2012

Testing Irreducibility of Trinomials over GF(2)

Steven Hayman
shayman@iwu.edu

TESTING IRREDUCIBILITY OF TRINOMIALS OVER $\mathbb{F}_2$

STEVEN HAYMAN

Abstract. The focus of this paper is testing the irreducibility of polynomials over $\mathbb{F}_q$. In particular there is an emphasis on testing trinomials over $\mathbb{F}_2$.

1. Introduction

The study of testing polynomials over finite fields for irreducibility was motivated by gathering evidence to support the conjecture that $x^n + x^3 + 1$ and $x^n + x^3 + x^2 + x + 1$ are simultaneously irreducible infinitely often over $\mathbb{F}_2$ [7]. Testing polynomials over finite fields for irreducibility has a number of cryptographic applications [6].

2. Main Results

In this paper we present three algorithms for testing polynomials for irreducibility over finite fields: the Ben-Or, Rabin [9], and Standard algorithms [4]. The author independently discovered the Standard algorithm before reading [4]. The Standard algorithm does seem to perform slightly better than the Ben-Or algorithm. Additionally, the Standard algorithm can be proven to have a better worst case complexity, and this result is supported by the timings given in Section 7.

We also present timings which suggest optimal parameter choices for the Ben-Or and Standard algorithms. Using the Ben-Or algorithm we have tabulated all irreducible trinomials up to degree 100,000. This data may help provide support for the conjectures found in [1].

3. Irreducible Polynomials over Finite Fields

In this section we give an overview of the algebraic concepts central to this paper. In particular we discuss finite fields, polynomials over finite fields, and finally irreducible polynomials over finite fields.

We begin by giving the definition of a field. Maybe the most important thing to note for readers unfamiliar with fields is that the set of real numbers with the operations of addition and multiplication is a familiar example of a field.

Definition 1 (Field)**.** A field is a set F with binary operations $+$ and $\times$ defined over F, and denoted by $(F, +, \times)$ that satisfy:

(1) For all $x, y, z \in F$ we have $(x + y) + z = x + (y + z)$.
(2) There exists $0 \in F$ such that $x + 0 = 0 + x = x$.
(3) For all $x \in F$ there exists $y \in F$ such that $x + y = y + x = 0$.
(4) For all $x, y \in F$ we have $x + y = y + x$.
(5) For all $x, y, z \in F \setminus \{0\}$ we have $(x \times y) \times z = x \times (y \times z)$.
(6) There exists $1 \in F \setminus \{0\}$ such that $x \times 1 = 1 \times x = x$.
(7) For all $x \in F \setminus \{0\}$ there exists $y \in F \setminus \{0\}$ such that $x \times y = y \times x = 1$.
(8) For all $x, y \in F$ we have $x \times y = y \times x$.
(9) For all $x, y, z \in F$ we have $x \times (y + z) = x \times y + x \times z$

If $(F, +, \times)$ is a field and F has a finite number of elements then $(F, +, \times)$ is called a finite field.

We have already stated $(\mathbb{R}, +, \times)$ is a field. Now we give a few more examples of fields.

Example 1. One can verify that $(\mathbb{Q}, +, \times)$ and $(\mathbb{C}, +, \times)$ are fields. If p is prime then $(\mathbb{Z}_p, +, \times)$ is a finite field. (Recall that $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, $+$ is addition modulo p, and $\times$ is multiplication modulo p.)

An important result from algebra is that finite fields can only have p^n elements, where p is prime. Furthermore, the finite field that contains p^n is unique up to an isomorphism. For this reason, these fields are denoted by the number of elements they have. To be concrete, the finite field $(\mathbb{Z}_2, +, \times)$ is denoted by $\mathbb{F}_2$. For the rest of this paper we will use this notation.

Now that we have given the definition of field, we can discuss polynomials over finite fields. It is hopefully the case that this a natural generalization from polynomials over the field $(\mathbb{R}, +, \times)$. The following definition can be found in [14].

Definition 2 (Polynomial)**.** Let F be a field. A polynomial over F is an expression of the form $\sum_{i=0}^{n} a_i x^i$ where $n \in \mathbb{N}$, each $a_i \in F$, and $a_n \neq 0$. Each a_i is called a coefficient. The degree of f, denoted $\deg(f)$, is n. The set of all polynomials over F is denoted by $F[x]$.

We proceed by providing a few examples of polynomials over a field.

Example 2. $x^2 + x + 1 \in \mathbb{F}_2[x]$ since 1, 1, and 1 are elements of $\mathbb{F}_2$. $3x^2 + 4x + 1 \in \mathbb{F}_5[x]$ since 3,4 and 1 are elements of $\mathbb{F}_5$.

Next we define irreducible and reducible polynomials. This definition can be best understood by relating it to prime and composite integers.

Definition 3 (Reducible/Irreducible)**.** Let F be a field. A polynomial $f(x) \in F[x]$ is reducible over F if and only if there is a $g(x) \in F[x]$ such that $g(x)$ divides $f(x)$ and $0 < \deg(g) < \deg(f)$. If such a $g(x)$ exists we call $g(x)$ a factor of $f(x)$. A polynomial is irreducible if and only if it is not reducible. The set of irreducible polynomials over F of degree n is denoted by $I_{F,n}$.

Example 3. Consider the polynomial $x^2 + 1 \in \mathbb{R}[x], \mathbb{C}[x]$ and $\mathbb{F}_2[x]$.

Over $\mathbb{R}$ we have that $x^2 + 1$ is irreducible since it cannot be factored as $(x - r_1)(x - r_2)$ where $r_1, r_2 \in \mathbb{R}$, and so $x^2 + 1 \in I_{\mathbb{F}_2,2}$. However, over $\mathbb{C}$ we have that $x^2 + 1$ is reducible since $x^2 + 1 = (x - i)(x + i)$ and $-i, i \in \mathbb{C}$.

We also have that $x^2 + 1$ is reducible over $\mathbb{F}_2$ since $x^2 + 1 = x^2 + 2x + 1 = (x + 1)^2$. (Remember that in $\mathbb{F}_2$, we have $2 = 0$).

It is worth noting that $c \in F$ with c dividing $f(x)$ does not necessarily mean that $f(x)$ is reducible. This is because $\deg(c) = 0$. For example 2 divides $2x + 4$ but $2x + 4$ is irreducible over $\mathbb{F}_5$.

One can fairly easily determine the reducibility of both monomial and binomials over $\mathbb{F}_2$.

Example 4. A monomial is a polynomial with a single term. Thus a monomial over $\mathbb{F}_2$ has the form x^r where $r \in \mathbb{N}$ and $r \geq 0$. When $r \geq 2$, we have that x divides x^r, and thus x^r is reducible over $\mathbb{F}_2$.

A binomial over $\mathbb{F}_2$ has the form $x^r + x^s$ with $r, s \in N$ and $r > s \geq 0$. If $s = 0$ we have $x^r + x^s = x^r + 1$. If this is the case, and $r \geq 2$ then $x + 1$ divides $x^r + 1$, and so $x^r + 1$ is reducible. If $s \neq 0$ then $r \geq s \geq 1$, and x divides $x^r + x^s$, and so $x^r + x^s$ is reducible.

We next consider trinomials of $\mathbb{F}_2$, that is $x^r + x^s + x^t$ where $r, s, t \in \mathbb{N}$ and $r \geq s \geq t \geq 0$. If $t \geq 1$ then x divides $x^r + x^s + x^t$ and so $x^r + x^s + x^t$ is reducible. However, if $t = 0$, we have $x^r + x^s + x^t = x^r + x^s + 1$. If this is the case, there is no general rule for determining reducibility. It is for this reason we will focus on $x^r + x^s + 1$.

Finally we present a simple fact about reducible polynomials.

Proposition 1. Let $f(x) \in F[x]$ with $\deg(f) = n$. If $f(x)$ is reducible then $f(x)$ has a factor of degree less than or equal to $\lfloor \frac{n}{2} \rfloor$.

Proof. Suppose $f(x)$ is reducible and has only factors of degree greater than $\lfloor \frac{n}{2} \rfloor$. Since $f(x)$ is reducible it follows that$f(x) = g(x)h(x)$ where $\deg(g) > \lfloor \frac{n}{2} \rfloor$ and $\deg(h) > \lfloor \frac{n}{2} \rfloor$.

First suppose n is even. We have that $\lfloor \frac{n}{2} \rfloor = \frac{n}{2}$. Then $\deg(f) = \deg(g)+\deg(h) > \frac{n}{2}+\frac{n}{2} = n$. But this is a contradiction since $\deg(f) = n$.

Next suppose that n is odd. We have that $\lfloor \frac{n}{2} \rfloor = \frac{n-1}{2}$. Then $\deg(g) > \frac{n-1}{2}$ and $\deg(h) > \frac{n-1}{2}$, or equivalently $\deg(g) \geq \frac{n+1}{2}$ and $\deg(h) \geq \frac{n+1}{2}$. Thus we have $\deg(f) = \deg(g)+\deg(h) \geq \frac{n+1}{2}+\frac{n+1}{2} = n+1$. But this is a contradiction since $\deg(f) = n$. □

4. Conditions for Irreducibility

All the tests for irreducibility presented in this paper are based on the following theorem. The theorem is well-known and will be given without proof, however the proof can be found in [14].

Theorem 1.

$$x^{q^n} - x = \prod_{f(x)\in\Phi_n} f(x)$$

where $\Phi_n := \{f(x) \in I_{\mathbb{F}_q,d} : d \text{ divides } n\}$.

More colloquially this is that over $\mathbb{F}_q$, $x^{q^n} - x$ is the product of all irreducible polynomials of degree d for all d dividing n.

Example 5. By Theorem 1, the product of all irreducible polynomials over $\mathbb{F}_2$ of degree d dividing 4 is

$$x^{2^4}-x = x(x+1)(x^2+x+1)(x^4+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1).$$

It follows from Theorem 1 that if $f(x)$ has a factor $g(x)$ of degree d then $g(x)$ divides $\gcd(f(x), x^{q^n} - x)$. This is the idea that will be used in the proof of the following Theorem.

Theorem 2. *Let* $f(x) \in \mathbb{F}_q[x]$ *with* $\deg(f) = n$. *We have that* $f(x)$ *is irreducible if and only if* $\gcd(f(x), x^{q^d} - x) = 1$ *for* $d \in \{1, 2, \ldots, \lfloor \frac{n}{2} \rfloor\}$.

Proof. Suppose $\gcd(f(x), x^{q^d} - x) \neq 1$ for $d \in \{1, 2, \ldots, \lfloor \frac{n}{2} \rfloor\}$. Let $g(x) = \gcd(f(x), x^{q^d} - x)$.

Since $g(x)$ divides $x^{q^d} - x$ then by Theorem 1 that $g(x)$ divides $\prod_{h(x)\in\Phi_d} h(x)$. It follows that $g(x) = \prod_{h(x)\in\Delta} h(x)$ where $\Delta \subseteq \Phi_d$.

Let $\hat{h}(x) \in \Delta$. Then $\hat{h}(x)$ divides $g(x)$ and $g(x)$ divides $f(x)$ so $\hat{h}(x)$ divides $f(x)$. Since $\hat{h}(x) \in \Delta$ and $\Delta \subseteq \Phi_d$ then $\hat{h}(x) \in \Phi_d$ and it follows that $0 < \deg(\hat{h}) \leq d < \lfloor \frac{n}{2} \rfloor$. By definition $f(x)$ is reducible.

Now suppose $f(x)$ is reducible. By Proposition 1 we have that $f(x)$ has a factor $g(x)$ with degree r where $r \leq \lfloor \frac{n}{2} \rfloor$. It follows from Theorem 1 that $g(x)$ divides $x^{q^r} - x$. Thus $\gcd(f(x), x^{q^r} - x) \neq 1$. □

Another theorem based on Theorem 1 can be used to determine irreducibility, and can be found in [17].

Theorem 3. *Let $f(x) \in \mathbb{F}_q[x]$ with $\deg(f) = n$. We have that $f(x)$ is irreducible if and only if $x^{q^n} = x \bmod f(x)$ and $\gcd(f(x), x^{q^{n/p}} - x) = 1$ for all prime p dividing n.*

Proof. Suppose $x^{q^n} \neq x \bmod f(x)$. Then $f(x)$ does not divide $x^{q^n} - x$. By Theorem 1, $f(x)$ does not divide $\prod_{h(x)\in\Phi_n} h(x)$ and it follows that $f(x)$ is reducible.

Suppose that $\gcd(f(x), x^{q^{n/p}} - x) \neq 1$ for some prime p dividing n. Since p is prime then $p \geq 2$ then it follows from Theorem 2 that $f(x)$ is reducible.

Suppose $f(x)$ is reducible and $x^{q^n} = x \bmod f(x)$. Since $f(x)$ is reducible, then $f(x)$ has a factor $g(x)$ with $0 < \deg(g) < n$. Also, since $f(x)$ divides $x^{q^n} - x$ and by Theorem 1, $f(x)$ divides $\prod_{h(x)\in\Phi_n} h(x)$. Since $g(x)$ divides $f(x)$ it follows that $g(x)$ divides $\prod_{h(x)\in\Phi_n} h(x)$. Thus $g(x) = \prod_{h(x)\in\Delta} h(x)$ where $\Delta \subseteq \Phi_n$.

Let $\hat{h}(x) \in \Delta$. Since $\hat{h}(x) \in \Delta$ and $\Delta \subseteq \Phi_n$ then $\hat{h}(x) \in \Phi_n$. Then $\deg(\hat{h})$ divides n. Since $\hat{h}(x) \in \Delta$ and $g(x) = \prod_{h(x)\in\Delta} h(x)$ it follows that $\hat{h}(x)$ divides $g(x)$. Furthermore, this means that $\deg(\hat{h}) \leq \deg(g)$, and so $\deg(\hat{h}) < n$. It follows that there exists a prime p dividing n such that $\deg(\hat{h})$ divides n/p. It follows from Theorem 1 that $\gcd(f(x), x^{q^{n/p}} - x) \neq 1$. □

5. Algorithms for Testing Irreducibility

In this section we introduce three algorithms for testing polynomials for irreducibility.

The first algorithm is attributed to Ben-Or and is based on Theorem 2. In order to implement Theorem 2, we need to compute $\gcd(f(x), x^{q^i} - x)$ for $i \in \{1, 2, \ldots, k\}$. In order to do this efficiently, we do our operations modulo $f(x)$.

Since $x^{q^i} - x$ can be very large, it is not practical to compute directly. However, by the Euclidean algorithm, we have $\gcd(f(x), x^{q^i} -$

$x) = \gcd(f(x), x^{q^i} - x \bmod f(x))$. From properties of mod, we have $\gcd(f(x), x^{q^i} - x \bmod f(x)) = \gcd(f(x), \left(x^{q^i} \bmod f(x)\right) - x)$.

Notice that we can compute $x^{q^i} \bmod f(x)$ from $x^{q^{i-1}} \bmod f(x)$ by taking the q-th modular power of $x^{q^{i-1}}$, i.e. $x^{q^i} \bmod f(x) = (x^{q^{i-1}} \bmod f(x))^q \bmod f(x)$.

The Ben-Or algorithm has a single loop which both builds up $x^{q^i} \bmod f(x)$ and computes $\gcd(f(x), x^{q^i} \bmod f(x) - x)$ for $i \in \{1, 2, \ldots, \lfloor \frac{n}{2} \rfloor\}$.

Algorithm 1: Ben-Or Algorithm

```
1  r(x) ← x                                   /* r(x) = x^{q^d} mod f(x) */
2  for d ← 1 to ⌊n/2⌋ do
3  |   r(x) ← r(x)^q mod f(x)
4  |   if gcd(f(x), r(x) − x) ≠ 1 then
5  |   |   return reducible
6  return irreducible
```

The second algorithm can be attributed to Rabin [17] and is based on Theorem 3. The Rabin algorithm is similar to the Ben-Or algorithm in that it uses a single loop to both build up $x^{q^{n/p}} \bmod f(x)$ and computes $\gcd(f(x), x^{q^{n/p}} \bmod f(x) - x) = 1$ for all prime p dividing n.

Algorithm 2: Rabin Algorithm

```
1  r(x) ← x                                   /* r(x) = x^{q^{n/p_d}} mod f(x) */
2  compute k distinct prime factors p_d of n in decreasing order
3  for d ← 1 to k do
4  |   r(x) ← r(x)^{q^{n/p_d} − n/p_{d−1}} mod f(x)
5  |   if gcd(f(x), r(x) − x) ≠ 1 then
6  |   |   return reducible
7  r(x) ← r(x)^{q^{n − n/p_k}} mod f(x)
8  if r(x) ≠ x then
9  |   return reducible
10 return irreducible
```

In order to analyze the Ben-Or and Rabin algorithms we need to state the cost of our operations. The operations that we need are q-th modular power, modular multiplication, and GCD. The input to these

TABLE 1. Cost of basic operations

Operation	$\mathbb{F}_q$
Modular Subtraction	$O(n)$
q-th Modular Power	$O(\mathrm{M}(n)(\log q))$
Modular Multiplication	$O(\mathrm{M}(n))$
GCD	$O(\mathrm{M}(n)\log n)$

operations are polynomials of degree less than or equal to n. The cost of these operations are in terms of arithmetic operations over $\mathbb{F}_q$ and summarized in Table 1.

Many of the operations in Table 1 are in terms of $M(n)$, the cost of multiplying two polynomials of degree less than or equal to n. The classical algorithm has $M(n) = O(n^2)$. Another algorithm due to Karatsuba has $M(n) = O(n^{\log_2 3})$. Finally, an algorithm due to Schönage has $M(n) = O(n \log n \log\log n)$. A good reference for these algorithms can be found in [10].

Over $\mathbb{F}_2$, Table 1 implies that the cost of 2nd modular power (modular squaring) is $O(\mathrm{M}(n))$. However, when reducing modulo a sparse polynomial, the cost becomes $O(n)$ [4].

We now present the complexities of the Ben-Or and Rabin algorithms. We first focus on the case of general polynomials over $\mathbb{F}_q$, and later focus on sparse polynomials over $\mathbb{F}_2$.

The worst and average case complexities of the Ben-Or are given below in Theorem 4. The worst case complexity can be found [9]. The average case complexity is due to Panario and Richmond [16].

Theorem 4. *Over $\mathbb{F}_q$, the Ben-Or algorithm has worst case complexity $O(n\,\mathrm{M}(n)(\log nq))$ and average case complexity $O((\log n)\,\mathrm{M}(n)(\log nq))$.*

The worst and average case analyses of the Rabin algorithm are given in Theorem 5. The worst case complexity can be found in [9]. The average case complexity is due to Panario et al. [15].

Theorem 5. *Over $\mathbb{F}_q$ the Rabin algorithm has worst case and average case complexity $O(n\,\mathrm{M}(n)(\log q))$. However, over $\mathbb{F}_2$, the Rabin algorithm on sparse polynomials has worst case complexity $O(n^2)$.*

If we fix q, then theorems 4 and 5 state that the Ben-Or and Rabin algorithms have $\log n$ factor difference in worst case complexity, but that the Ben-Or algorithm has a better average case complexity. However, when focusing on $\mathbb{F}_2$, the Rabin algorithm has a better worst case complexity.

The reason that the Ben-Or algorithm has a better average case complexity is that on average, polynomials have factors of degree less than $\log n$. Remember that the Ben-Or algorithm loops through each degree from 1 up to $\lfloor \frac{n}{2} \rfloor$ testing for factors. If a polynomial has a factor of degree less than $\log n$, then the Ben-Or algorithm will detect it in only $\log n$ iterations of its loop.

For the rest of this paper, let us focus on testing trinomials over $\mathbb{F}_2$. The average case complexity of the Ben-Or algorithm given in Theorem 4 is no longer applicable since we are not testing a general polynomial over $\mathbb{F}_q$. It is however conjectured that the Ben-Or algorithm has a better average case complexity than the Rabin algorithm when testing trinomials over $\mathbb{F}_2$. This is supported by the results given in Section 7. The Rabin algorithm has been proven to have a better worst case complexity. It is possible to mix these two algorithms to achieve the Ben-Or's average case complexity and the Rabin's worst case complexity. The idea of mixing these two algorithms has been used by several authors in testing special trinomials over $\mathbb{F}_2$ called primitive trinomials [4]. We will take the suggestion of [4] and refer to it as the Standard algorithm.

The basic idea of the Standard algorithm is to do the Ben-Or algorithm up until a bound β, and then switch over to the Rabin algorithm. Precisely, the Standard algorithm computes $\gcd(f(x), x^{2^i} \bmod f(x) - x)$ for $i \in \{1, 2, \dots \beta\}$. The algorithm continues by computing $\gcd(f(x), x^{2^{n/p}} \bmod f(x) - x)$ for all prime divisors p of n such that $n/p > \beta$ and checks to see if $x^{2^n} = x \bmod f(x)$.

We state the relationship between β and the worst case complexity of the Standard algorithm in Theorem 6. This result is probably known, but was not seen anywhere in the literature.

Theorem 6. *Over $\mathbb{F}_2$, the Standard algorithm on sparse polynomials has worse case complexity $O(\beta \, \mathrm{M}(n) \log n)$.*

Proof. The total number of squarings is n and the total number of GCDs is β. This gives a total cost of

$$n^2 + \beta \, \mathrm{M}(n) \log n = O(\beta \, \mathrm{M}(n) \log n)$$

arithmetic operations over $\mathbb{F}_2$. □

Notice that if $\beta = O(\frac{n^2}{\mathrm{M}(n) \log n})$ then the Standard algorithm has worst complexity $O(n^2)$, which is the same as the Rabin algorithm.

We will present empirical evidence in section 7 which suggests that $\beta = \frac{n}{\log n}$ is asymptotically best on average. Notice that by Theorem 6 this choice of β does not yield the optimal worst case complexity.

Algorithm 3: Standard Algorithm

```
1  r(x) ← x                                   /* r(x) = x^{q^d} mod f(x) */
2  for 1 ← d to β do
3  |   r(x) ← r(x)^q mod f(x)
4  |   if gcd(f(x), r(x) − x) ≠ 1 then
5  |   |   return reducible
6  Compute k distinct prime factors p_d of n in decreasing order with
   n/p_d > β
7  for d ← 1 to k do
8  |   r(x) ← r(x)^{q^{n/p_d − n/p_{d−1}}} mod f(x)
9  |   if gcd(f(x), r(x) − x) ≠ 1 then
10 |   |   return reducible
11 r(x) ← r(x)^{q^{n − n/p_k}} mod f(x)
12 if r(x) ≠ x then
13 |   return reducible
14 return irreducible
```

6. Improvements

In this section we introduce several improvements which can be applied to the Ben-Or and Standard algorithms.

The first improvement, which we apply to the standard algorithm, will be referred to as outer-level blocking [5]. Let $\{I_1, I_2, \ldots, I_k\}$ partition the set $\{1, 2, \ldots, \beta\}$. Instead of computing $\gcd(f(x), x^{2^d} \bmod f(x) - x)$ for $d \in \{1, 2, \ldots, \beta\}$ we can compute $\gcd(f(x), \prod_{d \in I_j} x^{2^d} \bmod f(x) - x)$ for $j \in \{I_1, I_2, \ldots, I_k\}$ where each $\prod_{d \in I_j} x^{2^d} - x$ is known as an interval polynomial.

In this way we trade β GCDs for k GCDs and β multiplications. Choosing $k = 1$ yields the best worst case complexity, however empirically this is not best on average.

The authors of [22] prove that on average having the end points of intervals be the cubes of integers is optimal. This result is for general polynomials over $\mathbb{F}_q$. In Section 7, we give evidence to support that this result is also true for trinomials over $\mathbb{F}_2$.

The second improvement known as inner-level blocking which can be applied to the Ben-Or and Standard algorithms can be found [5]. The

basic idea is that each $\prod_{d \in I_j} x^{2^d} \bmod f(x) - x$ can be computed in such a way that we trade multiplications for squarings.

There are several other improvements that can be used along with any irreducibility test. One such improvement is often attributed to Swan [19]. Swan's Theorem states exact conditions on a trinomial over $\mathbb{F}_2$ having an even number of factors.

Theorem 7 (Swan's Theorem)**.** *Let $n > k > 0$. Assume exactly one of n, k is odd. Then $x^n + x^k + 1$ has an even number of factors (and hence is reducible) over $\mathbb{F}_2$ in the following cases.*

1. *n is even, k is odd, $n \neq 2k$, and $nk/2 \equiv 0$ or $1 \bmod 4$*
2. *n is odd, k is even, $k \nmid 2n$, and $n \equiv \pm 3 \bmod 8$*
3. *n is odd, k is even, $k \mid 2n$, and $n \equiv \pm 1 \bmod 8$*

In all other cases $x^n + x^k + 1$ has an odd number of factors over $\mathbb{F}_2$.

A corollary to this theorem is that 5/8 of all trinomials over the form $x^n + x^k + 1$ have an even number of irreducible factors. The proof is given in Appendix A. This means that adding Swan's theorem to a tabulation of irreducible trinomials, we are required to test only 3/8 of the trinomials on average.

Another improvement that can be applied to any irreducibility test is called the repeated factor test. The idea is that if a polynomial $f(x)$ has a repeated factor, then this factor can be found by computing the GCD of $f(x)$ and it's formal derivative.

7. Results

In this section we present timings that compare irreducibility tests. All tests were implemented in C++ using NTL, GF2X and GMP [18, 3, 8], and run on a 3.3 GHz Intel Core i5 2500 processor. They were complied using GNU Compiler Collection (GCC) version 4.1.2.

Each test in this section (except for the last) states the CPU time (in seconds) it took to test all trinomials of the form $x^n + x^k + 1$ for irreducibility where $n > k > 0$ and $n < m$. For each of the tables a "-" refers to a computation that timed out.

In Table 2 we compare different choices of β for the Standard algorithm. The choices for β where chosen to provide a variety of different asymptotic functions, and are by no means exhaustive. The choice of $\beta = n/(\log n)$ appears to be asymptotically best.

Next we compare different ways of partitioning for the Ben-Or algorithm. We write n^2 to mean that the end points determining the partition were squares of integers. Similarly, n^3 means that the end

TABLE 2. Average case timings for different β

m	$\log n$	$(\log n)^2$	$\sqrt{n}$	$n/(\log n)$
1000	20	12	12	14
2000	204	106	113	121
4000	2400	1096	1141	1157
8000	32508	18369	13369	12736

TABLE 3. Average case timings for different partitions

m	n	n^2	n^3	n^4
1000	34	11	12	20
2000	402	118	119	198
4000	-	1251	1286	2108
8000	-	18277	14470	-

TABLE 4. Average case timings for irreducibility algorithms

m	Ben-Or	Rabin	Standard
1000	12	981	14
2000	119	9199	121
4000	1286	-	1157
8000	14470	-	12736

points were cubes of integers. The results are given in Table 3. Having the endpoints be cubes of integers was optimal in these tests. Note that is in agreement with [22].

In Table 4 we compare the Ben-Or, Rabin, and Standard algorithms. The Ben-Or and Standard algorithms both used the improvements described in Section 6. The Rabin algorithm is by far the worst on average. The Standard algorithm performs slightly better than the Ben-Or algorithm.

We finally compare the Ben-Or and Standard algorithms in the worst case, i.e. on irreducible polynomials. The results are given in Table 5. Again, both algorithms used the improvements described in Section 6. In this test, the timings are the CPU seconds required to test a single irreducible trinomial for irreducibility. We denote the degree of this polynomial by m.

TABLE 5. Worst case timings for irreducibility algorithms

m	Ben-Or	Rabin	Standard
20001	3	1	0
40001	26	0	2
80001	126	3	4

8. Conclusion

We have presented three algorithms for testing trinomials for irreducibility over $\mathbb{F}_2$ and their analyses. We have done experiments suggesting optimal parameters to these algorithms, and experiments comparing these three algorithms. We also have given a new theoretical result which is a corollary to Swan's theorem (see Appendix A).

9. Acknowledgements

I would like to thank Andrew Shallue for the immense amount of time he spent advising me on this project. I would like to thank Mark Liffiton for his helpful suggestions. Finally, I would also like to thank Andrew Shallue and Mark Liffiton for allowing me to use their computing resources, which were funded by Illinois Wesleyan start up funds.

References

[1] Ian F. Blake, Shuhong Gao, and Robert J. Lambert. Construction and distribution problems for irreducible trinomials over finite fields. In *Applications of finite fields (Egham, 1994)*, volume 59 of *Inst. Math. Appl. Conf. Ser. New Ser.*, pages 19–32. Oxford Univ. Press, New York, 1996.

[2] Antonia W. Bluher. A Swan-like theorem. *Finite Fields Appl.*, 12(1):128–138, 2006.

[3] R. Brent, P. Gaudry, E. Thomé, and P. Zimmermann. gf2x, 2008.

[4] Richard P. Brent, Samuli Larvala, and Paul Zimmermann. A fast algorithm for testing reducibility of trinomials mod 2 and some new primitive trinomials of degree 3021377. *Mathematics of Computation*, 72(243):1443–1452, 2003.

[5] Richard P. Brent and Paul Zimmermann. A multi-level blocking distinct-degree factorization algorithm. In *Finite fields and applications*, volume 461 of *Contemp. Math.*, pages 47–58. Amer. Math. Soc., Providence, RI, 2008.

[6] Richard P. Brent and Paul Zimmermann. The great trinomial hunt. *Notices Amer. Math. Soc.*, 58(2):233–239, 2011.

[7] Gove Effinger. Toward a complete twin primes theorem for polynomials over finite fields. In *Finite fields and applications*, volume 461 of *Contemp. Math.*, pages 103–110. Amer. Math. Soc., Providence, RI, 2008.

[8] Torbjrn Granlund et al. GNU multiple precision arithmetic library 4.1.2, December 2002. `http://swox.com/gmp/`.

[9] Shuhong Gao and Daniel Panario. Tests and constructions of irreducible polynomials over finite fields. In *Foundations of Computational Mathematics*, pages 346–361. Springer, Berlin, 1997.
[10] Joachim Von Zur Gathen and Jurgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.
[11] B. Hanson, D. Panario, and D. Thomson. Swan-like results for binomials and trinomials over finite fields of odd characteristic. *Des. Codes Cryptogr.*, 61(3):273–283, 2011.
[12] D. R. Heath-Brown. The distribution and moments of the error term in the Dirichlet divisor problem. *Acta Arith.*, 60(4):389–415, 1992.
[13] Ryul Kim and Wolfram Koepf. Parity of the number of irreducible factors for composite polynomials. *Finite Fields Appl.*, 16(3):137–143, 2010.
[14] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, Cambridge, 1994.
[15] Daniel Panario, Boris Pittel, Bruce Richmond, and Alfredo Viola. Analysis of rabin's irreducibility test for polynomials over finite fields. *Random Structures and Algorithms*, 19:525–551, October 2001.
[16] Daniel Panario and Bruce Richmond. Analysis of ben-or's polynomial irreducibility test. In *proceedings of the eighth international conference on Random structures and algorithms*, pages 439–456, New York, NY, USA, 1998. John Wiley & Sons, Inc.
[17] M.O. Rabin. Probabilistic algorithms in finite fields. In *SIAM J. Comput*, 1979.
[18] Victor Shoup. NTL: A library for doing number theory. `http://www.shoup.net/ntl`, 2003.
[19] Richard G. Swan. Factorization of polynomials over finite fields. *Pacific J. Math.*, 12:1099–1106, 1962.
[20] Uzi Vishne. Factorization of trinomials over Galois fields of characteristic 2. *Finite Fields Appl.*, 3(4):370–377, 1997.
[21] Joachim von zur Gathen. Irreducible trinomials over finite fields. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, ISSAC '01, pages 332–336, New York, NY, USA, 2001. ACM.
[22] Joachim vonzur Gathen, Daniel Panario, and Bruce Richmond. Interval partitions and polynomial factorization. *Algorithmica*, pages 1–35, 2011.

Appendix A. Corollary to Swan's Theorem

In 1962, Richard G. Swan proved the following theorem, which states exact conditions of n, k for $T_{n,k} = x^n + x^k + 1$ to have an even number of irreducible factors [19]. A polynomial with an even number of irreducible factors is reducible, and so Swan's Theorem has application in testing trinomials for irreducibility over $GF(2)$ [6]. A consequence of this theorem is that the probability of $T_{n,k}$ having an even number of irreducible factors is asymptotically $5/8$ as $n \to \infty$. Thus, adding Swan's theorem to an irreducibility test results in a speed up of $8/3$.

Theorem 1 (Swan's Theorem)**.** *Let $n > k > 0$. Assume exactly one of n, k is odd. Then $x^n + x^k + 1$ has an even number of factors (and hence is reducible) over $GF(2)$ in the following cases.*

(1) *n is even, k is odd, $n \neq 2k$, and $nk/2 \equiv 0$ or $1 \bmod 4$*
(2) *n is odd, k is even, $k \nmid 2n$, and $n \equiv \pm 3 \bmod 8$*
(3) *n is odd, k is even, $k \mid 2n$, and $n \equiv \pm 1 \bmod 8$*

In all other cases $x^n + x^k + 1$ has an odd number of factors over $GF(2)$.

In order to prove the probability of $T_{n,k}$ having an even number of irreducible factors, we will consider all the trinomials of the form $x^n + x^k + 1$ where $n < m$, and count the number with an even number of irreducible factors. In order to make the counting process easier, we prove the following proposition which maps case (1) to cases (1.a), (1.b), and (1.c).

Proposition 1. We have that n, k satisfy case (1) of Swan's Theorem if and only if n, k satisfy one of the following cases:

(1.a) $n \equiv 0 \bmod 8$ and k is odd
(1.b) $n \equiv 2 \bmod 8$, $k \equiv 1 \bmod 4$, and $n \neq 2k$
(1.c) $n \equiv 6 \bmod 8$, $k \equiv 3 \bmod 4$, and $n \neq 2k$

Proof. Suppose n, k satisfies (1) and does not satisfy (1.b) or (1.c). Since n is even then $n \equiv 0, 2, 4$ or $6 \bmod 8$. We prove that $n \not\equiv 2, 4, 6 \bmod 8$ and so $n \equiv 0 \bmod 8$.

Suppose that $n \equiv 2 \bmod 8$. Since k is odd, and $k \not\equiv 1 \bmod 4$ then $k \equiv 3 \bmod 4$. Then $n = 8m + 2$ and $k = 4l + 3$ where $m, l \in \mathbb{Z}$.

$$nk/2 = (8m + 2)(4l + 3)/2 = 4(4ml + l + 3m) + 3 \equiv 3 \bmod 4.$$

This contradicts (1) so $n \not\equiv 2 \bmod 8$.

Suppose that $n \equiv 4 \bmod 8$. Then $n = 8m + 4$ and $k = 2l + 1$ where $m, l \in \mathbb{Z}$. Then we have

$$nk/2 = (8m + 4)(2l + 1)/2 = 4(2ml + l + m) + 2 \equiv 2 \bmod 4.$$

This contradicts (1) so $n \not\equiv 4 \bmod 8$.

Suppose that $n \equiv 6 \bmod 8$. Since k is odd, and $k \not\equiv 3 \bmod 4$ then $k \equiv 1 \bmod 4$. Then $n = 8m + 6$ and $k = 4l + 1$ where $m, l \in \mathbb{Z}$. Then we have

$$nk/2 = (8m + 6)(4l + 1)/2 = 4(4ml + 3l + m) + 3 \equiv 3 \bmod 4.$$

This contradicts (1) so $n \not\equiv 6 \bmod 8$.

Now suppose n, k satisfies one of (1.a), (1.b), and (1.c). If n, k satisfy (1.a) then $n = 8$ and $k = 2l + 1$ where $m, l \in \mathbb{Z}$. Then we have

$$nk/2 = 8m(2l + 1)/2 = 4m(2k + 1) \equiv 0 \bmod 4.$$

If n, k satisfy (1.b) then $n = 8m + 2$ and $k = 2l + 1$ where $m, l \in \mathbb{Z}$. Then we have

$$nk/2 = (8m + 2)(4l + 1)/2 = 4(ml + m + l) + 1 \equiv 1 \bmod 4.$$

If n, k satisfy (1.c) then $n = 8m + 6$ and $k = 4l + 3$ where $m, l \in \mathbb{Z}$. Then we have

$$nk/2 = (8m + 6)(4l + 3)/2 = 4(ml + 3m + 3l + 2) + 1 \equiv 1 \bmod 4.$$

□

We also need to know the number of factors of $T_{n,k}$ where n, k are both even or both odd. When n, k are both even, we have that

$$T_{n,k} = x^n + x^k + 1 = (x^{n/2} + x^{k/2} + 1)^2$$

and thus has an even number of irreducible factors. When n, k are both odd we use the fact that a polynomial and its reciprocal polynomial have the same number of irreducible factors. Notice that when n, k are both odd, then n is odd and $n - k$ is even. Furthermore, $T_{n,n-k}$ is the reciprocal polynomial of $T_{n,k}$. Thus we can apply Swan's theorem to $T_{n,n-k}$. In order to count the number of $T_{n,k}$ that satisfy Swan's Theorem we double the number of $T_{n,k}$ that satisfy (2) or (3).

We are now ready to prove that the probability of $T_{n,k}$ having an even number of irreducible factors is asymptotically 5/8.

Corollary 1. *Let $n > k > 0$. Let $P(m)$ be the probability that $x^n + x^k + 1$ where $n < m$ has an even number of irreducible factors over $GF(2)$. Then as $m \to \infty$ we have that $P(m) \to 5/8$.*

Proof. The total number of trinomials $T_{n,k}$ where $n < m$ is

$$\sum_{n=2}^{m-1} \sum_{k=1}^{n-1} 1 = \sum_{n=2}^{m-1} (n - 1)$$

$$= \frac{(m-1)m}{2} - (m - 1) = \frac{m^2}{2} + \Theta(m).$$

The total number of trinomials $T_{n,k}$ where n, k are both even is

$$\sum_{\substack{2 \le n < m \\ n \equiv 0 \bmod 2}} \sum_{\substack{k=2 \\ k \equiv 0 \bmod 2}}^{n-2} 1 = \sum_{\substack{2 \le n < m \\ n \equiv 0 \bmod 2}} \left(\frac{n}{2} - 1\right) = \sum_{l=1}^{m/2} l + O(m)$$

$$= \frac{\frac{m}{2}(\frac{m}{2} + 1)}{2} + O(m) = \frac{m^2}{8} + O(m).$$

The total number of trinomials $T_{n,k}$ where n, k satisfy (1.a) is

$$\sum_{\substack{2 \le n < m \\ n \equiv 0 \bmod 8}} \sum_{\substack{k=1 \\ k \equiv 1 \bmod 2}}^{n-1} 1 = \sum_{\substack{2 \le n < m \\ n \equiv 0 \bmod 8}} \frac{n}{2} = \sum_{l=1}^{m/8} 4l + O(m)$$

$$= 4\frac{\frac{m}{8}(\frac{m}{8}+1)}{2} + O(m) = \frac{m^2}{32} + O(m).$$

The total number of trinomials $T_{n,k}$ where n, k satisfy (1.b) is

$$\sum_{\substack{2\le n<m \\ n\equiv 2 \bmod 8}} \sum_{\substack{1\le k<n \\ k\equiv 1 \bmod 4 \\ n\neq 2k}} 1 = \sum_{\substack{2\le n<m \\ n\equiv 2 \bmod 8}} \left(\frac{n}{4} + O(1)\right) = \sum_{l=1}^{m/8} 2l + O(m)$$

$$= \frac{m}{8}\left(\frac{m}{8}+1\right) + O(m) = \frac{m^2}{64} + O(m).$$

The total number of trinomials $T_{n,k}$ where n, k satisfy (1.c) is

$$\sum_{\substack{2\le n<m \\ n\equiv 6 \bmod 8}} \sum_{\substack{1\le k<n \\ k\equiv 3 \bmod 4 \\ n\neq 2k}} 1 = \sum_{\substack{2\le n<m \\ n\equiv 6 \bmod 8}} \left(\frac{n}{4} + O(1)\right) = \frac{m^2}{64} + O(m).$$

The total number of trinomials $T_{n,k}$ where n, k satisfy (2) is

$$\sum_{\substack{2\le n<m \\ n\equiv \pm 3 \bmod 8}} \sum_{\substack{k=2 \\ k\equiv 0 \bmod 2 \\ k\nmid 2n}}^{n-1} 1 = \sum_{\substack{2\le n<m \\ n\equiv \pm 3 \bmod 8}} \left(\sum_{\substack{k=2 \\ k\equiv 0 \bmod 2}}^{n-1} 1 - \sum_{\substack{k=2 \\ k\equiv 0 \bmod 2 \\ k\mid 2n}}^{n-1} 1 \right)$$

$$= \sum_{\substack{2\le n<m \\ n\equiv \pm 3 \bmod 8}} \left(\frac{n-1}{2} - (d(n)-1)\right)$$

Using the fact that $\sum_{n=1}^{m} d(n) = m\log m + O(m)$ [12],

$$= 2\sum_{l=1}^{m/8} 4l + O(m\log m) = 4\left(\frac{m}{8}\left(\frac{m}{8}+1\right)\right) + O(m\log m)$$

$$= \frac{m^2}{16} + O(m\log m)$$

The total number of trinomials $T_{n,k}$ where n, k satisfy (3) is

$$\sum_{\substack{2\le n<8 \\ n\equiv \pm 1 \bmod 8}} \sum_{\substack{k=2 \\ k\equiv 0 \bmod 2 \\ k\mid 2n}}^{n-1} 1 = \sum_{\substack{2\le n<8 \\ n\equiv \pm 1 \bmod 8}} (d(n)-1) = O(m\log m)$$

Then we have that

$$P(m) = \frac{(\frac{1}{8}+\frac{1}{32}+\frac{1}{64}+\frac{1}{64}+\frac{2}{16})m^2 + O(m\log m)}{\frac{m^2}{2} + \Theta(m)}$$

$$= 5/8 + O\left(\frac{\log m}{m}\right).$$

Furthermore, as $m \to \infty$ we have that $P(m) \to 5/8$. □

There are many Swans inspired theorem for special classes of polynomials over finite fields [20, 2, 13, 21, 11]. A good summary can be found in [11]. It is likely that similar techniques could be used to find the probability of those special classes having an even number of irreducible factors.