Encrypting with Elliptic Curve Cryptography

Reneé Brady
Florida A&M University

Naleceia Davis
Spelman College

Anna Tracy
University of the South

July 2010

Abstract

Today many people communicate via text messaging and "microblog" sites such as Twitter making security an issue of vital importance. We discuss how to use elliptic curves for encoding and encrypting these messages to communicate securely. In the process, we will use Unicode to encode text as a number, as well as the Koblitz method to encode text as a point on an elliptic curve over a finite field. We focus on the Diffie-Hellman and Massey-Omura methods of encrypting messages so that they may be transmitted securely via a key exchange. This research was completed during the 2010 Mathematical Sciences Research Institute's Undergraduate Program (MSRI-UP).

1 Introduction

Dating back to the early 1990s, text messaging has become one of the most popular mediums for communication between two parties. In addition to text messaging, today, many people communicate through sites such as Twitter. With these forms of communication comes the question of security. Through advancements in number theory, we are able to send secret messages even under the assumption that all of our communication is intercepted and read by an adversary.

Suppose that Alice wants to send Bob a message. Knowing that there is an eavesdropper, Eve, waiting to intercept and interpret the message, Alice must encode and encrypt the message. Alice can use the Koblitz, Diffie-Hellman, and Massey-Omura methods, to name a few, to ensure that it will be computationally infeasible for Eve to decipher the message. In this paper, we will discuss various methods of encoding and encryption using finite abelian groups and elliptic curves.

2 Background

2.1 Unicode and ASCII

The American Standard Code for Information Interchange (ASCII) is a character-encoding scheme based on the ordering of the English alphabet. ASCII takes all of the alphanumeric characters and returns them as integers from 0 to 2^8. While ASCII is based on the English language, Unicode, created in 1991, is a universal system. Unicode encodes every character and symbol from all of the world's languages as numbers. Unlike ASCII, Unicode writes characters as integers from 0 to 2^{16}. Each character in the Unicode system is assigned a value, which is later used to encode text. For example, the character "space" is represented by the value 32, capital A with 65, and lowercase b with 98.

2.2 Text Messaging and Twitter

Communication through text messaging and sites such as Twitter have limits on the number of characters per message. This limit derived from the 1985 epiphany of Friedhelm Hillebrand of Germany. After tapping out random sentences and questions on his typewriter, Hillebrand counted the number of letters, numbers, punctuation marks and spaces on the page. He noticed that each sentence and question was approximately 160 characters long. After analyzing postcards and other standard means of communication at that time, he concluded that this was the appropriate length for all messages.

This concept gave rise to the text limits that would be implemented for one of the most commonly used forms of communication today, text messaging. Conventionally, text messages are limited to 160 characters. The "microblog" site Twitter uses a text limit of 140 characters; the additional 20 characters are used to enable the user to send the message with a personal identification attachment. Hillebrand's idea may be over 25 years old, but it was the stepping stone to technical limitations that we use today in our most common forms of communication.

2.3 What is an Elliptic Curve?

Consider the cubic equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,\ a_i \in \mathbb{Q}. \tag{1}$$

Using the change of variables

$$X = x + \frac{{a_1}^2 + 4a_2}{12} \text{ and } Y = y + \frac{a_1x}{2} + \frac{a_3}{2},$$

we may always write (**??**) as a curve in the form

$$Y^2 = X^3 + AX + B, \tag{2}$$

where

$$A = \frac{24(a_1a_3 + 2a_4) - ({a_1}^2 + 4a_2)^2}{48}$$

and

$$B = \frac{216({a_3}^2 + 4a_6) - 36({a_1}^2 + 4a_2)(a_1a_3 + 2a_4) + ({a_1}^2 + 4a_2)^3}{864}.[?]$$

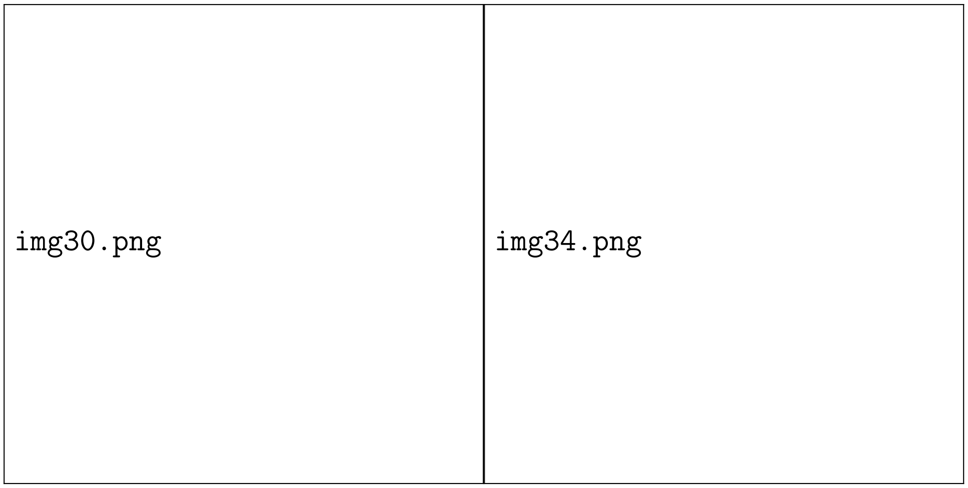

Figure 1: Graphical interpretation of nonsingular curves $y^2 = x(x-1)(x+1)$ and $y^2 = x^3 - x + 1$. []

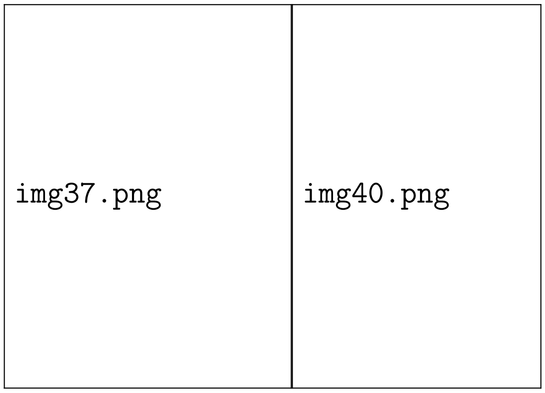

Figure 2: Graphical interpretation of singular curves $y^2 = x^2(x+1)$ and $y^2 = x^3$. []

The discriminant of a polynomial is an expression which provides information about its roots, whether we have repeated roots or complex ones. For the cubic equation (**??**), the discriminant is $-16(4A^3 + 27B^2)$. If the discriminant equals zero then the curve is called singular. Conversely, if the discriminant does not equal zero, then the curve is a nonsingular and has three distinct roots.

Definition 1. An *elliptic curve* E is a nonsingular cubic function of the form:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,\ a_i \in \mathbb{Q}.$$

Consider the elliptic curve E over the rationals. The set of rational points that satisfy E can be written in the form:

$$E(\mathbb{Q}) \simeq \{(X : Y) \,|\, Y^2 = X^3 + AX + B\} \cup \{\mathcal{O}\}, \tag{3}$$

where $\mathcal{O}$ represents the *"point at infinity"*, a formal point that comes with an elliptic curve. The idea is that we can draw a line, even tangent lines, to generate several

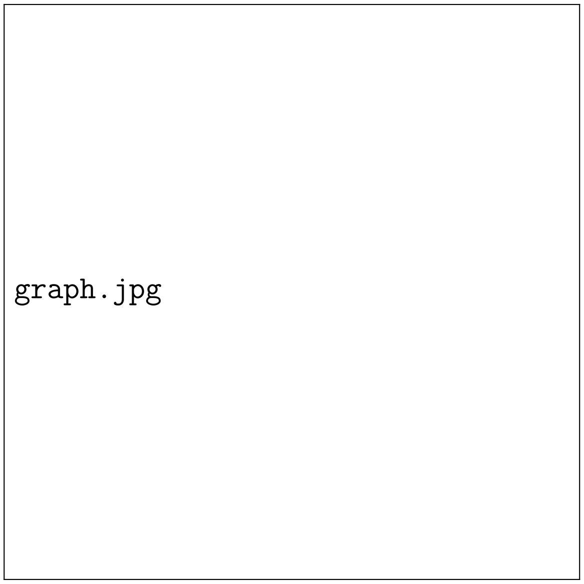

Figure 3: Solving for a third point on $y^2 = x^3 + 1$ graphically.

points from a few known ones. If P and Q are rational points on E, draw a line though them. If $P = Q$, then draw a line tangent to the curve at P. This line will intersect the curve at a third rational point $P * Q$. To facilitate the process, consider the following proposition, whose proof can be found in [**?**]

Proposition 1. *Suppose $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are two distinct points on an elliptic curve $E : y^2 = x^3 + Ax + B$, and that $x_1 \neq x_2$. Let L be the line through P and Q. Then L intersects the graph of E at exactly one other point $P * Q = (\lambda^2 - x_1 - x_2, \lambda x_3 + \nu)$, where $\lambda = \frac{(y_1 - y_2)}{(x_1 - x_2)}$, $\nu = y_1 - \lambda x_1$. and x_3 is the x-coordinate of the point $P * Q$.*

Example 1. Consider the curve $E : y^2 = x^3 + 1$.
Compute two points on the curve (See Figure 3), as $P = (0, 1)$ and $Q = (2, 3)$, by solving for y when $x = 0, 2$. The line through these two points is $y = x + 1$. By Proposition 1, $\lambda = 1$ and $\nu = 1$. So, the $x - coordinate$ of the $P * Q$ is $x_3 = \lambda^2 - x_1 - x_2 = 1 - 0 - 2 = -1$ and $y = \lambda x_3 + \nu = 1(-1) + 1 = 0$. Thus the point $P * Q$ is $(-1, 0)$ and hence have found three points on E.

2.4 The Group Law for Elliptic Curves

Definition 2. Given a nonempty set G and an operation $\circ$, the pair $(G, \circ)$ is a group if the following are satisfied:

1. Closure: if $a, b \in G$ then $a \circ b \in G$

2. Associativity: if $a, b, c \in G$ then $a \circ (b \circ c) = (a \circ b) \circ c$

3. There exist an identity element $e \in G$ such that $a \circ e = a = e \circ a$ for all $a \in G$

4. For all $a \in G$, there exist an inverse element $[-1]a \in G$ such that $a \circ [-1]a = e = [-1]a \circ a$. (Note: The operation $[m]a$ for an element $a \in G$ is defined as the following: $[m]a = \underbrace{a \circ a \circ ... \circ a}_{\text{m-times}}$.)

5. A group G is abelian if for every $a, b \in G$, $a \circ b = b \circ a$.

Definition 3. Suppose we have a set $S = \{x_1, x_2, ..., x_n\}$, then a group G is said to be a finitely generated abelian group if

1. G is abelian

2. Every $x \in G$ can be written as $x = [m_1]x_1 \circ [m_2]x_2 \circ ... \circ [m_n]x_n$ with $m_n \in \mathbb{Z}$,

If $a, b \in \mathbb{Z}$ and $n \in \mathbb{N}$, we say that a is *congruent to b modulo n* if $n|a - b$, and write $a \equiv b \bmod n$. We also let $n\mathbb{Z} = (n)$ be the subset of all $\mathbb{Z}$ consisting of all multiples of n. $\mathbb{Z}/n\mathbb{Z}$ is the set of equivalence classes of integers modulo n; every element $a \in \mathbb{Z}/n\mathbb{Z}$ is relatively prime to n (ie. $\gcd(a, n) = 1$). For example,

$$\mathbb{Z}/3\mathbb{Z} = \{\{\ldots, -3, 0, 3, \ldots\}, \{\ldots, -2, 1, 4, \ldots\}, \{\ldots, -1, 2, 5, \ldots\}\}$$

Definition 4. $(\mathbb{Z}/n\mathbb{Z})^*$ denotes the set under multiplication of nonzero elements $x \in \mathbb{Z}/n\mathbb{Z}$ such that $\gcd(x, n) = 1$

Theorem 1 (Lagrange's Theorem)**.** *For any finite group G, the order (number of elements) of every subgroup H of G divides the order of G. In other words,*

$$|G| = |G/H| \times |H|$$

Proposition 2. *The set $(\mathbb{Z}/n\mathbb{Z})^*$ is a finite abelian group*

Proof. To show that $(\mathbb{Z}/n\mathbb{Z})^*$ is a finite abelian group, we first let $G = \mathbb{Z}/n\mathbb{Z}$ and show that G is a group.

1. Closure: Suppose $a, b \in G$, then $\gcd(a, n) = 1$ and $\gcd(b, n) = 1$. Thus, by the Extended Euclidean Algorithm, there exists integers α, β, k_1, k_2 such that $\alpha a + k_1 n = 1$ and $\beta b + k_2 n = 1$. We want to show that if $ab \in G$, then $\gcd(ab, n) = 1$ and $\alpha a \beta b + kn = 1$.
   Multiplying the equations $\alpha a + k_1 n = 1$ and $\beta b + k_2 n = 1$ yields $ab\alpha\beta + \alpha a k_2 n + \beta b k_1 n + k_1 k_2 n^2$..........

2. Associativity: Since we are working with integers, the order in which the elements are multiplied does not matter. Thus, Associativity holds.

3. Identity: The identity element is 1 since any element multiplied by 1 is the original element.

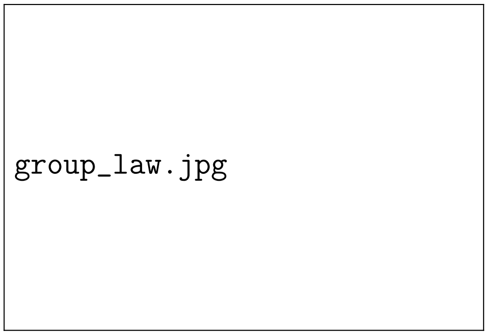

Figure 4: The graphical interpretation of The Group Law.

4. Inverses: Suppose $a \in G$. Then by the Extended Euclidean Algorithm, there exists $d, e \in Z$ such that $ad + en = 1$. Computing the module n on both sides, we see that $ad \equiv 1 \bmod n$. Hence G has inverses.

Hence, G is a group. Since we are working with integers and the order in which they are multiplied does not matter, G is also abelian. Hence,$(\mathbb{Z}/n\mathbb{Z})^*$ is an abelian group under multiplication. □

Theorem 2. *Denote K as either $\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$. Consider the elliptic curve*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,\ a_i \in K.$$

*Let * denote the composition law which takes two rational points P and Q and computes the point of intersection $P * Q$ of the curve E and the line through P and Q. Define the composition law $\oplus$ by $P \oplus Q = (P * Q) * \mathcal{O}$. This turns $(E(K), \oplus)$ into an abelian group.*

1. By definition, $E(K)$ is closed under $\oplus$, thus closure holds.

2. Commutativity is also satisfied since we are under addition.

3. The point $\mathcal{O}$ serves as the identity element since $P \oplus \mathcal{O} = (P * \mathcal{O}) * \mathcal{O}$

4. The additive inverse of a point P on E is $[-1]P = P \oplus \mathcal{O}$, the reflection of P over the $x - axis$

5. To show Associativity, we can look at the graphic representation of the Elliptic Curve, [].

Proposition 3. *Consider the elliptic curve E. The set of points on the curve over a finite field $\mathbb{F}_p$, $E(\mathbb{F}_p)$, is a finite abelian group.*

Proof. Since we are working with a finite field, then we have a finite number of points satisfying E. Let's begin by showing that $E(\mathbb{F}_p)$ is a group.

1. Consider points $P, Q \in E(\mathbb{F}_p)$, then $P \oplus Q \in E(\mathbb{F}_p)$ since we can always compute a third point by drawing a line intersecting both P and Q. Hence closure holds.

2. In order to show Associativity, consider the points $P, Q, R \in E(\mathbb{F}_p)$, and show that $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$. The graphical and algebraic proof can be found in [].

3. The identity element is $\mathcal{O}$ since $P \oplus \mathcal{O} = (P * \mathcal{O}) * \mathcal{O} = P$ for all $P \in E(\mathbb{F}_p)$.

4. Define $[-1]P = P*\mathcal{O}$, then $P \oplus [-1]P = (P*[-1]P)*\mathcal{O} = ((P)*(P*\mathcal{O}))*\mathcal{O} = \mathcal{O}$. Hence $[-1]P = P * \mathcal{O}$ is the additive inverse.

Hence, $E(\mathbb{F}_p)$ is a group. To show $E(\mathbb{F}_p)$ is abelian, show that for points $P, Q \in E(\mathbb{F}_p)$, $P \oplus Q = Q \oplus P$. Since E over the opration $*$ is commutative, $P \oplus Q = (P * Q) * \mathcal{O}$ and $Q \oplus P = (Q * P) * \mathcal{O} = (P * Q) * \mathcal{O}$. In other words, the order of finding point on E does not matter. Hence, $E(\mathbb{F}_p)$ is an abelian group. □

3 Encoding Messages

Suppose Alice wants to send Bob a message. Before she can select an abelian group and encrypt the message, she has to encode the message in such a way she can map it to the abelian group of her choice. She can do this either by encoding the message as an integer or as a point on an elliptic curve E. We will begin by considering how her message can be encoded using Unicode and proceed to explaining how to encode her message as a point on an elliptic curve using the Koblitz method.

3.1 Using Unicode to Write a String as a Number

A message, M, is a sequence of characters $\{m_1, m_2, \ldots, m_n\}$. Using either ASCII or Unicode, each character, m_k can be expressed as a number, a_k, where $0 \leq a_k < b$. Thus, M can be represented by the sequence $\{a_1, a_2, \ldots, a_n\}$. Recall from **??** that Unicode uses a 16-bit encoding system, base $b = 2^{16}$, while ASCII uses an 8-bit encoding system, base $b = 2^8$.

Using the following summation:

$$m = \sum_{k=1}^{n} a_k b^{k-1}$$

the sequence of numbers $\{a_1, a_2, \ldots, a_n\}$ can be combined to form one large number m. For our purposes, we will use the standard of Friedhelm Hillebrand discussed earlier so that n, the number of characters, will be less than or equal to 160; using this restriction, the summation m will be between 0 and b^{160}. This method can be used to map a message, M, to the abelian group $G = (\mathbb{Z}/n\mathbb{Z})^*$, for some n greater than the summation, m.

Given this number m, we can recover each a_k by using the following equation:

$$a_k = \left\lfloor \frac{m}{b^{k-1}} \right\rfloor \bmod b$$

where $\lfloor x \rfloor$ signifies the greatest integer less than x; for instance, $\lfloor 4.5 \rfloor = 4$.

Example 2. Consider the following message:

Hello World!

Unicode represents the string

$$\{\texttt{H, e, l, l, o, ,W, o, r, l, d, !}\}$$

as the list

$$\{72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 33\}.$$

Using the summation above where $n = 12$ and $b = 2^{16}$, we retrieve the number

$$m = 31609182056081481348633992424376689992778011045457429 20.$$

While the numeric representation of our message, m, can be encrypted in its current form, encoding it further allows more efficient encryption methods to be used. In particular, we use Koblitz's method to associate m with a point on a given elliptic curve. To accomplish this encoding, it is necessary to choose a prime, p, that is larger than m. Since the plain text message consists of as many as 160 characters, the encoding prime will need to be at least 2560 bits, that is, $2^{16 \cdot 160}$; this way m is guaranteed to be congruent to m modulo the encoding prime. To find a prime that is at least 2^{2560} bits, we use the following SAGE code:

```
sage: next_prime(2^2560)
4333002102749267793012357229951305291268519243122535662768313665470976 5
```

While in theory the above prime is ideal, in practice using primes this large is problematic due to the memory constraints of computers. If a smaller prime is used, the message must be broken down into short segments so that each segment is still smaller than the prime. These segments generally range from 192 to 521 bits in length in accordance with the recommendations of the National Institute of Standards and Technology. Once a length has been chosen and the message divided appropriately, the individual segments may then be mapped to separate points on the elliptic curve and subsequently be encrypted. For simplicity, in our paper we assume a prime larger than 2560 bits; however in practice, smaller primes are normally used.

3.2 Koblitz Encoding

Neal Koblitz, co-creator of elliptic curve cryptography, introduced a method to encode a message on an elliptic curve. In order to use Koblitz's method, we must first allow E to be an elliptic curve of the form $y^2 = x^3 + Ax + B$ over a finite field $\mathbb{F}_p$. We choose p such that the following conditions are satisfied:

1. p is a prime such that p does not divide $-16(4A^3 + 27B^2)$; this ensures that we have a nonsingular cubic curve over a finite field.

2. $p \equiv 3 \bmod 4$; this is done to simplify the formulas below.

3. p has more than 2560 bits; this is done so that we can encode a 160 character message as one point on the elliptic curve. (Note: If p has less than 2560 bits then the message will be divided into a series of smaller messages or chunks and have a series of points on the elliptic curve.)

To ensure message security, it is essential to choose a prime large enough such that the condition $p > m$ is satisfied and computing $|E(\mathbb{F}_p)|$ is difficult for a potential hacker. Due to the difficulty of finding large primes, we use the list provided by The National Institute for Standards and Technology (NIST) to find such primes and elliptic curves. NIST has formulated a list of 15 elliptic curves with prime fields of various sizes.

Lemma 1. *Given a prime $p \equiv 3$ mod 4 and an integer s, the congruence $y^2 \equiv s$ mod p holds if and only if both $s^{\frac{p+1}{2}} \equiv s$ mod p and $y \equiv \pm s^{\frac{p+1}{4}}$ mod p.*

Proof. Assume $y^2 \equiv s \bmod p$.
Recall Fermat's Little Theorem, which states $y^p \equiv y \bmod p$.
Raising both sides of $y^2 \equiv s \bmod p$ to the $\frac{p+1}{2}$ power yields the following

$$s^{\frac{p+1}{2}} \equiv (y^2)^{\frac{p+1}{2}} \equiv y^p \cdot y \equiv y \cdot y \equiv y^2 \equiv s \bmod p.$$

From the identity

$$(y^{\frac{p+1}{2}} - y)(y^{\frac{p+1}{2}} + y) = \ y^{p+1} - y^2 \equiv 0 \bmod p$$

we have the following

$$y \equiv \pm y^{\frac{p+1}{2}} \equiv \pm (y^2)^{\frac{p+1}{4}} \equiv \pm s^{\frac{p+1}{4}} \bmod p.$$

Conversely, assume $s^{\frac{p+1}{2}} \equiv s$ and $y \equiv \pm s^{\frac{p+1}{4}} \bmod p$. Then

$$y \ \equiv \pm s^{\frac{p+1}{4}} \qquad \Longrightarrow \qquad y^2 \ \equiv s^{\frac{p+1}{2}} \ \equiv s \bmod p.$$

□

We use the following algorithm to encode a message on an elliptic curve E:

1. Convert each character m_k into a number a_k using Unicode, where $b = 2^{16}$ and $0 \leq a_k < 2^{16}$.

2. Convert the message M into an integer using

$$m = \sum_{k=1}^{n} a_k b^{k-1}.$$

   In practice, we choose an n to be less than or equal 160 so that m satisfies $m \leq 2^{16 \cdot 160} < p$.

3. Fix a number d such that $d \leq \frac{p}{m}$. In practice, we choose the prime p large enough so that we can allow $d = 100$.

4. For integers $j = 0, 1, 2, \ldots, d-1$ we do the following loop:

   (a) Compute the x coordinate of a point on the elliptic curve as $x_j = (dm + j) \bmod p$, where $m = \lfloor \frac{x_j}{d} \rfloor$.

   (b) Compute $s_j = (x_j^3 + Ax_j + B) \bmod p$.

   (c) If $(s_j)^{\frac{p+1}{2}} \equiv s_j \bmod p$, then define the y coordinate of a point on the elliptic curve as $y_j = (s_j)^{\frac{p+1}{4}} \bmod p$. Return the point (x_j, y_j).

Thus, we are able to encode our message, M, as an element of the abelian group $G = E(\mathbb{F}_p)$.

SAGE can quickly implement this process using the following code:

```
def koblitz_encode(E, string):
    A = E.a4()
    B = E.a6()
    p = E.base_field().cardinality()
    b = 2^16
    m = str(string)
    n = len(m)
    m = sum(ord(m[k])*b^k for k in range(n))
    d = min(floor(p/m),100)
    for  j in range (d):
        x = (d*m + j) % p
        s = (x^3 + A*x + B) % p
        if s == power_mod(s,int((p+1)/2),p):
            y = power_mod(s, int((p+1)/4), p)
            return (E([x,y]))
```

In addition to encoding, SAGE can also be used to decode the point back to a string of characters using the following code:

```
def koblitz_decode(P):
    b = 2^16
    d = 100
    lst = []
    m = floor((P[0]/P[2]))/d) # (P[0]/P[2]) represents the x-coordinate of the point.
    while m != 0:
        lst.append(chr(m%b)) #converts m to a list of characters
        Nmb //= b  #replaces Nmb by floor(Nmb/b)
    return ''.join(lst)
```

The following examples show two different ways of encoding a message. Using the NIST list, we were able to find a prime p that satisfies $p > m$ for the first example. We will use the above code to encode the message.

```
sage: M = ‘I wish today was sunny. Yikes!!!’
sage: string = M
sage: E = p_521[3] #Taken from the NIST list, this particular curve can be
used for 32 Unicode characters or less
sage: koblitz_encode(E, string)
(6751471148059962009188727476457301655362732179755399412474316677896517601579
69685735858445191110663342458419095898296341121356574903498764893956761197702 :
26467411621663997524656470923974080889728039271831941163802483335694905768073
79087352229981883237545572112064856706602726961612204976269523532971366045559
332 : 1)
```

While the first strategy is ideal in theory, in practice, working with blocks of text with as many as 160 characters is computationally inefficient. Thus, it is more practical to separate the message into segments or “chunks” of 12 to 32 characters and encrypt them separately. The following shows an example of this:

```
sage: J = ’I wish today was’
sage: K = ’ sunny. Yikes!!!’
sage: E = p_256[3] #This curve can be used for 16 Unicode characters or less
sage: Koblitz(E, J)
(2031900276132948729732903467094810640313948757718598477079178721584322970330
0 : 1454047482809596794895120374517886573685948941474751628904948932041215698
636 : 1)
sage: Koblitz(E, K)
(5830684282950283332163559533681324589814285111106959033572124775115830070400 :
7554461311930554755126384355409742886178752477906031306690329014997979898243 3 :
1)
```

4 Encryption

Now that Alice and Bob are familiar with encoding messages as elements of finite abelian groups, they can choose to encrypt their messages in several ways. In this section, we will discuss two specific methods for which Alice and Bob can send encrypted messages. The first method, Diffie-Hellman, is the most efficient method, however the Massey-Omura method is the most secure of the two. We will begin by discussing the Diffie-Hellman method of encryption.

4.1 Diffie-Hellman

Say that Alice and Bob want to exchange messages with one another. Assuming that they have had no prior contact and their communication channels are open, they first need to verify that they are communicating with one another. They can do this using the following steps of the Diffie-Hellman Key Exchange.

1. Alice and Bob agree on a finite abelian group G and an element $g \in G$ such that $\gcd(g, |G|) = 1$.
2. They each choose private keys a and $b \in G$, respectively. The private keys must be relatively prime to the the order of G, $|G|$.
3. They each choose public keys α and $\beta \in G$ such that $a \cdot \alpha \equiv 1 \bmod |G|$ and $b \cdot \beta \equiv 1 \bmod |G|$.
4. Alice sends $[a]g$ to Bob and Bob sends $[b]g$ to Alice.
5. Alice and Bob each publish $[ab]g$ and verify that they are the same.

Now that they have verified that they are communicating with each other, Alice and Bob can do the following to exchange messages:

1. If Alice wants to send the message $M \in G$ to Bob, she will send $[\beta]M$.
2. Bob will then decode to find the original message by using his private key b to compute $[b \cdot \beta]M = M$.
3. If Bob wants to send a message, he can use a similar approach, but instead, use α.

For the Diffie-Hellman Key Exchange, we generally use the abelian group $G = (\mathbb{Z}/n\mathbb{Z})^*$. To find the order of G, $|G|$, we use the Euler-φ function.

Definition 5 (Euler's φ-function)**.** For n $\in \mathbb{N}$, let

$$\varphi(n) = \#\{a \in \mathbb{N} : a \leq n \text{ and } \mathrm{gcd(a, n)} = 1\}. \tag{4}$$

If p is any prime number, then

$$\varphi(p) = \#\{1, 2, \ldots, p-1\} = p - 1$$

Additionally, if $gcd(p, q) = 1$, then

$$\varphi(pq) = \varphi(p)\varphi(q).$$

The proof of this can be found in **??**]

To use the Diffie-Hellman Key Exchange, Alice and Bob will need to compute their respective public keys from their private keys. If the abelian group $G = (\mathbb{Z}/n\mathbb{Z})^*$. then the Extended Euclidean Algorithm (EAA) will be used for this computation. The steps for the EEA are as follows:

Start with integers a and b. Dividing a by b yields

$$a = b \cdot q_0 + r_0 \qquad \Longrightarrow \qquad \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} q_0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} b \\ r_0 \end{bmatrix}$$

for some quotient q_0 and remainder r_0. Dividing b by r_0 yields

$$b = r_0 \cdot q_1 + r_1 \qquad \Longrightarrow \qquad \begin{bmatrix} b \\ r_0 \end{bmatrix} = \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$

for some quotient q_1 and remainder r_1. Dividing r_0 by r_1 yields

$$r_0 = r_1 \cdot q_2 + r_2 \qquad \Longrightarrow \qquad \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \begin{bmatrix} q_2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

for some quotient q_2 and remainder r_2. Continuing this process results in the following

$$r_n = r_{n+1} \cdot q_{n+2} + r_{n+2} \qquad \Longrightarrow \qquad \begin{bmatrix} r_n \\ r_{n+1} \end{bmatrix} = \begin{bmatrix} q_{n+2} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{n+1} \\ r_{n+2} \end{bmatrix}$$

The basic idea is that we will have a decreasing sequence of positive integers $b > r_0 > r_1 > r_2 >, \ldots, > r_n > r_{n+1}$. At some point r_{n+2} will equal zero and $g = \text{gcd}(a, b) = r_{n+1}$. Multiplying the matrices produces

$$\begin{bmatrix} q_0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_n & 1 \\ 1 & 0 \end{bmatrix} = A$$

Observe:

1. A has $\mathbb{Z}$ coefficients.

2. The determinant of A = ± 1

Hence,

$$A^{-1} = \begin{bmatrix} x & y \\ s & t \end{bmatrix} \text{for some } x, y, s, \ t \in \mathbb{Z}$$

To compute A^{-1}, use

$$\begin{bmatrix} r_n \\ r_{n+1} \end{bmatrix} = \begin{bmatrix} q_{n+2} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{n+1} \\ r_{n+2} \end{bmatrix}.$$

and invert the matrix

$$\begin{bmatrix} r_{n+1} \\ r_{n+2} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & q_{n+2} \end{bmatrix} \begin{bmatrix} r_n \\ r_{n+1} \end{bmatrix}.$$

Compute the 2×2 matrix

$$\begin{bmatrix} x & y \\ s & t \end{bmatrix} = A^{-1} = \begin{bmatrix} 0 & -1 \\ -1 & q_{n+2} \end{bmatrix} \cdots \begin{bmatrix} 0 & -1 \\ -1 & q_1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ -1 & q_0 \end{bmatrix}$$

Using this,

$$A^{-1} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} x & y \\ s & t \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix} \qquad \Longrightarrow \qquad ax + by = g$$

The following is an example of the Diffie-Hellman Key Exchange.

Example 3. Alice and Bob begin by agreeing on the abelian group $G = (\mathbb{Z}/n\mathbb{Z})^*$ where n is the product of two large primes p and q. For the sake of simplicity, we will choose p and q to be the small primes 991 and 1801, respectively; thus, $n = 1784791$ and $|G| = \varphi(n) = \varphi(p)\varphi(q) = 1782000$.

Alice and Bob agree on $g = 179$ and they each choose private keys a and b to be 61 and 23, respectively. Using the Extended Euclidean Algorithm, Alice and Bob compute their public keys α and β to be 759541 and -309913, respectively; these will be used to send a message. (Since β is negative, we add $|G|$ to obtain the positive integer 1472087.)

Alice computes $[a]g = g^a \bmod |G|$ and sends the resulting number, 1306979, to Bob. Bob sends $[b]g = g^b \bmod |G| = 230939$ to Alice. Using their new messages, Alice computes $[ab]g = (g^b)^a \bmod |G| = 1775339$ and Bob each computes $[ba]g = (g^a)^b \bmod |G| = 1775339$. Since $[ab]g = [ba]g$, Alice and Bob know that they are communicating with one another!

Now, if Alice wants to send a message g to Bob, she will use his public key β to compute $[\beta]g = g^\beta \bmod |G| = 1207259$. Bob can decode to find the original message by calculating $[b\beta]g = (g^\beta)^b \bmod |G| = 179$.

Now that they have established that they are speaking with one another, they can send messages as shown in the proceeding example.

Example 4. Suppose Alice wants to send Bob the message 'hi'. Using the steps provided in Section**** to convert a string to a number, she computes $m = 6881384$. She then looks up Bob's public key β and sends Bob $[\beta]m = m^\beta \bmod |G| = 392144$. Bob can find m by calculating $[b\beta]m = (m^\beta)^b \bmod |G| = 6881384$. Using the equation:

$$a_k = \left\lfloor \frac{m}{b^{k-1}} \right\rfloor \bmod b$$

he can recover the original message 'hi'.

4.2 Massey-Omura

Suppose Alice wants to send a message to Bob over a public channel without publishing her encryption key α. In order to do this, Alice puts her message in box and places a lock on it. She then sends the box to Bob who puts his lock on it and sends it to back to Alice. Alice then takes her lock off and sends the box back to Bob. He then removes his lock, opens the box, and reads the message. This procedure can be implemented mathematically with the following Massey-Omura algorithm.

1. Alice and Bob agree on an abelian group G over a finite field.

2. Alice represents her message as an element $g \in G$.

3. Alice chooses a secret integer a with $\gcd(a, |G|) = 1$, computes $[a]g$, and sends it to Bob.

4. Bob chooses a secret integer b with $\gcd(b, |G|) = 1$, computes $[ba]g = [ab]g$, and sends it to Alice.

5. Alice computes $a^{-1} = \alpha \in G$ (see previous section). She computes $[\alpha ab]g = [b]g$ and sends it to Bob.

6. Bob computes $b^{-1} = \beta \in G$ and computes $[\beta b]g = g$.

In general, we choose our finite abelian group G to be a set of points on an elliptic curve E over a finite field $\mathbb{F}_p$, where E is one of the elliptic curves recommended by NIST. In this case, Alice can use the Koblitz method, discussed earlier, to encode her message as a point P on the elliptic curve.

SAGE can be used to implement the above algorithm.

```
def key(E):
    N = E.cardinality()
    #This randomly chooses the encryption key
    a = ZZ.random_element(N)
    while (gcd(a,N) != 1):
        a = ZZ.random_element(N)
    return a

def massey_omura(E, string):
     a = key(E)
     #This encodes a string as a point
     P = koblitz_encode(E, string)
     Q = a * P
     return ([a, Q])
```

The following code computes the decryption key α.

```
def inverse(a, E):
N = E.cardinality()
alpha = power_mod(a, -1, N)
return alpha
```

After receiving $[a]P$, Bob can compute his encryption key, b, using the `key` function and his decryption key, β, with the `inverse` function. Once the keys have been removed, Bob can retrieve the original message by using the `koblitz_decode` function defined in Section*****.

The following is an example of the Massey-Omura method.

```
sage: string = ’I wish today was’
sage: E = p_256[3] #Taken from the NIST list
sage: [a,aP]=massey_omura(E, string)
sage:a
7323619074119718633977085129784336559548400191795243506367902547572069
2170687
sage:aP
(111708789836929397634360019654638136289630753533185850630274419179133
534893979 : 862807601945502246955436936769157502637665913319417440610
0
8144281386680043733 : 1)
sage:b=key(E)
sage:b
5517455953852574574279454202387248311364097409982401503101084472275068
8866071
sage:alpha=inverse(a,E)
sage:alpha
1071084383619235364315413733251436452072515788292197824824582073078174
24399767
sage:beta=inverse(b,E)
sage:beta
3646273899740434103681750939376203837110085685169223454401076145463793
8356779
sage:baP=b*ap
sage:baP
(1871228119093174283025356075450764194244307151508158816685424906559 72
02515661 : 1082911406773579239480130188125338437525419764700006780050 4
1265940859744153277 : 1)
sage:bP=alpha*baP
sage:bP
(3361200873664857666832880446256452101778637048411469904233010381183 50
12404695 : 9798749532565104709182796368101291829027863727642446466783 1
101073174591177342 : 1)
sage:P=beta*bP
sage:P
(2031900276132948729732903467094810640313948757718598477079178721584 32
29703300 : 1143380417275466519678023265748896869564001944738155625666 2
8682376825882155315 : 1)
sage:koblitz_decode(P)
’I wish today was’
```

5 Conclusion

6 Acknowledgments

This work was conducted during the 2010 Mathematical Sciences Research Institute Undergraduate Program (MSRI-UP), a program supported by the National Science Foundation (grant No. DMS-0754872) and the National Security Agency (grant No. H98230-10-1-0233).

We also thank our mentor Dr. Edray Goins. Additionally, we thank Dr. Duane Cooper, Dr. Luis Lomelí, Ebony Harvey, Katie Ansaldi, and the MSRI staff.

References

[1] Author, Joe, *The Generic Math Book*, 2nd Ed., Springer-Verlag, New York, 1997.