Discrete Logarithm Based Cryptography with Abelian Varieties

April 22, 2015

Prepared in fulfillment of an undergraduate honours project by

Kevin Johnson

Supervised by

Carlisle Adams

The University of Ottawa
Faculty of Engineering

Abstract

In recent years there has been increased interest in public key cryptography utilizes the group of points on elliptic curves. In this report we develop notions central to cryptography using elliptic curves and generalize thess ideas to hyperelliptic curves. A discussion of even more general geometric objects, called abelian varieties, is also given. Explicit descriptions for algorithms needed to implement public key cryptography supplement the discussion. Throughout this report, all curves are defined over the field $\mathbb{F}_p$ where p is an odd prime.

Contents

All algorithms found in this report have been implemented in Python and can be found at www.github.com/kjson/AVcrypto. Furthermore, pseudocode for some of the longer algorithms presented can be found in section 5.

1 The Discrete Logarithm Problem

Given an arbitrary finite cyclic group G with group operation $\cdot$ and generator $g \in G$, discrete exponentiation by a in G is defined by

$$g^a = \overbrace{g \cdot g \cdots g}^{a \text{ times}}$$

If $y = g^a$ is known, computing a is called finding the discrete logarithm of y. With the method of fast exponentiation, y can be computed quickly, only $O(\log\, a)$ group operations. On the other hand, computing a can be much harder. In fact, in [5] it was shown that in an arbitrary group for which only the group operation and discrete exponentiation can be applied to group elements, computing discrete logarithms will take at least $O(\sqrt{|G|})$ operations. In most cases though, more structure is known about the group in use.

1.1 The Diffie-Hellman Key Exchange

This one-way property of discrete exponention has proven to be very useful for cryptographic purposes. The most notable of these is in the Diffie-Hellman Key Exchange Protocal in which two parties A and B wish to share a secret key k.

Algorithm 1 Diffie-Hellman Key Exchange Protocal

```
1: $A$ and $B$ share a publicly known group $G$ and generator $g$.
2: $A$ chooses a random private exponent $a$ and computes $g^a$.
3: $B$ chooses a random private exponent $b$ and computes $g^b$.
4: $A$ sends $g^a$ to $B$ and $B$ sends $g^b$ to $A$.
5: $A$ raises $g^b$ to their own private exponent $a$ to obtain $k = (g^b)^a = g^{ab}$.
6: $B$ raises $g^a$ to their own private exponent $b$ to obtain $k = (g^a)^b = g^{ba}$.
```

The two parties may now use k to communicate with any cryptographically secure communication protocol needing a symetric key. The described protocol relies on the hardness of computing g^{ab} given g^a and g^b, which is conjectured in [6] to be equivalent to computing discrete logarithms. This protocal has the nice property that it's applicable to any finite cyclic group. We look at some simple examples of groups which have already had significant use in cryptography.

1.2 A Brief History of the Groups $\mathbb{F}_p^*$ and $\mathbb{F}_{2^n}^*$

Given a finite field $\mathbb{F}$, the multiplicative units $\mathbb{F}^* = \mathbb{F} \backslash \{0\}$ form a finite cyclic group and thus may be used for discrete logarithm based cryptography. It is a standard result from algebra that every finite field has order p^n, where p is a prime and $n \in \mathbb{Z}^+$. We divide the discussion of the cryptographic properties of the group $\mathbb{F}_{p^n}^*$ into two cases; when $n = 1$ and when $n > 1$.

In the latter case, when working with finite fields of order p^n, the arithmetic is only really tractable when $p = 2$. In the 1980's, researchers at the University of Waterloo made attempts to construct discrete logarithm cryptosystems based on $\mathbb{F}^*_{2^{127}}$ which initially paralelled RSA in terms of bits of security. But in 1986, Don Coppersmith,a professor at Carleton University, devised an astonishing algorithm in which could compute discrete logarithms in the group $\mathbb{F}^*_{2^{127}}$ in about 5 minutes. Further attempts were made to increase the size to $n = 593$ but similar adaptations of Coppersmith's algorithm made researchers abandon public key cryptosystems based on the discrete logarithm problem in $\mathbb{F}^*_{2^n}$.

When $n = 1$, we have a group which is essentially just the non-zero integers mod a prime. As one might expect, when p is small, computing discrete logs in $\mathbb{F}^*_p$ can be done quickly with just trial exponentiation. When p is large though, say $p = 2^{1000}$, this method becomes completely intractable, even on todays fastest computers. That being said, there is an attack described in [1] which adapts the Number Field Sieve to solve discrete logs in $\mathbb{F}^*_p$. This attack has running time very similar to factoring

$$O(p) = e^{1.923(\log p)^{1/3}(\log\log p)^{2/3}}$$

This basically means that the bit length required for p in $\mathbb{F}^*_p$ based cryptosystems is the same as the bit length required for the modulus in RSA. In todays standards that means $p = 2^{2048}$. Although this is cryptographically viable, in practice using such large values of p has its limitations, such as bandwidth in network communications or memory in a hand-held devices. These two cases made researchers search for alternative groups with cryptographically strong properties.

2 Abelian Varieties

In this report we focus on groups which arise from the solution set of polynomial equations over finite fields. The most famous of these groups is the set of points on an elliptic curve characterized by the equation $y^2 = x^3+ax+b$ where $a, b \in \mathbb{F}_p$ satisfy $4a^3+27b^2 \not\equiv 0 \bmod p$. In recent years, groups arising from these equation have found significant success in public key cryptography even though it is still an open questions whether these kind of groups are actually cryptographically secure! One of the benefits of elliptic curves is that the fastest known attack on them has $O(\sqrt{N})$ complexity, where N is the order of the group. This means significantly smaller keys can be used in comparison to the key length of RSA. A natural question is

What about other polynomial equations?

It turns out that there are infinitely many groups which arise from the solution sets of polynomial equations. Which of these groups are cryptographically viable is a vast active area of research. First we develop some notations required to describe these groups.

Let K be a field. Let $A = K[x_1, ..., x_n]$ represent the polynomial ring in n variables over K. Given a subset $T \subseteq A$, we may define

$$Z(T) = \{P \in K^n \mid f(P) = 0 \text{ for all } f \in T\}$$

to be the set of all common zeros of polynomials in T. We call $Y \subseteq K^n$ an *algebraic set* if $Y = Z(T)$ for some subset $T \subseteq A$. A not so obvious but usual fact to keep in mind is that if $T \subseteq A$ and $J = \langle T \rangle$ is the ideal generated by elements of T, then $Z(T) = Z(J)$. We say an algebraic set Y is reducible if it can be written as the union of two smaller algebraic sets. For example, let $Y = Z(x^2 - yz, xz - x)$ as a subset of $\mathbb{C}^3$. That is, Y is the set of complex valued points in $\mathbb{C}^3$ which satisfy each of the equations $x^2 - yz$ and $xz - x$. Notice that

$$\begin{aligned} Y &= Z(x^2 - yz, xz - x) \\ &= Z(x^2 - yz, x(z - 1)) \\ &= Z(x^2 - yz, x) \cup Z(x^2 - yz, z - 1) \\ &= Z(yz, x) \cup Z(x^2 - yz, z - 1) \\ &= Z(y, x) \cup Z(z, x) \cup Z(x^2 - yz, z - 1) \end{aligned}$$

So Y is reducible in $\mathbb{C}^3$. If an algebraic set is not reducible, it is called irreducible. It is the irreducible algebraic set which we shall loosely call *algebraic varieties.* For example, take the curve $y^2 = x^3 + ax + b \in \mathbb{F}_p[x, y]$ where $4a^3 + 27b^2 \not\equiv 0 \bmod p$. Then $E = Z(y^2 = x^3 + ax + b)$ is an algebraic variety. In fact, it is the simplest example of an *abeilan variety* - A projective complete algebraic variety with a group structure on its points given by smooth morphisms. This definition is beyond what is important for this report, but essentially, an abelian variety is a set of solutions of polynomials which can be endowed with operations on its points which make it into a group. These are precisely the groups which arise from polynomial eqautions. For the remaining sections of this report, we will discuss abelian varieties of dimension 1. That is plane curves over finite fields.

3 Elliptic Curves

Let p be an odd prime and let $E = Z(y^2 - x^3 - ax - b)$. We call E an *elliptic curve* defined over $\mathbb{F}_p$ if $4a^3 + 27b^2 \not\equiv 0 \bmod p$. It was first realized by the mathematicians Abel and Jacobi in the 1700's that, remarkably, the points of E (along with a special point $\mathcal{O}$ called the point at infinity) can be made into a group using a very specific group operation. In this section we describe this group operation along with other algorithms needed for cryptographic purposes.

3.1 The Group Operation

Given two $\mathbb{F}_p$-rational points $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ which we want to add together, we first define the value

$$s = \begin{cases} \frac{y_1 - y_2}{x_1 - x_2} \bmod p & \text{if } P_1 \neq P_2 \\ \frac{3x_1^2 - a}{2y_1} \bmod p & \text{if } P_1 = P_2 \end{cases}$$

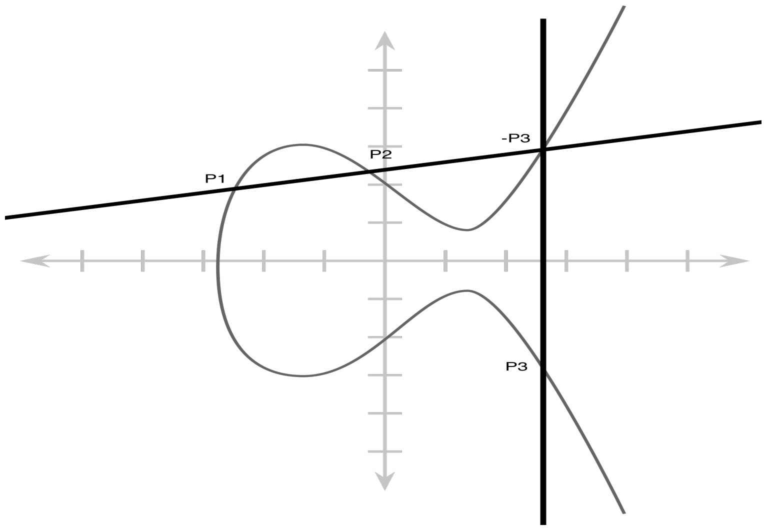
Then we define the coordinates of the point $P_3 = P_1 + P_2$ to be

$$x_3 = s^2 - (x_1 + x_2)$$
$$y_3 = s(x_3 - x_1) - y_1$$

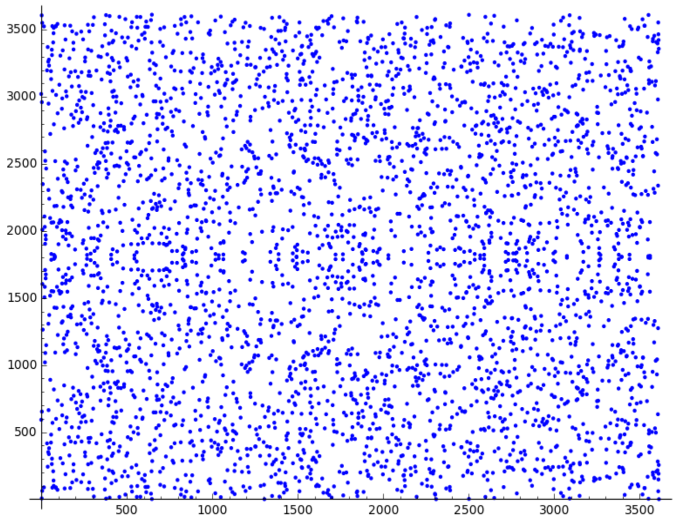
It's not immediately obvious that $P_3 = (x_3, y_3)$ is even a point on E and less obvious that this operation satisfies the axioms of a group. The prrof of associativity is quite lenthy but we will at least show that the operation is closed in E. Let $u = x_1 + x_2$. We verify that $y_3^3 = x_3^3 + ax_3 + b \bmod p$

$$\begin{aligned}
x_3^3 + ax_3 + b &= (s^2 - u)^3 + a(s^2 - u) + b \\
&= (s^4 - 2s^2u + u)(s^2 - u) + a(s^2 - u) + b \\
&= s^6 - s^4u - 2s^4u + 2s^2u^2 + us^2 - u^2 + a(s^2 - u) + b \\
&= s^6 - 3s^4u + s^2(2u^2 + u + a) - u^2 - au + b \\
&= s^2((s^4 - 2s^2u + u^2) - 2(s^2 - u)x_1 + x_1^2) - 2s^3y_1 - 2suy_1 - 2sx_1y_1 + y_1^2 \\
&= s^2((s^2 - u)^2 - 2(s^2 - u)x_1 + x_1^2) - 2s(s^3 - u - x_1)y_1 + y_1^2 \\
&= s^2(x_3 - x_1)^2 - 2s(x_3 - x_1)y_1 + y_1^2 \\
&= (s(x_3 - x_1) - y_1)^2 \\
&= y_3^2
\end{aligned}$$

There is also a nice pictoral way to visualize the addition over the real number $\mathbb{R}$. Given points P_1, P_2, we calculate P_3 by finding the point of intersection of the line from P_1 to P_2 and E and relecting it across the x axis.



One may wonder about the inherent geometry of this curve, perhaps this may be exploited to compute the descrete logarithms in E. Towards this, it should be noted that the previous curve is defined over the real numbers. The picture changes quite drastically over fintite fields. For example, take p to be the prime 3613. Then the elliptic curve given by the equation $y^2 = x^3 + 2x + 3$ is graphed as



So when the base field is finite, geometry is less apparent. The following algorithm describes how to add points on an elliptic curve.

Algorithm 2 The addition of two points $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ on an elliptic curve $E : y^2 + x^3 + ax + b$

```
1: function Add($E$,$P_1$, $P_2$)
2:     if $P_1 = \mathcal{O}$ then
3:         return $P_2$
4:     else if $P_2 = \mathcal{O}$ then
5:         return $P_1$
6:     else if $P_1 = P_1$ then
7:         $s \leftarrow (3x_1^2 - a)(2y_1)^{-1} \bmod p$
8:     else
9:         if $x_1 \neq x_2$ then
10:            $s \leftarrow (y_1 - y_2)(x_1 - x_2)^{-1} \bmod p$
11:        else
12:            return $\mathcal{O}$
13:        end if
14:    end if
15:    $x_3 \leftarrow s^2 - x_1 - x_2 \bmod p$
16:    $y_3 \leftarrow y_1 + s(x_3 - x_1) \bmod p$
17:    return $P_3 = (x_3, y_3)$
18: end function
```

Note that in steps 7 and 10, modular inverses must be calculated. Also note that if $x_1 = x_2$ but $P_1 \neq P_2$, then the second coordinates satisfy $y_1 = -y_2$. So the line from P_1 to P_2 is just a vertical line at x_1. This is taken to be the point at infinity $\mathcal{O}$. Also, when implementing this algorithm, the point at infinity $\mathcal{O}$ may be instantiated with any variable because no arithmetic is ever performed on it.

3.2 Scalar Multiplication of a Point

For most discrete logarithm protocols (such as Diffie-Hellman or DSA), we require to add point P to itself many times in order to perform discrete exponentiation. That is, given an integer m we need to calculate

$$[m]P = \overbrace{P + P + \cdots + P}^{m \text{ times}}$$

fast in order to be cryptographically reasonable. The following algorithm uses fast exponentiation by squaring in $E(\mathbb{F}_p)$ to achieve this.

Algorithm 3 Scalar multiplication of a point P by an integer m

```
function ScalarMult($m$,$P$)
    if $m = 0$ then
        return $\mathcal{O}$
    else if $m = 1$ then
        return $P$
    else if $m \equiv 0 \bmod 2$ then
        return ScalarMult($m/2$,$P + P$)
    else
        return $P$ + ScalarMult($m - 1$,$P$)
    end if
end function
```

3.3 Finding Points

Now that we have developed the arithmetic on an elliptic curve E, the next step is to find points on E. If $y^2 = x^3 + ax + b$ for $a, b \in \mathbb{F}_p$, then finding $\mathbb{F}_p$-rational points on E is equivalent to determining if $x^3 + ax + b$ is a square mod p. This is a classical problem in number theory which can be reformulated to determing the value of the *Legendre Symbol* of a mod p, which is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a square mod } p \\ -1 & \text{if } a \text{ is not a square mod } p \\ 0 & \text{if } p \text{ divides } a \end{cases}$$

where (in our case) $a = x^3+ax+b$. The following five properties let us determine whether a is a square mod p in polynomial time. Let $a, b \in \mathbb{Z}$ and p, q be odds primes.

(i) If $a \equiv b \bmod p$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$

(ii) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{p}\right)$

(iii) $\left(\frac{-1}{p}\right) = 1$ if $p \equiv 1 \bmod 4$, and $\left(\frac{-1}{p}\right) = -1$ if $p \equiv 3 \bmod 4$

(iv) $\left(\frac{2}{p}\right) = 1$ if $p \equiv \pm 1 \bmod 8$, and $\left(\frac{2}{p}\right) = -1$ if $p \equiv \pm 3 \bmod 8$

(v) If p, q are distinct, then

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right) \text{ if } p \text{ or } q \equiv 1 \bmod 4$$

$$\left(\frac{p}{q}\right) = -\left(\frac{q}{p}\right) \text{ if } p \equiv q \equiv 3 \bmod 4$$

For example, to determine if 105 is a square mod 227, we simply compute

$$\begin{aligned}
\left(\frac{105}{227}\right) &\overset{\text{(ii)}}{=} \left(\frac{3}{227}\right)\left(\frac{5}{227}\right)\left(\frac{7}{227}\right)\\
&\overset{\text{(v)}}{=} (-1)\left(\frac{227}{3}\right)\left(\frac{227}{5}\right)(-1)\left(\frac{227}{7}\right)\\
&\overset{\text{(i)}}{=} \left(\frac{2}{3}\right)\left(\frac{2}{5}\right)\left(\frac{3}{7}\right)\\
&\overset{\text{(iv)}}{=} (-1)(-1)(-1)\left(\frac{7}{3}\right)\\
&\overset{\text{(i)}}{=} (-1)\left(\frac{1}{3}\right)\\
&= -1
\end{aligned}$$

So 105 is a not a square mod 227. The process of determining if an integer a is a square mod p is fairly quick but doesn't actually tell us the squareroot of a. If $\left(\frac{a}{p}\right) = 1$, then we may find y such that $y^2 = a \bmod p$ by breaking the calcultions into two cases: whether $p \equiv 3$ or $p \equiv 1 \bmod 4$. First, if $p \equiv 3 \bmod 4$, then $y = a^{(p+1)/4}$ satisfies $y^2 \equiv a \bmod p$. The second case where $p \equiv 1 \bmod 4$ is more involved. Pick random r such that $\left(\frac{r^2-4a}{p}\right) = -1$ and write $d = r^2 - 4a$. let $\alpha = \frac{r+\sqrt{d}}{2}$ and $\alpha^{\frac{p+1}{2}} = \frac{y+v\sqrt{d}}{2}$ where y, v are the coefficients of the elements 1 and $\sqrt{d}$ in the $\frac{p+1}{2}$-th power expansion of α. Then y satisfies $y^2 \equiv a \bmod p$. Putting all this together we obtain the following algorthm which finds random points on an elliptic curve E.

Algorithm 4 Find random points on elliptic curve $E : y^2 - x^3 - ax - b$ modulo an odd prime p

```
1: function RandomPoints($E$,$p$)
2:     $c \leftarrow x^3 + ax + b$
3:     pick random $x \in \{1, \ldots, p-1\}$ with Legendre($c$,$p$) $= 1$
4:     if $p \equiv 3 \bmod 4$ then
5:         $y \leftarrow c^{(p+1)/4}$
6:     else
7:         pick random $x \in \{1, \ldots, p-1\}$ with Legendre($r^2 - 4c$,$p$) $= -1$
8:         $d \leftarrow r^2 - 4c, \alpha \leftarrow \frac{r+\sqrt{d}}{2}, y \leftarrow \alpha^{\frac{p+1}{2}} - \frac{1}{2}v\sqrt{d}$
9:     end if
10:    return $(x, y)$
11: end function
```

3.4 Counting Points

When using an elliptic curve E for discrete logarithm based cryptosystems, it is importance to know the number of points on E. This is because in 1978 Stephan Pohlig and Martin Hellman came up with an attack, described in [3], which uses the order of the group to solve discrete logs. The attack proceeds as follows: Given G, a finite cyclic group with order N, we may factor $N = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s}$ where $p_1, p_2, ..., p_s$ are primes and $e_1, e_2, ..., e_s \in \mathbb{Z}^+$. All the subgroups $G_1, G_2, ..., G_s$ of G will have order $p_1^{e_1}, p_2^{e_2}, ..., p_s^{e_s}$ respectively. Given an element $h = g^a \in G$, the Pohlig-Hellman attack solves for a in the smaller subgroups $G_1, G_2, ..., G_s$ and uses the Chinese remainder theorem to piece these solutions back together to solve for a in the bigger group G. What they noticed is that when using this method, the complexity of solving for a in G (which can be done in $O(\sqrt{N})$ bit operations) gets reduced to $O(p_1^{e_1/2}) + O(p_2^{e_2/2}) + O(p_s^{e_s/2})$, i.e. the sum of the complexities of solving in the smaller groups. This means a group's bits of security is only as high as the bits of security of its largest subgroup. Therefore we want the order of the groups that we use to be prime, or at least a very small multiple of a prime, to avoid this attack.

With this is mind, we need an algorithm which calculates the number of points on an elliptic curve E and thus the order of the group E. Intuitively, the number of points must be much less than p^2, for this would correspond to a curve that zeros every point in $\mathbb{Z}_p^2$. In fact, a much better estimate is given by Helmut Hasse. Denote the number of $\mathbb{F}_p$ rational points on E by $\#E(\mathbb{F}_p)$, then

$$\#E(\mathbb{F}_p) = p + 1 + t \tag{1}$$

where $|t| \leq 2\sqrt{p}$. So essentially finding $\#E(\mathbb{F}_p)$ is equivalent to finding t. The first tractable algorithm to do so was designed by Rene Schoof in 1985 and was inspired by a very special map called *the map of Frobeneous*

$$\begin{aligned} \phi &: E \to E \\ (x, y) &\mapsto (x^p, y^p) \end{aligned}$$

which happens to satisfy the equation

$$\phi^2 + p = -t\phi \tag{2}$$

in the endomorphism ring of E. That is, (2) is an equation of group homomorphisms from $E \to E$. The idea is to find t in the interval $[-2\sqrt{p}, 2\sqrt{p}]$ which gives an equivalnce in (2). The problem is when p is large (300 bits), this interval is just too big to brute force. Schoof's idea was to reduce this guess work by computing t satisfying (2) in the l-torsion subgroups $E[l] \subset E$ (the subgroup of all points P satisfying $[l]P \equiv \mathcal{O}$). Once this is done for a sufficient number of small primes l, the Chinese Remainder Theorem can be used to recover t. A detailed explanation of Schoof's algorithm can be found in algorithm 5 in section 5.

4 Hyperelliptic Curves

Unlike elliptic curves, when the genus of a curve is greater than one, then the set of points on it will not always form a group.

4.1 The Jacobian of a Hyperelliptic Curve

Luckily, there is another way to form an abelian group with hyperelliptic curves. Indeed, let D be the set of all formal finite sums

$$\sum_i m_i P_i$$

where $m_i \in \mathbb{F}_p$ and P_i are points on the curve H. We call elements of D divisors of H. Given a rational function f in $\mathbb{F}_p(H)$, we can define the corresponding divisor to f as

$$(f) = \sum_i m_i P_i$$

where P_i are the zeros and poles of f with number of multiplicities m_i. Divisors of this form are called principal divisors and we let P denote the subset of all of them in D. If we define the operation on D by

$$\sum_i m_i P_i + \sum_i m_i' P_i = \sum_i (m_i + m') P_i$$

then D becomes and abelian group. Unfortunetly, this group is far too large and unstructured for cryptographic purposes. So we consider the subgroup D^0 of all divisors of D whose coefficients sum to 0. That is, divisors $\sum_i m_i P_i$ such that $\sum_i m_i = 0$. This subgroup is still infinite, but that can be remedied by defining two divisors D_1, D_2 of D^0 to be equal if $D_1 - D_2$ is equal to the divisor of a rational function on H. That is, $D_1 - D_2 = (f)$ for $f \in \mathbb{F}_p(H)$. This new quotient group, denoted

$$J = D^0/P$$

is called the jacobian of the curve H and is a finite cyclic group. Since elements of J are in fact equivalence classes of divisors we use the symbol $[D]$ to represent them. This will be the group used to build hyperelliptic cryptosystems.

4.2 Representation of Divisors

Athough the Jacobian J of an hyperelliptic curve H is a finite abelian group, elements of J can be very hard to represent. To make the group operation in J reasonable, we ustilize the Mumford representation of a divisor which is described as follows. We call a divisor D *semi-reduced* if

$$D = \sum_{P \in H \setminus \mathcal{O}} m_P P - \sum_{P \in H \setminus \mathcal{O}} m_P \mathcal{O}$$

satisfies

(i) $m_i \geq 0$

(ii) If $P \neq P'$ and $m_P > 0$, then $m_{P'} = 0$

(iii) If $P = P'$ and $m_P > 0$, then $m_p = 0$

A semi-reduced divisor D is completely reduced if $D = \sum_{P \in H} m_P \leq \mathfrak{g}$. It turns out that for every equivalence class $[D]$ in J there exists an unique reduced divisor D' such that $[D\prime] = [D]$. An interesting consequence is that every element in J can be represented with the sum of g points. So for a hyperelliptic curves of genus 2, there are elements of J which cannot be represented with less than 2 points. In the case of elliptic curves, the genus is 1, so $J \cong E$. This also means that acurately storing all reduced divisors is equivalent to storing all of J on a computer! Although this is convenient, the presentation is still combersome. Ideally, we would like to represent divisors algebraically, which is what the Mumford representation allows us to do. Let D be a semi-reduced with points $P_i = (x_i, y_i)$. We associate to D polynomials $a, b \in \mathbb{F}_p[x]$ such that

$$a(x) = \prod_{i}^{r} (x - x_i)$$

$$b(x_i) = y_i \ 1 \leq i \leq r$$

where $\deg b < \deg a$ and $(x - x_i)^{k_i} \mid b - y_i$, if k_i is the multiplicity of P_i. Denote this representation $D \overset{\text{def}}{=} \operatorname{div}(a, b)$.

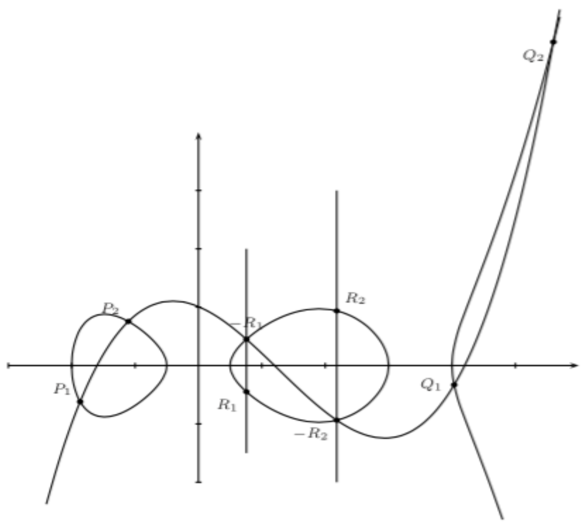
4.3 The Group Operation

Now that we have a computationally feasable way of storing divisors on H computationally, the next question is how to define the arithmetic on H. The group operation, as described in [2], is defined on divisors in thier Mumford representation

Algorithm 5 Addition of two divisors $D_1 = \operatorname{div}(u_1, v_1), D_2 = \operatorname{div}(u_2, v_2)$ in the Jacobian of a hyperelliptic curve H defined by $y^2 = f(x)$ of genus $\mathfrak{g}$ over $\mathbb{F}_p$

```
 1: function ADD($H$,$(u_1, v_1)$,$(u_2, v_2)$)
 2:     $d_1 \leftarrow \gcd(u_1, u_2)$, find $e_1, e_2 \in \mathbb{F}_p[x]$ with $d_1 = e_1 u_1 + e_2 u_2$
 3:     $d \leftarrow \gcd(d_1, v_1 + v_2)$, find $c_1, c_2 \in \mathbb{F}_p[x]$ with $d = c_1 d_1 + c_2(v_1 + v_2)$
 4:     $s_1 \leftarrow c_1 e_1, s_2 \leftarrow c_1 e_2$ and $s_3 \leftarrow c_2$
 5:     $u \leftarrow \frac{u_1 u_2}{d^2}$ and $v \leftarrow \frac{s_1 u_2 v_1 + s_3(v_1 v_2 + f(x)}{d}$ mod $u$
 6:     while $\deg(u) > \mathfrak{g}$ do
 7:         $u \leftarrow \frac{f - v^2}{u}$ and $v \leftarrow -v$ mod $u$
 8:     end while
 9:     divide $u$ by its leading coefficient so that it becomes monic
10:     return (u,v)
11: end function
```

Again there is some intuition to be gained when considering hyperelliptic curves defined over the reals. Let H be a hyperelliptic curve of genus 2, defined by the equation $y^2 = f(x)$ where $\deg(f) = 5$. Recall that some elements in J can only be represented by two points. Then for P_1, P_2, Q_1, Q_2 be 4 distinct points on H. Then from the formal sums $P_1 + P_2, Q_1 + Q_2$ we obtain a new sum $R_1 + R_2$ as



In an analogous way to elliptic curves, scalar multiplication of elements of J may be computed using exponentiation by squaring.

4.4 Counting Points

Just like in the case of elliptic curves, it is essential to determine the number of elements in J over $\mathbb{F}_p$. We may use the generalized Hasse-Weil bound for a curve of genus $\mathfrak{g}$ to estimate the number of points in $J(\mathbb{F}_p)$. That is,

$$\sqrt{p} - 1)^{2\mathfrak{g}} + 1 \le \#J(\mathbb{F}_p) \le \sqrt{p} - 1)^{2\mathfrak{g}} - 1$$

An algorithm very analogous to Schoof's algorithm which works well for hyperelliptic curves of genus $\mathfrak{g} = 2$. Let ϕ be the Frobeneous endomorphism, then for genus 2 curves, we get a similar eqaution of enddmorphism

$$\phi^4 + s_1\phi^3 + s_2\phi^2 - s_1 q\phi + q^2 \equiv \mathcal{O} \tag{3}$$

In the reverse direction, knowledge of $\#J(\mathbb{F}_p)$ almost determines (3) exactly. For example when the genus is 2, $\#J(\mathbb{F}_p) - p^2 - 1 = s_2 - s_1(q+1)$. So the idea of the algorithm is to compute (3) modulo small primes l working in $J[l]$. Then Once this has been done, (3) can be recovered using the Chinese remainder theorem in the exact same way as in Schoof's algorithm. A detailed implementation of this algorithm has been created in python but not included as pseudocode due to length.

The real advantage here is that the number of points in $J(\mathbb{F}_p)$ can be almost g times greater than the number of points on a well suited elliptic curve $E(\mathbb{F}_p)$. Even when $g = 2$ this is a significant advantage. The bottleneck to using hyperelliptic curves is the group operation. It's arithmetic is combersome compared with that of elliptic curves and this is only exacerbated when the size of p is large.

5 Algorithms

Algorithm 6 The Legendre symbol of an integer a modulo a prime p

```
function Legendre($a$,$p$)
    $s \leftarrow 1$
    for $q \in$ FACTORS($a$) do
        $s \leftarrow s*$LegendreForPrime($q$,$p$)
    end for
    function LegendreForPrime($q$,$p$)
        if $q = 1$ then
            return 1
        else if $q \equiv 0 \bmod p$ then
            return 0
        else if $q = -1$ then
            if $p \equiv 1 \bmod 4$ then
                return 1
            else
                return $-1$
            end if
        else if $q = 2$ then
            if $p \equiv \pm 1 \bmod 8$ then
                return 1
            else
                return $-1$
            end if
        else if $q > p$ then
            return LegendreForPrime($q \bmod p$,$p$)
        else if $q \equiv 1 \bmod 4$ or $p \equiv 1 \bmod 4$ then
            return LegendreForPrime($p$,$q$)
        else
            return $-$LegendreForPrime($p$,$q$)
        end if
    end function
    return $s$
end function
```

Algorithm 7 Schoofs algorithm to calculate the number of points on elliptic curve $\mathbb{E} : y^2 - x^3 - ax - b$ over $\mathbb{F}_p$

```
 1: function ORDER($\mathbb{E}$,$p$)
 2:     $S \leftarrow$ set of odd primes such that $\prod_{l \in S} l > 4\sqrt{p}$
 3:     $C \leftarrow \emptyset$
 4:     if $\gcd(x^p - x, x^3 + ax + b) = 1$ then
 5:         $C \leftarrow C \cup \{(2, 1)\}$
 6:     else
 7:         $C \leftarrow C \cup \{(2, 0)\}$
 8:     end if
 9:     for $l \in S$ do
10:         $\psi_l \leftarrow$ DIVISIONPOLYNOMIAL($l$)
11:         $p_l = p \bmod l$
12:         $(x', y') \leftarrow (x^{p^2}, y^{p^2}) \oplus [p_l](x, y)$
13:         for $t \in \{1, 2, ..., \frac{l-1}{2}\}$ do
14:             $(x_t, y_t) = [t](x, y)$
15:             if $x' = x_t \bmod \psi_l$ then
16:                 if $\frac{(y' - y_t)}{y} = 0 \bmod \psi_l$ then
17:                     $C \leftarrow C \cup \{(l, t)\}$
18:                 else
19:                     $C \leftarrow C \cup \{(l, -t)\}$
20:                 end if
21:                 Next $l$
22:             end if
23:         end for
24:         if $p = w^2 \bmod l$ for some $w \in Z$ then
25:             if gcd(numerator($x^p - x_w$), $\psi_l$) = 1 then
26:                 $C \leftarrow C \cup \{(l, 0)\}$
27:             else
28:                 if gcd(numerator($\frac{y^p - y_w}{y}$), $\psi_l$) = 1 then
29:                     $C \leftarrow C \cup \{(l, -2w)\}$
30:                 else
31:                     $C \leftarrow C \cup \{(l, 2w)\}$
32:                 end if
33:             end if
34:         else
35:             $C \leftarrow C \cup \{(l, 0)\}$
36:         end if
37:     end for
38:     return CHINESEREMAINDERTHEOREM(C)
39: end function
```

6 References

1 An Commeine and Igor Semaev, *An Algorithm to Solve the Discrete Logarithm Problem with the Number Field Sieve*, Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, vol, 3958, pg 174-190, 2006.

2 Henri Cohen and Gerhard Frey *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, Chapman and Hall/CRC, Boca Raton FL, 2006, ISBN 1-58488-518-1.

3 Stephan Pohlig and Martin E. Hellman, *An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance*, IEEE Transactions on Information Theory, vol. m24, NO. 1, January 1978, pg 106.

4 Robin Hartshorne, *Algebraic Geometry, Graduate Texts in Mathematics, vol. 52*, Springer-Verlag, New York, 1977, ISBN 0-387-90244-9.

5 Victor Shoup, *Lower bounds for discrete logarithms and related problems*, Theory and Application of Cryptographic Techniques, 1997, pp. 256 - 266.

6 Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **IT-22** (1976), no. 644-654.