliptic curve over $\mathbb{F}_p$ has kq points, where k is small and q is prime?" addresses a cryptographic need to find the most secure elliptic curves for encryption possible. With Conjectures 1 and 2, we can choose our prime p by selecting the highest likelihood of finding an elliptic curve over $\mathbb{F}_p$ with a number of points equal to k times a prime, i.e., the largest value $c_p \cdot f_k \cdot P_k$.

8 References

[1] Galbraith, Steven D., McKee, James. *The Probability that the Number of Points on an Elliptic Curve over a Finite Field is Prime.* J. London Math. Soc. (2) 62, 2000, 671-684.

[2] Schoof, René. *Counting Points on Elliptic Cures over Finite Fields.* Journal de Théorie des Nombres de Bordeaux 7, 1995, 219-254.

[3] Cohen, Henri. *A Course in Computational Algebraic Number Theory.* Springer-Verlag, 1993.

[4] Chen, Rong-Jaye. *Schoof-Elkies-Atkin Algorithm.* Department of Computer Science, National Chiao Tung University, 2008.

[5] Hoffstein, Jeffrey, Pipher, Jill, Silverman, J.H. *An Introduction to Mathematical Cryptography.* Springer, 2008.

[6] Holden, Joshua. *A Tour of Public Key Cryptography (and of Number Theory).* Rose-Hulman Institute of Technology, Lecture - March 17, 2001.

[7] *NSA Suite B Cryptography.* National Security Agency, Government of the United States of America, posted January 2009, reviewed June 2014.

[8] *Recommended Elliptic Curves for Federal Government Use.* Computer Security Resource Center, Computer Security Division, National Institute of Standards and Technology, Government of the United States of America, 1999.

[9] Sutherland, Andrew. *Lecture Notes on Elliptic Curves, Lecture 8.* MIT Mathematics, 2013.

[10] Ritzenhaler, Christophe. *Elliptic curves and applications to cryptography.* Université de Marseille, 2011-2012.

[11] Cox, David A. *Primes of the form $x^2 + ny^2$: Fermat, Class Field Theory, and Complex Multiplication.* Wiley, 1989.