Research Summery

Alec Zabel-Mena

February 23, 2020

In being conditionally accepted to the PR-LSAMP research program, the research proposed for me was to study certain elliptic curves, and to count their rational points. The goal was to determine whether or not the curves were optimal (or near optimal), and maximal (or near maximal). Both terms depend on the Hasse bound of rational points on the curve which states that $|N - (q+1)| \leq 2\sqrt{q}$, where q is the prime characteristic of some finite field (or the extension of some finite field). In my reseerch, I studied when $q = 2, 3$.

The first objective was to understand the zeta function of elliptic curves and how one can find the number of rational points of the curve over any extension of a finite field. After understanding those two things, understanding the Hasse bound was next. Afterwords, I was to prepare an algorithm for finding certain elliptic curves, finding their zeta functions, and the number of rational points over some extension field. I was also to test whetherr or not the curves were maximal (or near maximal) and whether they were optimal (or near optimal). The code is displayed below.

```
    # Here is an algorithm for finding optimal and maximal elliptic curves, suitable
    # for cryptogtraphy.

from __future__ import print_function # Just import some of the print function's
                                      # propertie to make output bearable.

q = 2
r = 2 #r will be replaced by any suitable power when computing the extension fields

for a6 in range(q):
    for a4 in range(q):
        for a3 in range(q):
            for a2 in range(q):
                for a1 in range(q):
                    b2=a1^2+4*a2; b4=a1*a3+2*a4; b6=a3^2+4*a6; b8=a1^2*a6-a1*a3*a4+
                        a2*a3^2+4*a2*a6-a4^2

                    delta=-b2^2*b8-8*b4^3-27*b6^2+9*b2*b4*b6
                    if delta % q == 0:
```

```
    print("")
else:
    E = EllipticCurve(GF(q),[a1,a2,a3,a4,a6])
    N1 = E.cardinality()
    a = q+1-N1

    cmplxpoly.<T> = PolynomialRing(CC)
    f(T) = 1-a*T+q*T^2
    Z(T) = f(T)/((1-T)*(1-q*T))
    raiz = f(T).roots()
    alpha = [p for p,e in raiz][0]
    beta = [p for p,e in raiz][1]

    Nr=q^r+1-alpha^r-beta^r

    print(E,":")
    print("    N_1=", N1, "|", "Z(T)=",Z(T),"|", "N_r=",Nr,  "\n")
```