Irreducible Quadratics over $\mathbb{Z}_n$

by

Robyn Hearn

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Honours Bachelor of Science

in the
Department of Mathematics
Faculty of Science

Approval

Name:	**Robyn Hearn**
Degree:	**Honours Bachelor of Science (Mathematics)**
Title:	**Irreducible Quadratics over $\mathbb{Z}_n$**
Supervisory Committee:	**Michael Monagan** Supervisor Professor
	Jonathan Jedwab Professor
Date Approved:	**December 14, 2018**

Abstract

It is commonly known that there are $\binom{n}{2}$ irreducible, quadratic polynomials over $\mathbb{Z}_n$ when n is a prime. Then it is natural to ask how many irreducible quadratics there are over $\mathbb{Z}_n$ without the condition that n is prime. The counting methods that solve the case where n is prime, rely on field axioms which $\mathbb{Z}_n$ does not generally satisfy. In this thesis, we relate the problem of counting reducible, quadratic polynomials to the simpler problem of counting squares. In the construction of this count, we find an algorithm to generate all quadratic polynomials over $\mathbb{Z}_n$ and categorize them as reducible or irreducible. This algorithm is computationally less expensive than the naive cubic algorithm for generating all irreducible quadratics.

Keywords: Algebra; Combinatorics; Ring(s); Field(s); Irreducible; Polynomial(s)

Acknowledgements

Many thanks to my research supervisor Michael Monagan for his support and contributions. Thank you as well to Jonathan Jedwab for his help in clearly communicating my work. Also to Simon Fraser University, KEY Big Data, and the Natural Sciences and Engineering Research Council for this opportunity and for financial support.

Table of Contents

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Introductory courses in field and ring theory often present the following problem.

(1) *Show that the number of irreducible quadratic monic polynomials in $\mathbb{Z}_p[x]$ is $\frac{1}{2}p(p-1)$, where p is a prime.* [2]

One method of solving this problem is to count all quadratic monic polynomials and subtract the number of those which are reducible. The total number of quadratic monic polynomials is p^2 since there are p choices for both of the coefficients b, c in $x^2 + bx + c$. In the potential factorization $(x + \alpha)(x + \beta)$, there are p choices for which $\alpha = \beta$ and $\binom{p}{2}$ choices for which $\alpha \neq \beta$. Therefore, the number of quadratic monic polynomials which are reducible is $\frac{1}{2}p(p+1)$, so the number which are irreducible is $p^2 - \frac{1}{2}p(p+1) = \frac{1}{2}p(p-1)$. Then a natural question is

Do we require that p is prime for the method to carry over?

By the above method, we see that $\mathrm{GF(p^k)[x]}$ has $\frac{1}{2}p^k(p^k - 1)$ irreducible quadratic monic polynomials for each natural number k. However, if $\mathrm{GF(p^k)} \neq \mathbb{Z}_{\mathrm{p^k}}$, so this thesis will focus on the following question:

(2) *For a positive integer n, how many irreducible quadratic monic polynomials are there in $\mathbb{Z}_n[x]$?*

Consider the following example, which illustrates the difference between problems (1) and (2).

Example 1. *We classify every quadratic monic polynomial over $\mathbb{Z}_4$ as reducible or irreducible.*

Reducible			*Irreducible*
x^2	$=$	$(x+0)^2 = (x+2)^2$	x^2+1
x^2+x	$=$	$(x+0)(x+1)$	x^2+2
x^2+2x	$=$	$(x+0)(x+2)$	x^2+x+1
x^2+3x	$=$	$(x+0)(x+3)$	x^2+x+3
x^2+2x+1	$=$	$(x+1)^2 = (x+3)^2$	x^2+2x+2
x^2+3x+2	$=$	$(x+1)(x+2)$	x^2+2x+3
x^2+3	$=$	$(x+1)(x+3)$	x^2+3x+1
x^2+x+2	$=$	$(x+2)(x+3)$	x^2+3x+3

The number of irreducible quadratics is not $\frac{1}{2}4(4-1)=6$*, but* 8*.*

Our count of irreducible quadratic monic polynomials in $\mathbb{Z}_p[x]$ requires that for all fixed b, c in $\mathbb{Z}_p$, the set of congruence

$$\alpha + \beta \equiv b \pmod{p}$$

$$\alpha\beta \equiv c \pmod{p}$$

has a unique solution when α, β in $\mathbb{Z}_p$. This is not necessarily true in $\mathbb{Z}_n$, so $\frac{1}{2}n(n+1)$ overcounts the number of reducible quadratic monic polynomials (as observed in Example 1.)

1.2 Computational Experiments

Our investigation into this question began with computational experiments using the following algorithm on various values of n. Algorithm 1 is a straight forward way to count reducible polynomials but has cubic time complexity. Since the algorithm requires so much time, our experiments began in Maple, but were promptly transferred to C using primarily bit-wise operations, and finally the work was parallelized in Cilk C to get enough raw data to work with. More details are included in Appendix A.

Algorithm 1 Generate All Reducible Quadratic Monic Polynomials in $\mathbb{Z}_n[x]$

```
Input: $n \in \mathbb{Z}^+$
1: $P \leftarrow []$;    $i \leftarrow 0$
2: \\Generate all quadratic monic polynomials
3: for $b \in \mathbb{Z}_n$ do
4:     for $c \in \mathbb{Z}_n$ do
5:         \\Look for a root, $x_0 \in \mathbb{Z}_n$
6:         for $x_0 \in \mathbb{Z}_n$ do
7:             if $x_0^2 + bx_0 + c \equiv 0 \pmod{n}$ then
8:                 \\If a root is found, increase count by 1
9:                 $P[i] \leftarrow \{x^2 + bx + c\}$;    $i{+}{+}$
10:                break
11:            end if
12:        end for
13:    end for
14: end for
15: return $P$
```

We sought to find a pattern in the number of polynomials output by this algorithm. In the case of $n = 2^k$, define C_k as the number of reducible quadratic polynomials over $\mathbb{Z}_n$ as determined by Algorithm 1, and define D_k as the difference between C_k and the count which (1) suggests, $\frac{1}{2}2^k(2^k - 1)$.

k	1	2	3	4	5	6	7	8	9
C_k	1	8	36	160	656	2688	10816	43520	174336
D_k	0	2	8	40	160	672	2688	10880	43520

Table 1.1: Output of Algorithm 1 with input $n = 2^k$ on increasing values of k

Curiously, neither of these sequences were found in the On-Line Encyclopedia of Integer Sequences [3], but the following relationship is observed.

$$C_k = \begin{cases} D_{k+1} & \text{if } k \text{ is even} \\ D_{k+1} - 2^{k-1} & \text{if } k \text{ is odd.} \end{cases} \tag{1.1}$$

Substituting $D_k = C_k - \frac{1}{2}p^k(p^k - 1)$, and solving for a closed from, (1.1) gives the following conjecture.

Conjecture 1. *Let $C(n)$ denote the number of irreducible quadratic monic polynomials in $\mathbb{Z}_n[x]$. Then for each natural number k,*

$$C(2^k) = \begin{cases} \frac{1}{3}2^{2k+1} - \frac{4}{6}2^k & \textit{if } k \textit{ is even} \\ \frac{1}{3}2^{2k+1} - \frac{5}{6}2^k & \textit{if } k \textit{ is odd.} \end{cases}$$

The same procedure was run for $n = p^k$ for a variety of small primes p, though a pattern does not emerge as clearly. Throughout this paper, we will discuss powers of 2 separately from powers of odd primes, the reason for which will become clear.

$p = 3$						
k	1	2	3	4	5	6
output	3	45	432	4050	36693	331695
output $-\frac{1}{2}p^k(p^k-1)$	0	9	81	810	7290	66339

$p = 5$						
k	1	2	3	4	5	6
output	10	350	9000	227500	5693750	
output $-\frac{1}{2}p^k(p^k-1)$	0	50	1250	32500	812500	

$p = 7$						
k	1	2	3	4	5	6
output	21	1323	65856	3241350		
output $-\frac{1}{2}p^k(p^k-1)$	0	147	7203	360150		

Table 1.2: Output of Algorithm 1 with input $n = p^k$ on a variety of $p, k \in \mathbb{N}$

Table 1.2 has missing entries where computation was excessively expensive; this highlights the impracticality of Algorithm 1, and motivates a more efficient algorithm.

1.3 Further Motivation and Approach

The concept of irreducible polynomials is reminiscent of prime numbers; that is, when we count irreducible polynomials, we are counting the "building blocks" of polynomials. Further, irreducible polynomials over integer rings are useful in the construction of fields, so an efficient construction of irreducible polynomials could provide an efficient construction of fields. The focus of this thesis is the proof of Conjecture 1 and of similar expressions for $n = p^k$ where p is an odd prime, and for composite n. From this proof we have an efficient construction of irreducible polynomials. This construction relies on the relationship between quadratic polynomials and squares as described in [1]. In Section 2 we discuss this relationship between quadratic polynomials and squares, as well as a count of quadratic residues and squares in $\mathbb{Z}_n$ using methods from [4]. Section 3 contains our main theorems, their constructive proofs, and an exploration of efficiency. Since this relationship between quadratic polynomials and squares does not generalize to larger degree polynomials, we also explore some closely related problems in hopes of finding a more general counting approach. To this end, Section 4 will consider counting and generating all roots of these polynomials.

Chapter 2

Counting Squares

2.1 Definitions & Examples

The basic definitions and notation conventions regarding rings and fields follow the text of [2]. In particular, $\mathbb{Z}_n$ denotes the set of integers from 0 to $n-1$ together with addition and multiplication modulo n. Definitions and notation regarding squares and quadratic residues follow the text of [4].
An integer a in the range $0 \leq a < n$ is a *square* in $\mathbb{Z}_n$ if $x^2 = a$ has a solution in $\mathbb{Z}_n$. The square a is also a *quadratic residue* if a is invertible in $\mathbb{Z}_n$; that is, if $\gcd(a, n) = 1$.

Example 2.
In $\mathbb{Z}_{2^4}$, the squares are $0, 1, 4, 9$ and the quadratic residues are $1, 9$.
In $\mathbb{Z}_{2^5}$, the squares are $0, 1, 4, 9, 16, 17, 25$ and the quadratic residues are $1, 9, 17, 25$.

2.2 Quadratic Polynomials & Squares

We begin with the question of how squares are related to quadratic polynomials; the answer is presented in 1964 text by Trygve Nagell, [1].

Lemma 2. *Let a, b, c, n be integers such that $a \neq 0$. Then $ax^2 + bx + c \equiv 0 \pmod{n}$ has a solution if and only if the system of congruences*

$$\begin{cases} y^2 \equiv b^2 - 4ac \pmod{4an} \\ y \equiv b \pmod{2a} \end{cases}$$

has a solution.

Proof.

(⇒) Consider

$$ax^2 + bx + c \equiv 0 \pmod{n} \tag{2.1}$$

where a, b, c, n are integers and $a \neq 0$. Then congruence (2.1) is equivalent to

$$\begin{aligned} 0 &= ax^2 + bx + c + kn \text{ where } k \text{ is some integer} \\ &= 4a^2x^2 + 4abx + 4ax + 4akn \\ &= (2ax + b)^2 - b^2 + 4ac + k(4an). \end{aligned}$$

Let $y = 2ax + b$. Then

$$y \equiv b \pmod{2a}$$

and

$$y^2 \equiv b^2 - 4ac \pmod{4an}$$

directly follow.

($\Leftarrow$) Consider

$$y^2 \equiv b^2 - 4ac \pmod{4an} \tag{2.2}$$

and

$$y \equiv b \pmod{2a} \tag{2.3}$$

where a, b, c, n are integers.

By (2.3), $y - b = j(2a)$ for some integer j, so $2a \mid y - b$. Let $x = (y - b)/2a$; then $x \in \mathbb{Z}$ and

$$y = 2ax + b. \tag{2.4}$$

Hence congruence (2.2) is equivalent to

$$\begin{aligned} 0 &= y^2 - b^2 + 4ac + k(4an) && \text{where } k \text{ is some integer} \\ &= (2ax + b)^2 - b^2 + 4ac + k(4an) && \text{(by (2.4))} \\ &= 4a^2x^2 + 4abx + b^2 - b^2 + 4ac + k(4an). \end{aligned}$$

Dividing by $4a$ we get

$$0 = ax^2 + bx + c + kn$$

which implies

$$ax^2 + bx + c \equiv 0 \pmod{n}.$$

□

Example 3. *Congruence* $x^2 + 3x + 8 \equiv 0 \pmod{9}$ *has solutions*

$$x = 2 \text{ and } x = 4.$$

The set of congruences $\{y^2 \equiv 3^2 - 4(8) \pmod{4(9)}, y \equiv 3 \pmod{2}\}$ *has solutions*

$$y = 7, y = 11, y = 25, \text{ and } y = 29.$$

Then solve $y = 2x + 3$ to get

$$x = 2, x = 4, x = 11 \equiv 2 \pmod 9, \textit{ and } x = 13 \equiv 4 \pmod 9.$$

2.3 Number of Squares in an Integer Ring

Motivated by the relationship between squares and quadratic polynomials, we now count squares in integer rings. The lemmas and theorems in this section come from a 1996 paper by Walter D. Stangl, [4] and are crucial to our study of squares.

Lemma 3. *For $k \geq 3$ and p a prime, the number of squares in $\mathbb{Z}_{p^k}$ is equal to the sum of the number of quadratic residues in $\mathbb{Z}_{p^k}$ and the number of squares in $\mathbb{Z}_{p^{k-2}}$.*

Proof. We want to show that every square in $\mathbb{Z}_{p^k}$ which is not a quadratic residue is in one-to-one correspondence with a square in $\mathbb{Z}_{p^{k-2}}$. To this end, we will construct a bijection between the sets

$$S := \{s \in \mathbb{Z} \colon 0 \leq s < p^k, \exists\, x \text{ such that } x^2 \equiv s \pmod{p^k}, \text{ and } \gcd(s, p^k) > 1\}, \text{ and}$$
$$S' := \{s \in \mathbb{Z} \colon 0 \leq s < p^{k-2} \text{ and } \exists\, x \text{ such that } x^2 \equiv s \pmod{p^{k-2}}\}.$$

Define $f \colon S \to S'$ by $f(s) = s/p^2$. f is a mapping from S to S' because for $s \in S$, s/p^2 is in S' by the following:
Since $s \in S$, there exist integers x, r such that

$$x^2 = s + rp^k. \tag{2.5}$$

Since $\gcd(s, p^k) > 1$, p divides the right side of (2.5), so p also divides the left side. Since p is prime, p^2 divides x^2, the left side of (2.5), so p^2 also divides the right side. Then we have

$$(x/p)^2 = s/p^2 + rp^{k-2},$$

and s/p^2 is an integer less than p^{k-2} since $s < p^k$.
Claim: f is surjective.
Let s' be an element of S'. There exist integers y, t such that

$$y^2 = s' + tp^{k-2}.$$

Then multiplication by p^2 gives

$$(yp)^2 = s'p^2 + tp^{k-2}$$

and $s'p^2$ is an integer less than p^k since $s' < p^{k-2}$. So $s'p^2$ is in S and $f(s'p^2) = s'$.
Claim: f is injective.

Let s_1, s_2 be elements of S such that $f(s_1) = f(s_2)$. Then

$$s_1/p^2 = s_2/p^2,$$

and multiplication by p^2 gives $s_1 = s_2$. □

Example 4.
As seen in Example 2, $\mathbb{Z}_{2^4}$ has 2 quadratic residues which are 1 and 9.
$\mathbb{Z}_{2^2}$ has 2 squares, 0 and 1.
Then as seen in the proof of Lemma 3, $\mathbb{Z}_{2^4}$ has 4 squares: $1, 9, 2^2 \times 0 = 0$, and $2^2 \times 1 = 4$.
Similarly, $\mathbb{Z}_{2^5}$ has 4 quadratic residues and $\mathbb{Z}_{2^3}$ has 3 squares, so $\mathbb{Z}_{2^5}$ has 7 squares.

Recall that we shall consider the odd primes separately from 2. We will first consider the squares in $\mathbb{Z}_{2^k}$.

Lemma 4. *The equation $x^2 = u$ has exactly 0 or 4 solutions in $\mathbb{Z}_{2^k}$ when u is a unit and $k \geq 3$.*

Proof. Let k be some integer greater than or equal to 3, and u a unit in $\mathbb{Z}_{2^k}$.
Suppose $x^2 \equiv a \pmod{2^k}$ has a solution, b. Then clearly $-b$ is also a solution.
Claim: The solutions $b, -b$ are distinct.
Suppose toward a contradiction that $b \equiv -b \pmod{2^k}$. Then $2b \equiv 0 \pmod{2^k}$ which implies $\gcd(b, 2^k) > 1$. But we require that $\gcd(b^2, 2^k) = 1$ since u is a unit; hence, the claim holds.
We have that $2^{k-1} \pm b$ are also solutions since

$$\begin{aligned}(2^{k-1} \pm b)^2 &\equiv 2^{2k-2} \pm 2^k b + b^2 \pmod{2^k}\\ &\equiv b^2 \pmod{2^k}.\end{aligned}$$

Claim: The solutions $2^{k-1} + b$ and $2^{k-1} - b$ are distinct from $b, -b$, and each other.
These solutions are distinct in $\mathbb{Z}_{2^k}$ by the following:
$2^{k-1} + b \equiv 2^{k-1} - b \implies b \equiv -b$
$2^{k-1} + b \equiv b \implies 2^{k-1} \equiv 0$
$2^{k-1} - b \equiv b \implies 2^{k-2} \equiv b$ which contradicts $\gcd(u, 2^k) = 1$
$2^{k-1} + b \equiv -b \implies 2^{k-2} \equiv -b$ which contradicts $\gcd(u, 2^k) = 1$
$2^{k-1} - b \equiv -b \implies 2^{k-1} \equiv 0.$

Then $x^2 \equiv u \pmod{2^k}$ has at least four distinct solutions.

Now suppose toward a contradiction that there exists a fifth distinct solution, c. Then $b^2 \equiv c^2$ which implies $b^2 - c^2 \equiv (b - c)(b + c) \equiv 0$ in $\mathbb{Z}_{2^k}$. Since b, c are both odd, $b + c$ is of the form $(2m + 1) + (2n + 1) = 2(m + n + 1)$ for some non-negative integers m, n. Then we have $(b - c) \times 2(m + n + 1) \equiv 0 \pmod{2^k}$, so $(b - c)(m + n + 1) \equiv 0 \pmod{2^{k-1}}$.

Hence either $b-c$ is a multiple of 0 or 2^{k-1} or $m+n+1$ is a multiple of 2^{k-1} all of which imply $c \equiv \pm b$.

$\square$

Example 5.
In $\mathbb{Z}_{2^4}$, $x^2 = 3$ has no solution since 3 is not a square (seen in Example 2).
However, $x^2 = 9$ has solutions $3, 5, 11,$ and 13.

Lemma 5. *The number of quadratic residues in $\mathbb{Z}_2$ and in $\mathbb{Z}_4$ is 1. For $k > 2$, the number of quadratic residues in $\mathbb{Z}_{2^k}$ is 2^{k-3}.*

Proof. Clearly the only quadratic residue in $\mathbb{Z}_2$ is 1, and the same is true of $\mathbb{Z}_4$.
Now consider $k > 2$. The odd numbers make up the units in $\mathbb{Z}_{2^k}$ of which there are 2^{k-1}. By Lemma 4, the units of $\mathbb{Z}_{2^k}$ can be split into equivalence classes of 4 elements with equal squares; that is, if we square each unit, we will generate each square exactly 4 times. Then there are $2^{k-1}/4 = 2^{k-3}$ of these squares which are, more precisely, quadratic residues in $\mathbb{Z}_{2^k}$.

$\square$

Note that this matches our results from Example 2. We now have all that we need to count the number of squares in $\mathbb{Z}_{2^k}$.

Theorem 6. *The number of squares in $\mathbb{Z}_{2^k}$ is*

$$S(2^k) = \begin{cases} \frac{2^{k-1}+4}{3} & \text{if } k \text{ is even} \\ \frac{2^{k-1}+5}{3} & \text{if } k \text{ is odd.} \end{cases}$$

Proof. We use proof by induction on k.
Consider the first base case $k = 1$. In $\mathbb{Z}_2$, $0^2 = 0$ and $1^2 = 1$, so 0 and 1 are both squares.
Also, $\frac{2^{1-1}+5}{3} = \frac{6}{3} = 2$, so the theorem holds for $k = 1$.
Suppose the theorem holds for natural numbers less than or equal to k.
We also requre a second base case, $k = 2$. In $\mathbb{Z}_4$, $0^2 = 2^2 = 0$ and $1^2 = 3^2 = 1$, so 0 and 1 are both squares.
Also, $\frac{2^{2-1}+4}{3} = \frac{6}{3} = 2$, so the theorem also holds for $k = 2$.
Case $k+1$ is even:
Then by Lemma 3, $S(2^k) = S(2^{k-2}) + Q(2^k)$ where $Q(2^k)$ is the number of quadratic

residues in $\mathbb{Z}_{2^k}$. By Lemma 5, $Q(2^k) = 2^{k-3}$, so from our induction hypothesis we have

$$\begin{aligned} S(2^{k+1}) &= S(2^{k-1}) + Q(2^{k+1}) \\ &= \frac{2^{k-2}+4}{3} + 2^{k-2} \\ &= \frac{4 \times 2^{k-2} + 4}{3} \\ &= \frac{2^k + 4}{3}. \end{aligned}$$

Case $k+1$ is odd:
Again, by Lemmas 3 and 5 we have

$$\begin{aligned} S(2^{k+1}) &= S(2^{k-1}) + Q(2^{k+1}) \\ &= \frac{2^{k-2}+5}{3} + 2^{k-2} \\ &= \frac{4 \times 2^{k-2} + 5}{3} \\ &= \frac{2^k + 5}{3}. \end{aligned}$$

□

Example 6.
Theorem 6 matches what we have seen of $\mathbb{Z}_{2^4}$ *which has* $S(2^4) = \frac{2^{4-1}+4}{3} = 4$ *squares, namely* $\{0, 1, 4, 9\}$.
Similarly, we saw that $\mathbb{Z}_{2^5}$ *has* $S(2^5) = \frac{2^{5-1}+5}{3} = 7$ *squares, namely* $\{0, 1, 4, 9, 16, 17, 25\}$.

We have determined the number of quadratic residues and squares in $\mathbb{Z}_{p^k}$ where $p = 2$ and now reconsider with p an odd prime.

Lemma 7. *The number of quadratic residues of* $\mathbb{Z}_{p^k}$ *where* p *is an odd prime is*

$$Q(p^k) = \frac{p^k - p^{k-1}}{2}.$$

Proof. We use the Euler ϕ function on p^k to see that $\mathbb{Z}_{p^k}$ has $p^k - p^{k-1}$ units; each of these units are a power of the primitive root of p^k, say α. Then the units α^{2j} for $0 \leq j < \frac{\phi(p^k)}{2}$ are distinct quadratic residues. Hence there are $\frac{p^k - p^{k-1}}{2}$ quadratic residues in $\mathbb{Z}_{p^k}$.

□

In the proof of Lemma 7, we rely on primitive roots. Note that there are no primitive roots of 2^k when $k > 2$ which is why, as discussed above, we handled $\mathbb{Z}_{2^k}$ separately. Now we can write our last theorem for counting squares.

Theorem 8. *The number of squares in $\mathbb{Z}_{p^k}$ when p is an odd prime is*

$$S(p^k) = \begin{cases} \frac{p^{k+1}+p+2}{2(p+1)} & \text{if } k \text{ is even} \\ \frac{p^{k+1}+2p+1}{2(p+1)} & \text{if } k \text{ is odd.} \end{cases}$$

Proof. As in Theorem 6, we use proof by induction on k.
Consider first base case $k = 1$. In $\mathbb{Z}_p$, 0 is the only non-unit and is a square, so $S(p) = Q(p) + 1$. Then the theorem holds for $k = 1$ if

$$Q(p) + 1 = \frac{p^{1+1} + 2p + 1}{2(p+1)}.$$

Observe that

$$\begin{aligned} Q(p) + 1 - \frac{p^{1+1} + 2p + 1}{2(p+1)} &= \frac{p-1}{2} + 1 - \frac{p^2 + 2p + 1}{2(p+1)} \\ &= \frac{(p+1)^2 - (p^2 + 2p + 1)}{2(p+1)} \\ &= 0. \end{aligned}$$

As in Theorem 6, we require a second base case, $k = 2$. In $\mathbb{Z}_{p^2}$, 0 is also the only non-unit and is a square, so $S(p^2) = Q(p^2) + 1$. Then the theorem holds for $k = 1$ if

$$Q(p^2) + 1 = \frac{p^{2+1} + p + 2}{2(p+1)}.$$

Observe that

$$\begin{aligned} Q(p^2) + 1 - \frac{p^{2+1} + p + 2}{2(p+1)} &= \frac{p^2 - p}{2} + 1 - \frac{p^3 + p + 2}{2(p+1)} \\ &= \frac{p(p-1)(p+1) + 2(p+1) - p^3 - p - 2}{2(p+1)} \\ &= 0. \end{aligned}$$

Suppose the theorem holds for natural numbers less than or equal to k.
Case $k + 1$ is even:
Claim

$$S(p^{k+1}) = \frac{p^{(k+1)+1} + p + 2}{2(p+1)}.$$

By Lemma 3,

$$S(p^{k+1}) - \frac{p^{(k+1)+1} + p + 2}{2(p+1)} = S(p^{k-1}) + Q(p^{k+1}) - \frac{p^{k+2} + p + 2}{2(p+1)}. \tag{2.6}$$

By Lemma 7, the RHS of (2.6) is equal to

$$S(p^{k-1}) + \frac{p^{k+1} - p^k}{2} - \frac{p^{k+2} + p + 2}{2(p+1)}.$$

Then by the induction hypothesis,

$$\begin{aligned} S(p^{k+1}) - \frac{p^{(k+1)+1} + p + 2}{2(p+1)} &= \frac{p^k + p + 2}{2(p+1)} + \frac{p^{k+1} - p^k}{2} - \frac{p^{k+2} + p + 2}{2(p+1)} \\ &= \frac{p^k + p + 2 + (p+1)(p^{k+1} - p^k) - (p^{k+2} + p + 2)}{2(p+1)} \\ &= 0. \end{aligned}$$

Case $k+1$ is odd:
Claim

$$S(p^{k+1}) = \frac{p^{(k+1)+1} + 2p + 1}{2(p+1)}.$$

By Lemma 3,

$$S(p^{k+1}) - \frac{p^{(k+1)+1} + 2p + 1}{2(p+1)} = S(p^{k-1}) + Q(p^{k+1}) - \frac{p^{k+2} + 2p + 1}{2(p+1)}. \tag{2.7}$$

By Lemma 7, the RHS of (2.7) is equal to

$$S(2^{k-1}) + \frac{p^{k+1} - p^k}{2} - \frac{p^{k+2} + 2p + 1}{2(p+1)}.$$

Then by the induction hypothesis,

$$\begin{aligned} S(p^{k+1}) - \frac{p^{(k+1)+1} + 2p + 1}{2(p+1)} &= \frac{p^k + 2p + 1}{2(p+1)} + \frac{p^{k+1} - p^k}{2} - \frac{p^{k+2} + 2p + 1}{2(p+1)} \\ &= \frac{p^k + 2p + 1 + (p+1)(p^{k+1} - p^k) - (p^{k+2} + 2p + 1)}{2(p+1)} \\ &= 0. \end{aligned}$$

□

Example 7.
We see that $\mathbb{Z}_{3^3}$ *has* $S(3^3) = \frac{3^{3+1} + 2\times 3 + 1}{2(3+1)} = 11$ *squares, namely* $\{0, 1, 4, 7, 9, 10, 13, 16, 19, 22, 25\}$.
$Q(3^3) = \frac{3^3 - 3^{3-1}}{2} = 9$ *of these are quadratic residues, and those are every square except for* 0 *and* 9.
Similarly, $\mathbb{Z}_{5^2}$ *has* $S(5^2) = \frac{5^{2+1} + 5 + 2}{2(5+1)} = 11$ *squares, namely* $\{0, 1, 4, 6, 9, 11, 14, 16, 19, 21, 24\}$.
$Q(5^2) = \frac{5^2 - 5^{2-1}}{2} = 10$ *of these are quadratic residues, and those are every square except for* 0.

2.4 Generate Squares

From the constructive proofs of the above lemmas and theorems, we have the following algorithms for generating all quadratic residues and then all squares in an integer ring.

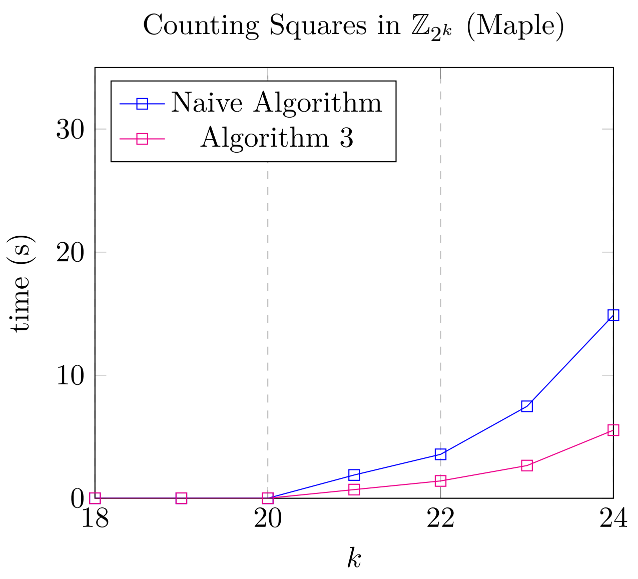
Algorithm 2 Generate Quadratic Residues in $\mathbb{Z}_{p^k}$

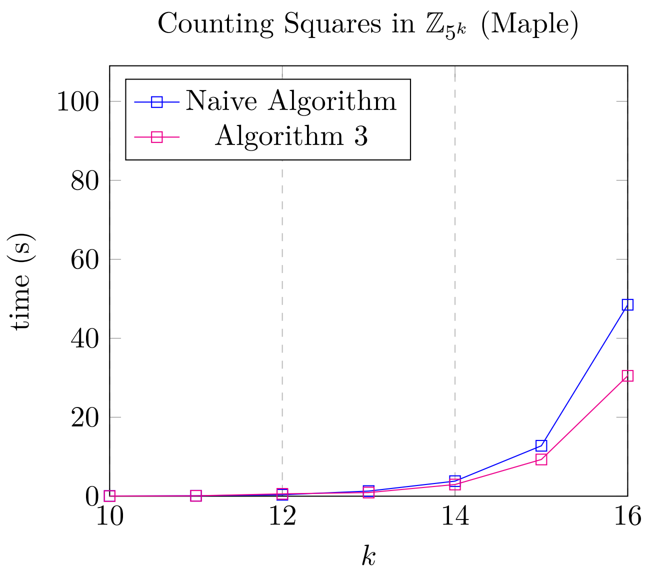
```
Input: $p, k \in \mathbb{Z}^+$ where $p$ is prime
 1: if $p = 2$ then
 2:     if $k < 4$ then
 3:         return 1
 4:     end if
 5:     return $\{(ip+1)^2 \mod n : i = 0, 1, ..., p^{k-2} - 1\}$
 6: end if
 7: $\alpha \leftarrow$ a primitive root modulo $p^k$
 8: $S \leftarrow []$
 9: $S[0] \leftarrow \alpha^2$
10: for $i$ from 1 to $\frac{\phi(p^k)}{2}$ do
11:     $S[i] \leftarrow (\alpha^2 S[i-1]) \mod p^k$
12: end for
13: return $\{S\}$
```

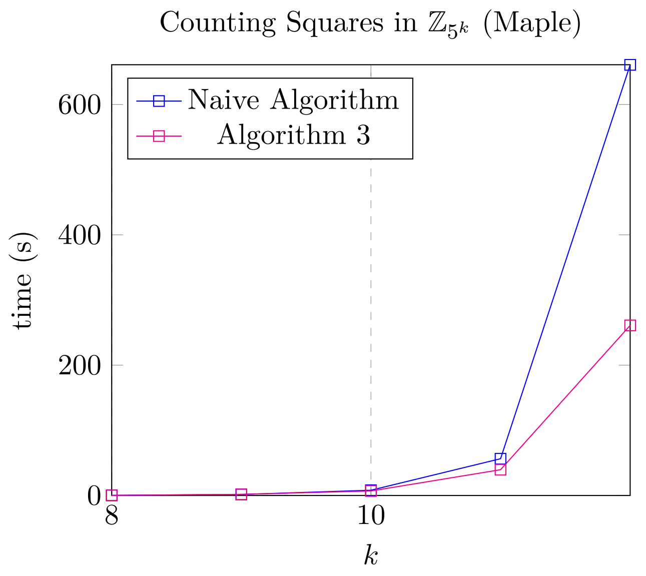
Algorithm 3 Generate Squares in $\mathbb{Z}_{p^k}$

```
Input: $p, k \in \mathbb{Z}^+$ where $p$ is prime
 1: if $k = 0$ then
 2:     return $\{0\}$
 3: else if $k = 1$ then
 4:     return $\{0\} \cup$ output of Algorithm 2 with input $p, 1$
 5: end if
 6: $T \leftarrow$ output of Algorithm 3 with input $p, k-2$
 7: $S \leftarrow$ output of Algorithm 2 with input $p, k$
 8: return $S \cup \{ip^2 \mod p^k : i \in T\}$
```

Algorithm 3 is more efficient than the intuitive approach of simply squaring every element in $\mathbb{Z}_{p^k}$. These algorithms were implemented in Maple alongside this intuitive approach, and below is a comparison of their run-time. More information can be found in Appendix B.


Counting Squares in $\mathbb{Z}_{2^k}$ (Maple)
Naive Algorithm
Algorithm 3
time (s)
0
10
20
30
18
20
22
24
k


Counting Squares in $\mathbb{Z}_{5^k}$ (Maple)
Naive Algorithm
Algorithm 3
time (s)
0
20
40
60
80
100
10
12
14
16
k


Counting Squares in $\mathbb{Z}_{5^k}$ (Maple)
Naive Algorithm
Algorithm 3
time (s)
0
200
400
600
8
10
k

Chapter 3

Main Results and Corollaries

3.1 Counting Results

Finally, we answer our original inquiry with the following theorem. Refer to Theorems 6 and 8 for definitions of $S(2^k)$ and $S(p^k)$.

Theorem 9. *Suppose n has prime factorization*

$$n = 2^{\alpha_0} p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k},$$

where p_i are odd primes and α_i are non-negative integers. Then the number of irreducible, quadratic, monic polynomials in $\mathbb{Z}_n[x]$ is

$$C(n) = n^2 - \frac{n}{2} S(2^{\alpha_0+2}) \prod_{i=1}^{k} S(p_i^{\alpha_i})$$

where $S(n)$ is defined as in Theorems 6 and 8.

Proof. As in Problem (1), we count all quadratic monic polynomials and subtract the number of those which are reducible. The number of quadratic monic polynomials, x^2+bx+c in $\mathbb{Z}_n[x]$ is n^2 since there are n choices for each of b and c. To count those that are reducible, we count the number of b, c pairs such that $x^2 + bx + c \equiv 0 \pmod{n}$ has a solution.
By Lemma 2, this is equal to the number of b, c pairs such that

$$\begin{cases} y^2 \equiv b^2 - 4c \pmod{4n} \\ y \equiv b \pmod{2}. \end{cases}$$

By the Chinese Remainder Theorem,

$$y^2 \equiv b^2 - 4c \pmod{2^{\alpha_0+2} p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}} \tag{3.1}$$

has a solution which is unique modulo n if and only if

$$y^2 \equiv b^2 - 4c \pmod{2^{\alpha_0+2}} \tag{3.2}$$

has a solution, and

$$y^2 \equiv b^2 - 4c \pmod{p_i^{\alpha_i}} \tag{3.3}$$

has a solution for all $1 \leq i \leq k$.
We have from Theorem 5 that there are $S(2^{\alpha_0+2})$ distinct values of $b^2 - 4c$ for which 3.2 has a solution. From Theorem 7, for each i there are $S(p_i^{\alpha_i})$ distinct values of $b^2 - 4c$ such that 3.3 has a solution. Then there are $S(2^{\alpha_0+2})\prod_{i=1}^{k} S(p_i^{\alpha_i})$ distinct values of $b^2 - 4c$ such that 3.1 has a solution. Let s be one of the $S(2^{\alpha_0+2})\prod_{i=1}^{k} S(p_i^{\alpha_i})$ squares in $\mathbb{Z}_n$. Then we solve for b, c pairs in $\mathbb{Z}_n$ such that, $s = b^2 - 4c$ and $y \equiv b \pmod 2$, so there are $\frac{n}{2}$ choices for b, and c is uniquely determined to be $\frac{b^2-y^2}{4}$.
Finally we see that there are

$$\frac{n}{2} S(2^{\alpha_0+2}) \prod_{i=1}^{k} S(p_i^{\alpha_i})$$

reducible monic quadratic polynomials over $\mathbb{Z}_n$ and subtract this from the total number of monic quadratic polynomials over $\mathbb{Z}_n$.

□

In the special case that n is a power of a single prime, we have the following formulas.

Corollary 10.

$$C(2^i) = \begin{cases} \frac{1}{3}2^{2i+1} - \frac{4}{6}2^i & \textit{if } i \textit{ is even} \\ \frac{1}{3}2^{2i+1} - \frac{5}{6}2^i & \textit{if } i \textit{ is odd.} \end{cases}$$

Proof. By Theorem 9,

$$C(2^i) = 2^{2i} - \frac{2^i}{2} S(2^{i+2}).$$

By Theorem 6,

$$S(2^{i+2}) = \begin{cases} \frac{2^{(i+2)-1}+4}{3} & \text{if } i \text{ is even} \\ \frac{2^{(i+2)-1}+5}{3} & \text{if } i \text{ is odd.} \end{cases}$$

Then if i is even,

$$\begin{aligned} C(2^i) &= 2^{2i} - 2^{i-1}\frac{2^{i+1}+4}{3} \\ &= \frac{2^{2i+1}}{3} - \frac{2 \times 2^i}{3}. \end{aligned}$$

And if i is odd,

$$\begin{aligned} C(2^i) &= 2^{2i} - 2^{i-1}\frac{2^{i+1}+5}{3} \\ &= \frac{2^{2i+1}}{3} - \frac{5\times 2^i}{6}. \end{aligned}$$

□

Note that Corollary 10 matches Conjecture 1!

Corollary 11. *For odd prime p,*

$$C(p^i) = \begin{cases} p^{2i} - \frac{p^i(p^{i+1}+p+2)}{2(p+1)} & \textit{if } i \textit{ is even} \\ p^{2i} - \frac{p^i(p^{i+1}+2p+1)}{2(p+1)} & \textit{if } i \textit{ is odd.} \end{cases}$$

Proof. By Theorem 9,

$$C(p^i) = p^{2i} - \frac{p^i}{2}S(2^2)\times S(p^i).$$

By Theorem 6 $S(2^2) = \frac{2^{2-1}+4}{3} = 2$, and by Theorem 8

$$S(p^i) = \begin{cases} \frac{p^{i+1}+p+2}{2(p+1)} & \text{if } i \text{ is even} \\ \frac{p^{i+1}+2p+1}{2(p+1)} & \text{if } i \text{ is odd.} \end{cases}$$

Then if i is even,

$$\begin{aligned} C(p^i) &= p^{2i} - \frac{p^i}{2}\times 2\times\frac{p^{i+1}+p+2}{2(p+1)} \\ &= p^{2i} - \frac{p^i(p^{i+1}+p+2)}{2(p+1)}. \end{aligned}$$

And if i is odd,

$$\begin{aligned} C(p^i) &= p^{2i} - \frac{p^i}{2}\times 2\times\frac{p^{i+1}+2p+1}{2(p+1)} \\ &= p^{2i} - \frac{p^i(p^{i+1}+2p+1)}{2(p+1)}. \end{aligned}$$

□

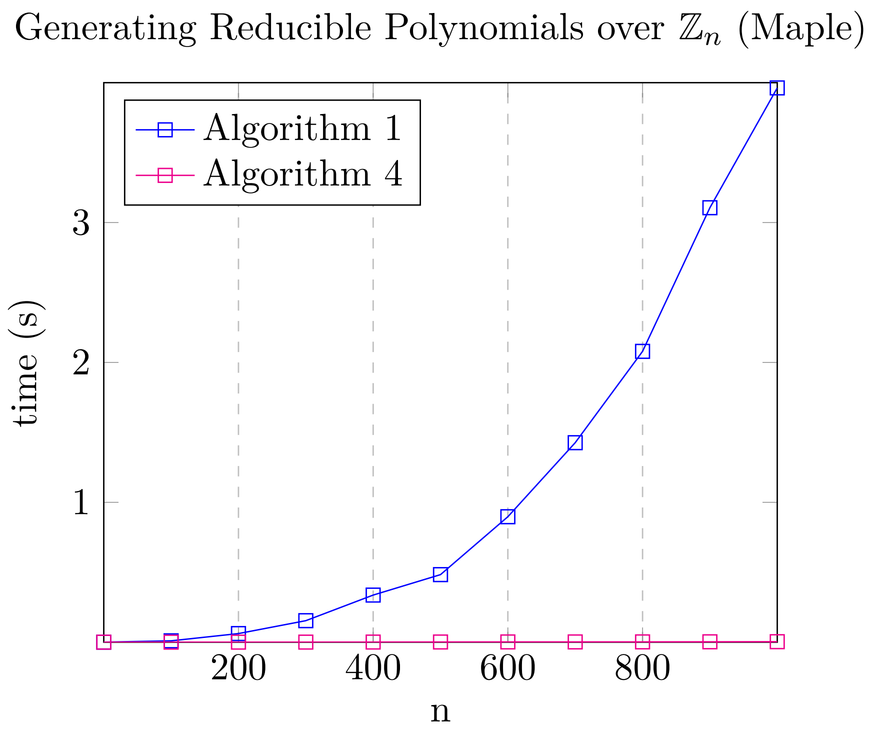
3.2 Construction Results

From the proof of Theorem 9, we have the following algorithm for generating irreducible quadratics over $\mathbb{Z}_n$.

Algorithm 4 New Method to Generate Reducible Quadratic Monic Polynomials in $\mathbb{Z}_n[x]$

```
1:  $P \leftarrow []$;   $i \leftarrow 0$
2:  for $s \in \{x^2 \mod 4n : x \in \mathbb{Z}_{4n}\}$ do
3:      if $2|s$ then
4:          $B \leftarrow \{b \in \mathbb{Z}_n : 2 \mid b\}$
5:      else
6:          $B \leftarrow \{b \in \mathbb{Z}_n : 2 \nmid b\}$
7:      end if
8:      for $b \in B$ do
9:          $c \leftarrow \frac{b^2-s}{4} \mod n$
10:         $P[i] \leftarrow \{x^2 + bx + c\}$;   $i++$
11:     end for
12: end for
13: return $P$
```

This algorithm is computationally less expensive than the naive Algorithm 1 as shown in Figure 3.2. More details can be found in Appendix A.



Chapter 4

Roots

4.1 Lifting

The reader should now see why the method used in this paper does not generalize to larger degree polynomials, so we hope to find a more general approach for this same count. For the sake of further exploration, we consider the problem of counting roots. An efficient way to generate roots of polynomials in $\mathbb{Z}_{p^k}[x]$ is to "lift roots" from $\mathbb{Z}_p[x]$. Returning to Nagell's text, we see that

"If we know the solution of the congruence

$$f(x) \equiv 0 \pmod{p^\alpha} \quad (\alpha \geq 1), \tag{4.1}$$

it is possible to deduce the solutions of the congruence

$$f(x) \equiv 0 \pmod{p^{\alpha+1}}\text{"} \tag{4.2}$$

by the following algorithm, the details of which are given in Appendix C.

Algorithm 5 Lift root $r \in \mathbb{Z}_{p^\alpha}$ of $f(x)$ in $\mathbb{Z}[x]$ to root(s) of $f(x)$ in $\mathbb{Z}_{p^{\alpha+1}}$

Input: prime p, $\alpha \in \mathbb{N}$, $f \in \mathbb{Z}[x]$, and $r \in \mathbb{Z}$ such that $0 \le r < p^\alpha$ and $f(r) \equiv 0 \pmod{p^\alpha}$

Output: $\{r' \in \mathbb{Z} : 0 \le r' < p^{\alpha+1}, f(r') \equiv 0 \pmod{p^{\alpha+1}}, \text{and } r' \equiv r \pmod{p^\alpha}\}$

```
1: $g \leftarrow f'(r) \mod p$
2: $h \leftarrow (-f(r)/p^\alpha) \mod p$
3: if $g = 0$ and $h = 0$ then
4:     return $\{r + t \cdot p^\alpha : t \in \mathbb{Z}$ and $0 \le t < p\}$
5: else if $g = 0$ or $h = 0$ then
6:     return $\emptyset$
7: else
8:     $t = h \cdot g^{-1} \mod p$
9:     return $\{r + t \cdot p^\alpha\}$
10: end if
```

Example 8.

Consider calling Algorithm 5 with input

$$p = 7, \alpha = 1, f = x^2 + 2x + 1, \text{ and } r = 6.$$

Since $f' = 2x + 2$, *line* 1 *assigns* $g \leftarrow 0$, *and line* 2 *assigns* $h \leftarrow 0$.
Then line 3 *returns* $\{6, 13, 20, 27, 34, 41, 48\}$.
We can check that in fact $f(r) \equiv 0 \pmod{7^2}$ *for all* r *in* $\{6, 13, 20, 27, 34, 41, 48\}$.
Consider calling Algorithm 5 with input

$$p = 3, \alpha = 3, f = x^2 + 23x + 4, \text{ and } r = 11.$$

Since $f' = 2x + 23$, *line* 1 *assigns* $g \leftarrow 0$, *and line* 2 *assigns* $h \leftarrow 1$.
Then line 6 *returns* $\emptyset$.
We can check that in fact f *has no root in* $\mathbb{Z}_{3^4}$ *which is congruent to* 11 *modulo* p.
Consider calling Algorithm 5 with input

$$p = 5, \alpha = 2, f = x^2 + 15x + 9, \text{ and } r = 9.$$

Since $f' = 2x + 15$, *line* 1 *assigns* $g \leftarrow 3$, *and line* 2 *assigns* $h \leftarrow 1$.
Then $t = 3$, *so line* 9 *returns* $\{9 + 2 \cdot 5^2 = 59\}$.
We can check that in fact, $f(59) \equiv 0 \pmod{5^3}$.

Theorem 12. *Algorithm 5 is correct.*

Proof. Any solution r' to 4.2 must also be a solution to 4.1 since

$$f(r') = cp^{\alpha+1} = (cp)p^{\alpha}$$

for some integer c. Then r' is of the form $r + tp^{\alpha}$ where r as a solution to 4.1, $0 \leq r < p^{\alpha}$, $t \in \mathbb{Z}$, and $0 \leq t < p$. So we want to show that Algorithm 5 will "lift" the root r to $r' = r + tp^{\alpha}$, a solution to 4.2.
By Taylor's theorem

$$\begin{aligned} f(r + tp^{\alpha}) &= f(r) + f'(r)tp^{\alpha} + \frac{f''(r)tp^{\alpha+1}}{2} + \dots \\ &\equiv f(r) + f'(r)tp^{\alpha} \mod p^{\alpha+1}. \end{aligned}$$

Then Algorithm 5 should solve for t such that

$$f(r) + f'(r)tp^{\alpha} \equiv 0 \mod p^{\alpha+1}.$$

In fact, Algorithm 5 solves

$$gt \equiv h \mod p$$

where

$$g = f'(r), \qquad h = \frac{-f(r)}{p^{\alpha}}.$$

This is equivalent by the following:

$$\begin{aligned} & f(r) + f'(r)tp^{\alpha} \equiv 0 \mod p^{\alpha+1} \\ \iff \quad & f(r) + f'(r)tp^{\alpha} + dp^{\alpha+1} = 0 \\ \iff \quad & f(r) + dp^{\alpha+1} = -f'(r)tp^{\alpha} \\ \iff \quad & -\frac{f(r)}{p^{\alpha}} - dp = f'(r)t \\ \iff \quad & f'(r)t \equiv \frac{-f(r)}{p^{\alpha}} \mod p. \end{aligned}$$

Also note that we have $0 \leq r < p^{\alpha}$ and $0 \leq t < p$, so $0 \leq r + tp^{\alpha} < p^{\alpha+1}$ as desired.
We also want to show that every r' which satisfies the set conditions of the output of Algorithm 5 is indeed in the output of Algorithm 5. Let r_0 be one such integer.

Then we have

1. $0 \leq r_0 < p^{\alpha+1}$,
2. $f(r_0) \equiv 0 \pmod{p^{\alpha+1}}$, and
3. $r_0 \equiv r \pmod{p^{\alpha}}\}$.

By (2), there exists a solution to 4.1, ϵ and an integer t such that $r_0 = \epsilon + tp^{\alpha}$. By (3), ϵ must be r, so $r_0 = r + tp^{\alpha}$. Then r_0 is in the output of Algorithm 5 with input p, α, f, r. □

Finally, we have the following algorithm to generate all roots of $f(x)$ in $\mathbb{Z}_{p^k}$.

Algorithm 6 Generate all roots in $\mathbb{Z}_{p^k}$ of $f(x)$ in $\mathbb{Z}[x]$

Input: prime p, $k \in \mathbb{N}$, and $f \in \mathbb{Z}[x]$

Output: $\{r \in \mathbb{Z} : 0 \leq r < p^k$ and $f(r) \equiv 0 \pmod{p^k}\}$

```
1: $R \leftarrow$ roots of $f$ modulo $p$
2: if $k = 1$ then
3:     return R
4: else
5:     for $i$ from 1 to $k-1$ do
6:         $R \leftarrow \bigcup_{r \in R}$ output of Algorithm 5 with input $p, i, f, r$
7:     end for
8:     return $R$
9: end if
```

Theorem 13. *Algorithm 6 is correct.*

Proof. Let R denote the output of Algorithm 6 with input p, k, f. From Theorem 12, every element in R is a root of f modulo p^k. Then it suffices to show that R contains every root of f modulo p^k.

We proceed by induction on k. As a base case, consider $k = 1$; the claim trivially holds by lines 1 through 3 of Algorithm 6. Suppose that the output of Algorithm 6 with input $p, k-1, f$ returns every root of f modulo p^{k-1}. We want to show that the output of Algorithm 6 with input p, k, f returns every root of f modulo p^k. By the induction hypothesis, the $k-2$nd iteration of the for-loop in Algorithm 6 assigns R as the set of every root of f modulo p^{k-1}. By Theorem 12, Algorithm 5 with input $p, k-1, f, r$ returns every root of f modulo p^k which are congruent to r modulo p^{k-1}. Then the $k-1$st iteration of the for-loop assigns R as the set of every root of f modulo p^k. □

Chapter 5

Conclusion

By considering the squares of integer rings, we have now answered the question: how many irreducible quadratic monic polynomials are there are over $\mathbb{Z}_n$ for all natural numbers $n > 1$? In doing so, we have produced an algorithm which efficiently generates these polynomials. A primary application of irreducible polynomials is in the construction of extension fields, so Algorithm 4 can be further used to efficiently generate ideals and extension fields. In the current case of quadratic polynomials, we are limited to extension fields of size p^2 for p prime. A natural next problem for this area of research is

For positive integers n, k, how many irreducible monic polynomials of degree k are there in $\mathbb{Z}_n[x]$? How can we efficiently generate them?

If our method could be generalized from $k = 2$ to all integers k, we would be able to quickly construct extension fields of size p^{k-1} for prime p and integer $k \geq 4$. Since the relationship between polynomials and squares would not be so clear in this generalized case, we would likely require an entirely new approach. Perhaps the presented method for counting roots would be a good starting point.

Bibliography

[1] Trygve Nagell. *Introduction to Number Theory.* Chelsea Publishing Company, New York, second edition, 1964.

[2] Norman R. Reilly. *Introduction to Applied Algebraic Systems.* Oxford University Press, New York, 2009.

[3] N.J.A. Sloane. On-line encyclopedia of integer sequences. `https://oeis.org/`. Accessed: 2018-11-15.

[4] Walter D. Stangl. Counting squares in $\mathbb{Z}_n$. *Mathematics Magazine*, 69(4):285–289, 1996.

Appendix A

Implementation of Algorithms 1 and 4

A.1 Algorithm 1 with n a power of 2 implemented in Cilk

```
#include <stdio.h>
#include <stdlib.h>

int * array( int n ) { return( malloc( n*sizeof(int) ) ); }

void printarray( int *A, int n ) {
    int i;
    printf("[");
    for( i=0; i<n; i++ ) { if( i!=0 ) printf(","); printf("%ld",A[i]); }
    printf("]\n");
    return;
}

int sumArray( int *P, int size ) {
int i,sum;
    sum = 0;
    for( i=0; i<size; i++ ) { sum = sum + P[i]; }
    return sum;
}

void printdata( int *N, int *C, int *D, int numTrials ) {
    printf("\nN = ");
    printarray(N,numTrials);
    printf("C = ");
    printarray(C,numTrials);
    printf("D = ");
    printarray(D,numTrials);
}

cilk int parallelCount( int n, int l, int m )  {
int N,a,b,f,IRRED,y,mask,z;
     N = 0;
     mask = n-1; // assuming n = 2^i
     for( a=0; a<n; a++ ) {
         for( b=0; b<n; b++ ) {
             //f := x^2+a*x+b;
             IRRED = 1;
             for( y=0; y<n && IRRED; y++ ) {
                 //if( ((y*y+a*y+b) % n) == 0 ) IRRED = 0;
                 if( ((y*y+a*y+b) & mask) == 0 ) IRRED = 0;
             }
             if( IRRED ) N++;
         }
      }
      return N;
}


cilk int main( int argc, char *argv[] ) {
    int i,j,numTrials,NumCores,step,MinI,MaxI,k;
    int *N; int *C; int *D; int *P;
    MinI = 1;
    MaxI = 2;
    numTrials = MaxI-MinI+1;
    N = array(numTrials); C = array(numTrials); D = array(numTrials);

    NumCores = 1; //default 1 core
    if( argc > 1) sscanf(argv[1],"%d\n", &NumCores );
    printf("NumCores = %d\n",NumCores);

    P = array(NumCores);

    k=0;
    for( i=MinI; i<=MaxI; i++ ) {
        N[k] = 1 << i;
        C[k] = 0;
        if (N[k]<4) {
            C[k] = count(N[k]);
        } else {
            step = N[k]/NumCores;
            for( j=0; j<NumCores; j++ ) {
               P[j] = spawn parallelCount(N[k],j*step,(j+1)*step);
            }
            sync;
            C[k] = sumArray(P,NumCores);
        }
        D[k] = (N[k]-N[k]*N[k])/2 + C[k];
        k++;
    }

    printdata(N, C, D, numTrials);

    return 1;
}
```

A.2 Algorithm 1 with n a power of an odd prime implemented in C

```c
#include <stdio.h>
#include <stdlib.h>

FILE *f;

int * array( int n ) { return( malloc( n*sizeof(int) ) ); }

void printarray( int *A, int n ) {
    int i;
    printf("[");
    for( i=0; i<n; i++ ) { if( i!=0 ) printf(","); printf("%d",A[i]); }
    printf("]\n");
    return ;
}

int power( int b, int e ) {
    if (e==0) return 1;
    else return b*power(b,e-1);
}

int count( int n )  {
int N,a,b,IRRED,y,z;
     N = 0;
     for( a=0; a<n; a++ ) {
         for( b=0; b<n; b++ ) {
             //f := x^2+a*x+b;
             IRRED = 1;
             for( y=0; y<n; y++ ) {
                 fprintf(f, "x^2 + %dx + %d\r\n\r\n", a, b);
                 if( ((y*y+a*y+b) % n) == 0 )  { IRRED = 0; break; }
             }
             if( IRRED ) {N++; fprintf(f, "N = %d\r\n", N);}
         }
      }
      return N;
}


int main() {
    int a,b,c,d,n,i,r;
    int *A;

    f = fopen("output.txt", "w");
    if (f == NULL) {
        printf("Error opening file output.txt\n");
        exit(1);
    }

    for( i=1; i<=3; i++ ) {
        n = power(3,i);
        c = count(n);
        printf("n=%d  #=%d \n",n,c);
    }
    return 1; }
```

A.3 Algorithm 4 Implemented in Maple

```
> #generate all monic, quadrative polynomials over Z_{p^k}
> AllPolynomials := proc(p,k)
> local n,b,c,P;
> P := {};
> n := p^k;
> for b from 0 to n-1 do
> for c from 0 to n-1 do
> P := P union {x^2 + b*x + c};
> od;
> od;
> return P;
> end proc:
> IrrPolynomials := proc(p,k)
> local s,S,b,B,c,P,n,s1,s2;
> n := p^k;
> P := {};
> if p=2 then
> S := Squares(p,k+2);
> for s in S do
> if type(s,even) then
> B := {seq(2*i,i=0..(n/2)-1)};
> else
> B := {seq(2*i+1,i=0..(n/2)-1)};
> fi;
> for b in B do
> c := (b^2 - s)/4 mod n;
> P := P union {x^2 + b*x + c};
> od;
> od;
> else
> for s1 in Squares(2,2) do
> for s2 in Squares(p,k) do
> s := chrem([s1,s2],[4,p^k]);
> if type(s,even) then
> B := {seq(2*i,i=0..(n/2))};
> else
> B := {seq(2*i+1,i=0..(n/2))};
> fi;
> for b in B do
> c := (b^2 - s)/4 mod n;
> P := P union {x^2 + b*x + c};
> od;
> od;
> od;
> fi;
> return (AllPolynomials(p,k) minus P);
> end proc:
```

Appendix B

Implementation of Algorithms 2 and 3

B.1 Algorithm 2 implemented in Maple

```
>  with(NumberTheory):
>  Quads := proc(p,k)
>  local S,i,alpha,n;
>  n := p^k;
>  if p=2 then
>  if k<4 then return {1}; fi;
>  return {seq((p*i + 1)^2 mod n,i=0..p^(k-2)-1)};
>  fi;
>  alpha := PrimitiveRoot(p);
>  S := Array(0..phi(n)/2);
>  S[0] := alpha^2;
>  for i from 1 to phi(n)/2 do
>  S[i] := (alpha^2 * S[i-1]) mod n;
>  od;
>  return {seq(S[i],i=0..phi(n)/2)};
>  end proc:
```

B.2 Algorithm 3 implemented in Maple

```
>  #A new implementation to generate all squares in Z_{p^k}
>  Squares := proc(p,k)
>  local S,T,i,n;
>  n := p^k;
>  if k=0 then return {0}; fi;
>  if k=1 then return {0} union Quads(p,1); fi;
>  T := Squares(p,k-2);
>  S := Quads(p,k);
>  return S union {seq((i*p^2) mod n, i in T)};
>  end proc:
```

B.3 Example

```
>  p := 3;
>  k := 4;
>  Squares(p,k);
```

$$p := 3$$

$$k := 4$$

$$\{0, 1, 4, 7, 9, 10, 13, 16, 19, 22, 25, 28, 31, 34, 36, 37, 40, 43, 46, 49, 52, 55, 58, 61, 63, 64, 67, 70, 73, 76, 79\}$$

Appendix C

Implementation of Algorithms 5 and 6

C.1 Algorithm 5 Implemented in Maple

```
> #lifts a single root (r) of function (f with derivative F) in Z_{p^alpha}[x]
to root(s) of f in Z_{p^{alpha+1}}[x]
> liftRoot := proc(f,F,p,alpha,r)
> local a,ainv,b,T,t;
> a := Eval(F,x = r) mod p;
> b := -(eval(f,x = r)/p^alpha);
> b := b mod p;
> if (a=0 and b=0) then return [seq(r + t*p^alpha, t=0..p-1)];
> fi;
> if (a=0 or b=0) then return NULL; fi;
> ainv := 1/a mod p;
> t := b*ainv mod p;
> return r + t*p^alpha;
> end proc:
```

C.2 Algorithm 6 Implemented in Maple

```
> #lifts root(s) (R) of function (f) in Z_{p^alpha}[x] to root(s) of f in
Z_{p^{alpha+1}}[x]
> PadicLift := proc(f,F,p,alpha,R)
> return [seq(liftRoot(f,F,p,alpha,r), r in R)];
> end proc:
> #Uses p-adic lifting to find the roots of function (f) in Z_{p^alpha}[x]
> findRoots := proc(f,p,alpha)
> local rootsZp,i,R,F;
> rootsZp := Roots(f) mod p;
> R := [seq(rootsZp[i,1], i=1..nops(rootsZp))];
> if alpha = 1 then return rootsZp; fi;
> F := diff(f,x);
> for i from 1 to alpha-1 do
> R := PadicLift(f,F,p,i,R);
> od;
> end proc:
```

C.3 Examples

```
>  #A reducible example
>  f := x^6 + 2*x^5 + 2*x^3 + 6*x + 3;
>  p := 3;
>  alpha := 4;
>  findRoots(f,p,alpha);
```

$$f := x^6 + 2\,x^5 + 2\,x^3 + 6\,x + 3$$

$$p := 3$$

$$\alpha := 4$$

$$[41]$$

```
>  #An irreducible example
>  f := x^6 + 3*x^5 + 2*x^3 + 15*x + 7;
>  p := 3;
>  alpha := 4;
>  findRoots(f,p,alpha);
```

$$f := x^6 + 3\,x^5 + 2\,x^3 + 15\,x + 7$$

$$p := 3$$

$$\alpha := 4$$

$$[]$$