TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Properties of Primitive Roots of Prime Numbers and Applications

by Ioannis Konidakis

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DIPLOMA DEGREE OF

ELECTRICAL AND COMPUTER ENGINEERING

December, 2022

THESIS COMMITTEE
Professor George N. Karystinos, *Thesis Supervisor*
Professor Aggelos Bletsas
Associate Professor Vasileios Samoladas

Abstract

In this work we derive a few properties of primitive roots of prime numbers and use them to generate an algorithm that functions as a Lehmer pseudo-random number generator, as well as a solution to the discrete logarithm problem. Specifically, we capitalize patterns in the exponentiation of primitive roots to achieve the calculation of the primitive root's power sequence by mainly using additions instead of multiplications. Our new generated algorithm has three different forms, their difference lying on their initialization step. We test our algorithm as a Lehmer pseudo-random number generator against a multiplication-based brute-force algorithm, achieving satisfying results for all three initialization forms. Finally, we use our algorithm to solve the discrete logarithm problem, testing it against Shanks' Baby Step - Giant Step algorithm. The results show that Shanks' algorithm, even for small numbers, achieves a lower cost than all forms of our algorithm. However, our algorithm could be implemented in Baby Step - Giant Step to potentially reduce its total cost.

Acknowledgements

First of all, I would like to thank my parents for their constant support throughout the course of my studies.
I would like to thank my thesis supervisor, professor George N. Karystinos, as this thesis would not exist without his encouragement to pursue my idea. I would also like to thank the rest of my thesis committee for taking the time to review my work, especially associate professor Vasileios Samoladas, who at times provided new perspective and ideas for the potential applications of this thesis.
Finally, I would like to thank my friends, who supported me through all the ups and downs of my life, sharing moments and experiences that shaped me as a human being.

In the loving memory of my late cousin, Katerina.

Contents

Chapter 1

Introduction

Primitive roots of prime numbers have been used in many applications in the field modular arithmetic, as well as cryptography. We aspire to create a Lehmer (pseudo-) random number generator, following the sequence

$$X_{k+1} \equiv a \cdot X_k \pmod{m},$$

where $m \in \mathbb{P}$ and a is a primitive root of m.
To do so, we tried to find patterns in the cyclic sequence of primitive root generators $\langle a \rangle$ in $GF(m^i)$ which would reduce the number of multiplications used for the calculation, replacing them with additions. In our efforts, we managed to find some interesting properties which we collocate in this thesis.
In addition, we tried to apply the same technique on the discrete logarithm problem. This problem's cracking complexity is widely used in cryptography, with its most well-known application being W. Diffie's and M. Hellman's public key cryptography protocol [1][2], a method currently used in a variety of Internet services and which later on inspired other important algorithms such as ElGamal[3], RSA[4] etc.
Ever since, many algorithms have emerged for solving the discrete logarithm problem, but while some algorithms achieve a sub-exponential complexity, it remains a calculation that cannot be done in real time.
Nowadays, modern research has turned on quantum cryptography, which is capable of making such "impossible" calculations in real time (e.g. Shor's algorithm[5]).

Outline:

We will firstly present some theorems and definitions which are prerequisites for the understanding of our thesis. Consequently, we will develop the observations we made, firstly on primitive roots of prime numbers in finite fields $\mathbb{Z}_p$, and then their generalized version on primitive elements of Galois fields $GF(p^m)$. Afterwards, we will describe our algorithm with its 3 different forms:

- Simple Form (S.F.)
- Generalized Form (G.F.)
- Enhanced 3-term Form (E3TF)

Finally, we will suggest 2 applications for our algorithm. Firstly, as a congruential pseudo-random number generator, and secondly, as an algorithm for computing discrete logarithms in Finite Fields. We will present the efficiency of our algorithm, both in theory and in practice and draw final conclusions.

Chapter 2

A Number-Theoretic Prelude

In this chapter, we will present some well known theorems and definitions used in Number Theory, to set some common ground for the disposition of this thesis. The proof of our statements is omitted as they are considered fundamental [6].

2.1 Division

Theorem 2.1.1 (Division Algorithm) *Let $a \in \mathbb{Z}$, $n > 0$, then $\exists$ unique $q \in \mathbb{Z}$ and $r \in \mathbb{Z}_n$, such that*

$$a = qn + r$$

where q is the quotient and r the remainder of dividing a with n.
Note that when $a \geq 0$ then $q \geq 0$, and when $a < 0$ then $q < 0$.

Definition 2.1.1 (Remainder Operator) *For any $a \in \mathbb{Z}$ and $n \neq 0$, we define $r_n[a]$ as the remainder of the division of a with n.*

$$a = qn + r_n[a], \text{ where } r_n[a] \in \mathbb{Z}_{|n|}.$$

Theorem 2.1.2 *If $n \neq 0$, then for all $a, b \in \mathbb{Z}$:*

1. $r_n[an] = 0$
2. $r_n[a + b] = r_n[r_n[a] + b] = r_n[r_n[a] + r_n[b]]$
3. $r_n[ab] = r_n[r_n[a]b] = r_n[r_n[a]r_n[b]]$

Definition 2.1.2 (Divisibility Operator) *For any $a \neq 0$ and $b \in \mathbb{Z}$, if $r_a[b] = 0$ then a divides b, which can be symbolized as $a|b$. If $r_a[b] \neq 0$, or a does not divide b, then we write $a \nmid b$.*

Definition 2.1.3 (Common Divisor) *$d \neq 0$ is a **common divisor** of $a, b \in \mathbb{Z}$ if $d|a$ and $d|b$.*

If $a \neq 0$ or $b \neq 0$, then the set $S = \{d > 0 : \ d|a \text{ and } d|b\}$ there exists a greatest element of S.

Definition 2.1.4 (Greatest Common Divisor (GCD)) *If $a \neq 0$ or $b \neq 0$, then their greatest common divisor is*

$$(a, b) = max\{d > 0 : \ d|a \text{ and } d|b\}.$$

Definition 2.1.5 *Integers a, b are relatively prime if $(a, b) = 1$.*

2.2 Congruence

Definition 2.2.1 *Given an integer $n > 0$, a and b are said to be congruent modulo n, if n divides $a - b$, or*

$$a \equiv b \ (mod\ n) \Leftrightarrow n|a - b.$$

Theorem 2.2.1 (Congruence Properties) *The following are true.*

1. $a = b \Leftrightarrow a \equiv b \ (mod\ n), \forall n > 0$
2. $a \equiv 0 \ (mod\ n) \Leftrightarrow n|a$
3. $a \equiv a \ (mod\ n)$
4. $a \equiv b \ (mod\ n) \Leftrightarrow b \equiv a \ (mod\ n)$
5. $a \equiv b \ (mod\ n)$ *and* $b \equiv c \ (mod\ n) \Rightarrow a \equiv c \ (mod\ n)$
6. $a \equiv r_n[a] \ (mod\ n)$
7. $a \equiv b \ (mod\ n) \Leftrightarrow r_n[a] = r_n[b]$
8. $a \equiv b \ (mod\ n) \Rightarrow a + c \equiv b + c \ (mod\ n)$
9. $a \equiv b \ (mod\ n) \Rightarrow ax \equiv bx \ (mod\ n)$
10. $a \equiv b \ (mod\ n) \Rightarrow a^k \equiv b^k \ (mod\ n), \forall k > 0$
11. $ac \equiv bc \ (mod\ n)$ *and* $(c, n) = 1 \Rightarrow a \equiv b \ (mod\ n)$

Definition 2.2.2 (Linear Congruence) *The relation $ax \equiv b \ (mod\ n)$, where x is unknown, is called linear congruence relation.*

Theorem 2.2.2 *Let $ax \equiv b \ (mod\ n)$.*

1. *The equation has a solution if and only if $(a, n)|b$*
2. *If $(a, n)|b$, then the equation has (a, n) non-congruent solutions modulo n.*

Corollary 2.2.2.1 *If $(a, n) = 1$, then the linear congruent equation $ax \equiv b \ (mod\ n)$ has a unique solution modulo n.*

2.3 Prime Numbers

Definition 2.3.1 (Prime Number) *A natural number $p > 1$ is called prime if its only divisors are 1 and p. In any other case (if it is not prime), it is called composite number.*

Definition 2.3.2 *A common symbolism for the set of all prime numbers is $\mathbb{P}$.*

Theorem 2.3.1 $p \in \mathbb{P},\ p|ab \Rightarrow p|a$ *or* $p|b$.

Theorem 2.3.2 (Fundamental Theorem of Arithmetic) *Any $n > 1$ can be represented uniquely as a product of prime numbers.*

Definition 2.3.3 (Euler's ϕ Function) *We define Euler's function as* $\phi : \mathbb{N}^* \to \mathbb{R}$, *with*

$$\phi(n) = |\mathcal{T}_n|, \text{ where } \mathcal{T}_n = \{m \in \{1, 2, \ldots, n\} \ : \ (m, n) = 1\}, \ n \geq 1.$$

Corollary 2.3.2.1 *For any* $n \geq 1$, *there exist exactly* $\phi(n)$ *elements* $x \in \mathbb{Z}_n$, *such that* $(x, n) = 1$.

Corollary 2.3.2.2 *We can deduce*

$$p \in \mathbb{P} \Leftrightarrow \phi(p) = p - 1$$
$$p \notin \mathbb{P} \Leftrightarrow \phi(p) < p - 1.$$

Theorem 2.3.3 (Euler's Theorem) $n \geq 1$, $(a, n) = 1 \Rightarrow a^{\phi(n)} \equiv 1 \ (mod\ n)$.

Theorem 2.3.4 *Fermat's little theorem states that if* p *is a prime number, then for any integer* a, *the number* $a^p - a$ *is an integer multiple of* p.

$$a^p \equiv a \ (mod\ p)$$

In the special case that a *is not divisible by* p, *Fermat's little theorem is equivalent to the statement that*

$$a^{p-1} \equiv 1 \ (mod\ p),$$

which we can also see is a special case of Euler's theorem.

2.4 Order and Primitive Roots

Definition 2.4.1 (Order) *We define order of an integer* a *modulo* $n > 1$ *as the smallest* $k > 0$ *such that* $a^k \equiv 1 \ (mod\ n)$, *and we write* $order_n(a)$.

Lemma 2.4.1 *Order of an integer* a *modulo* $n > 1$ *is defined if and only if* $(a, n) = 1$.

Theorem 2.4.2 *If* $order_n(a) = k$ *and* $h \geq 0$, *then*

1. $b \equiv a \ (mod\ n) \Rightarrow order_n(b) = k$
2. $a^h \equiv 1 \ (mod\ n) \Leftrightarrow k | h$
3. $k | \phi(n)$
4. $\forall i, j \in \mathbb{N}, \ a^i \equiv a^j \ (mod\ n) \Leftrightarrow i \equiv j \ (mod\ k)$
5. $\forall i, j \in \ \{1, 2, \ldots, k\}, \ i \neq j \Rightarrow a^i \not\equiv a^j \ (mod\ n)$

Definition 2.4.2 (Primitive Root) *If* $order_n(a) = \phi(n)$, *then* a *is a primitive root of* n.

Theorem 2.4.3 *Let* $n > 1$, *with* $\mathcal{T}_n = \{a_1, a_2, \ldots, a_{\phi(n)}\}$, *and* $a \in \mathbb{Z}$ *a primitive root of* n. *Then,*

$$\mathcal{T}_n = \{r_n[a], r_n[a^2], \ldots, r_n[a^{\phi(n)}]\}.$$

Corollary 2.4.3.1 *Let $n > 1$ and $a \in \mathbb{Z}$ a primitive root of n. Then n has $\phi(\phi(n))$ primitive roots in $\mathbb{Z}_n$, which form the set $\{r_n[a^i] : \ i \in \mathcal{T}_{\phi(n)}\}$.*

Theorem 2.4.4 (Primitive Roots of Prime Numbers) *Let $p \in \mathbb{P}$, then:*

1. *$p \not| c$ and $ac \equiv bc \ (mod\ p) \Rightarrow a \equiv b \ (mod\ p)$*
2. *$ab \equiv 0 \ (mod\ p) \Rightarrow a \equiv 0 \ (mod\ p)$ or $b \equiv 0 \ (mod\ p)$*

Theorem 2.4.5 *If $p \in \mathbb{P}, d > 0$ and $d|p-1$, then there exist exactly $\phi(d)$ elements $x \in \mathbb{Z}_p$, such that $order_p(x) = d$.*

Corollary 2.4.5.1 *$p \in \mathbb{P}$ has exactly $\phi(p-1)$ primitive roots in $\mathbb{Z}_p$.*

Corollary 2.4.5.2 *If $p \in \mathbb{P}, d > 0$ and $d|p-1$ then the polynomial congruence $x^d - 1 \equiv 0 \ (mod\ p)$ has exactly d solutions in $\mathbb{Z}_p$.*

2.5 Algebraic Structures

Definition 2.5.1 (Group) *A group $< G, * >$ is the set G together with a closed binary operation $*$, such that the following requirements (group axioms) are satisfied.*

1. *$(a * b) * c = a * (b * c)$ (Associativity)*
2. *$\exists e \in G$, such that $a * e = e * a = a, \ \forall a \in G$ (Identity element)*
3. *$\forall a \in G$, $\exists a' \in G$ such that $a' * a = a * a' = e$ (Inverse element)*

Definition 2.5.2 (Commutative or Abelian Group) *A group $< G, * >$ that also satisfies the commutative property,*

$$\forall a, b \in G, \ a * b = b * a$$

is called commutative group or abelian group.

Definition 2.5.3 *Let $< G, * >$ a group with $a \in G$. Then, $\forall n > 0$ we define*

$$a^n = \underbrace{a * a * a * \ldots * a}_{n}$$
$$a^{-n} = \underbrace{a' * a' * a' * \ldots * a'}_{n}$$
$$a^0 = e$$

Definition 2.5.4 (Subgroup) *Given a group G under a binary operation $*$, H is called a subgroup of G if it also forms a group under $*$ and $H \subseteq G$.*

Definition 2.5.5 *$< G, * >, 7a \in G$. Then, for $n \in \mathbb{Z}$, the set $< a >= \{a^n\} \subseteq G$ is a subgroup of G and it is called cyclic subgroup of G generated by a.*

Definition 2.5.6 *For $< G, * >$, if $\exists a \in G$ such that $G =< a >$, then G is called cyclic group.*

Definition 2.5.7 (Rings) *A set $< R, +, \cdot >$ with 2 closed binary operations is called a ring if it satisfies the following requirements:*

1. $< R, + >$ *is an abelian group (or commutative group)*

2. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ *(associative under multiplication)*

3. $a \cdot (b + c) = a \cdot b + a \cdot c$ *(multiplication is distributive)*

If $\forall a, b \in R,\ a \cdot b = b \cdot a$, *then the ring is commutative.*
If $\exists$ *element* $1 \in R$ *such that* $1 \cdot a = a \cdot 1 = a,\ \forall a \in R$, *then the ring has an identity element, and thus is a monoid under multiplication.*

Definition 2.5.8 (Integral Domain) *An integral domain* $< R, +, \cdot >$ *is a commutative ring with an identity element, such that* $\forall a, b \in R,\ a \cdot b = 0 \Rightarrow a = 0$ *or* $b = 0$.

Definition 2.5.9 (Field) *A field* $< F, +, \cdot >$ *is an integral domain* $< F, +, \cdot >$, *where* $\forall\ a \in F - \{0\},\ \exists a' \in F - \{0\}$, *such that* $a \cdot a' = 1$.
A field with a finite number of elements is called a finite field.

Definition 2.5.10 (Polynomial) *Let* $< R, +, \cdot >$ *a ring. A polynomial with coefficients in* R *is an expression*

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \leftrightarrow [a_0, a_1, a_2, \ldots, a_n],$$

where $a_i \in R,\ i = 0, 1, \ldots, n,\ n \geq 0$.
If $a_n = 1$, *the polynomial is called monic.*

Definition 2.5.11 (Polynomial Ring) *Let* $< R, +, \cdot >$ *a ring. Polynomial ring is the ring* $< R[x], \oplus, \odot >$, *where*

$$R[x] = \Big\{ \sum_{i=0}^{n} a_i x^i,\ a \in R,\ n = 0, 1, 2, \ldots \Big\}$$

Note that $\oplus, \odot$ *are scalar addition and multiplication.*

Corollary 2.5.0.1 *If* $< R, +, \cdot >$ *is an integral domain, then* $< R[x], \oplus, \odot >$ *is also an integral domain.*

Theorem 2.5.1 *Let* $< F, +, \cdot >$ *a field ,* $< F[x], \oplus, \odot >$ *an integral domain, and polynomials* $f(x), g(x) \in F[x]$, $g(x) \neq 0$. *Then, there exist unique polynomials* $q(x), r(x) \in R[x]$ *such that*

$$f(x) = q(x)g(x) + r(x) \text{ and}$$
$$deg(r(x)) < deg(g(x))$$

Definition 2.5.12 (Polynomial Remainder) *Let* $< F, +, \cdot >$ *a field and polynomials* $f(x), g(x) \in F[x]$, $g(x) \neq 0$. *Then,* $f(x) = q(x)g(x) + r(x)$. *We define* $r_g[f] = r$ *the remainder.*
Consequently, $g|f$ *if and only if* $r_g[f] = 0$.

Theorem 2.5.2

$$r_g[a \odot b] = r_g[r_g[a] \odot b] = r_g[a \odot r_g[b]] = r_g[r_g[a] \odot r_g[b]]$$
$$r_g[a \oplus b] = r_g[a \oplus r_g[b]] = r_g[r_g[a] \oplus b] = r_g[a] \oplus r_g[b]$$

Definition 2.5.13 (Ideal) *Let* $< R, +, \cdot >$ *a ring and* $I \subseteq R,\ I \neq 0$. I *is an ideal of* R *if*

1. $< I, + >$ *is a group*

2. $\forall\ a \in I$ *and* $r \in R$, $a \cdot r \in I$

Definition 2.5.14 (Quotient ring) *Let* $< F, +, \cdot >$ *a field,* $< F[x], +, \cdot >$ *an integral domain, and a polynomial* $f(x) \in F[x]$*, with* $deg(f(x)) > 0$*. Then, we define as quotient ring the set* $F[x]/f(x) = \{r(x) :\ deg(r(x)) < deg(f(x))\} \subseteq F[x]$*, with operations*

$$\oplus :\ r_1(x) \oplus r_2(x) = r_1(x) + r_2(x)$$
$$\odot :\ r_1(x) \odot r_2(x) = r_{f(x)}[r_1(x)r_2(x)]$$

Theorem 2.5.3 *The quotient ring* $< F[x]/f(x), \oplus, \odot >$ *is a commutative ring with an identity element.*

Definition 2.5.15 (GCD in Polynomials) *Let* F *a field and* $f(x), g(x) \in F[x]$ *with* $f(x) \neq 0$ *or* $g(x) \neq 0$*. The greatest common divisor of* $f(x)$ *and* $g(x)$ *is* $(f(x), g(x)) = d(x) \in F[x]$*, if and only if:*

1. $d(x)$ *is monic*
2. $d(x)|f(x)$ *and* $d(x)|g(x)$
3. *if* $c(x)|f(x)$ *and* $c(x)|g(x)$*, then* $deg(c(x)) \leq deg(d(x))$

Theorem 2.5.4 *Let* F *field. If* $f(x), g(x) \in F[x]$ *and* $f(x) \neq 0$ *or* $g(x) \neq 0$*, then:*

1. *There exists (f(x),g(x))*
2. *There exist (not unique) polynomials* $u(x)$ *and* $v(x) \in F[x]$*, such that* $(f(x), g(x)) = f(x)u(x) + g(x)v(x)$

Definition 2.5.16 (Relatively Prime Polynomials) *If* $(f(x), g(x)) = 1$*, then* $f(x), g(x)$ *are relatively prime.*

2.6 Galois Fields

Definition 2.6.1 *Let* F *a field and* $f(x) \in F[x]$*.* $f(x)$ *is irreducible over* $F[x]$ *if* $\nexists g(x), h(x) \in F[x]$*, with* $deg(g(x)) < deg(f(x))$ *and* $deg(h(x)) < deg(f(x))$*, such that*

$$f(x) = g(x) \cdot h(x).$$

Theorem 2.6.1 (Fundamental Theorem of Algebra (or d'Alembert–Gauss theorem)) *Every* $f(x) \in \mathbb{C}[x]$ *with* $deg(f(x)) \geq 2$ *is not irreducible over* $\mathbb{C}[x]$*.*

Theorem 2.6.2 (Creating Finite Fields) *Let* F *a field,* $f(x) \in F[x]$ *irreducible over* $F[x]$ *. Then, the quotient ring* $F[x]/f(x)$ *is a field.*

Definition 2.6.2 *We will now refer to any finite field as a galois field, symbolizing it* $GF(p)$*. For example, field* $\mathbb{Z}_2$ *would be* $GF(2)$*.*

Definition 2.6.3 *Let* $GF(q)$ *a Galois field, with* $q = p^m$*,* $p \in \mathbb{P}$*.* $\forall b \in GF(q), b \neq 0$*, the multiplicative order of* b *if defined as*

$$ord(b) = min\{n > 0 :\ b^n = 1\}$$

Definition 2.6.4 (Primitive Element) *An element $b \in GF(q)$ is called a primitive element if $ord(b) = q - 1$.*
If $q \in \mathbb{P}$ (hence $m = 1$), then $ord(b) = order_p(b)$

Theorem 2.6.3 *Let F field with $|F| = q = p^m$. Then:*

1. $\forall b \in F^*, ord(b) | q - 1$
2. *If $t | q - 1$ and $t > 0$, then there exist exactly $\phi(t)$ elements $b \in F$, such that $ord(b) = t$*
3. *F has $\phi(q - 1)$ primitive elements*

Definition 2.6.5 (Primitive Polynomial) *A polynomial $p(x) \in F[x]$, with $deg(p(x)) = m$ is called primitive polynomial if :*

1. *It is monic*
2. *It is irreducible over $F[x]$*
3. *$min\{n > 0 : \ p(x) | x^n - 1\} = q^m - 1$*

Chapter 3

Observations on Primitive Roots of Prime Numbers

In this chapter we will present and prove some observations that we have made on the exponentiation of primitive roots for some prime number p. We will then attempt to generalize our observations for primitive elements of Galois fields $GF(p^m)$.
Some of the observations will be used in chapter 4 to develop our new algorithm, while some other observations are just presented for their theoretical interest.

3.1 Supplements in exponentiation

Theorem 3.1.1 *Let s be the supplement of α modulo $p \in \mathbb{P},\ p > 2$, meaning $a^i + s \equiv 0\ (mod\ p),\ i \in \mathbb{Z}$.*
We will prove that $s = a^{\frac{p-1}{2}+i}$, and thus

$$a^i + a^{\frac{p-1}{2}+i} \equiv 0\ (mod\ p),$$

or, equivalently

$$r_p[a^i] + r_p[a^{\frac{p-1}{2}+i}] = p.$$

e.g. Let $a = 12$ a primitive root of $p = 17$,

a^i :	12^1	12^2	12^3	12^4	12^5	12^6	12^7	12^8
‖	‖	‖	‖	‖	‖	‖	‖	‖
$r_n[a^i]$:	12	8	11	13	3	2	7	16

	12^9	12^{10}	12^{11}	12^{12}	12^{13}	12^{14}	12^{15}	12^{16}
	‖	‖	‖	‖	‖	‖	‖	‖
	5	9	6	4	14	15	10	1

	12	8	11	13	3	2	7	16
+	5	9	6	4	14	15	10	1
	17	17	17	17	17	17	17	17

Proof:
From Fermat's little theorem - Euler's theorem,

$$a^{p-1} \equiv 1\ (\text{mod } p) \xrightarrow{\text{Let } p>2}$$

$$a^{\frac{p-1}{2}} \equiv \begin{cases} -1\ (\text{mod } p) \\ 1\ (\text{mod } p) \end{cases}$$

Assuming $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$, then $a^{p-1} \equiv a^{\frac{p-1}{2}} \pmod{p}$.
Since the powers of primitive roots are generators of $\mathbb{Z}_p^*$ with period $\phi(p) = p-1$ (as shown in 2.4.3), there cannot be duplicates unless the exponents have a distance of $k \cdot p$, $k \in \mathbb{Z}$. Hence, $a^{\frac{p-1}{2}} \not\equiv 1 \pmod{p}$, as $a^{p-1} \equiv 1 \pmod{p}$ and $p-1 \not\equiv \frac{p-1}{2} \pmod{p-1}$.
Subsequently,

$$\begin{aligned}
&a^{\frac{p-1}{2}} \equiv -1 \pmod{p} \Leftrightarrow \\
&a^{\frac{p-1}{2}} + 1 \equiv 0 \pmod{p} \overset{i\in\mathbb{Z}}{\Longleftrightarrow} \\
&a^i(a^{\frac{p-1}{2}} + 1) \equiv 0 \pmod{p} \Leftrightarrow \\
&a^i + a^{\frac{p-1}{2}+i} \equiv 0 \pmod{p} \Leftrightarrow \\
&r_p[a^i + a^{\frac{p-1}{2}+i}] = r_p[0] \Leftrightarrow \\
&r_p[a^i + a^{\frac{p-1}{2}+i}] = 0 \Leftrightarrow \\
&r_p[r_p[a^i] + r_p[a^{\frac{p-1}{2}+i}]] = 0 \Leftrightarrow \\
&r_p[a^i] + r_p[a^{\frac{p-1}{2}+i}] = kp,\ k \in \mathbb{Z}.
\end{aligned}$$

Considering that for any exponent j, $r_p[a^j] > 0$ and $r_p[a^j] < p$, $k = 1$, and so

$$r_p[a^i] + r_p[a^{\frac{p-1}{2}+i}] = p.$$

3.2 Unitary Difference

Theorem 3.2.1 *Let a be a primitive root of $p \in \mathbb{P}$. There exists unique k, such that*

$$a^k - a^{k-1} \equiv 1\ (mod\ p),\ k \in \mathbb{Z}_p$$

Proof:

$$\begin{aligned}
&a^k - a^{k-1} \equiv 1 \pmod{p} \Leftrightarrow \\
&a^{k-1} \cdot (a-1) \equiv 1 \pmod{p}
\end{aligned}$$

We know that $((a-1), p) = 1$, and thus $x \cdot (a-1) \equiv b \pmod{p}$ has a unique solution modulo p, as shown in 2.2.
Additionally, $a^{k-1} \pmod{p}$ can take any value in $\mathbb{Z}_p^*$, and since 0 cannot be the solution to the equation $x \cdot (a-1) \equiv 1 \pmod{p}$, there exists k such that $r_p[a^{k-1}]$ is equal to the solution.
Consequently, there exists unique k such that

$$a^k - a^{k-1} \equiv 1 \pmod{p},\ k \in \mathbb{Z}_p.$$

Theorem 3.2.2 *Let a be a primitive root of $p \in \mathbb{P}$, there exist unique $k', j, j' \in \mathbb{Z}_p$, such that*

$$\begin{aligned}
a^{k'} - a^{k'-1} &\equiv -1\ (mod\ p), \\
a^j + a^{j-1} &\equiv 1\ (mod\ p), \\
a^{j'} + a^{j'-1} &\equiv -1\ (mod\ p).
\end{aligned}$$

Proof is trivial.

3.3 Supplementary Exponents

Theorem 3.3.1 *Let a be a primitive root of $p \in \mathbb{P}$. For any exponent i, $i \in \mathbb{Z}$,*

$$a^i \equiv a - 1 \ (mod\ p) \Leftrightarrow a^{p-i} - a^{p-i-1} \equiv 1 \ (mod\ p).$$

Proof:
Let $i \in \mathbb{Z}$, such that $a^i \equiv a - 1 \pmod{p}$, then

$$\begin{aligned} a^i &\equiv a - 1 \pmod{p} \Leftrightarrow \\ a^i - a &\equiv -1 \pmod{p} \xLeftrightarrow{*a^{p-i-1} \neq 0} \\ a^{p-i-1+i} - a^{p-i-1+1} &\equiv -a^{p-i-1} \pmod{p} \Leftrightarrow \\ a^{p-1} - a^{p-i} &\equiv -a^{p-i-1} \pmod{p} \Leftrightarrow \\ a^{p-i} - a^{p-i-1} &\equiv a^{p-1} \pmod{p} \Leftrightarrow \\ a^{p-i} - a^{p-i-1} &\equiv 1 \pmod{p} \end{aligned}$$

Hence, for $k \in \mathbb{Z}_p$, such that $a^k - a^{k-1} \equiv 1 \pmod{p}$, and $i \in \mathbb{Z}_p$, such that $a^i \equiv a - 1 \pmod{p}$, $k + i = p$.

Theorem 3.3.2 *Let a be a primitive root of $p \in \mathbb{P}$. For any exponent $i', u, u' \in \mathbb{Z}$,*

$$\begin{aligned} a^{i'} &\equiv -(a-1) \ (mod\ p) \Leftrightarrow a^{p-i'} - a^{p-i'-1} \equiv -1 \ (mod\ p), \\ a^{u} &\equiv (a+1) \ (mod\ p) \Leftrightarrow a^{p-u} + a^{p-u-1} \equiv 1 \ (mod\ p), \\ a^{u'} &\equiv -(a+1) \ (mod\ p) \Leftrightarrow a^{p-u'} + a^{p-u'-1} \equiv -1 \ (mod\ p). \end{aligned}$$

Corollary 3.3.2.1 *For exponents $k, k', j, j', i, i', u, u' \in \mathbb{Z}_p$, such as those defined in theorems 3.2.1,3.2.2,3.3.1,3.3.2,*

$$k + i = k' + i' = j + u = j' + u' = p.$$

3.4 Further Association of Exponents

Theorem 3.4.1 *Let a be a primitive root of $p \in \mathbb{P}$ and $k_{++}, k_{+-}, k_{-+}, k_{--}, i_{++}, i_{+-}, i_{-+}, i_{--} \in \mathbb{Z}_p$, such that*

$$\begin{aligned} a^{k_{++}} + a^{k_{++}-1} &\equiv +1 \ (mod\ p), \\ a^{k_{+-}} + a^{k_{+-}-1} &\equiv -1 \ (mod\ p), \\ a^{k_{-+}} - a^{k_{-+}-1} &\equiv +1 \ (mod\ p), \\ a^{k_{--}} - a^{k_{--}-1} &\equiv -1 \ (mod\ p), \\ a^{i_{++}} &\equiv a + 1 \ (mod\ p), \\ a^{i_{+-}} &\equiv a - 1 \ (mod\ p), \\ a^{i_{-+}} &\equiv -a + 1 \ (mod\ p), \\ a^{i_{--}} &\equiv -a - 1 \ (mod\ p). \end{aligned}$$

Then,

$$\begin{aligned} a^{k_{++}+1} - a^{k_{++}-1} &\equiv a^{i_{+-}} \ (mod\ p), \\ a^{k_{+-}+1} - a^{k_{+-}-1} &\equiv a^{i_{-+}} \ (mod\ p), \\ a^{k_{-+}+1} - a^{k_{-+}-1} &\equiv a^{i_{++}} \ (mod\ p), \\ a^{k_{--}+1} - a^{k_{--}-1} &\equiv a^{i_{--}} \ (mod\ p). \end{aligned}$$

Proof:
We will only provide proof for the first congruence, as the rest can be easily produced using the same process.

$$\begin{aligned} a^{k_{++}+1} - a^{k_{++}-1} &= a^{k_{++}+1} + a^{k_{++}} - a^{k_{++}} - a^{k_{++}-1} \\ &= a^{k_{++}+1} + a^{k_{++}} - (a^{k_{++}} + a^{k_{++}-1}) \\ &\equiv a(a^{k_{++}} + a^{k_{++}-1}) - (a^{k_{++}} + a^{k_{++}-1}) \pmod{p} \\ &\equiv a(1) - (1) \pmod{p} \\ &\equiv a - 1 \pmod{p} \\ &\equiv a^{i_{+-}} \pmod{p} \end{aligned}$$

3.5 Generalized Unitary Difference

Theorem 3.5.1 *Let a be a primitive root of $p \in \mathbb{P}$, then $\exists k_{\pm\pm}, j_{\pm\pm} \in \mathbb{Z}_p$, such that*

$$\begin{aligned} a^{k_{++}} + a^{j_{++}} &\equiv +1 \ (mod\ p) \\ a^{k_{+-}} + a^{j_{+-}} &\equiv -1 \ (mod\ p) \\ a^{k_{-+}} - a^{j_{-+}} &\equiv +1 \ (mod\ p) \\ a^{k_{--}} - a^{j_{--}} &\equiv -1 \ (mod\ p). \end{aligned}$$

Proof is omitted, as it is very similar to 3.2.1.

3.6 Extension to Galois Fields

We will attempt to show that the above theorems 3.1.1, 3.2.1, 3.2.2, 3.3.1, 3.3.2, 3.4.1 stand when generalizing our assumptions to any finite field.
As we presented in Chapter 2, a Galois Field, or finite field, is a field that contains a finite number of elements.

Theorem 3.6.1 (Supplements in exponentiation in $GF(q)$) *Let F be a field, and $f(x) \in F[x]$ irreducible polynomial in $F[x]$. Assuming $q = p^m$, where $p \in \mathbb{P}$, $p > 2$, and $a \in GF(q)$ a primitive element, then*

$$r_{f(x)}\left[a^i + a^{\frac{q-1}{2}+i}\right] = 0.$$

The proof is identical to 3.1, except this time the operators $(+, \cdot)$ are under the irreducible polynomial $f(x)$.
Notice that p needs to be greater than 2 for this property, as $2 \not| (q-1)$ if $q = 2^m$.

Theorem 3.6.2 (Unitary Difference in $GF(q)$) *Let F be a field, and $f(x) \in F[x]$ irreducible polynomial in $F[x]$. Assuming $q = p^m$, where $p \in \mathbb{P}$, and $a \in GF(q)$ a primitive element, then there exists $k \in \mathbb{Z}_q$, such that*

$$r_{f(x)}[a^k - a^{k-1}] = 1$$

Proof:

$$r_{f(x)}[a^k - a^{k-1}] = 1 \xLeftrightarrow{associative\ \cdot}$$
$$r_{f(x)}[a^{k-1}(a-1)] = 1$$

Since $F[x]/f(x)$ is a field, we know there exists an identity element, hence $\exists\lambda$ such that $(a-1)\cdot\lambda = 1$. Since $\langle a\rangle$ is a generator of $GF(q)-\{0\}$, and $\lambda \neq 0$, $\exists k$ such that $a^{k-1} = \lambda$. Hence $\exists k$ such that $r_{f(x)}[a^k - a^{k-1}] = 1$.

Theorem 3.6.3 *Let F be a field, and $f(x) \in F[x]$ irreducible polynomial in $F[x]$. Assuming $q = p^m$, where $p \in \mathbb{P}$, and $a \in GF(q)$ a primitive element, then there exist $k', j, j' \in \mathbb{Z}_q$, such that*

$$\begin{aligned} r_{f(x)}[a^{k'} - a^{k'-1}] &= -1, \\ r_{f(x)}[a^{j} + a^{j-1}] &= 1, \\ r_{f(x)}[a^{j'} + a^{j'-1}] &= -1. \end{aligned}$$

Proof is omitted.

Theorem 3.6.4 (Generalized Unitary Difference in $GF(q)$) *Let F be a field, and $f(x) \in F[x]$ irreducible polynomial in $F[x]$. Assuming $q = p^m$, where $p \in \mathbb{P}$, and $a \in GF(q)$ a primitive element, then there exist $k_{\pm\pm}, j_{\pm\pm} \in \mathbb{Z}_q$, such that*

$$\begin{aligned} r_{f(x)}[a^{k_{++}} + a^{j_{++}}] &= +1, \\ r_{f(x)}[a^{k_{+-}} + a^{j_{+-}}] &= -1, \\ r_{f(x)}[a^{k_{-+}} - a^{j_{-+}}] &= +1, \\ r_{f(x)}[a^{k_{--}} - a^{j_{--}}] &= -1. \end{aligned}$$

Proof:
We will provide proof for the first statement, as the rest is trivial.

$$\begin{aligned} r_{f(x)}[a^{k_{++}} + a^{j_{++}}] = 1 &\overset{associative\ \cdot}{\Longleftrightarrow} \\ r_{f(x)}[a^{k_{++}}(a^0 + a^{j_{++}-k_{++}})] = 1 &\Leftrightarrow \\ r_{f(x)}[a^{k_{++}} r_{f(x)}[1 + a^{j_{++}-k_{++}}]] = 1 & \end{aligned}$$

Let $r_{f(x)}[1 + a^{j_{++}-k_{++}}] = c$. Since $F[x]/f(x)$ is a field, there exists an identity element λ such that $r_{f(x)}[a^{k_{++}} \cdot \lambda] = 1$, hence $\exists k, j$ such that $c = \lambda \Rightarrow r_{f(x)}[a^{k_{++}} + a^{j_{++}}] = 1$.

Chapter 4

Creating a new Algorithm

In this chapter we will use some of the theorems we proved in chapter 3, to generate a new algorithm that can capitalize on properties that primitive elements of prime numbers have, to calculate the cyclic sequence $\langle a \rangle$ using mainly additions.
Firstly, it is essential to make some definitions, for our writings to be easier to understand.

Definition 4.0.1 *Let a be a primitive root of $p \in \mathbb{P}$. We will define the powers of a as terms of a sequence, meaning*

$$a_i = a^i, \ i \in \mathbb{Z}.$$

Definition 4.0.2 *We will now standardize our definition on exponents that satisfy some of the requirements of theorems in chapter 3.*
Let a be a primitive root of $p \in \mathbb{P}$. There exist exponents $k_{++}, k_{+-}, k_{-+}, k_{--}, i_{++}, i_{+-}, i_{-+}, i_{--} \in \mathbb{Z}_p$, such that

$$\begin{aligned}
a_{k_{++}} + a_{k_{++}-1} &\equiv +1 \ (mod\ p),\\
a_{k_{+-}} + a_{k_{+-}-1} &\equiv -1 \ (mod\ p),\\
a_{k_{-+}} - a_{k_{-+}-1} &\equiv +1 \ (mod\ p),\\
a_{k_{--}} - a_{k_{--}-1} &\equiv -1 \ (mod\ p),\\
a_{i_{++}} &\equiv +(a+1) \ (mod\ p),\\
a_{i_{+-}} &\equiv +(a-1) \ (mod\ p),\\
a_{i_{-+}} &\equiv -a+1 \ (mod\ p),\\
a_{i_{--}} &\equiv -a-1 \ (mod\ p).
\end{aligned}$$

Definition 4.0.3 *Let $k_{++}, k_{+-}, k_{-+}, k_{--}, i_{++}, i_{+-}, i_{-+}, i_{--} \in \mathbb{Z}_p$, as defined in 4.0.2, we will define those exponents as **Points of Interest**, and thus the set PoI as*

$$PoI = \{k_{++}, k_{+-}, k_{-+}, k_{--}, i_{++}, i_{+-}, i_{-+}, i_{--}\}.$$

4.1 Cyclic Sequences in Consecutive Terms

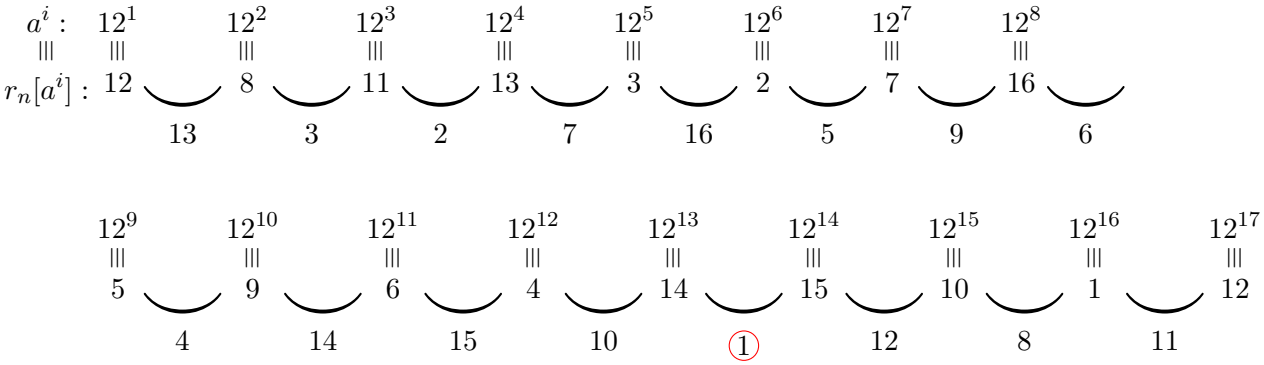
Using the observation of 3.2.1, we can deduce that the differences of consecutive terms a_k, a_{k-1}, is the cyclic sequence shifted by a term $s \in \mathbb{Z}$.

$$\begin{aligned}
a_0 &\equiv 1 \ (\text{mod } p)\\
&\equiv a_{k_{-+}} - a_{k_{-+}-1} \ (\text{mod } p)
\end{aligned}$$

Hence,

$$\begin{aligned}
a_1 &\equiv a(a_{k_{-+}} - a_{k_{-+}-1}) \pmod p \\
&\equiv a_{k_{-+}+1} - a_{k_{-+}} \pmod p \\
a_2 &\equiv a_{k_{-+}+2} - a_{k_{-+}+1} \pmod p \\
a_3 &\equiv a_{k_{-+}+3} - a_{k_{-+}+2} \pmod p \\
&\vdots \\
a_n &\equiv a_{k_{-+}+n} - a_{k_{-+}+n-1} \pmod p.
\end{aligned}$$

e.g. $a = 12$, $p = 17$,



$$\begin{aligned}
a_0 &\equiv a_{14} - a_{13} \pmod p \\
a_1 &\equiv a_{15} - a_{14} \pmod p \\
a_2 &\equiv a_{16} - a_{15} \pmod p \\
a_3 &\equiv a_{17} - a_{16} \pmod p \\
&\vdots \\
a_n &\equiv a_{14+n} - a_{14+n-1} \pmod p.
\end{aligned}$$

Similarly,

$$\begin{aligned}
a_n &\equiv a_{k_{++}+n} + a_{k_{++}+n-1} \pmod p \\
a_n &\equiv -(a_{k_{+-}+n} + a_{k_{+-}+n-1}) \pmod p \\
a_n &\equiv -(a_{k_{--}+n} - a_{k_{--}+n-1}) \pmod p
\end{aligned}$$

In addition, similar relations can be found using the points $i \in \mathbb{Z}_p$.

$$\begin{aligned}
a_n &\equiv a_{i_{++}+n} - a_{n+1} \pmod p \\
a_n &\equiv -(a_{i_{+-}+n} - a_{n+1}) \pmod p \\
a_n &\equiv a_{i_{-+}+n} + a_{n+1} \pmod p \\
a_n &\equiv -(a_{i_{--}+n} + a_{n+1}) \pmod p
\end{aligned}$$

4.1.1 Simple Form of the Algorithm (S.F.)

We can now see that if we know all the powers of our base up to an index which has "certain properties," we can compute the next terms using only addition.

$$
\begin{aligned}
a_{k_{++}+n} &\equiv a_n - a_{k_{++}+n-1} \pmod{p} \\
a_{k_{+-}+n} &\equiv -a_n - a_{k_{+-}+n-1} \pmod{p} \\
a_{k_{-+}+n} &\equiv a_n + a_{k_{-+}+n-1} \pmod{p} \\
a_{k_{--}+n} &\equiv -a_n + a_{k_{--}+n-1} \pmod{p} \\
a_{i_{++}+n} &\equiv a_n + a_{n+1} \pmod{p} \\
a_{i_{+-}+n} &\equiv -a_n + a_{n+1} \pmod{p} \\
a_{i_{-+}+n} &\equiv a_n - a_{n+1} \pmod{p} \\
a_{i_{--}+n} &\equiv -a_n - a_{n+1} \pmod{p}
\end{aligned}
$$

So, our algorithm can be described as calculating the powers of the base using multiplications until we reach an exponent $e = \min(PoI)$. After e, we can use one of the above relations (depending on the categorization of e) to calculate the rest of the sequence with additions, increasing the value of n by one after each calculation. Particularly, after the successful search

Algorithm 1: Simple Form

```
e ← 1
while e ∉ PoI do
    a_{e+1} ← mod(a_e · a, p)
    e ← e + 1
end
index ← 1
if e is some k_{±±} then
    while a_{e+index} ≠ 1 do
        a_{e+index} ← mod(±a_{index} ± a_{e+index−1}, p)
        index ← index + 1
    end
else
    /* e is some i_{±±} */
    while a_{e+index} ≠ 1 do
        a_{e+index} ← mod(±a_{index} ± a_{index+1}, p)
        index ← index + 1
    end
end
```

of $e = \min(PoI)$, our algorithm creates a window W in memory, which stores e values. The window shifts one position to the right after each calculation, storing the newest term of the sequence a_{end} and deleting the oldest.
For example, on the first calculation after finding e, the window will be

$$W = \{a_1, a_2, \ldots, a_e\},$$

on the second calculation it will be

$$W = \{a_2, a_3, \ldots, a_{e+1}\},$$

whereas, on the last calculation it will be

$$W = \{a_{p-e}, a_{p-e+1}, a_{p-e+2}, \ldots, a_{p-1}\}.$$

S.F. in Practice:

e.g. $a = 12$, $p = 17$:
Initialization Step:

$$\begin{array}{ccc} 12^1 & 12^2 & 12^3, \\ \| & \| & \| \\ 12 & 8 & 11 \end{array} \quad \Rightarrow i_{+-} = 3$$

Additions Step:

1st iteration:

$$\begin{array}{cccc} 12^1 & 12^2 & 12^3 & 12^4 \\ \| & \| & \| & \| \\ 12 & 8 & 11 & 13 \end{array}$$

2nd iteration:

$$\begin{array}{ccccc} 12^1 & 12^2 & 12^3 & 12^4 & 12^5 \\ \| & \| & \| & \| & \| \\ 12 & 8 & 11 & 13 & 3 \end{array}$$

3rd iteration:

$$\begin{array}{cccccc} 12^1 & 12^2 & 12^3 & 12^4 & 12^5 & 12^6 \\ \| & \| & \| & \| & \| & \| \\ 12 & 8 & 11 & 13 & 3 & 2 \end{array}$$

...

Theorem 4.1.1 (Theoretical boundary for PoI) *Let a be a primitive root of $p \in \mathbb{P}$, for any set PoI as defined in 4.0.3,*

$$min(PoI) \leq \lceil \frac{p-1}{4} \rceil.$$

Proof:
Let $4|(p-1)$, then we can define the following sets:

$$A = \left\{1, 2, \ldots, \frac{p-1}{4}\right\}$$
$$B = \left\{\frac{p-1}{4} + 1, \frac{p-1}{4} + 2, \ldots, \frac{p-1}{2}\right\}$$
$$C = \left\{\frac{p-1}{2} + 1, \frac{p-1}{2} + 2, \ldots, \frac{3}{4}(p-1)\right\}$$
$$D = \left\{\frac{3}{4}(p-1) + 1, \frac{3}{4}(p-1) + 2, \ldots, p-1\right\}$$

By definition, A, B, C, D are foreign and $A \cup B \cup C \cup D = \mathbb{Z}_p^*$.
Let some $k = \min(\text{PoI}) \Leftrightarrow k = \min(k_{++}, k_{+-}, k_{-+}, k_{--}, i_{++}, i_{+-}, i_{-+}, i_{--})$***, and let $k \notin A \Rightarrow k \in B \cup C \cup D$.

- If $k \in B$, then there exists i such that $i + k = p$ and from 3.3.2, $i \in \{i_{++}, i_{+-}, i_{-+}, i_{--}\}$, depending on k.
  Moreover, $\frac{p-1}{4} + 1 \leq k \leq \frac{p-1}{2}$, and thus $k_{min} = \frac{p-1}{4} + 1$, $k_{max} = \frac{p-1}{2}$. Consequently,

$$\begin{aligned} i_{min} &= p - k_{max} \\ &= p - \frac{p-1}{2} \\ &= p - 1 - \frac{p-1}{2} + 1 \\ &= \frac{p-1}{2} + 1 \end{aligned}$$

and

$$\begin{aligned} i_{max} &= p - k_{min} \\ &= p - \frac{p-1}{4} - 1 \\ &= p - 1 - \frac{p-1}{4} \\ &= \frac{3}{4}(p-1). \end{aligned}$$

Therefore, $i \in C$. However, from 3.4.1 we know $\exists\ i'$, such that $a_{i'} \equiv -a_i \pmod{p}$ and since $i \in C$, from 3.1, $i' \in A$ which contradicts our assumption that k is the minimum PoI.

- If $k \in C$, from 3.4.1 $\exists k'$, such that $a_{k'} \equiv -a_k \pmod{p}$ and from 3.1 $k' \in A$ which again contradicts our original assumption that k is the minimum PoI.
- If $k \in D$, then there exists i such that $i + k = p$ and from 3.3.2, $i \in \{i_{++}, i_{+-}, i_{-+}, i_{--}\}$, depending on k.
  Moreover, $\frac{3}{4}(p-1) + 1 \leq k \leq p - 1$, and thus $k_{min} = \frac{3}{4}(p-1) + 1$, $k_{max} = p - 1$. Consequently,

$$\begin{aligned} i_{min} &= p - k_{max} \\ &= p - p + 1 \\ &= 1 \end{aligned}$$

and

$$\begin{aligned} i_{max} &= p - k_{min} \\ &= p - \frac{3}{4}(p-1) - 1 \\ &= p - 1 - \frac{3}{4}(p-1) \\ &= \frac{1}{4}(p-1). \end{aligned}$$

Therefore, $i \in A$ and thus $i = \min(\text{PoI})$ which contradicts our assumption.

Hence, if $4|(p-1)$, $\min(\text{PoI}) \leq \frac{p-1}{4}$.
Assuming $4 \not| (p-1)$, then $4|(p+1)$ and we can define the following sets:

$$\begin{aligned} A' &= \left\{1, 2, \ldots, \frac{p+1}{4}\right\} \\ B' &= \left\{\frac{p+1}{4} + 1, \frac{p+1}{4} + 2, \ldots, \frac{p-1}{2}\right\} \\ C' &= \left\{\frac{p-1}{2} + 1, \frac{p-1}{2} + 2, \ldots, \frac{3}{4}(p+1)\right\} \\ D' &= \left\{\frac{3}{4}(p+1) + 1, \frac{3}{4}(p+1) + 2, \ldots, p-1\right\} \end{aligned}$$

Following the same procedure as in case $4|(p-1)$, we can deduce $k \in A'$ and thus $\min(\text{PoI}) \leq \frac{p+1}{4}$.
Finally, we can combine the two cases and acquire $\min(\text{PoI}) \leq \lceil \frac{p-1}{4} \rceil$.

4.1.2 Generalized Simple Form of the Algorithm (G.F.)

As shown in 3.5.1, there exists j, such that $a_k \pm a_j \equiv \pm 1 \pmod{p}$. Consequently, we can identify $a_i \equiv \pm a \pm 1 \pmod{p}$ as a special case of 3.5.1, with $j = 1$. Using this, we can produce a new, generalized set of equations.
For each triplet $(a_n, k_{\pm\pm}, j)$, with $n, k_{\pm\pm}, j \in \mathbb{Z}$ and $k_{\pm\pm} > j$,

$$\begin{aligned}
a_n &\equiv a_{k_{++}+n} + a_{j+n} \pmod{p} \Leftrightarrow a_{k_{++}+n} \equiv a_n - a_{j+n} \pmod{p} \\
a_n &\equiv -a_{k_{+-}+n} - a_{j+n} \pmod{p} \Leftrightarrow a_{k_{+-}+n} \equiv -a_n - a_{j+n} \pmod{p} \\
a_n &\equiv a_{k_{-+}+n} - a_{j+n} \pmod{p} \Leftrightarrow a_{k_{-+}+n} \equiv a_n + a_{j+n} \pmod{p} \\
a_n &\equiv -a_{k_{--}+n} + a_{j+n} \pmod{p} \Leftrightarrow a_{k_{--}+n} \equiv -a_n + a_{j+n} \pmod{p}.
\end{aligned}$$

Essentially, we state that if we find any 2 terms in our sequence that satisfy

$$\theta_1 a_x + \theta_2 a_y \equiv 1 \pmod{p},\ \theta_{1,2} \in \{-1, 1\},$$

we can produce the rest of the sequence using only additions through the above set of equations.

e.g. $a = 3$, $p = 17$

a^i :	3^1	3^2	3^3	3^4	3^5	3^6	3^7	3^8
$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$
$r_n[a^i]$:	3	9	10	13	5	15	11	16

3^9	3^{10}	3^{11}	3^{12}	3^{13}	3^{14}	3^{15}	3^{16}
$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$	$\|\|\|$
14	8	7	4	12	2	6	1

$$\begin{aligned}
a_0 &\equiv a_7 - a_3 \pmod{p} \\
a_1 &\equiv a_8 - a_4 \pmod{p} \\
a_2 &\equiv a_9 - a_5 \pmod{p} \\
a_3 &\equiv a_{10} - a_6 \pmod{p} \\
&\vdots \\
a_n &\equiv a_{7+n} - a_{3+n} \pmod{p}.
\end{aligned}$$

Definition 4.1.1 *Our new definition of the generalized set of Points of Interest consists of pairs* $(k_{\pm\pm}, j) \in \mathbb{Z}$*:*

$$GPoI = \{(k_{++}, j_1), (k_{+-}, j_2), (k_{-+}, j_3), (k_{--}, j_3)\}$$

So, our generalized algorithm tries all possible values of j for each new exponent e computed. After a GPoI pair is found, the rest of the sequence can again be computed with additions, according to the categorization of the GPoI.

Algorithm 2: Generalized Simple Form

```
e ← 1
while True do
    a_{e+1} ← mod(a_e · a, p)
    e ← e + 1
    j ← e − 1
    while j > 0 do
        if (e, j) ∈ GPoI then
            break
        end
        j ← j − 1
    end
end
index ← 1
while a_{e+index} ≠ 1 do
    a_{e+index} ← mod(±a_{index} ± a_{j+index}, p)
    index ← index + 1
end
```

4.1.3 Enhanced Form

We will show that our general idea can be expanded into using multiple terms of the cyclic sequence.
Once more, we will start by assuming a combination of q terms of the sequence is equal to 1.

$$\pm a_{k_1} \pm a_{k_2} \pm \ldots \pm a_{k_q} \equiv 1 \pmod{p}$$

Then,

$$\begin{aligned} a_1 \equiv a(\pm a_{k_1} \pm a_{k_2} \pm \ldots \pm a_{k_q}) \pmod{p} &\equiv \pm a_{k_1+1} \pm a_{k_2+1} \ldots \pm a_{k_q+1} \pmod{p} \\ a_2 &\equiv \pm a_{k_1+2} \pm a_{k_2+2} \ldots + \pm a_{k_q+2} \pmod{p} \\ &\cdot \\ &\cdot \\ &\cdot \\ a_n &\equiv \pm a_{k_1+n} + \pm a_{k_2+n} \ldots + \pm a_{k_q+n} \pmod{p}. \end{aligned}$$

As we can see, we have 2^q different possible combinations for q possible Points of Interest. Undeniably, it would not be practical for our algorithm to look into every possible combination, as this would dramatically increase the number of operations.
Consequently, we will examine an enhanced version of our algorithm which takes into consideration combinations of 2 AND 3 terms (Enhanced 3-Term Form - E3TF).

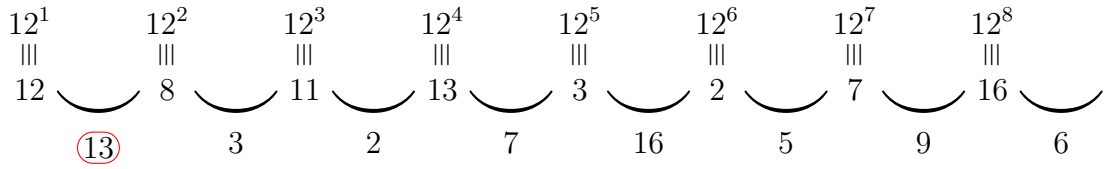
Definition 4.1.2 *Let a be a primitive root of $p \in \mathbb{P}$. There exist exponents such that*

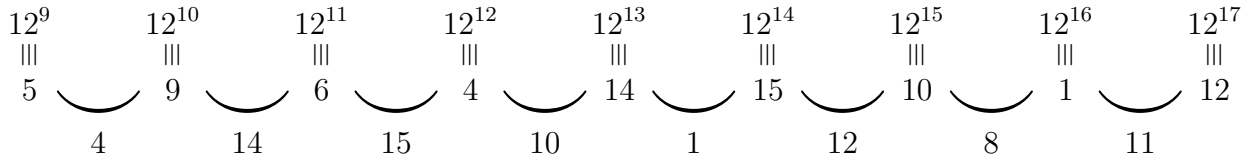
$$\begin{aligned}
a_{k_{+++}} &\equiv 1 - a_{j_1} - a_{j_2} \\
a_{k_{++-}} &\equiv -1 - a_{j_1} - a_{j_2} \\
a_{k_{+-+}} &\equiv 1 - a_{j_1} + a_{j_2} \\
a_{k_{+--}} &\equiv -1 - a_{j_1} + a_{j_2} \\
a_{k_{-++}} &\equiv 1 + a_{j_1} - a_{j_2} \\
a_{k_{-+-}} &\equiv -1 + a_{j_1} - a_{j_2} \\
a_{k_{--+}} &\equiv 1 + a_{j_1} + a_{j_2} \\
a_{k_{---}} &\equiv -1 + a_{j_1} + a_{j_2}.
\end{aligned}$$

Thus, our new set of equations

$$\begin{aligned}
a_{k_{+++}+n} &\equiv a_n - a_{j_1+n} - a_{j_2+n} \\
a_{k_{++-}+n} &\equiv -a_n - a_{j_1+n} - a_{j_2+n} \\
a_{k_{+-+}+n} &\equiv a_n - a_{j_1+n} + a_{j_2+n} \\
a_{k_{+--}+n} &\equiv -a_n - a_{j_1+n} + a_{j_2+n} \\
a_{k_{-++}+n} &\equiv a_n + a_{j_1+n} - a_{j_2+n} \\
a_{k_{-+-}+n} &\equiv -a_n + a_{j_1+n} - a_{j_2+n} \\
a_{k_{--+}+n} &\equiv a_n + a_{j_1+n} + a_{j_2+n} \\
a_{k_{---}+n} &\equiv -a_n + a_{j_1+n} + a_{j_2+n}.
\end{aligned}$$

e.g. $a = 12$, $p = 17$





$$\begin{aligned}
a^2 - a^1 \equiv a^1 + 1 \pmod{p} &\Rightarrow a^3 \equiv a^2 + a^2 + a^1 \pmod{p} \\
&\Rightarrow a^4 \equiv a^3 + a^3 + a^2 \pmod{p} \\
&\dots
\end{aligned}$$

Definition 4.1.3 *Our new definition of the enhanced set of Points of Interest for the 3-term algorithm will consist of triplets $(k_{\pm\pm\pm}, j_1, j_2)$:*

$$\begin{aligned}
\textit{PoI-3T} = \{&(k_{+++}, j_1, j_2), (k_{++-}, j_1, j_2), (k_{+-+}, j_1, j_2), (k_{+--}, j_1, j_2), \\
&(k_{-++}, j_1, j_2), (k_{-+-}, j_1, j_2), (k_{--+}, j_1, j_2), (k_{---}, j_1, j_2)\}
\end{aligned}$$

Algorithm 3: Enhanced 3-term Form

```
e ← 1
while True do
    a_{e+1} ← mod(a_e · a, p)
    e ← e + 1
    j_1 ← e − 1
    while j_1 > 0 do
        if (e, j_1) ∈ GPoI then
            break
        end
        j_2 ← e − 1
        while j_2 > 0 do
            if (e, j_1, j_2) ∈ PoI-3T then
                break
            end
            j_2 ← j_2 − 1
        end
        j_1 ← j_1 − 1
    end
end
index ← 1
while a_{e+index} ≠ 1 do
    a_{e+index} ← mod(±a_{index} ± a_{j+index}, p)
    index ← index + 1
end
```

Our algorithm, as we can see, looks into the possibility of finding an exponent j such that $j = \min(GPoI)$, and if it fails it tries to find a triplet (k, j_1, j_2). If that also fails, it moves on to another value of j.
The number of possible tuples is thus $2^2 + 2^3 = 4 + 8 = 12$.

4.2 Simulating Initialization Step

In this section, we will explore the efficiency of our algorithm in locating $\min(PoI)$.

4.2.1 Initialization Performance

The following results showcase the distribution of $\frac{\min(PoI)}{p-1}$ for primes up to 14 bits.

Simple Form:

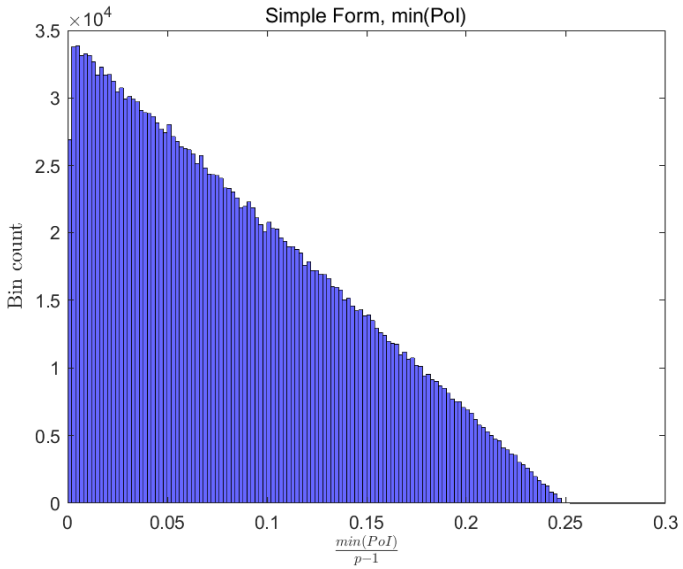


Figure 4.1: Distribution of $\frac{\min(PoI)}{p-1}$ for primes and their primitive roots up to 14 bits.

As we can see, the boundary of $\min(PoI)$ is, as expected, $0.25 \cdot (p-1)$, and the distribution is linearly decreasing.
Next, we depicted the categorization of our minimum Points of Interest along the primes we studied.

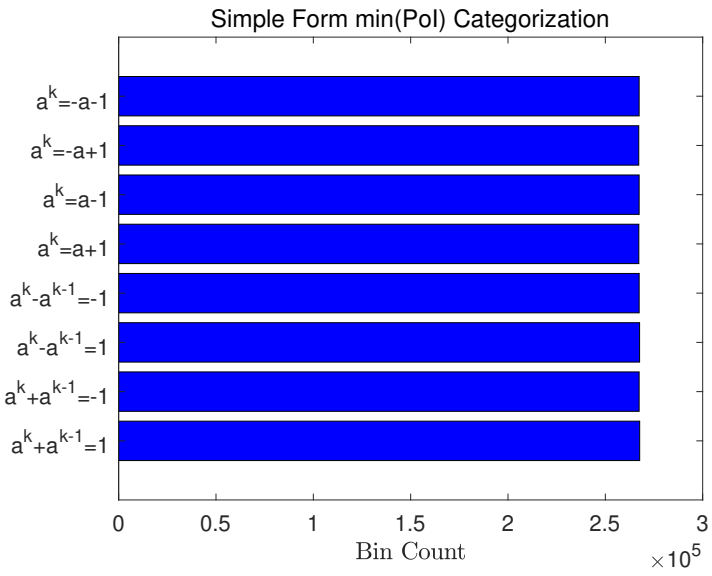


Figure 4.2: Categorization of spotted $\min(PoI)$

The exponent's categories are clearly uniformly distributed, and thus equiprobable.

Generalized Form:

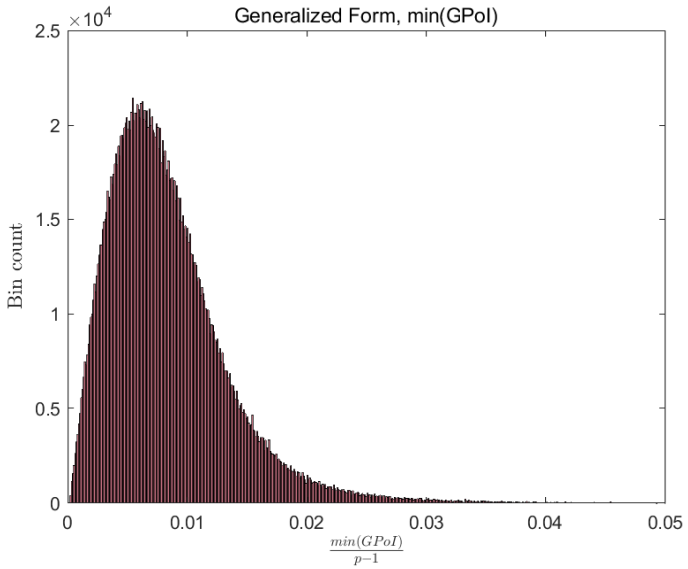


Figure 4.3: Distribution of $\frac{\min(GPoI)}{p-1}$ for primes and their primitive roots up to 15 bits.

Here, the boundary of $\min(GPoI)$ seems to be approaching $0.05 \cdot (p-1)$.
Note that this time, $\min(GPoI) \sim \mathcal{N}(\mu, \sigma)$, with $\mu \approxeq 0.008 \cdot (p-1)$, $\sigma \approxeq 0.005 \cdot (p-1)$.

Again, we depicted the categorization of our minimum Generalized Points of Interest along the primes we studied.

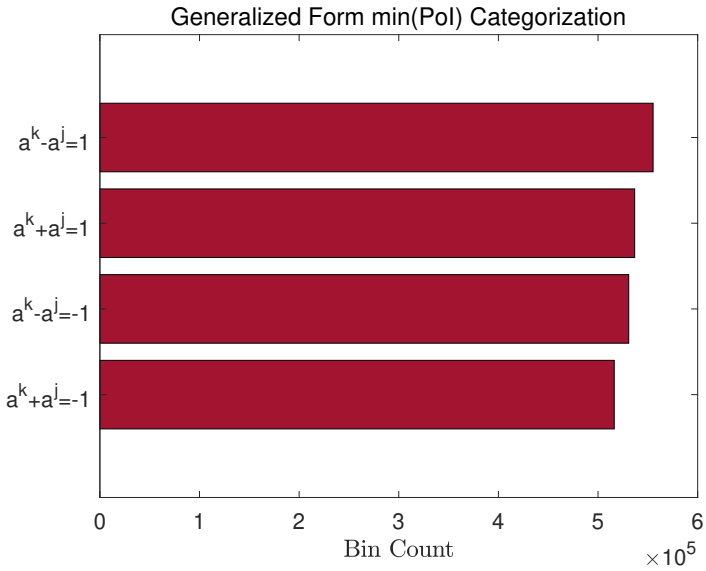


Figure 4.4: Categorization of spotted $\min(GPoI)$

Our $\min(GPoI)$ seems to be slightly more likely to come from exponents that satisfy $a^k - a^j \equiv 1 \pmod{p}$.

Enhanced 3-term Form:

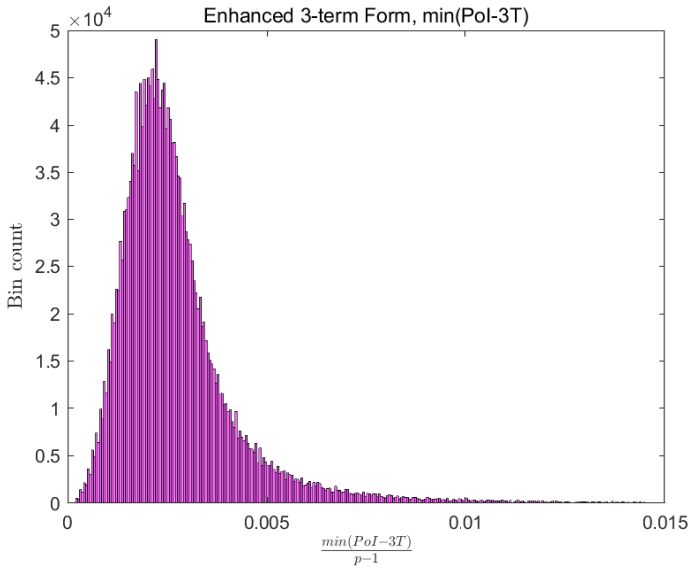


Figure 4.5: Distribution of $\frac{\min(PoI-3T)}{p-1}$ for primes and their primitive roots up to 15 bits.

The boundary of $\min(PoI - 3T)$ seems to be approaching $0.015 \cdot (p-1)$.
Again, $\min(PoI - 3T) \sim \mathcal{N}(\mu, \sigma)$, with $\mu \approxeq 0.003 \cdot (p-1)$, $\sigma \approxeq 0.002 \cdot (p-1)$.

The categorization of our minimum Enhanced 3-term Points of Interest along the primes we studied is as follows.

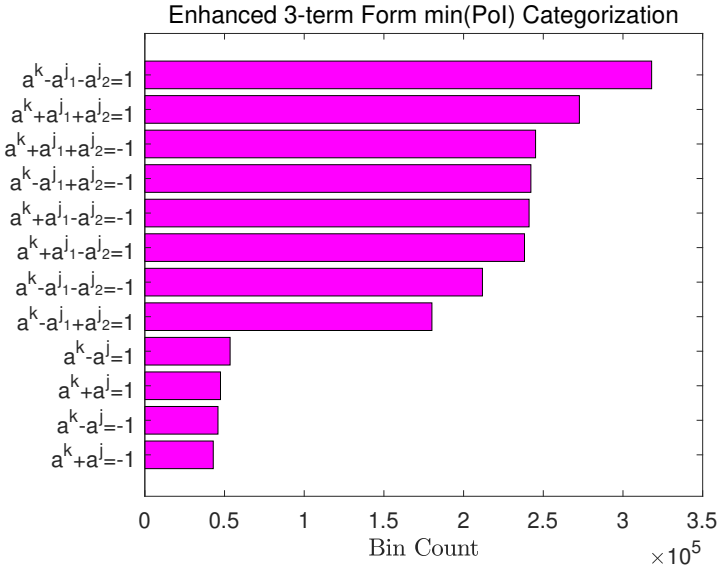


Figure 4.6: Categorization of spotted $\min(PoI - 3T)$

As expected, due to the way our algorithm functions (checks all possible triplets for a given exponent couple (k, j)), we found way more triplets (k, j_1, j_2) than couples. Furthermore, our most found triplet was (k, j_1, j_2), such that $a^k - a^{j_1} - a^{j_2} \equiv 1 \pmod{p}$, or using definition 4.1.2 (k_{--+}, j_1, j_2).

4.2.2 Initialization Costs

Afterwards, we depicted the cost of each algorithm's attempt to locate the above Points of Interest. We evaluate the cost as the number of additions, multiplications and conditional operators used in each algorithm's initialization step. Each dot in a stem diagram depicts the number of operations in one simulation.

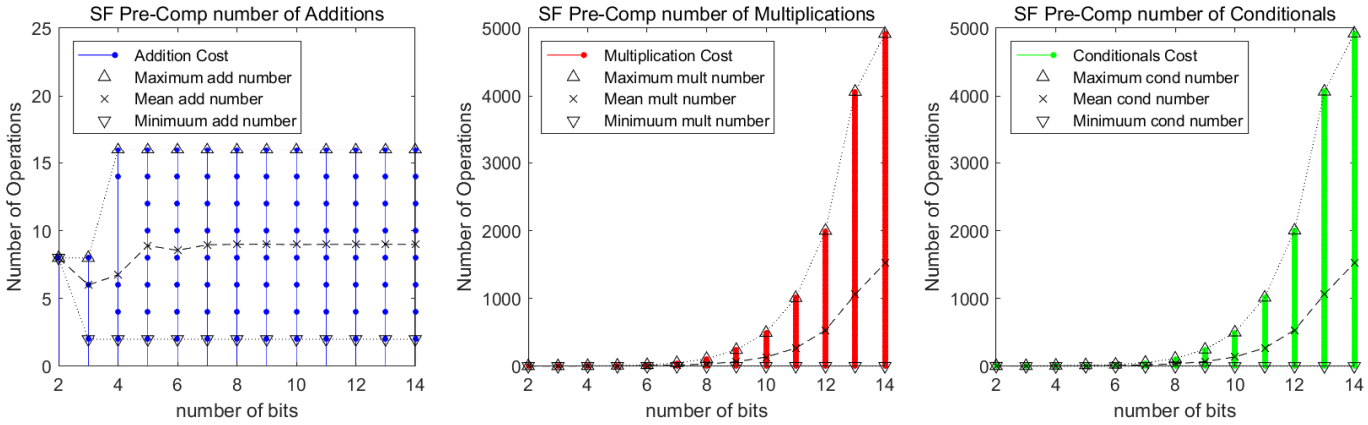


Figure 4.7: SF algorithm

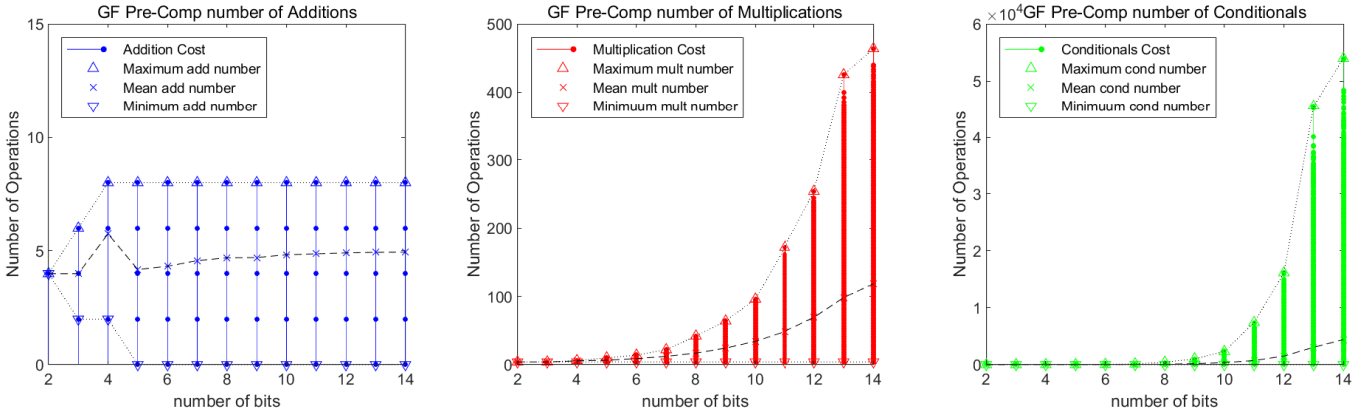


Figure 4.8: GF algorithm

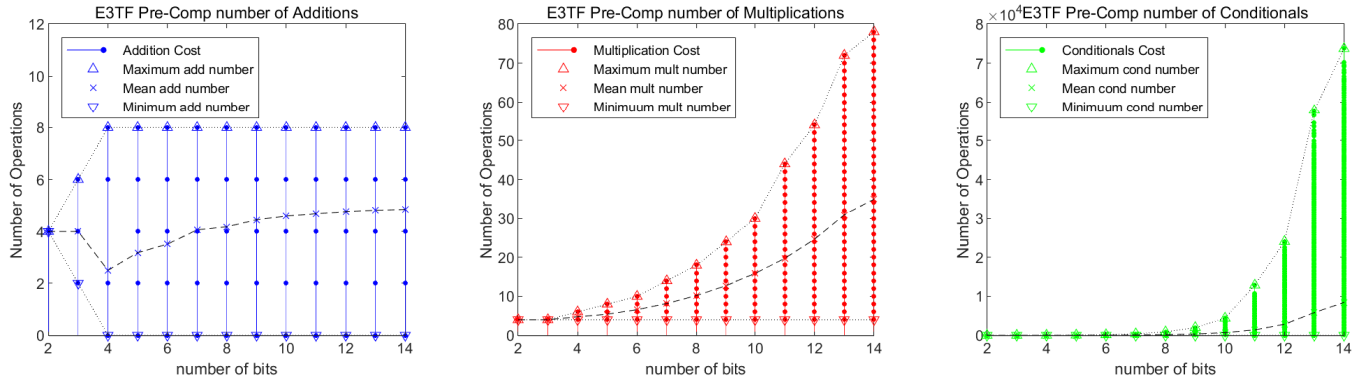


Figure 4.9: E3TF algorithm

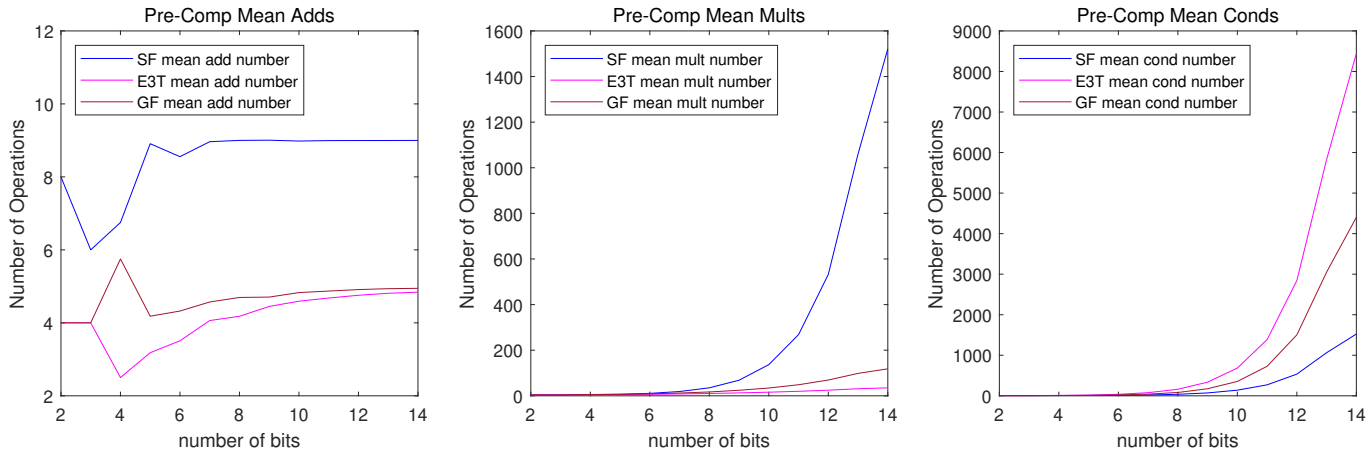


Figure 4.10: Mean Cost of each algorithm

As we can see, GF and E3TF have much less multiplications than SF, though they pay the price in their use of conditional operators.

Chapter 5

Applications

At this point, we can understand that our "new" algorithm is essentially an improved brute force algorithm for the calculation of the cyclic sequence produced by the powers of a primitive root modulo p. It is categorized as "brute force," as the number of operations is not reduced, but rather the difficulty of the operation is improved by using mostly additions instead of multiplications.
We will now investigate some prospective applications for such an algorithm, as well as compare it to the currently used methods.

5.1 Lehmer Random Number Generator

We propose our new algorithm as a potentially fast Lehmer Random Number generator.

5.1.1 Linear Congruential Generators and Pseudo-random Numbers

A linear congruential generator (LCG) is an algorithm that yields a sequence of pseudo-randomized numbers calculated using the recurrence relation[7]:

$$X_{n+1} \equiv (aX_n + c) \pmod{m}$$

where X is the sequence of pseudo-random values, $m > 0$ is the modulus, $a \in (0, m)$ is the multiplier, $c \in (0, m)$ the increment, and X_0 is the seed.
When it comes to choosing the parameters, there are 3 possible pathways[7]:

(a) Set $c = 0$ and m prime (often referred to as Lehmer RNG [8] or Multiplicative Congruential Generator)

(b) Set $c = 0$ and m a power of 2

(c) Set $c \neq 0$

We will only be dealing with form (a) of the algorithm (LRN).

5.1.2 Lehmer Random Number Generator

By setting the increment $c = 0$ and m prime, we obtain [8]

$$X_{n+1} \equiv aX_n \pmod{m}.$$

As we can see, the largest possible period of the sequence is $m-1$, a period which is achieved when a is a primitive element of GF(m).

5.1.3 Brute Forcing Randomness

The most obvious way to produce the random sequence generated from the above congruence is through brute-forcing it, calculating each term through modular multiplication.
Clearly, this would result in a time complexity of $2 \cdot (m-1)$ multiplications (assuming modular multiplication needs 1 multiplication and 1 division)[9], and thus $\mathcal{O}(2(m-1)(n\log_2 n)) = \mathcal{O}(mn\log_2 n) = \mathcal{O}(2^n n\log_2 n)$[10], where n is the number of bits in m.

Algorithm 4: Brute LRN

$i \leftarrow 1$
while $a_{end} \neq 1$ **do**
  $a_{i+1} \leftarrow \mathrm{mod}(a_i \cdot a, p)$
  $i \leftarrow i + 1$
end
$out \leftarrow \mathrm{mod}(a \cdot seed, p)$

5.1.4 An Addition-based Implementation

Our implementation for this application is pretty simple. We follow the steps of our algorithm, as described in Chapter 4, either using the S.F., G.S.F. or the E.3T.F. We then multiply every calculated number with the initial seed X_0 and output the whole sequence.

$$
\begin{aligned}
X_0 &= seed \\
X_1 &\equiv a \cdot X_0 \pmod{p} \\
X_2 &\equiv a(a \cdot X_0) \pmod{p} \\
X_3 &\equiv a(a^2 \cdot X_0) \pmod{p} \\
&\vdots \\
X_n &\equiv a^n \cdot X_0 \pmod{p}
\end{aligned}
$$

5.1.5 Simulations

We tested the performance of S.F., G.F. and E.3T.F., against the brute force algorithm. We evaluate the performance based on the number of additions, multiplications and conditional operators used.

Additions, Multiplications and Conditionals

We showcase the number of operators used in each algorithm, as the number of bits of our prime number increases. Each dot in a stem diagram depicts the number of operations in one simulation. We simulated all primitive roots for primes up to 14 bits.

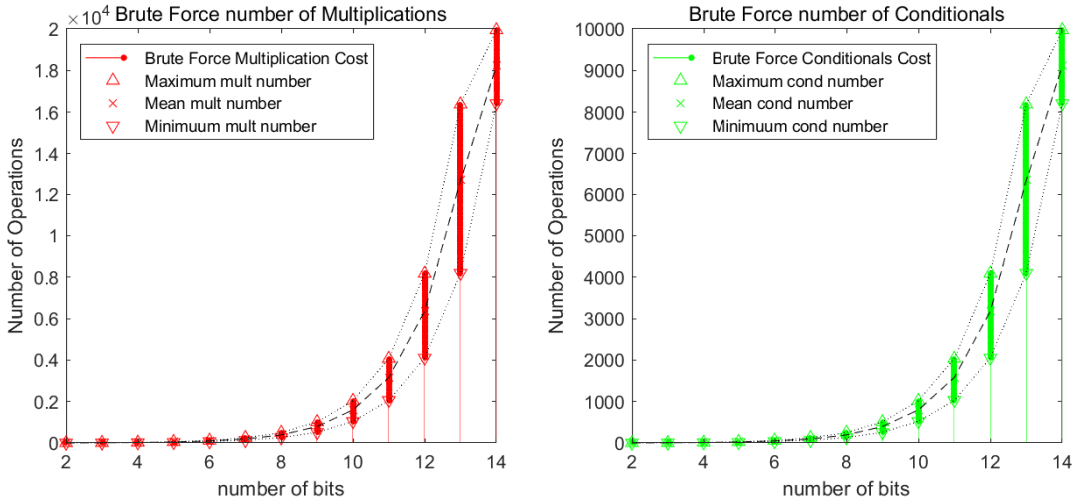


Figure 5.1: Brute Force

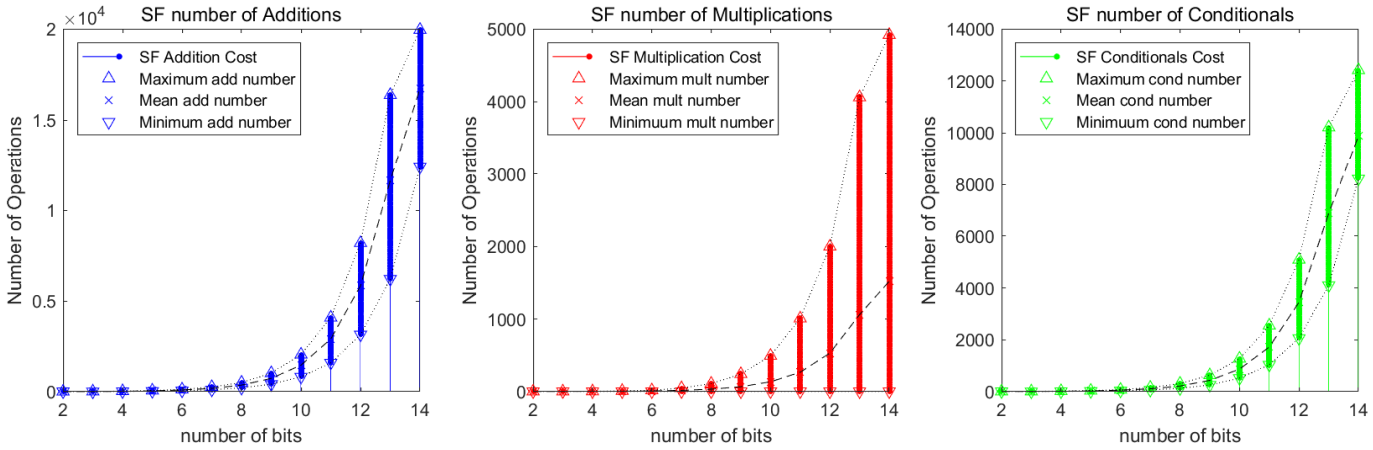


Figure 5.2: SF algorithm

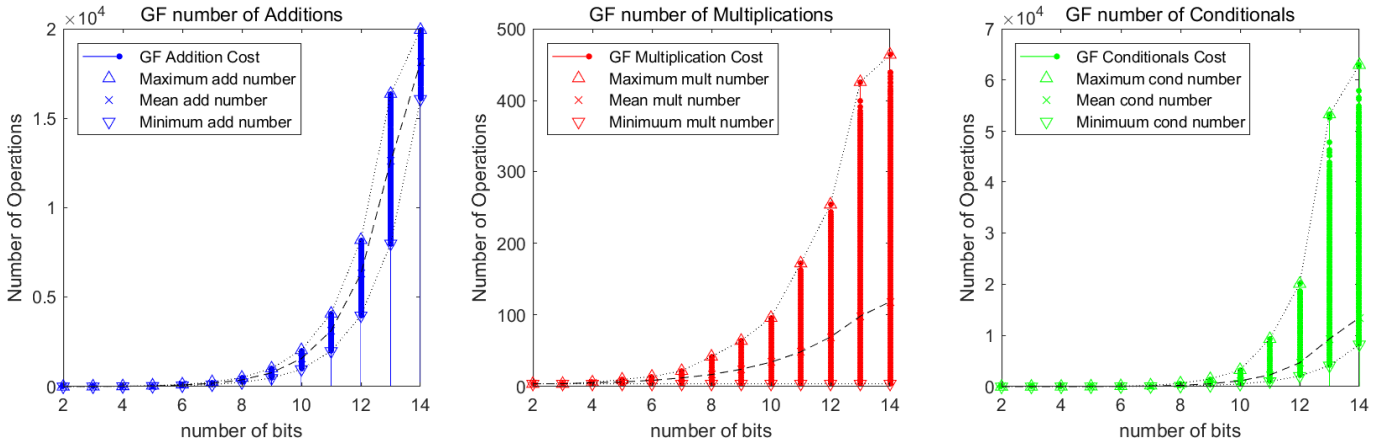


Figure 5.3: GF algorithm

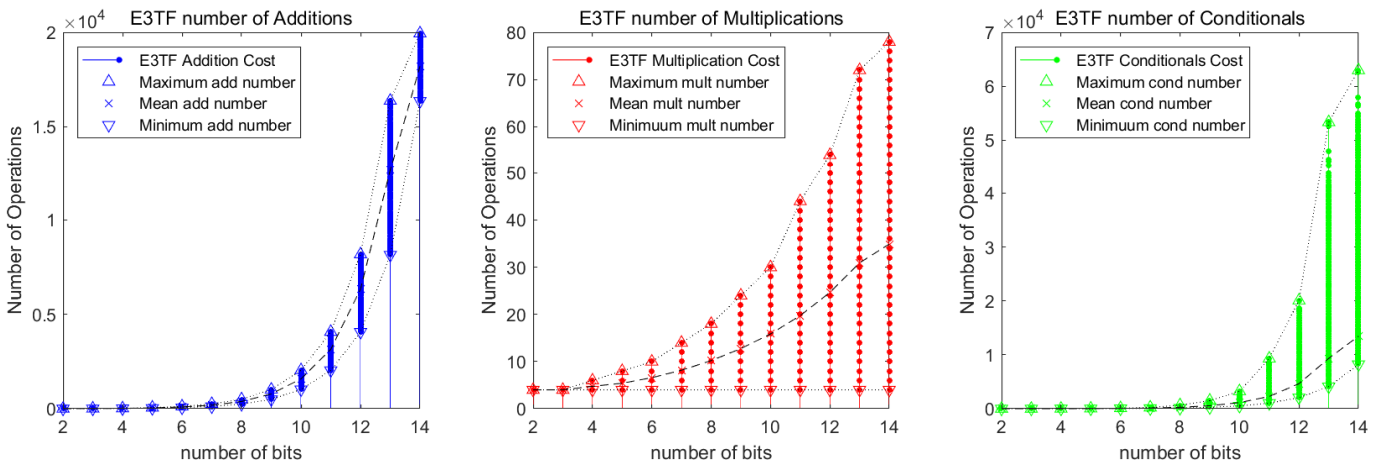


Figure 5.4: E3TF algorithm

We can see that E3TF algorithm has the fewest multiplications. Moreover, brute force and SF algorithms have the fewest conditionals, while the number of additions is similar for all 3 of our algorithms. Finally, we can observe that the number of multiplications for the brute force algorithm is approximately 4 times larger than SF (as expected), and insurmountably larger than GF and E3TF algorithms.

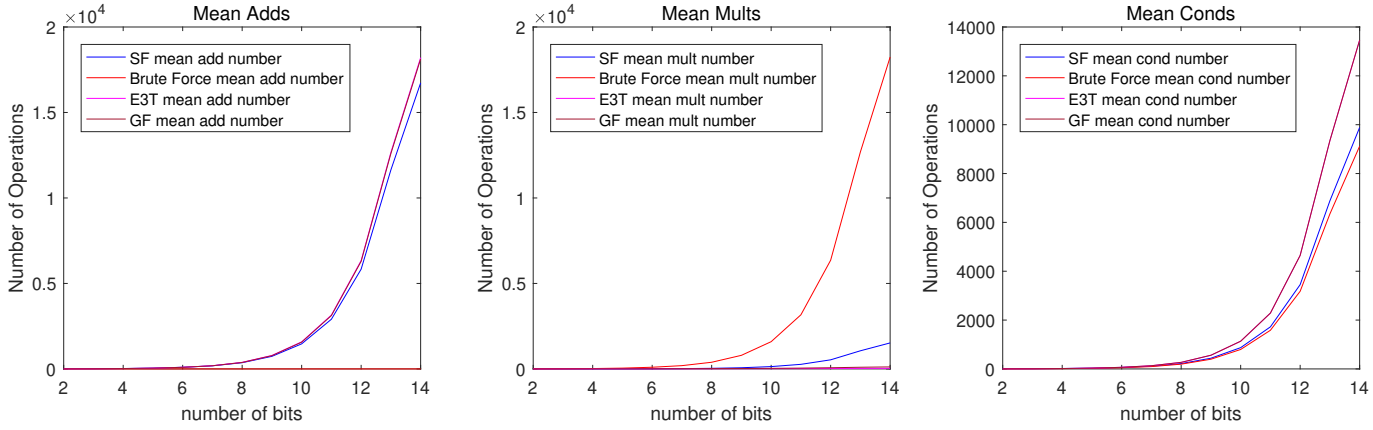


Figure 5.5: Mean number of Operations for each algorithm

Conclusion

If we take into account that a multiplication has a complexity of $\mathcal{O}(n\log n)$, using the complexity of Harvey-Hoeven multiplication algorithm[10], while additions and conditional operations are much "cheaper," all forms of our algorithms managed to acquire a better performance than brute force.

5.2 Discrete Logarithm

The first thing coming to mind when thinking about powers of primitive elements of some GF(q) is the discrete logarithm. We will firstly express the difficulty of the problem, as well as its importance in modern cryptography, and then attempt to propose our new algorithm which despite not being competitive to the current ones, shows some interesting prospect for future work.

5.2.1 The Discrete Logarithm Problem

As shown in chapter 1, given some group $G(+, \cdot)$ with its identity element 1, and a, b elements of G, the integer k that solves the equation $b^k = a$ is termed a discrete logarithm of a to the base b.[9]

5.2.2 Use in Cryptography

Diffie-Hellman

This method ensures secure exchange of cryptography keys over a public channel and was one of the first public-key protocols.[1][2]
We will describe a simple implementation of the protocol over the multiplicative group of integers modulo p, where p is prime and g is a primitive root modulo p, though this can be generalized to finite cyclic groups.
Suppose Alice and Bob want to establish a common cryptographic key, unknown to any outsiders.

- Alice and Bob publicly agree to use a modulus $p = 17$ and base $g = 12$ (p.r. modulo 17)
- Alice chooses a secret integer $i = 12$, and sends Bob $A = g^i \pmod p$, through the public channel
- Bob chooses a secret integer $j = 9$, and sends Alice $B = g^j \pmod p$, through the public channel
- Bob computes $s = A^j \pmod p$, in our example $s = 4^9 \pmod{17} = 4$
- Alice computes $s = B^i \pmod p$, in our example $s = 5^{12} \pmod{17} = 4$
- The secret key shared by Alice and Bob is now $g^{i \cdot j} \pmod p$, in our example $s = 4$.

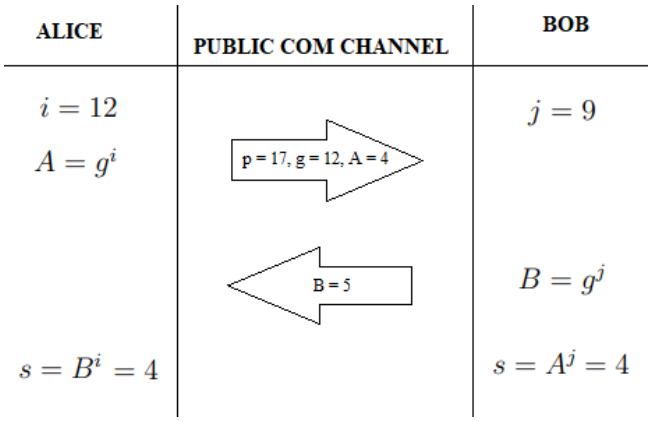


Figure 5.6: Diffie - Hellman schematic

Now, suppose an eavesdropper, Eve, wants to steal the secret key. Then, Eve would have to solve the discrete logarithm problem for one of the two computations, $A = g^i \pmod p \Rightarrow i = \text{ind}_g\ A \pmod p$ or $B = g^j \pmod p \Rightarrow j = \text{ind}_g\ B \pmod p$. The strength of the scheme comes from the fact that this problem takes extremely long time to compute by any known algorithm just from the knowledge of p, g, $g^i \pmod p$, and $g^b \pmod p$[9].

5.2.3 Algorithms

Brute Force

An obvious-naive way to solve this problem is through sheer computation of the cyclic sequence of GF(p).
This would result in exponential time complexity, as it would need a worst case of $p-1$ multiplications, or $\mathcal{O}(2^n n\log\, n)$[10] where $n = \log_2 p$ the number of bits of p.

Baby Step - Giant Step

This algorithm, developed by Shanks, is a generic algorithm well suited to a variety of problems in group computation. It is one of the standard methods for computing discrete logarithms, and is the basis of more general algorithms for determining group structure.[11] [12]
Let G a group of order q. Given as input g and $y \in \langle g\rangle$, $y = g^x$, we can imagine the elements of $\langle g\rangle$ laid out in a circle as

$$1 = g^0, g^1, g^2, \ldots, g^{q-2}, g^{q-1}, g^q = 1,$$

and we know that y must lie on this circle.
We will try to "mark off" the circle at intervals of size $t = \lfloor\sqrt{q}\rfloor$. We compute and record the $\lfloor q/t\rfloor + 1 = \mathcal{O}(\sqrt{q})$ elements

$$g^0, g^t, g^{2t}, \ldots, g^{\lfloor q/t\rfloor \cdot t}.$$

These are the giant steps. Note that the gap between any consecutive marks on the circle is at most t. In addition, we know that $y = g^x$ lies in one of those gaps. We are thus guaranteed that one of the t elements

$$y \cdot g^0 = g^x,\ y \cdot g^1 = g^{x+1}, \ldots,\ y \cdot g^t = g^{x+t},$$

will be equal to one of the initial points we marked. These are the baby steps.
Say $y \cdot g^i = g^{k\cdot t}$. Then, we can easily compute x as $x = kt - i\ (\mathrm{mod} q)$.

e.g. $g = 12, p = 17,\ x = 13, g^x = 14,\ t = \sqrt{16} = 4$ (Giant Steps)

$$12^0 = 1,\ 12^4 = 13,\ 12^8 = 16,\ 12^12 = 4,\ 12^16 = 1$$

(Baby Steps)

$$14 \cdot 12^0 = 14,\ 14 \cdot 12^1 = 15,\ 14 \cdot 12^2 = 10,\ 14 \cdot 12^3 = 1$$

Hence, $12^x \cdot 12^3 = 12^{16} \Rightarrow x + 3 = 16 \Rightarrow x = 13$.

Pseudocode for the algorithm used in a finite field $GF(p)$:

Algorithm 5: Baby Step - Giant Step

Data: A finite field $GF(p)$, $p \in \mathbb{P}$ of order $q = p - 1$, having a generator g and an element y.
Result: A value x satisfying $g^x = y$.
$t \leftarrow \lfloor \sqrt{q} \rfloor$
for $j = 1, j \leq \lfloor q/t \rfloor, j++$ **do**
  Compute $g^j \pmod p$ and store the pair (j, g^j) in a table
end
$temp \leftarrow 0$
$i \leftarrow 0$
while $temp \neq any\ g^{j_k}$ **do**
  $temp \leftarrow y \cdot g^i \pmod p$
  $i \leftarrow i + 1$
end
return $j_k \cdot t - (i - 1) \pmod q$

The algorithm requires $\mathcal{O}(\sqrt{q})$ exponentiations and multiplications in $GF(p)$, and each exponentiation can be done in time $\mathcal{O}(\text{polylog}(q))$ using an efficient exponentiation algorithm. Sorting the $\mathcal{O}(\sqrt{q})$ pairs (j, g_j) can be done in time $\mathcal{O}(\sqrt{q} \cdot \log q)$, and we can then use binary search to check whether y_i is equal to some g_k in time $\mathcal{O}(\log q)$.[12]
The overall algorithm thus runs in time $\mathcal{O}(\sqrt{q} \cdot \text{polylog}(q)) = \mathcal{O}(\sqrt{2^n} \cdot \text{polylog}(2^n))$, where $n = \log_2 q$, the number of bits of q (hence, the number of bits of $p - 1$).

5.2.4 Implementing our Algorithm

The implementation of our algorithm is again similar to the original implementations in Chapter 4, (S.F., G.S.F. or E.F.).
The only difference is the stopping criterion, which, given $a^x \equiv \beta \pmod p$, is when
$a^{k+n} \equiv \beta \pmod p \Rightarrow \text{ind}_a \beta = k + n$, where k is the given point of interest and $n \in \mathbb{Z}_{\text{I}}$.
Our algorithm's computational complexity is obviously still exponential, but perhaps for relatively small numbers $p \in \mathbb{P}$ we could manage to achieve performances close to Baby Step - Giant Step. Additionally, We can again expect our algorithm to perform better than the naive brute force algorithm.

5.2.5 Simulations

We tested the performance of S.F., G.F. and E.3T.F., against the brute force and Baby Step - Giant Step algorithm. The costs are modelled similarly to the previous simulations.

Additions, Multiplications and Conditionals

First, we showcase the number of additions, multiplications and conditional operators used in each algorithm. Again, each dot in a stem diagram depicts the number of operations in one simulation. We simulated all primitive roots a for primes p up to 14 bits, and tried to solve $a^x \equiv b \pmod p$, where b is a random value in $[1, p - 1]$.

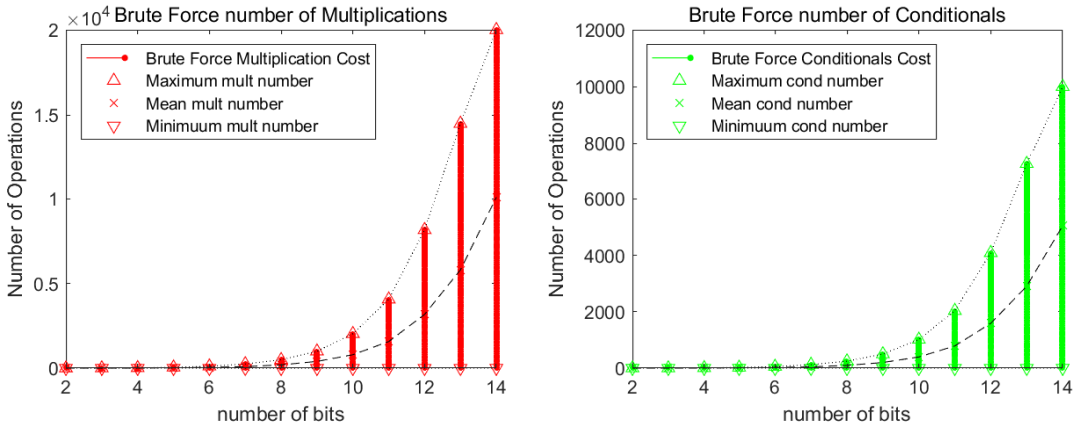


Figure 5.7: Brute Force

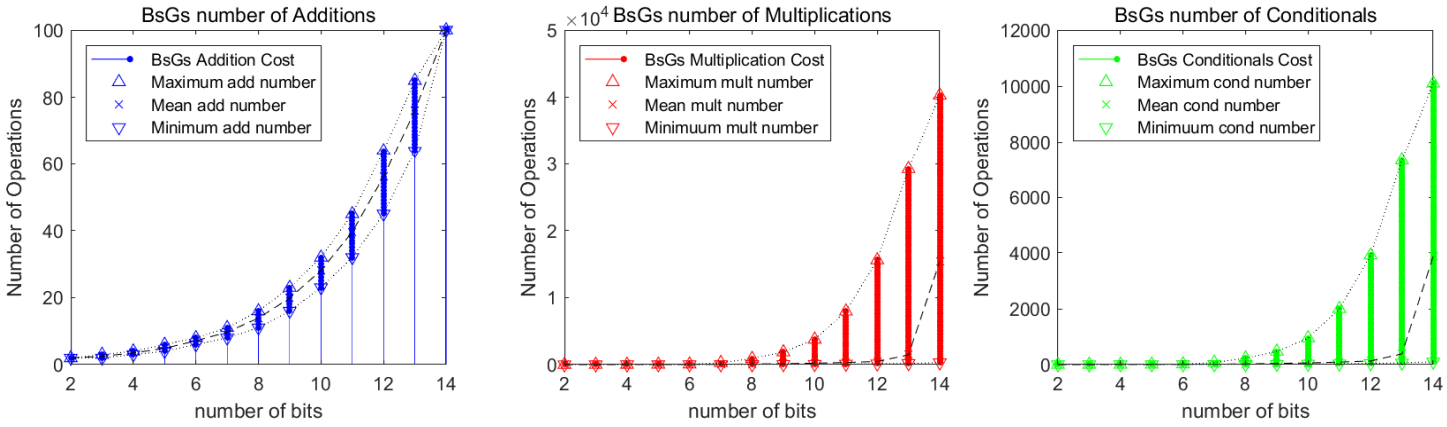


Figure 5.8: Baby Step - Giant Step

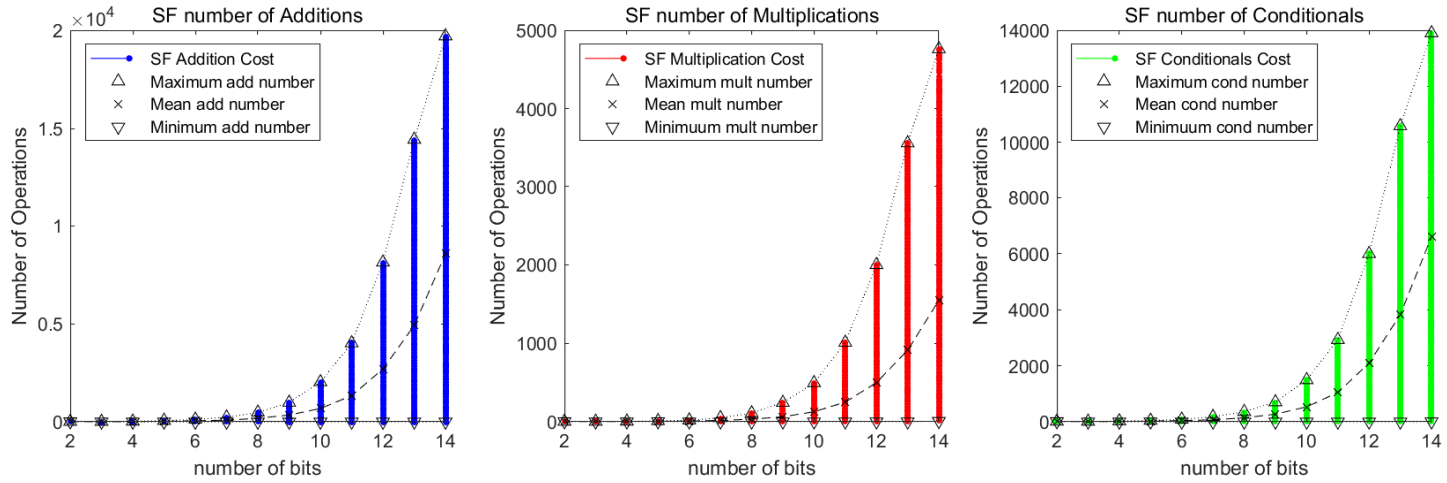


Figure 5.9: SF

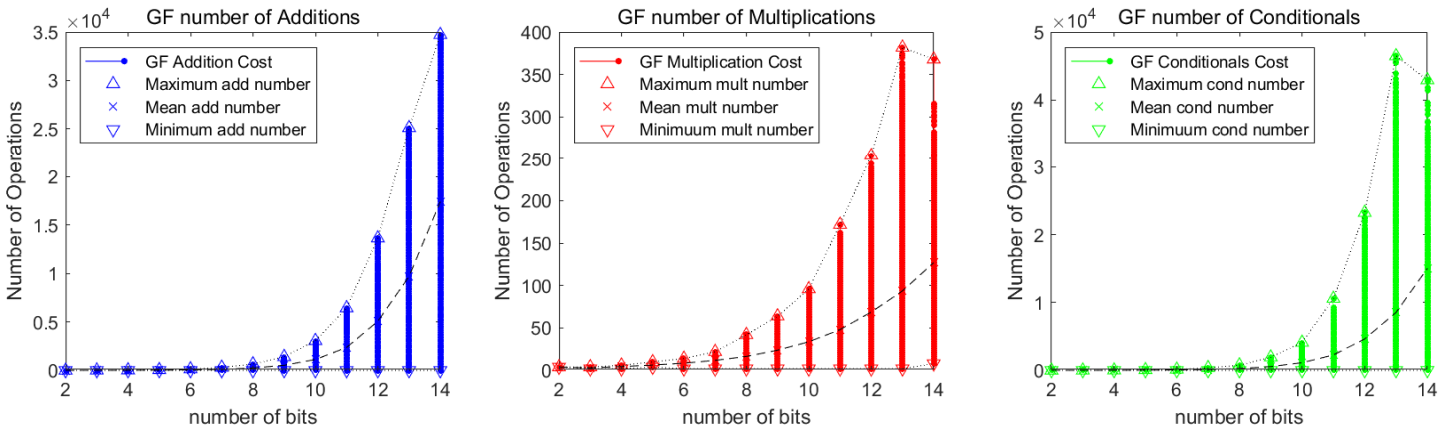


Figure 5.10: GF

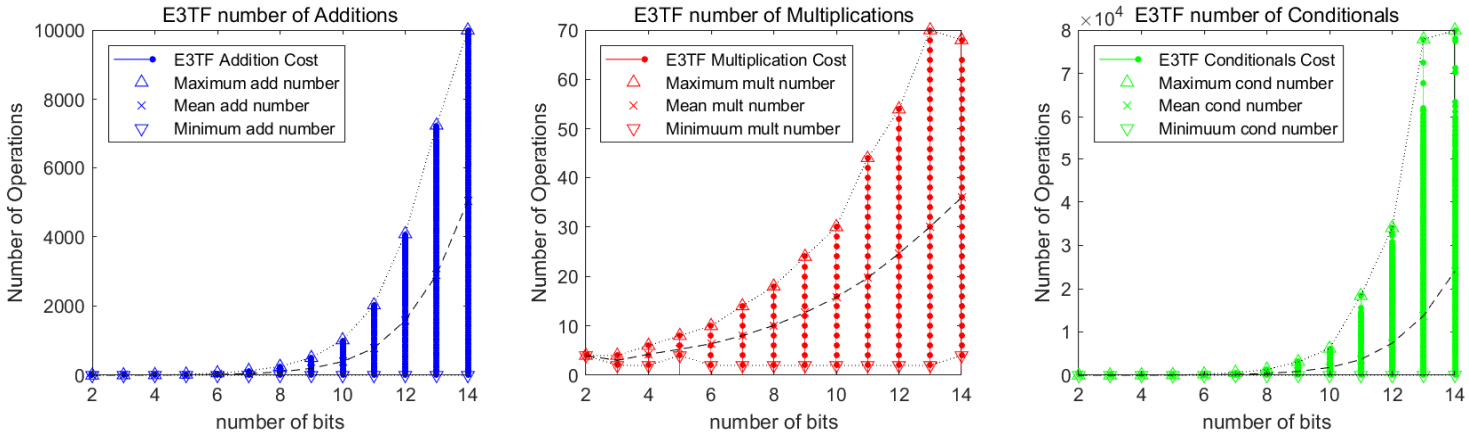


Figure 5.11: E3TF

We can see that E3TF algorithm has the fewest multiplications, while Baby Step - Giant step has the most. We can also observe some odd behaviour of Baby Step - Giant Step algorithm for 14-bit prime numbers, as there is an inexplicable increase of multiplications and conditional operators. We cannot provide a thorough explanation for this event, but a very probable reason would be the small sample of 14 bit numbers used in our experiments, due to lack of resources.

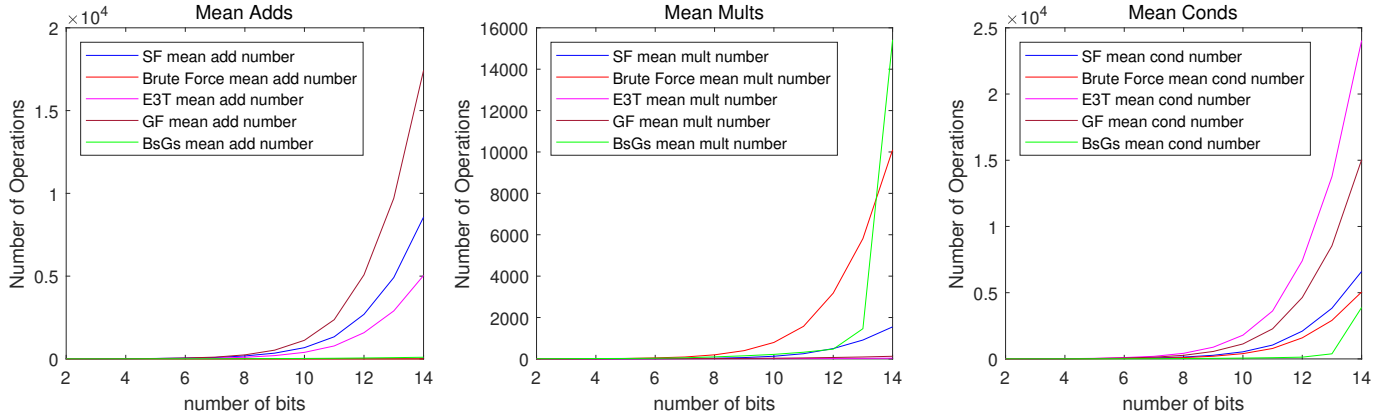


Figure 5.12: Mean number of Operations for each algorithm

Conclusion

Again, if we take into account the computational complexity of each operation, for prime numbers up to 13 bits, Baby Step-Giant Step algorithm seems to perform significantly better than the rest, while our algorithms still manage to "cost" less than the multiplication-based brute force algorithm.

Chapter 6

Conclusions and Future work

In this thesis, our goal was to present some observations on primitive roots of prime numbers. We successfully proved that our observations are also valid for every primitive element of a finite field $GF(p^m)$. Then, using

$$a^x = \pm a^k \pm a^j \ (\text{mod}(\ p),$$

for $k = \min(PoI)$ and some $j \in \mathbb{Z}_p, k, j < x$, we proposed an algorithm that could calculate $\langle a \rangle$, replacing multiplications with additions. We proposed 3 different versions of the algorithm:

- Simple Form
- Generalized Form
- Enhanced 3-term Form

Each version capitalized on properties we proved.

We used our algorithm as a potentially "fast" Lehmer pseudo-random number generator[8], and after simulations, we concluded that all forms of our algorithm are indeed faster than brute force, in terms of operations' cost.

Another suggested application for our algorithm was the discrete logarithm problem, a problem with applications in many public key cryptography methods, the most common one being Diffie-Hellman key exchange[1][2].
We showcased Shanks' Baby Step-Giant Step algorithm[11] on cracking DH, and concluded that even for small values of primes, our algorithm performs worse than BsGs. We observed some anomalies on the operational cost of Baby Step - Giant Step algorithm for 14-bit prime numbers, but we attributed them to the small sample of 14-bit numbers used in the simulations. Given time and resources, further research on this odd behavior of the algorithm for "mid-sized" numbers would perhaps provide a more thorough explanation.

There are many interesting pathways going forwards, as this topic has many applications. Starting from the theoretic perspective of our algorithm, as we can see in figures 4.3 and 4.5, the distribution of $\frac{\min(PoI)}{p-1}$ is Gaussian. Hence, we would be interested in further researching the reasons behind this distribution, and perhaps use the statistical properties of the distribution to more effectively find Points of Interest.

Another important factor that we have not taken into account throughout this thesis is dealing with memory. If we were to ever apply this algorithm in real life, we would have to

think of "smart" ways to store the information needed.

Furthermore, our algorithm is a fast way to compute continuous powers of a primitive root. This means that we could apply this to Shanks' Baby Step-Giant Step, as it also contains a step in which it computes continuous powers of a primitive root, and perhaps slightly improve its runtime. Again, we do not expect astonishing results, as we only improve the non-dominant term complexity-wise.

Finally, even though the theory behind prime numbers is relatively old, one cannot say that we know everything there is to know about them. Having said that, I believe that in time we could stumble across something new, something that could possibly tie everything we observed together.

> *"I think prime numbers are like life. They are very logical but you could never work out the rules, even if you spent all your time thinking about them."*
>
> *Mark Haddon,*
> *"The Curious Incident of the Dog in the Night-Time"*

Bibliography

[1] R. C. Merkle, "Secure communications over insecure channels," *Communications of the ACM*, vol. 21, no. 4, pp. 294–299, 1978.

[2] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pp. 365–390, 2022.

[3] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[5] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, pp. 124–134, Ieee, 1994.

[6] G. N. Karystinos, *Syllabus for subject: Number Theory and Cryptography.* TUC, 2021.

[7] D. E. Knuth, "Fundamental algorithms," *The art of computer programming*, vol. 1, pp. 261–268, 1997.

[8] D. H. Lehmer, "Mathematical methods in large-scale computing units," in *Proceedings of the Second Symposium on Large Scale Digital Computing Machinery*, (Cambridge, United Kingdom), pp. 141–146, Harvard University Press, 1951.

[9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, "Handbook of applied cryptography," pp. 71, 103–113, 2001.

[10] D. Harvey and J. Van Der Hoeven, "Integer multiplication in time o(nlogn)," 2019.

[11] D. Shanks, "Class number, a theory of factorization, and genera," in *Proc. of Symp. Math. Soc., 1971*, vol. 20, pp. 415–440, 1971.

[12] J. Katz and Y. Lindell, "Introduction to modern cryptography," pp. 352–353, 2020.