FINITE FIELD ARITHMETIC AND ITS APPLICATIONS TO CRYPTOGRAPHY

ANUBHAV SHARMA

Abstract. Fields are sets with properties that allow us to perform arithmetic within them. Cryptography, literally defined as secret writing, is used to ensure confidentiality and authentication of information when transferring over an insecure channel. In this paper, we explore the applications of finite field arithmetic in cryptography by explaining the Advanced Encryption Standard (private-key) and Elliptic Curve Cryptography (public-key).

Contents

Date: April 28, 2023.

This document is a joint thesis submitted to the Department of Mathematics and Statistics and Department of Computer Science at Haverford College in partial fulfillment of the requirements for a double major in Mathematics and Computer Science.

1. Introduction

Performing arithmetic over the real numbers is very common. While we may have taken it for granted, arithmetic over real numbers follows certain properties. For instance, adding or multiplying two numbers, regardless of their order, gives the same result. Sets like the real numbers are examples of algebraic structures called fields. Fields have two binary operations and follow certain properties that allow us to perform arithmetic within them. While fields like the real numbers or rationals all contain infinitely many elements, there are fields with finitely many elements called finite fields.

On the other hand, cryptography is the idea of securely transferring information from one party to another over an insecure channel. The general scheme follows the sender encrypting the message, then sending it through the channel, and finally, the receiver decrypting the message to recover the original message. The encryption and decryption relies on a secret key. Based on the types of keys used, cryptography is divided into two types: private key cryptography and public key cryptography. In a private or symmetric key cryptosystem, the sender and the receiver use the same (or very similar) key for both encryption and decryption. In contrast, different keys are used for encryption and decryption for public (or asymmetric) key cryptosystem.

Another important ingredient for cryptography is the encryption function that when given a key, converts the original message to an encrypted text. One of the ways to construct these encryption functions is to employ arithmetic over finite fields. In this paper, by laying the foundations for field theory and cryptography, we look at the application of field arithmetic to cryptography by explaining the Advanced Encryption Standard (AES) and Elliptic Curve Digital Signature Algorithm (ECDSA).

As an example for a private key cryptosystem, we cover the Advanced Encryption Standard (AES) which was approved by the National Institute of Standards and Technology (NIST) in 2001 [16]. AES is one of the most popular symmetric key algorithms and has been used in web messaging like WhatsApp and Facebook Messenger [3], iMessages on iPhones [9], as well as cloud services like Amazon Web Services (AWS) [1]. The algorithm of AES relies on arithmetic within the finite field of 2^8 elements.

For a public key cryptosystem, Elliptic Curve cryptography is discussed. We will see how elliptic curves over finite fields can be used to define the Discrete Logarithm Problem and eventually be used to construct a digital signature scheme, namely the Elliptic Curve Digital Signature Algorithm

(ECDSA). Digital signatures can be used to ensure authentication between two parties. ECDSA is also widely used including authentication of commands between AWS Key Management Service [1], signatures for iMessages in iPhone [9] as well as to verify untraceable payments for digital cash like Bitcoin [10].

2. Finite Fields

This chapter is intended to provide the mathematical preliminaries necessary to understand the cryptographic applications in latter sections. We begin this chapter by covering the necessary background in field theory and polynomials over fields. Then, we will prove some important theorems including the Fundamental Theorem of Field Theory and extending it to prove the existence and uniqueness of finite fields.

2.1. Introduction to Finite Fields.

In this section, we will introduce fields and finite fields as well as cover some examples.

Definition 2.1 (Field)**.** A nonempty set $\mathbb{F}$ is a **field** if it has two binary operations, addition $(+)$ and multiplication $(\cdot)$, satisfying following properties:

(1) (Closure) For a, b $\in \mathbb{F}$,
$$a+b \in \mathbb{F} \quad \text{and} \quad a \cdot b \in \mathbb{F}.$$
(2) (Associativity) For a, b $\in \mathbb{F}$,
$$a+(b+c)=(a+b)+c \text{ and } a\cdot(b\cdot c)=(a\cdot b)\cdot c.$$
(3) (Additive Identity) There exists an element $0 \in \mathbb{F}$ for every element $a \in \mathbb{F}$ such that $a+0=a$.
(4) (Multiplicative Identity) There exists an element $1 \in \mathbb{F}$ for every element $a \in \mathbb{F}$ such that $a \cdot 1 = a$.
(5) (Additive Inverse) For every element $a \in \mathbb{F}$, there exists a unique element $-a \in \mathbb{F}$ such that $a+(-a)=(-a)+a=0$. The element $-a$ is called the additive inverse of a.
(6) (Multiplicative Inverse) For every element $a \in \mathbb{F}-\{0\}$, there exists an element $a^{-1} \in \mathbb{F}$ such that $a \cdot a^{-1} = a^{-1} \cdot a = 1$. The element a^{-1} is called the multiplicative inverse of a.
(7) (Commutativity) For a, b $\in \mathbb{F}$,
$$a+b=b+a \text{ and } a\cdot b = b \cdot a.$$
(8) (Distributivity) For a, b, c $\in \mathbb{F}$,
$$a\cdot(b+c)=a\cdot b+a\cdot c \text{ and } (a+b)\cdot c = a\cdot c+b\cdot c.$$

Example 2.2. The set of integers $\mathbb{Z}$ is not a field because it does not satisfy property (6). For instance, 7 doesn't have a multiplicative inverse in $\mathbb{Z}$ since $1/7 \notin \mathbb{Z}$.

Example 2.3. Under the ordinary operations of addition and multiplication, the set of rational numbers $\mathbb{Q}$, the set of real numbers $\mathbb{R}$, and the set of complex numbers $\mathbb{C}$ all form a field. All these fields contain infinitely many elements.

Example 2.4. Consider another set $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ where 0 serves as the additive identity and 1 as the multiplicative identity. Addition and Multiplication are closed, associative, and distributive under modular arithmetic (mod 5). The additive inverses of $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ are $\{0, 4, 3, 2, 1\} \in \mathbb{Z}_5$ respectively. The multiplicative inverses of $\mathbb{Z}_5{}^{\times} = \{1, 2, 3, 4\}$ are $\{1, 3, 2, 4\} \in Z_5{}^{\times}$ respectively.

We can be convinced that $\mathbb{Z}_5$ above satisfies all field properties and hence, is a field. In fact, it contains finitely many elements. This makes a good transition into the core idea of the thesis: finite fields.

Definition 2.5 (Finite Fields; Ch. 3 Sec 2 of [15])**.** A field with m elements ($m \in \mathbb{N}$) is called a **finite field** of order m.

We saw $\mathbb{Z}_5$ which was a finite field with 5 elements. Below we will look at a slightly more interesting example of finite field with 4 elements.

Example 2.6. Consider a set of polynomials $\{0, 1, x, x + 1\}$ where two binary operations (addition and multiplication) are defined as:

$+$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

$\cdot$	1	x	$x+1$
1	1	x	$x+1$
x	x	$x+1$	1
$x+1$	$x+1$	1	x

This is another example of a finite field which is explained in more detail in Example 2.25.

Definition 2.7 (Characteristic of a Field; Sec 22.1 of [12])**.** A field $\mathbb{F}$ has **characteristic** n if n is the smallest positive integer such that for every nonzero element α in $\mathbb{F}$, we have

$$\underbrace{\alpha + \alpha + \cdots + \alpha}_{\text{n times}} = 0 \text{ or simply, } \alpha n = 0$$

If no such integer exists, then $\mathbb{F}$ has characteristic 0.

Example 2.8. The characteristic of the finite field $\mathbb{Z}_5$ is 5 because for any element $k \in \mathbb{Z}_5{}^{\times}$, $k + k + k + k + k = 5k \equiv 0 \pmod 5$ and 5 is the smallest positive integer to satisfy this criteria. For instance, $1 + 1 + 1 + 1 + 1 = 0 \pmod 5$.

2.2. **Polynomials over Fields.** All of us have probably seen polynomials with real numbers as coefficients. Now that we know that the set of real numbers is just an example of a field, field properties allow us to extend our "well-known" polynomials to polynomials over fields.

Definition 2.9 (Polynomials over a field; Sec 17.1 of [12])**.** Any expression of the form

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

where $a_i \in \mathbb{F}$, is called a polynomial over the field $\mathbb{F}$ with **indeterminate** x. If n is the largest non-negative number for which $a_n \neq 0$, we say that the **degree** of f is n. We will denote the set of all the polynomials in x with coefficients in the field $\mathbb{F}$ by $\mathbb{F}[x]$.

Example 2.10. Let $f(x) = 3x^2+3x+5$ and $g(x) = 4x+3$ be two polynomials over $\mathbb{Z}_7$. The degree of $f(x)$ is 2 and of $g(x)$ is 1. Adding $f(x)$ and $g(x)$ gives us $f(x) + g(x) = (3x^2 + 3x + 5) + (4x + 3) = 3x^2 + 7x + 8 \equiv 3x^2 + 1$. Multiplying $f(x)$ and $g(x)$ gives us $f(x) \cdot g(x) = (3x^2 + 3x + 5) \cdot (4x + 3) = 12x^3 + 21x^2 + 29x + 15 \equiv 5x^3 + x + 1$.

In a similar way, we can define derivatives for our polynomials over fields. Unlike for polynomials over reals, derivatives for polynomials over fields cannot be explained using the definition of the limit.

Definition 2.11 (Derivatives; Ch 19 of [7])**.** Let $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \in \mathbb{F}[x]$. The **derivative** of $f(x)$ denoted by $f'(x)$ is a polynomial

$$f'(x) = na_n x^{n-1} + (n-1)a_{n-1}x^{n-2} + \cdots + a_1 \in \mathbb{F}[x].$$

Not only we can add and multiply the polynomials over fields but also we can divide them using the Division Algorithm below.

Theorem 2.12 (Division Algorithm; Sec 17.2 of [12])**.** *Let $f(x)$ and $g(x)$ be polynomials in $\mathbb{F}[x]$, where $\mathbb{F}$ is a field and $g(x)$ is a nonzero polynomial. Then there exist unique polynomials $q(x), r(x) \in \mathbb{F}[x]$ such that*

$$f(x) = g(x)q(x) + r(x),$$

where degree(r(x)) < degree(g(x)).

Example 2.13. Let $f(x) = 2x^2 + x$ and $g(x) = 3x + 1$ be two polynomials over $\mathbb{Z}_5$. By long division, noting that addition and multiplication are in $\mathbb{Z}_5$, it can be found that $q(x) = 4x + 4$ and $r(x) = 1$. Therefore, by the division algorithm,

$$f(x) = 2x^2 + x = (3x + 1)(4x + 4) + 1.$$

In the Division Algorithm, the polynomial $r(x)$ is called the remainder and there is usually a faster way of computing $r(x)$ without having to perform the long division through the Division Algorithm. The Remainder Theorem below shows us how the remainder can be computed in a special case.

Theorem 2.14 (Remainder Theorem; Thm 16.2 of [7])**.** *Let $\mathbb{F}$ be a field, $a \in \mathbb{F}$, and $f(x) \in \mathbb{F}[x]$. Then $f(a)$ is the remainder in the division of $f(x)$ by $(x - a)$.*

Now, we will see below what happens when the remainder $r(x)$ is the zero polynomial.

Theorem 2.15 (Factor Theorem; Thm 16.2 of [7])**.** *Let $\mathbb{F}$ be a field, $a \in \mathbb{F}$, and $f(x) \in \mathbb{F}[x]$. The element a is a zero of $f(x)$ if and only if $(x - a)$ is a factor of $f(x)$.*

For readers interested in the proof of above theorems, we direct them to Theorem 16.2 of [7].

Note that the Factor Theorem follows from the Remainder Theorem and the Division Algorithm. The element a is a zero of the polynomial when $f(a) = 0$. By the Remainder Theorem, we know that the remainder is zero, by the Division Algorithm $(x - a)$ must be a factor of $f(x)$.

From the Division Algorithm, we saw how a large polynomial can be broken down into different factors if the remainder is 0. Similar to when performing any arithmetic within the fields, we need to make sure that each of the resulting polynomial lies over the field considered. For instance, in the Division Algorithm, all polynomials $f(x)$, $g(x)$, $q(x)$, and $r(x)$ were in the same ring $\mathbb{F}[x]$ and particularly, $r(x)$ was zero if we are interested in factorization. We will now explore when this is not always possible over fields by looking at irreducible polynomials.

Definition 2.16 (Irreducible polynomials; Sec 17.3 of [12])**.** A non-constant polynomial $f(x) \in \mathbb{F}[x]$ is **irreducible** over a field $\mathbb{F}$ if $f(x)$ cannot be expressed as a product of two polynomials $g(x)$ and $h(x)$ in $\mathbb{F}[x]$, where the degrees of $g(x)$ and $h(x)$ are both smaller than the degree of $f(x)$.

Example 2.17. The polynomial $f(x) = x^2 + 1$ is irreducible over $\mathbb{R}[x]$ because we cannot factor $f(x)$ into $(x - a_1)(x - a_2)$ where a_1 and a_2 belong to $\mathbb{R}$.

2.3. Fundamental Theorem of Field Theory.

In the previous section, we concluded that not all polynomials can be reduced over a particular field. Fortunately, there are ways to reduce these irreducible polynomials if we consider a large enough field. That is the general idea behind the Kronecker's Theorem which is also called the Fundamental Theorem of Field Theory. We will first see what these "large" fields or, more formally, Extension Fields mean.

Definition 2.18 (Extension Fields; Ch 19 of [7])**.** A field $\mathbb{E}$ is an **extension field** of a field $\mathbb{F}$ if $\mathbb{F} \subseteq \mathbb{E}$ and the operations of $\mathbb{F}$ are those of $\mathbb{E}$ restricted to $\mathbb{F}$. Also, $\mathbb{F}$ is called a **subfield** of $\mathbb{E}$.

Example 2.19. $\mathbb{R}$ is an extension field of $\mathbb{Q}$ and $\mathbb{C}$ is an extension field of $\mathbb{R}$.

Example 2.20. When looking at field extensions, one caveat to consider is that not all subsets of a finite field $\mathbb{F}$ are necessarily subfields of $\mathbb{F}$. For instance, in Example 2.35, we will see that $\mathbb{F}_8 = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$. In Example 2.25, we will see that $\mathbb{F}_4 = \{0, 1, x, x+1\}$. We know that $\mathbb{F}_2 = \{0, 1\}$. Just by looking at sets, we can see that $\mathbb{F}_2 \subseteq \mathbb{F}_4 \subseteq \mathbb{F}_8$. While $\mathbb{F}_2$ is a subfield of both $\mathbb{F}_4$ and $\mathbb{F}_8$, $\mathbb{F}_4$ is not a subfield of $\mathbb{F}_8$. For readers interested in the details, we refer them to Theorem 21.4 of [7].

In Example 2.20, we saw that both $\mathbb{F}_4$ and $\mathbb{F}_8$ are extension fields of $\mathbb{F}_2$. This raises another issue: how large of an extension field should we consider when we are attempting to reduce an irreducible polynomial over a larger field. The concept of splitting fields addresses this issue.

Definition 2.21 (Splitting Fields; Ch 19 of [7])**.** Let $\mathbb{E}$ be an extension field of a field $\mathbb{F}$ and let $f(x) \in \mathbb{F}[x]$ with degree of at least 1. We say that $f(x)$ **splits** in $\mathbb{E}$ if there are elements $a \in \mathbb{F}$ and $a_1, a_2, \ldots, a_n \in \mathbb{E}$ such that

$$f(x) = a(x - a_1)(x - a_2) \cdots (x - a_n).$$

We call $\mathbb{E}$ a **splitting field** for $f(x)$ over $\mathbb{F}$ if $\mathbb{E} = \mathbb{F}(a_1, a_2, \ldots, a_n)$. That is, $\mathbb{E}$ is the smallest field that contains $\mathbb{F}$ and all of the elements $a_1, \ldots, a_n$.

Example 2.22. We will go back to our Example 2.17. The polynomial $f(x) = x^2 + 1$ in $\mathbb{R}[x]$ cannot be reduced over $\mathbb{R}$ but can be factored over $\mathbb{C}$ as

$$f(x) = (x - i)(x + i).$$

We know that $\mathbb{C}$ is the smallest field larger that $\mathbb{R}$ that contains i and $-i$. In other words, $\mathbb{C} = \mathbb{R}(i, -i)$. Thus, $\mathbb{C}$ is a splitting field for $f(x)$ over $\mathbb{R}$.

Before we go on to proving the Fundamental Theorem of Field Theory, we need to dive a little deeper into how we can construct a field from polynomials. We will begin by defining ideals in $\mathbb{F}[x]$.

Definition 2.23 (Ideals in polynomials over fields; Ch 17 of [12])**.** Let $\mathbb{F}$ be a field. A principal ideal in $\mathbb{F}[x]$ is an ideal $\langle p(x) \rangle$ generated by some polynomial $p(x)$, that is,

$$\langle p(x) \rangle = \{p(x)q(x) \mid q(x) \in \mathbb{F}[x]\}.$$

Intuitively, ideal $\langle p(x) \rangle$ generated by some polynomial $p(x)$ is the set of all the multiples of $p(x)$ in $\mathbb{F}[x]$. The ideals of irreducible polynomials can be used to construct fields using following theorem.

Theorem 2.24 (Constructing fields from ideals; Theorem 17.5 from [7])**.** *Let $\mathbb{F}$ be a field and $p(x)$ be an irreducible polynomial over $\mathbb{F}$. Then $\mathbb{F}[x]/\langle p(x)\rangle$ is a field, where*

$$\mathbb{F}[x]/\langle p(x)\rangle = \{f(x) + \langle p(x)\rangle \mid f(x) \in \mathbb{F}[x]\}.$$

For readers interested in a proof, we refer them to Theorem 17.5 in [7]. Instead, we will focus on an example below.

Example 2.25. The polynomial $p(x) = x^2 + x + 1$ is irreducible in $\mathbb{Z}_2[x]$. Therefore, by the above theorem, $\mathbb{Z}_2[x]/\langle x^2 + x + 1\rangle$ must be a field. Note that $\mathbb{Z}_2[x]/\langle x^2 + x + 1\rangle = \{ax + b + \langle x^2 + x + 1\rangle \mid a, b \in \mathbb{Z}_2\}$. This field has four elements $\{0, 1, x, x + 1\}$.
Adding coset representations in $\mathbb{F}[x]/\langle x^2 + x + 1\rangle$ results in:

$+$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Similarly, multiplying the coset representations in $\mathbb{F}[x]/\langle x^2 + x + 1\rangle^\times$ results in:

$\cdot$	1	x	$x+1$
1	1	x	$x+1$
x	x	$x+1$	1
$x+1$	$x+1$	1	x

Note that x and $(x + 1)$ are multiplicative inverses of each other. Our irreducible polynomial is $x^2 + x + 1$ which implies $x^2 + x = -1 \equiv 1 \pmod 2$. Hence, the product $x(x + 1) = x^2 + x \equiv 1$. Likewise, we can verify all properties of fields listed in Definition 2.1 for $\mathbb{F}[x]/\langle x^2 + x + 1\rangle$.

Now, we have the necessary ingredients to prove the Kronecker's Theorem (Fundamental Theorem of Field Theory).

Theorem 2.26 (Kronecker's Theorem; Thm 19.1 of [7])**.** *Let $\mathbb{F}$ be a field, and let $f(x)$ be a non-constant polynomial in $\mathbb{F}[x]$. Then there is an extension field $\mathbb{E}$ of $\mathbb{F}$ in which $f(x)$ has a zero.*

While the structure of this proof has been taken from Thm 19.1 of [7], the exposition was enhanced by the author.

Proof. Given a polynomial $f(x) \in \mathbb{F}[x]$, we first factorize as $f(x) = p_1(x)p_2(x)\cdots p_k(x)$ where all $p_i(x)$ are irreducible factors. We take any one of those irreducible factors, denoted by $p(x)$. By Theorem 2.24, we know that $\mathbb{F}[x]/\langle p(x)\rangle$ is a field, which will be our candidate for the extension field $\mathbb{E}$. Now, we need to show that $p(x)$ has a zero in $\mathbb{F}[x]/\langle p(x)\rangle$.

We claim that polynomial $x + \langle p(x)\rangle$ is a zero of $p(x)$ in this extension field $\mathbb{E} = \mathbb{F}[x]/\langle p(x)\rangle$. Let $p(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_0$, for some $a_i \in \mathbb{F}$. Therefore,

$$\begin{aligned}
p(x + \langle p(x)\rangle) &= a_n(x + \langle p(x)\rangle)^n + a_{n-1}(x + \langle p(x)\rangle)^{n-1} + \cdots + a_0 \\
&= a_n(x^n + \langle p(x)\rangle) + a_{n-1}(x^{n-1} + \langle p(x)\rangle) + \cdots + a_0 \\
&= a_nx^n + a_n\langle p(x)\rangle + a_{n-1}x^{n-1} + a_{n-1}\langle p(x)\rangle + \cdots + a_0 \\
&= a_nx^n + a_{n-1}x^{n-1} + \cdots + a_0 + (a_n + a_{n-1} + \cdots + a_1)\langle p(x)\rangle \\
&= a_nx^n + a_{n-1}x^{n-1} + \cdots + a_0 + \langle p(x)\rangle \\
&= p(x) + \langle p(x)\rangle \\
&= 0 + \langle p(x)\rangle.
\end{aligned}$$

Since $p(x)$ has a zero in $\mathbb{F}[x]/\langle p(x)\rangle$ and $p(x)$ was one of the factors of $f(x)$, $f(x)$ also has a zero in $\mathbb{F}[x]/\langle p(x)\rangle$. $\square$

Given any polynomial over a field, Kronecker's Theorem allows us to construct an extension field where the polynomial is reducible. An example might help us understand this better.

Example 2.27. Let $f(x) = x^5 + x^2 + 2x + 2 \in \mathbb{Z}_3[x]$. This can be factored as $f(x) = (x^2 + 2)(x^3 + x + 1)$. Both $(x^2 + 2)$ and $(x^3 + x + 1)$ are irreducible over $\mathbb{Z}_3$. Let us take $(x^3 + x + 1)$ as our $p(x)$. Our extension field of $\mathbb{Z}_3[x]$ is $\mathbb{Z}_3[x]/\langle x^3 + x + 1\rangle$. Note that this field is in the form of

$$\mathbb{Z}_3[x]/\langle x^3 + x + 1\rangle = \{ax^2 + bx + c + \langle x^3 + x + 1\rangle \mid a, b, c \in \mathbb{Z}_3\}.$$

The claim is that $x + \langle x^3 + x + 1\rangle$ is a root $f(x)$ within this extension field $\mathbb{Z}_3[x]/\langle x^3 + x + 1\rangle$. Let us compute $f(x + \langle x^3 + x + 1\rangle)$:

$$\begin{aligned}
&f(x + \langle x^3 + x + 1\rangle) \\
&= (x + \langle x^3 + x + 1\rangle)^5 + (x + \langle x^3 + x + 1\rangle)^2 + 2(x + \langle x^3 + x + 1\rangle) + 2 \\
&= (x^5 + \langle x^3 + x + 1\rangle) + (x^2 + \langle x^3 + x + 1\rangle) + (2x + \langle x^3 + x + 1\rangle) + 2 \\
&= x^5 + x^2 + 2x + 2 + \langle x^3 + x + 1\rangle \\
&= (x^2 + 2)(x^3 + x + 1) + \langle x^3 + x + 1\rangle \\
&= 0 + \langle x^3 + x + 1\rangle.
\end{aligned}$$

Hence, $f(x)$ has a zero in $\mathbb{Z}_3[x]/\langle x^3 + x + 1\rangle$.

2.4. **Criterion for Unique Zeros.**

We began our discussion by attempting to reduce polynomials over fields and used Kronecker's Theorem in the last section to prove that it is always possible to construct an extension field where a polynomial over a field has at least one zero. Note that this extension field may not necessarily be the splitting field. For splitting fields, we need to have every single root of the polynomial to be included in the extension field. In this section, we will explore when the roots of the polynomial are unique.

Definition 2.28 (Separable polynomials; Ch 22 of [12])**.** Let $\mathbb{F}$ be a field. A polynomial $f(x) \in \mathbb{F}[x]$ of degree n is **separable** if it has n distinct roots in the splitting field of $f(x)$.

Example 2.29. The polynomial $f(x) = x^2 + 1$ in $\mathbb{R}[x]$ is separable since it has two (degree of the polynomial) distinct roots i and $-i$ in the splitting fields $\mathbb{C}$ over $\mathbb{R}$.

Theorem 2.30 (Criterion for Unique Zeros; Lemma 22.5 of [12])**.** *Let $\mathbb{F}$ be a field and $f(x) \in \mathbb{F}[x]$. Then $f(x)$ is separable if and only if $f(x)$ and $f'(x)$ are relatively prime. Recall that relatively prime means that the two polynomials do not share a common polynomial factor of degree greater than 0.*

The structure of this proof was inspired by Lemma 22.5 of [12].

Proof. $\Rightarrow$: Let $f(x)$ be a separable polynomial of degree n. Then, by Definition 2.28, $f(x)$ has n distinct roots in the splitting field of $f(x)$ over $\mathbb{F}[x]$. In this extension field, $f(x)$ can be factored as $f(x) = a(x-a_1)(x-a_2)\cdots(x-a_n)$ with $a_i \neq a_j$ for $i \neq j$. Taking the derivative of $f(x)$ by using the product rule, we get,

$$\begin{aligned}
f'(x) = {} & a(x-a_2)(x-a_3)(x-a_4)\cdots(x-a_i)\cdots(x-a_n) \\
& + a(x-a_1)(x-a_3)(x-a_4)\cdots(x-a_i)\cdots(x-a_n) \\
& + a(x-a_1)(x-a_2)(x-a_4)\cdots(x-a_i)\cdots(x-a_n) \\
& + \cdots \\
& + a(x-a_1)(x-a_2)\cdots(x-a_{i-1})(x-a_{i+1})\cdots(x-a_n) \\
& + \cdots \\
& + a(x-a_1)(x-a_2)(x-a_3)\cdots(x-a_i)\cdots(x-a_{n-1}).
\end{aligned}$$

Given an arbitrary factor $(x-a_i)$ in $f(x)$, we can see that $(x-a_i)$ appears on every term of $f'(x)$ except the i^{th} term. Therefore, it is not a factor of $f'(x)$. Hence, $f(x)$ and $f'(x)$ do not have any common factors.
$\Leftarrow$: We will prove the contrapositive of the other half by showing that if $f(x)$ is not separable, then $f(x)$ and $f'(x)$ have a common factor. Since $f(x)$ is not separable it can be written as $f(x) = (x-a)^k g(x)$, where $k > 1$. Then,

$$f'(x) = k(x-a)^{k-1}g(x) + (x-a)^k g'(x).$$

Clearly, $(x-a)^{k-1}$ is a common factor for $f(x)$ and $f'(x)$. □

2.5. **Existence and Uniqueness of Finite Fields.**

Now we will focus our exposition on finite fields and prove the existence and the uniqueness of finite fields. We need to first strengthen our Definition 2.7 on the characteristic of finite fields.

Theorem 2.31 (Prime Characteristic of Fields; Sec 22.1 and Thm 16.19 of [12])**.** *If $\mathbb{F}$ is a finite field, then the characteristic of $\mathbb{F}$ is p, where p is prime.*

This proof was taken from Thm 16.19 of [12].

Proof. Let the characteristic of $\mathbb{F}$ be p. For the nonzero element $1 \in \mathbb{F}$, it must be the case that $p \cdot 1 = 0$. Suppose p is not prime, then $p = mn$ for some m and n such that $1 < m < p$ and $1 < n < p$. Therefore, $(mn) \cdot 1 = 0$ which implies, $(m \cdot 1)(n \cdot 1) = 0$. Since $\mathbb{F}$ is a field and has no zero divisors, it must be that either $m \cdot 1 = 0$ or $n \cdot 1 = 0$. As 1 is an element in the field, without loss of generality, $m \cdot 1 = 0$ and this is a contradiction to our original assumption that characteristic of $\mathbb{F}$ is p since $m < p$. Hence, p must be prime. □

The fact that the characteristic of a finite field is always prime allows us to get a nice relation when we raise the sum or product of the field elements to the power of p.

Theorem 2.32 (Frobenius endomorphism; Proposition 35 of [5])**.** *Let $\mathbb{F}$ be a field of characteristic p and $n \in \mathbb{N}$. Then for any $a, b \in \mathbb{F}$,*

$$(a+b)^{p^n} = a^{p^n} + b^{p^n}, \text{ and } (ab)^{p^n} = a^{p^n} b^{p^n}.$$

The structure of the proof for the base case was inspired from Proposition 35 of [5]. The inductive case was worked out by the author.

Proof. We will proceed by induction for this proof.

Base Case: $n = 1$

We need to prove that $(a+b)^p = a^p + b^p$ and $(ab)^p = a^p b^p$. The Binomial Theorem states that,

$$(a+b)^p = a^p + \binom{p}{1} a^{p-1} b + \cdots + \binom{p}{k} a^{p-k} b^k + \cdots + b^p \text{ where,}$$

$$\binom{p}{k} = \frac{p!}{r!(p-k)!}.$$

Since $\mathbb{F}$ is a field, from Theorem 2.31, we know that p is prime. Then, this translates to,

$$\binom{p}{k} = \frac{p!}{r!(p-k)!} = \frac{p(p-1)!}{r!(p-k)!}.$$

Since p is prime, $r!(p-k)!$ cannot cancel p in the numerator and the coefficient will be $p \cdot \left(\frac{(p-1)!}{r!(p-k)!}\right)$. Note that the characteristic of the field is p so all these intermediate terms with multiples of p is 0. Hence, $(a+b)^p = a^p + b^p$. By properties of fields (Definition 2.1), $(ab)^p = a^p b^p$ is trivially true.

Inductive Case: $n = k$
Suppose this holds for all k where $1 \leq k \leq n$, we need to prove that the same is the case for $k+1$. So by the induction hypothesis, this relation holds for all k where $1 \leq k \leq n$, and

$$(a+b)^{p^{k+1}} = ((a+b)^p)^{p^k} = (a^p + b^p)^{p^k} = (a^p)^{p^k} + (b^p)^{p^k} = a^{p^{k+1}} + b^{p^{k+1}}.$$

$$(ab)^{p^{k+1}} = (ab)^{p^k}(ab)^p = (a^{p^k} b^{p^k} a^p b^p) = (a^{p^k} a^p)(b^{p^k} b^p) = a^{p^{k+1}} b^{p^{k+1}}.$$

Hence,

$$(a+b)^{p^n} = a^{p^n} + b^{p^n}, \text{ and } (ab)^{p^n} = a^{p^n} b^{p^n}.$$

□

Remark: Note that while proving the Frobenius endomorphism, we started with assumption of a field but only used some of the field properties like commutativity. In Theorem 2.33, we will employ this flexibility by using Frobenius endomorphism to a set that satisfies essential but not all field properties.

Theorem 2.33 (Existence and Uniqueness of Finite Fields; Thm 22.6 of [12])**.** *For every prime p and every positive integer n, there exists a unique finite field $\mathbb{F}$ with p^n elements which is isomorphic to the splitting field of $x^{p^n} - x$ over $\mathbb{Z}_p$.*

The structure of the introduction and statements on closure and inverses were adapted from Thm 22.6 of [12] while the conclusion on uniqueness of fields was inspired from Thm 21.1 of [7]. The exposition was enhanced by the author.

Proof. Let $\mathbb{F}$ be the splitting field of $f(x) = x^{p^n} - x$ over $\mathbb{Z}_p$. Now, the formal derivative of $f(x)$ is $f'(x) \equiv p^n x^{p^n-1} - 1$. Since the coefficient of x^{p^n-1} is p^n, in $\mathbb{Z}_p$, $f'(x) = -1$. Therefore, $f'(x)$ and $f(x)$ do not have any common factors of deg > 1 so by using Theorem 2.30, $f(x)$ is separable. That is, it has p^n distinct roots in its splitting field $\mathbb{F}$. We will call these roots $\alpha_1, \alpha_2 \cdots \alpha_{p^n}$. By Kronecker's Theorem (Theorem 2.26), we can be sure that $\mathbb{F} = \mathbb{Z}_p[\alpha_1, \alpha_2 \cdots \alpha_{p^n}]$.

Under the same binary operators addition $(+)$ and multiplication $(\cdot)$ of $\mathbb{Z}_p$, our extension $\mathbb{F}$ satisfies the fundamental field properties like existence of additive and multiplicative identity, commutativity, associativity, and distributivity. Below we will be proving other field properties. Given two arbitrary roots, say α, and β, we know that $0 = \alpha^{p^n} - \alpha$, and $0 = \beta^{p^n} - \beta$. This implies, $\alpha^{p^n} = \alpha$ and $\beta^{p^n} = \beta$. Now, the following is true:

(1) (Closure) $\alpha + \beta \in \mathbb{F}$.
$\alpha + \beta = \alpha^{p^n} + \beta^{p^n} = (\alpha + \beta)^{p^n}$ [From Thm 2.32]

Thus, $f(\alpha+\beta) = (\alpha+\beta)^{p^n} - (\alpha+\beta) = (\alpha+\beta) - (\alpha+\beta) = 0$.
Hence, $(\alpha+\beta)$ is a root of $f(x)$.

(2) (Closure) $\alpha \cdot \beta \in \mathbb{F}$.
$\alpha \cdot \beta = \alpha^{p^n} \cdot \beta^{p^n} = (\alpha \cdot \beta)^{p^n}$ [From Thm 2.32]
Thus, $f(\alpha \cdot \beta) = (\alpha \cdot \beta)^{p^n} - (\alpha \cdot \beta) = (\alpha \cdot \beta) - (\alpha \cdot \beta) = 0$.
Hence, $(\alpha \cdot \beta)$ is a root of $f(x)$.

(3) Additive inverse $-\alpha \in \mathbb{F}$. [Without Loss of Generality]
If p is odd, p^n is also odd and thus,
$f(-\alpha) = (-\alpha)^{p^n} - (-\alpha) = -\alpha^{p^n} + \alpha = -(\alpha^{p^n} - \alpha) = 0$.
If $p = 2$,
$f(-\alpha) = (-\alpha)^{2^n} - (-\alpha) = \alpha + \alpha = 0$. [Since $\alpha \equiv -\alpha \pmod 2$]
Either way, $-\alpha$ is a root of $f(x)$.

(4) Multiplicative inverse $\alpha^{-1} \in \mathbb{F}$ for $\alpha \neq 0$.
Let $\alpha^{-1} = \alpha^{p^n-1}$. We can see that $(\alpha^{-1})^{p^n} = (\alpha^{p^n})^{-1} = \alpha^{-1}$.
Thus, $f(\alpha^{-1}) = (\alpha^{-1})^{p^n} - \alpha^{-1} = \alpha^{-1} - \alpha^{-1} = 0$.
Hence, α^{-1} is a root of $f(x)$.

Hence, $\mathbb{F}$ is a field and specifically, a splitting field of $x^{p^n} - x$ over $\mathbb{Z}_p$. For uniqueness, let $\mathbb{K}$ be any field with order p^n. Now, $\mathbb{K}^\times$ is a multiplicative group with order $p^n - 1$. Therefore, any nonzero element α, satisfies $\alpha^{p^n-1} = 1$. That is, $\alpha^{p^n} = \alpha$ or $\alpha^{p^n} - \alpha = 0$. Since any nonzero element α in $\mathbb{K}$ is a zero of $f(x)$ and 0 in $\mathbb{K}$ is trivially a zero of $f(x)$, K is a splitting field of $f(x)$. However, any two splitting fields of $f(x)$ over a field $\mathbb{F}$ must be isomorphic. For readers interested in the details of isomorphism, we direct them to Corollary to Thm 19.4 in [7] or Corollary 21.36 in [12]. □

Definition 2.34 (Galois Field)**.** The unique finite field with p^n elements is called **Galois field** and represented by GF(p^n) or $\mathbb{F}_{p^n}$.

From Thm 2.31, we know that all finite fields must have prime characteristic, so Galois fields cover every finite field in algebra. Furthermore, we now know that finite fields exist and in fact are unique for a given prime p and $n \in \mathbb{N}$.

Example 2.35. The roots of $f(x) = x^{2^3} - x$ over $\mathbb{Z}_2[x]$ are elements of $\mathbb{F}_8$. The given polynomial is $f(x) = x^8 - x$. Just by inspection, we know that 0 and 1 are zeros of the polynomial. Therefore, by Thm 2.15, x and $(x-1)$ must be factors of $f(x)$. Note that in $\mathbb{Z}_2$, $-1 \equiv 1$ and thus, $(x-1) \equiv (x+1)$. Using Thm 2.14 and 2.12, we know that there exists $g(x)$ such that, $f(x) = x(x+1)g(x)$. By long division, we find that $g(x) = (x^6 + x^5 + x^4 + x^3 + x^2 + x + 1)$. Further, $g(x)$ can be factored as $(x^3 + x^2 + 1)(x^3 + x + 1)$. Altogether,

$$f(x) = x(x+1)(x^3 + x^2 + 1)(x^3 + x + 1).$$

Let α be a root of $(x^3 + x^2 + 1)$ and hence, a root of $f(x)$. Then, by long division, we can find that α, α^2, and $\alpha^2 + \alpha$ are the roots of the factor

(x^3+x+1). Using the closure property and some long division, we find that the roots of $f(x)$ are 0, 1, α, $\alpha+1$, α^2, α^2+1, $\alpha^2+\alpha$, and $\alpha^2+\alpha+1$. Now we can consider the following addition table for these roots:

$+$	0	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
0	0	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
1	1	0	$\alpha+1$	α	α^2+1	α^2	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$
α	α	$\alpha+1$	0	1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$	α^2	α^2+1
$\alpha+1$	$\alpha+1$	α	1	0	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$	α^2+1	α^2
α^2	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$	0	1	α	$\alpha+1$
α^2+1	α^2+1	α^2	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$	1	0	$\alpha+1$	α
$\alpha^2+\alpha$	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$	α^2	α^2+1	α	$\alpha+1$	0	1
$\alpha^2+\alpha+1$	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$	α^2+1	α^2	$\alpha+1$	α	1	0

Similarly, we can consider the following multiplication table for any of these roots except 0:

$\cdot$	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
1	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
α	α	α^2	$\alpha^2+\alpha$	α^2+1	$\alpha^2+\alpha+1$	1	$\alpha+1$
$\alpha+1$	$\alpha+1$	$\alpha^2+\alpha$	α^2+1	1	α	$\alpha^2+\alpha+1$	α^2
α^2	α^2	α^2+1	1	$\alpha^2+\alpha+1$	$\alpha+1$	α	$\alpha^2+\alpha$
α^2+1	α^2+1	$\alpha^2+\alpha+1$	α	$\alpha+1$	$\alpha^2+\alpha$	α^2	1
$\alpha^2+\alpha$	$\alpha^2+\alpha$	1	$\alpha^2+\alpha+1$	α	α^2	$\alpha+1$	α^2+1
$\alpha^2+\alpha+1$	$\alpha^2+\alpha+1$	$\alpha+1$	α^2	$\alpha^2+\alpha$	1	α^2+1	α

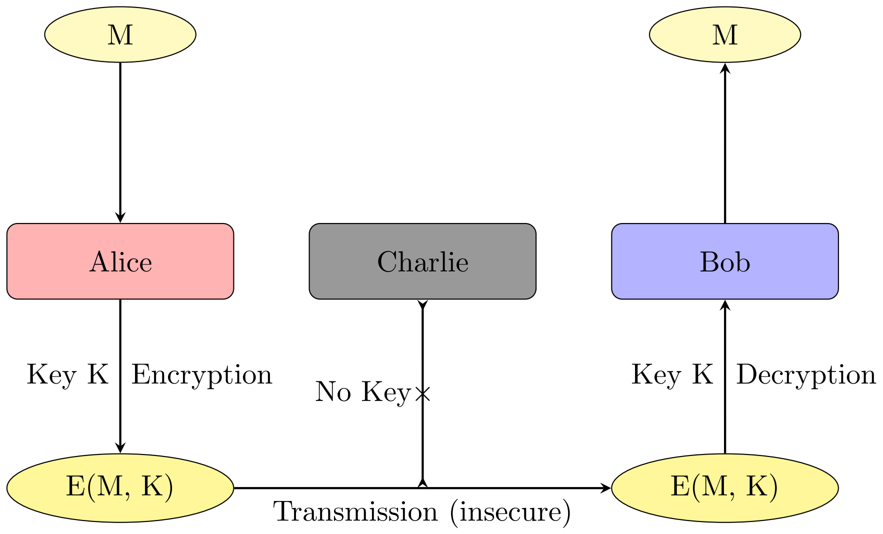
From the above tables, we can convince ourselves that the roots form a field. In short, the splitting field of $f(x) = x^{2^3} - x$ over $\mathbb{Z}_2[x]$ is isomorphic to

$$\mathrm{GF}(2^3) \equiv \mathbb{F}_8 \equiv \mathbb{Z}_2[x]/\langle x^3+x^2+1\rangle.$$

3. Cryptography

In this chapter, we will switch gears to learning about cryptography. The literal translation of the word Cryptography is "secret writing". It has existed for thousands of years, making information exchange more secure. Given two parties who wish to communicate via an insecure channel, the main goal of cryptography is to make sure that an adversary third party is unable to retrieve (or at least comprehend) the information exchanged. In all examples that follow, we will assume that Alice is trying to securely send message to Bob while Charlie is a third party trying to steal the information. Cryptography has been around for a while but in the modern world, it is more specific to communication over the internet between two users.

3.1. **Private (or symmetric) key cryptography.** Cryptography is classified into two categories: private, or symmetric, key cryptography and public, or asymmetric, key cryptography. In symmetric key cryptography, it is assumed that the Alice and Bob possess a common *secret key* that Charlie does not have access to. When sending a piece of information, Alice uses the *secret key* K to **encrypt** the message. The encrypted message is sent over the insecure channel to Bob. On the other end, Bob uses the same or in some cases, a new key that can be easily derived from the old key, to **decrypt** the message. However, even if Charlie is able to retrieve the message from the communication channel, he is not able to comprehend the message as he doesn't possess the key, by our underlying assumption. For message M and secret key K, if we represent encryption scheme as $E(M,K)$ and decryption scheme as $D(M,K) = E^{-1}(M,K)$, the following diagram represents the core idea behind symmetric key cryptography:



Consider the following simple example of symmetric key cryptography. Let A be the set of uppercase letters in the English alphabet. Define a mapping function $h(x)$ from A to $\mathbb{Z}_{26}$ as follows:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Now given a key $K \in \mathbb{Z}_{26}^{\times}$ and a letter $M \in A$ define an encryption scheme $E(M,K) = h^{-1}((h(M) + K) \bmod 26)$ and decryption scheme $D(M,K) = h^{-1}((h(M) - K) \bmod 26)$. For simplicity, let us assume that each letter is encrypted independently.

Suppose Alice and Bob agree on a key $K = 13$ and Alice wishes to send a message "MATH". She will encrypt the message as:

$$\begin{aligned} E(\text{``}M\text{''}, K) &= h^{-1}((h(\text{``}M\text{''}) + K) \bmod 26) \\ &= h^{-1}(12 + 13) \bmod 26) \\ &= h^{-1}(25 \bmod 26) \\ &= h^{-1}(25) \\ &= \text{``}Z\text{''}. \end{aligned}$$

Similarly, “A”, “T”, and “H” are encrypted to “N”, “G”, “U” respectively and the encrypted version of word “MATH” become “ZNGU”. This is then communicated via the insecure channel. When Bob receives the message, he decrypts it according to the defined decryption scheme as:

$$\begin{aligned} D(\text{“}Z\text{”}, K) &= h^{-1}((h(\text{“}Z\text{”}) - K) \bmod 26) \\ &= h^{-1}(25 - 13) \bmod 26) \\ &= h^{-1}(12 \bmod 26) \\ &= h^{-1}(12) \\ &= \text{“}M\text{”}. \end{aligned}$$

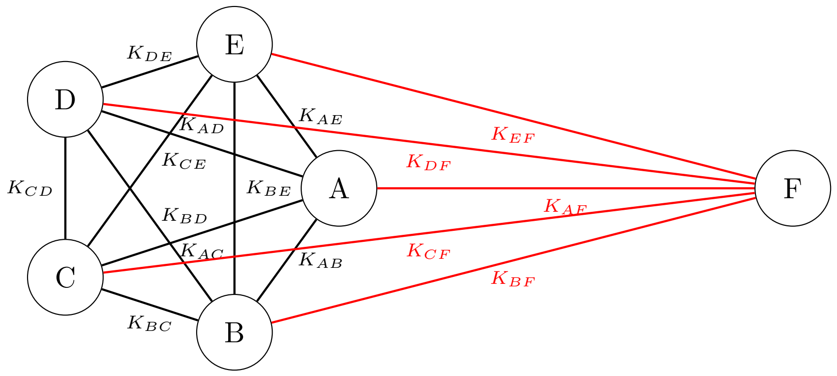
Likewise, Bob recovers the intended plain-text “MATH”. Even if Charlie intercepts the message during transmission, he recovers “ZNGU” and is not able to easily comprehend the message as he lacks the key $K = 13$. Of course, he could try all possible values of K and probably figure out the message in some time. This is why the size of key is important factor in deciding the strength of the cryptographic algorithm.

Note that the decryption scheme in a cryptography system must be inverse of the encryption scheme so as to recover the intended plain-text by the recipient. The general example presented above is known as Caesar cipher and when $K = 13$, it is specifically referred to as the ROT13 cipher. Some other examples of private key cryptography are Data Encryption Standard (DES) and Advanced Encryption Standard (AES). AES utilizes arithmetic in a finite field and will be studied in detail in Section 4.

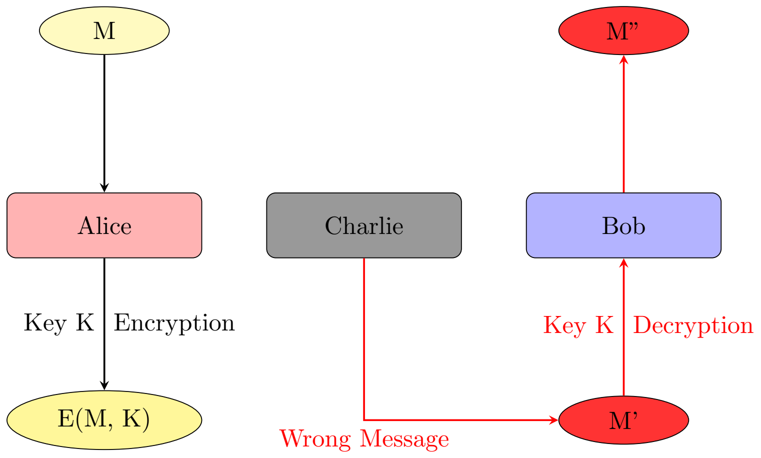
3.2. **Public (or asymmetric) key cryptography.**

While private key cryptography was widely used until the 1970s, as referenced in [10], it posed some problems such as Bob and Alice needed to exchange the key before exchanging information. Also, as the number of people (n) in the communication network increases, the number of keys required grows significantly. This can be illustrated with an example below.

Consider a communication channel with a group of 5 people: A, B, C, D, and E. Each of them possess a secret key with one another, that is, the keys are K_{AB}, K_{AC}, $\ldots$, K_{DE} totaling to 10 keys. Now suppose F wants to join the communication group. F needs to setup five additional keys (marked with red), one with each of existing parties: K_{AF}, K_{BF}, $\ldots$, K_{EF}. This brings up the total number of keys to 15. The number of keys required to be distributed can add up quickly for communication in large groups.



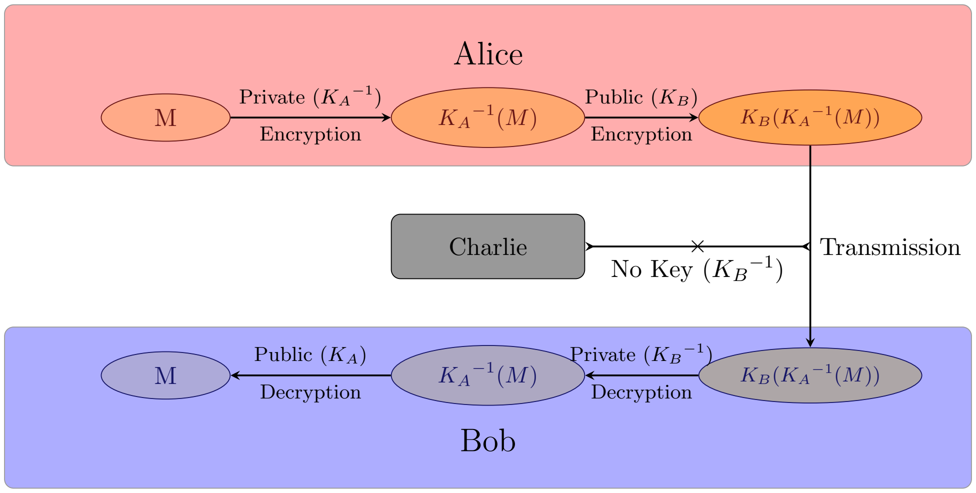
In addition to the key distribution problem, while the information could be protected, the source of the sender couldn't be verified. That is, Bob cannot be confident if the message actually came from Alice or if it was altered during transmission. The following illustration shows how Bob can receive a wrong message M'' but is unable to verify if Alice actually sent M''.



To address the key distribution problem and the problem of digital signatures, public key cryptography was introduced. In public key cryptography, there are two types of keys, the public key and the private key. Everyone has access to the public keys while only the individuals have access to their private keys. For instance, only Alice possesses her private key ${K_A}^{-1}$ and only Bob possesses his private key ${K_B}^{-1}$. Everyone has access to Alice's public key K_A and Bob's public K_B. The encryption key and decryption key are not the same and therefore, public key cryptography is also called asymmetric key cryptography.

Now if Alice wants to send Bob a message M, Alice can encrypt the message with Bob's public key K_B. The encrypted message $K_B(M)$ is sent over the communication channel and when received by Bob, he can use his private key ${K_B}^{-1}$ to recover the message as ${K_B}^{-1}(K_B(M)) = M$. The encrypted message cannot be read by Charlie as he doesn't have Bob's private key. The idea is that while everyone can encrypt the message to the sender, only the recipient can decrypt it. This scheme solves the key distribution problem. When a new member is added to the communication network, only two keys (public and private) need to be generated: the private key is distributed to the individual and the public key is published. This is much more efficient than private key cryptography where we need a unique set of keys for each possible combination of communication.

While the above scheme guarantees confidentiality, it still doesn't guarantee authentication. That is, Bob still can't be sure that the message he received was from Alice. This is because as Bob's public key is public, Charlie or anyone can send a message to Bob pretending to be Alice. Thus, we can improve the scheme by Alice encrypting the message first by her private key and then encrypting it again using Bob's public key. Then Bob can recover the message by first decrypting using his private key and then decrypting using Alice's public key to recover message M. If decrypting using Alice's public key gives the intended message, Bob can be sure that the message was sent by Alice as only she has access to her private key. The illustration below shows a general example of public key cryptography:



Note that the fundamental assumption here is that it is unfeasible to know private key K^{-1} when the public key K is known. Public key systems employ trapdoor one-way functions for encryption. These are functions that are easy to compute in one direction but whose inverse is infeasible to compute unless key K is known. For more information on trapdoor functions and a detailed implementation of public key cryptosystems, Section 2.1 of [10] can be consulted. Some common examples of public key cryptosystems are

RSA algorithm ([14]), Diffie-Hellman key exchange ([6]), and Elliptic-curve cryptography. Elliptic-curve cryptography will be studied in Section 5.

While public-key cryptography is more secure, it is more difficult to implement than private-key cryptosystems. According to [10], if Bob wants to send Alice a long message, he first uses an asymmetric cryptosystem to send Alice the key for a symmetric cryptosystem, and then he uses the symmetric cryptosystem to encrypt his message using best of both worlds.

4. Advanced Encryption Standard (AES)

In this chapter, we will look at a specific example of a popular private key cryptosystem: the Advanced Encryption Standard (AES). The Advanced Encryption Standard (AES) is a cryptographic algorithm approved by the Federal Information Processing Standard (FIPS) to protect electronic data. It is heavily used in web messaging like WhatsApp and Facebook Messenger [3], iMessages in iPhone [9] as well as cloud services like Amazon Web Services (AWS) [1].

It employs a symmetric block cipher to encrypt and decrypt information using the **Rijndael algorithm** specificed in [4]. AES processes data in blocks of 128 bits. That is, the input is 128 bits long and the output is also 128 bits long, where each bit is 0 or 1. The algorithm may be used with three different key lengths of 128 bits, 192 bits, and 256 bits thereby referred as AES-128, AES-192, and AES-256 respectively [16].

The basic unit of processing in the AES algorithm is called a **byte** which is a sequence of eight bits treated as a single entity [16]. The information to be encrypted in bytes for the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between braces in the order $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. According to [16], these bytes are interpreted as finite field elements in GF(2^8) using a polynomial representation:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{n=0}^{7} b_i x^i.$$

For example, $\{10100011\}$ identifies the specific finite field element $x^7 + x^5 + x + 1$.

Since there are a lot of bits, it is also conventional to represent them in hexadecimal values as follows:

Bits	Hex	Bits	Hex	Bits	Hex	Bits	Hex
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

For example, $\{10100011\} = \{1010\}\{0011\} = \{a3\}$.

The message to be encrypted is first translated to bits using American Standard Code for Information Interchange (ASCII) and grouped into blocks of 128 bits. Since each block is 128 bit and 1 byte is 8 bits, there are 16 bytes to consider in the AES algorithm. These 16 bytes are internally arranged as two-dimensional array which is called the **state** and represented by s [16]. The state consists of four rows of bytes. The input is copied into the state array where encryption operations are carried out and finally copied to the output array. Corresponding states and output for a 128-bit (16 bytes) input as $\{in_0, in_1, \ldots, in_{15}\}$ is given in the table (from [16]) below:

Input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

State bytes

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

Output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

More generally, for N_b bytes per row, the input array is copied to the State array according to following scheme:

$$s[r, c] = in[r + 4c] \text{ for } 0 \leq r < 4 \text{ and } 0 \leq c < N_b.$$

Similarly, after cipher operations are applied, the State array is copied to the output array as follows:

$$out[r + 4c] = s[r, c] \text{ for } 0 \leq r < 4 \text{ and } 0 \leq c < N_b.$$

Each column in the state array is called a **word** [16]. In the above example, there are four columns in the State array and therefore, four words, as follows:

$$\begin{aligned}
w_0 &= s_{0,0}s_{1,0}s_{2,0}s_{3,0} \\
w_1 &= s_{0,1}s_{1,1}s_{2,1}s_{3,1} \\
w_2 &= s_{0,2}s_{1,2}s_{2,2}s_{3,2} \\
w_3 &= s_{0,3}s_{1,3}s_{2,3}s_{3,3}
\end{aligned}$$

4.1. **Binary Operations: Addition and Multiplication.** So far we have defined that every byte is a finite field element in GF(2^8). Recall that a field is incomplete without its binary operations. In this section, we will define the two binary operations: addition and multiplication for the field elements.

4.1.1. *Addition.* In AES, adding two bytes is achieved by adding two polynomials in GF(2^8). Recall that adding of two polynomials in GF(2^8) is regular addition of the coefficients for corresponding powers expressed in $\mathbb{Z}_2$.

Example 4.1. Suppose we want to add two bytes $\{10100011\}$ and $\{01101001\}$.

$$\begin{aligned}&\{10100011\} + \{01101001\}\\ &= (x^7 + x^5 + x + 1) + (x^6 + x^5 + x^3 + 1)\\ &= (x^7 + x^6 + x^3 + x)\\ &= \{11001010\}.\end{aligned}$$

Note that this is equivalent to applying an XOR operator which is simply addition in mod 2. Recall that an XOR operator is defined as:

a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

If we use XOR in the above example, we get

$$\{10100011\} \oplus \{01101001\} = \{11001010\}.$$

4.1.2. *Multiplication.* In AES, multiplication of two bytes is achieved by multiplying two polynomials in GF(2^8) modulo the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ [16]. One way to quickly check that $m(x)$ is irreducible is by showing that it has no zeros in $\mathbb{F}_2$: the elements of $\mathbb{F}_2$ are 0 and 1 and clearly $m(1) \neq 0$ and $m(0) \neq 0$. Expressing in terms of ideals, our finite field is $\mathbb{Z}_2[x]/\langle x^8 + x^4 + x^3 + x + 1\rangle$.

Example 4.2. Suppose we want to multiply bytes $\{10100011\}$ and $\{01101001\}$.

$$\begin{aligned}&\{10100011\} \cdot \{01101001\}\\ &= (x^7 + x^5 + x + 1) \cdot (x^6 + x^5 + x^3 + 1)\\ &= (x^{13} + x^{12} + x^{11} + x^8 + x^4 + x^3 + x + 1).\end{aligned}$$

Now performing long division by $x^8 + x^4 + x^3 + x + 1$ gives

$$\begin{aligned}&(x^{13} + x^{12} + x^{11} + x^8 + x^4 + x^3 + x + 1) \bmod (x^8 + x^4 + x^3 + x + 1)\\ =\ &(x^5 + x^4 + x^3 + x^2 + x).\end{aligned}$$

Hence, the result is

$$\{10100011\} \cdot \{01101001\} = \{00111110\}.$$

Unlike addition, there is no simple operation at the byte level that corresponds to this multiplication. However, the multiplication in this finite field can be simplified into byte level interpretation by considering multiplication by polynomial x.

Theorem 4.3 (Multiplication by $\{02\}$; Section 4.2.1 of [16]). *In $GF(2^8)$ in AES, multiplication of a polynomial by $p(x) = x$ (or in byte-level $\{00000010\} = \{02\}$)is equivalent to a left shift or a left shift followed by subsequent bitwise XOR operator with $\{00011011\} = \{1b\}$. More compactly, if the byte is represented as $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, multiplication by $\{00000010\}$ is equivalent to $\{b_6b_5b_4b_3b_2b_1b_00\} \oplus b_7 \cdot \{00011011\}$.*

Proof. Given a byte $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, the equivalent polynomial representation is $b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$. Now multiplication of $b(x)$ by polynomial x is given by

$$\begin{aligned}
& x \cdot b(x) \\
&= x \cdot (b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0) \bmod \\
&\quad (x^8 + x^4 + x^3 + x + 1) \\
&= (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod \\
&\quad (x^8 + x^4 + x^3 + x + 1).
\end{aligned}$$

Case I: $b_7 = 0$.

$$\begin{aligned}
& x \cdot b(x) \\
&= (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod \\
&\quad (x^8 + x^4 + x^3 + x + 1) \\
&= (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \\
&\equiv \{b_6b_5b_4b_3b_2b_1b_00\}
\end{aligned}$$

This is exactly a left shift of byte $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$.

Case II: $b_7 = 1$.

$$\begin{aligned}
& x \cdot b(x) \\
&= (x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod \\
&\quad (x^8 + x^4 + x^3 + x + 1) \\
&= (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + \\
&\quad (x^8 \bmod (x^8 + x^4 + x^3 + x + 1)) \\
&= (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1) \\
&\equiv \{b_6b_5b_4b_3b_2b_1b_00\} \oplus \{00011011\}
\end{aligned}$$

This is exactly a left shift of the byte followed by XOR with $\{00011011\}$.

Hence, $\{b_7b_6b_5b_4b_3b_2b_1b_0\} \cdot \{00000010\} = \{b_6b_5b_4b_3b_2b_1b_00\} \oplus b_7 \cdot \{00011011\}$. □

4.2. **Polynomials with Word coefficients.** In the above section, we showed how each byte in AES is an element of GF(2^8). In this section, we will look at polynomials in GF(2^8). Recall that each column in the state array is called a word. That is, a word contains 4 bytes. Now given a word $[a_0, a_1, a_2, a_3]$, define a four-term polynomial where $a_i \in \mathbb{F}_{2^8}$ as:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

The coefficients a_i are themselves finite field elements in $\mathbb{F}_{2^8}$.

4.2.1. *Addition.* Given two words $[a_0, a_1, a_2, a_3]$ and $[b_0, b_1, b_2, b_3]$, the addition of words is performed as addition of polynomials in GF(2^8); that is, addition of coefficients of the like terms. Since coefficients themselves are finite field elements, this eventually translates to an XOR operation between the bytes. Mathematically,

$$\begin{aligned}&[a_0, a_1, a_2, a_3] + [b_0, b_1, b_2, b_3]\\&= (a_3x^3 + a_2x^2 + a_1x + a_0) + (b_3x^3 + b_2x^2 + b_1x + b_0)\\&= (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0).\end{aligned}$$

4.2.2. *Multiplication.* The multiplication of two words is performed in a similar fashion as multiplication of two polynomials. However, that result is a polynomial with degree 6 and not a word. Thus, it is reduced to a word equivalent by taking the product modulo a polynomial $m(x) = x^4 + 1$ for the AES algorithm. For more details, we refer the readers to [16].

Theorem 4.4 (Equivalency of mod by $x^4 + 1$; Equation 4.10 of [16])**.**

$$x^i \ mod \ (x^4 + 1) = x^{i \ mod \ 4}, \qquad where \ i \in \mathbb{N}.$$

Proof. We will prove this by induction for $i \in \mathbb{N}$. For the base case, we will consider cases where $i = 0, 1, 2, 3,$ and 4.

$$\begin{aligned}&x^0 \bmod (x^4 + 1) = 1 \bmod (x^4 + 1) = 1 = x^0 = x^{0 \bmod 4}.\\&x^1 \bmod (x^4 + 1) = x \bmod (x^4 + 1) = x = x^1 = x^{1 \bmod 4}.\\&x^2 \bmod (x^4 + 1) = x^2 \bmod (x^4 + 1) = x^2 = x^{2 \bmod 4}.\\&x^3 \bmod (x^4 + 1) = x^3 \bmod (x^4 + 1) = x^3 = x^{3 \bmod 4}.\\&x^4 \bmod (x^4 + 1) = 1 \bmod (x^4 + 1) = 1 = x^0 = x^{4 \bmod 4}.\end{aligned}$$

Now for the inductive case where $i > 4$, we can write i as $i = 4k + c$ where $k \in \mathbb{Z}$ and $c = i \bmod 4$. In that case,

$$\begin{aligned}
&x^i \bmod (x^4 + 1) \\
&= x^{(4k+c)} \bmod (x^4 + 1) \\
&= (x^{4k} \cdot x^c) \bmod (x^4 + 1) \\
&= (x^{4k} \bmod (x^4 + 1)) \cdot (x^c \bmod (x^4 + 1)) \\
&= (x^4 \bmod (x^4 + 1))^k \cdot (x^c \bmod (x^4 + 1)) \\
&= 1^k \cdot (x^c \bmod (x^4 + 1)) \\
&= x^c \bmod (x^4 + 1).
\end{aligned}$$

Since $c \in \{0, 1, 2, 3\}$, this reduces to the respective base case so the equality is true regardless. $\square$

Given two words $[a_0, a_1, a_2, a_3]$ and $[b_0, b_1, b_2, b_3]$,

$$\begin{aligned}
&[a_0, a_1, a_2, a_3] \cdot [b_0, b_1, b_2, b_3] \\
= &((a_3x^3 + a_2x^2 + a_1x + a_0) \cdot (b_3x^3 + b_2x^2 + b_1x + b_0)) \bmod (x^4 + 1) \\
= &((a_3 \cdot b_3)x^6 + \\
&((a_3 \cdot b_2) \oplus (a_2 \cdot b_3))x^5 + \\
&((a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3))x^4 + \\
&((a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3))x^3 + \\
&((a_2 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus (a_0 \cdot b_2))x^2 + \\
&((a_1 \cdot b_0) \oplus (a_0 \cdot b_1))x + \\
&(a_0 \cdot b_0)) \\
&\ \bmod (x^4 + 1) \\
= &(a_3 \cdot b_3)x^2 + \\
&((a_3 \cdot b_2) \oplus (a_2 \cdot b_3))x + \\
&((a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)) + \\
&((a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3))x^3 + \\
&((a_2 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus (a_0 \cdot b_2))x^2 + \\
&((a_1 \cdot b_0) \oplus (a_0 \cdot b_1))x + \\
&(a_0 \cdot b_0)) \\
= &((a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3))x^3 + \\
&((a_2 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus (a_0 \cdot b_2) \oplus (a_3 \cdot b_3))x^2 + \\
&((a_1 \cdot b_0) \oplus (a_0 \cdot b_1) \oplus (a_3 \cdot b_2) \oplus (a_2 \cdot b_3))x + \\
&((a_0 \cdot b_0) \oplus (a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)) \\
= &[((a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3)), \\
&((a_2 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus (a_0 \cdot b_2) \oplus (a_3 \cdot b_3)), \\
&((a_1 \cdot b_0) \oplus (a_0 \cdot b_1) \oplus (a_3 \cdot b_2) \oplus (a_2 \cdot b_3)), \\
&((a_0 \cdot b_0) \oplus (a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3))].
\end{aligned}$$

When $a(x)$ is a fixed polynomial and the resulting product is expressed as $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$, the coefficients can be written compactly in matrix form as:

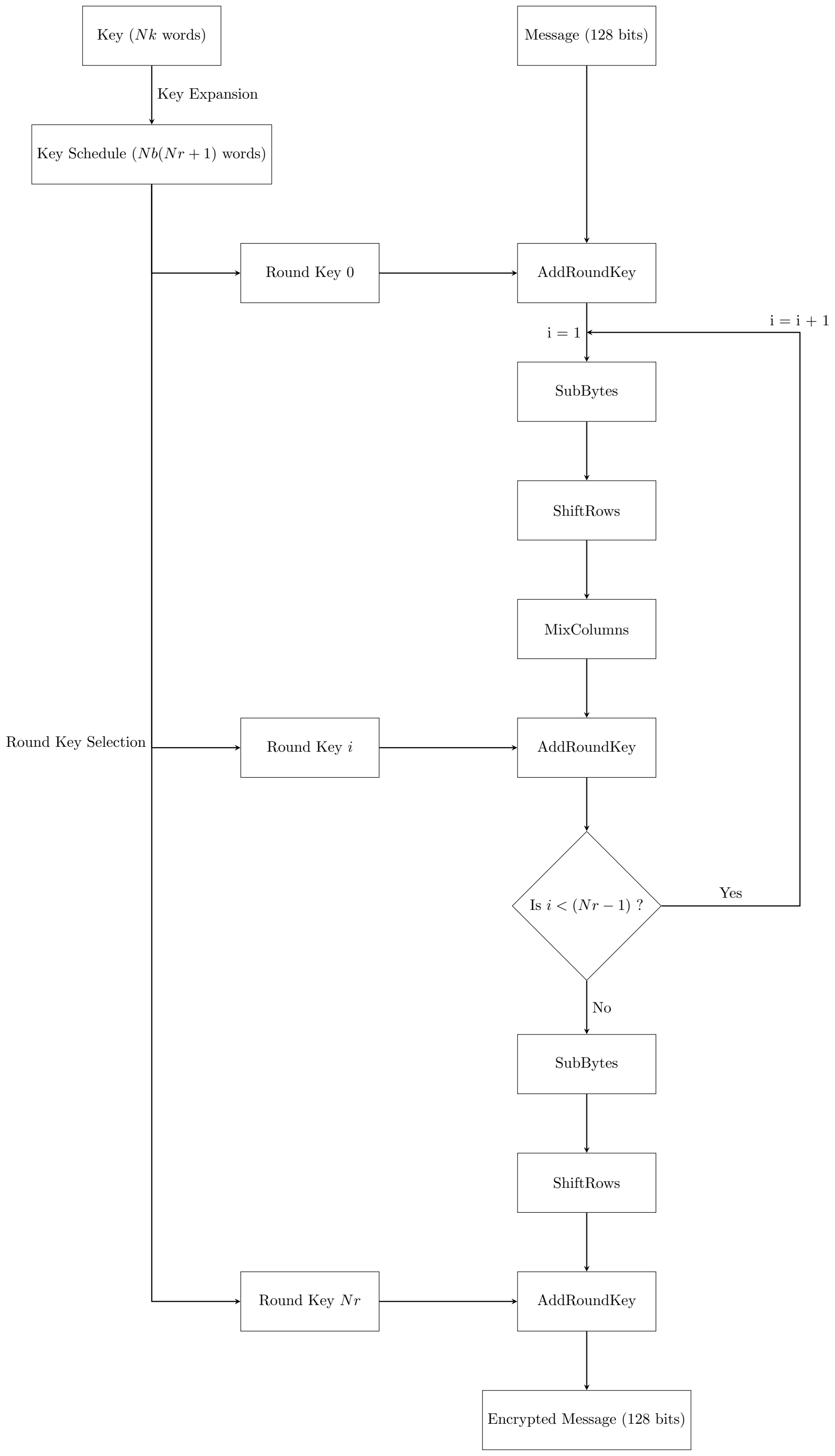
$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

In GF(2^8), $x^4 + 1 = (x + 1)^4$. Therefore, $x^4 + 1$ is not an irreducible polynomial in GF(2^8) and multiplication by a fixed four-term polynomial $a(x)$ is not necessarily invertible [16]. In fact, as specified in Section 2.1.7 in [11], a polynomial $a(x)$ has an inverse if the polynomial $(x + 1)$ does not divide it. We will see later in Section 4.3.3 that we choose a particular $a(x)$ that satisfies this condition and is invertible in the ring $\mathbb{F}_{2^8}[x]/\langle x^4 + 1\rangle$.

4.3. **Encryption.** We have all the prerequisite to now go into the details of encryption. The 128-bit input is first transformed into the state table as specified in Section 4. Then the first round key is added to the input similar to that described in Section 4.3.4. Details on how to generate a round key will be defined in Section 4.5. After initial addition of the round key, the encryption algorithm includes four transformations applied for a certain number of rounds. The number of rounds depends upon the key length as described in table below from [16]:

AES type	Key Length *(Nk)*	Block size *(Nb)*	Rounds *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Every round is identical and composed of following four transformations, except the final round where the *MixColumns()* step is omitted. The following flowchart summarizes the encryption process in AES:


Key (Nk words)
Key Expansion
Key Schedule ($Nb(Nr+1)$ words)
Message (128 bits)
Round Key 0
AddRoundKey
i = 1
i = i + 1
SubBytes
ShiftRows
MixColumns
Round Key Selection
Round Key i
AddRoundKey
Is $i < (Nr-1)$?
Yes
No
SubBytes
ShiftRows
Round Key Nr
AddRoundKey
Encrypted Message (128 bits)

4.3.1. *SubBytes().* The first step in every round is *SubBytes()* which performs independent transformation on each byte using finite field arithmetic. It comprises the following two steps:

(1) Take the multiplicative inverse of the element in the finite field GF(2^8). The element $\{00\}$ is mapped to itself, by definition.
(2) Apply the following affine transformation over GF(2):

$$b_i{}' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

for $0 \leq i < 8$, where b_i is the i^{th} bit of the byte resulting from step (1) and c_i is the i^{th} bit with the value $\{01100011\} = \{63\}$. For byte $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, this can be written more compactly in the matrix form as:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

In cryptography algorithms, involution functions provide symmetry between encryption and decryption, that is, the function is same as its inverse. However, the *SubBytes()* step is not an involution as *InvSubBytes()* (Section 4.4.1) is not the same as *SubBytes()*.

Example 4.5. Consider transformation of byte $\{00000001\}$. The multiplicative inverse is $\{00000001\}$. Applying the affine transformation results in

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

The result is $\{01111100\} = \{7c\}$. Hence, after the *SubBytes()* step, $\{01\}$ transforms to $\{7c\}$.

Recall that the bytes are elements of finite fields; that is, there are a finite (2^8) number of them and every element (except $\{00\}$) has a multiplicative inverse. Therefore, the two transformations described above can be reduced to a simple substitution represented in S-box Table 1. Going back to the

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

TABLE 1. S-box Table: For a byte $\{xy\}$ in hexadecimal, the transformed byte after applying *SubBytes()*.

above example $\{01\}$, the substituted value can be found in the row with index 0 and column with index 1 which is $\{7c\}$.

4.3.2. *ShiftRows()*. In this step, every row of the state is cyclically shifted to the left by the value determined by the row index. In other words, row 0 is not shifted but row 1, 2, and 3 are each shifted by 1, 2, and 3 units respectively. The table below shows the effect of *ShiftRows()* on the State array:

Before Shifting

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

After Shifting

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

The *ShiftRows()* operation is also not an involution.

4.3.3. *MixColumns()*. After row transformations, the algorithm operates on columns. As described in Section 4.2, each column of the state is treated as a four-term polynomial in GF(2^8). For this step from [16], the word polynomial is multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$ given by:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}.$$

Since x^4+1 is not irreducible in GF(2^8), not all ring elements has an inverse. However, this fixed polynomial $a(x)$ is chosen in a way to be invertible. The inverse is given in Section 4.4.3. The selection of this fixed polynomial $a(x)$ was done using the criteria of invertibility, diffusion, and performance. Multiplication of the polynomial by coefficients like 0 or 1 need no computation. From Theorem 4.3, multiplication by 2 can be computed and multiplication by 3 is simply XOR of multiplication by 2 and the operand [11]. For details on diffusion, we refer the reader to Section 3.4.3 of [11].

Suppose the word is $[s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}]$ and the result after *MixColumns()* is represented by $[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}]$. As described in Section 4.1.2, this can be written in matrix form as:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$
$$= \begin{bmatrix} (\{02\} \cdot s_{0,c}) \oplus (\{03\} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s_{0,c} \oplus (\{02\} \cdot s_{1,c}) \oplus (\{03\} \cdot s_{2,c}) \oplus s_{3,c} \\ s_{0,c} \oplus s_{1,c} \oplus (\{02\} \cdot s_{2,c}) \oplus (\{03\} \cdot s_{3,c}) \\ (\{03\} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \cdot s_{3,c}) \end{bmatrix}.$$

The *MixColumns()* operation is also not an involution especially as we need to multiply the result with the inverse of fixed polynomial $a(x)$ to invert the function and $a(x)$ is not identity.

4.3.4. *AddRoundKey().* This is the final step for each round. The corresponding round key is added to the State by simple XOR operation. This can be represented as:

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round\ *\ Nb\ +\ c}]$$

for $0 \leq c < Nb$ and $0 \leq round \leq Nr$.
Note that first round key is added at $round = 0$ even before *SubBytes()*. *AddRoundKey()* happens for every round after that. The method to generate round keys $[w_{round\ *\ Nb\ +\ c}]$ will be described in Section 4.5.

The *AddRoundKey()* operation is an involution as the XOR operation is an involution.

4.4. **Decryption.** Decryption is the inverse process of encryption, where we convert the encrypted message back to plaintext. Therefore, the order of transformations for decryption should be reverse of the encryption transformations. As given in [4], in decryption the inverse of a round should be given by:

```
InvRound(State, RoundKey) {
AddRoundKey(State, RoundKey);
```

```
    InvMixColumns(State);
    InvShiftRows(State);
    InvSubBytes(State);
    }
```

For the final round, since we omit the *MixColumns()* step:

```
    InvFinalRound(State, RoundKey) {
    AddRoundKey(State, RoundKey);
    InvShiftRows(State);
    InvSubBytes(State);
    }
```

Note that in decryption, the final round happens first which is then followed by subsequent general rounds and finally followed by a Round Key Addition. An example taken from [4], for a two-round variation, the Rijndael algorithm code looks like:

```
    AddRoundKey(State, ExpandedKey+2*Nb);
    InvShiftRows(State);
    InvSubBytes(State);
    AddRoundKey(State, ExpandedKey+Nb);
    InvMixColumns(State);
    InvShiftRows(State);
    InvSubBytes(State);
    AddRoundKey(State, ExpandedKey);
```

Since *ShiftRows()* just transposes the bytes and has no effect on the byte values, while *SubBytes()* works on independent bytes [4], *ShiftRows()* and *SubBytes()* commute. This results in following sequence:

```
    AddRoundKey(State, ExpandedKey+2*Nb);
    InvSubBytes(State);
    InvShiftRows(State);
    AddRoundKey(State, ExpandedKey+Nb);
    InvMixColumns(State);
    InvSubBytes(State);
    InvShiftRows(State);
    AddRoundKey(State, ExpandedKey);
```

Furthermore, the column mixing operations *MixColumns()* and *InvMixColumns()* are linear with respect to the column input, that is *InvMixColumns(State* $\oplus$ *RoundKey)* = *InvMixColumns(State)* $\oplus$ *InvMixColumns(RoundKey)* [16]. This means that the sequence

```
AddRoundKey(State, RoundKey);
InvMixColumns(State);
```

can be replaced by:

```
InvMixColumns(State);
AddRoundKey(State, InvRoundKey);
```

Adding this change as well results in the following sequence of transformations for our two-round inverse cipher [4]:

```
\\initial key addition
AddRoundKey(State, ExpandedKey+2*Nb);
\\ a general round
InvSubBytes(State);
InvShiftRows(State);
InvMixColumns(State);
AddRoundKey(State, Inv_ExpandedKey+Nb);
\\ final round
InvSubBytes(State);
InvShiftRows(State);
AddRoundKey(State, ExpandedKey);
```

The pseudo-code can be generalized to any number of rounds. Once we apply *InvMixColumns()* to all the round keys except the first and the last one, the decryption follows same structure to that of encryption. The individual transformations will be explained below.

4.4.1. *InvSubBytes().* This is the reverse of the the *SubBytes()* operation. We need to apply the inverse of the affine transformation described in Section 4.3.1 and take the multiplicative inverse of the byte in GF(2^8). Similar to S-box in Table 1, we can represent this inverse through inverse S-box in Table 2.

4.4.2. *InvShiftRows().* As the name suggests, this is the reverse of the *ShiftRows()* operation described in Section 4.3.2. Every row of the State is cyclically shifted by the value determined by the row index. In other words, row 0 is not shifted but row 1, 2, and 3 are each shifted by 3, 2, and 1 units respectively. The table below shows the effect of *ShiftRows()* on the State array:

Before Shifting

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

After Shifting

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,3}$	$s_{1,0}$	$s_{1,1}$	$s_{1,2}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,1}$	$s_{3,2}$	$s_{3,3}$	$s_{3,0}$

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

TABLE 2. Inverse S-box Table: For a byte $\{xy\}$ in hexadecimal, the transformed byte after applying *InvSubBytes()*.

4.4.3. *InvMixColumns().* *InvMixColumns()* reverses the mixing of columns done by *MixColumns()*. For this step from [16], the word polynomial is multiplied modulo x^4+1 with the inverse of the polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ in Section 4.3.3 given by:

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}.$$

Suppose the word is $[s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}]$ and the result is represented by $[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}]$. As described in Section 4.1.2, this can be written in matrix form as:

$$\begin{aligned}\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} &= \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \\ &= \begin{bmatrix} (\{0e\} \cdot s_{0,c}) \oplus (\{0b\} \cdot s_{1,c}) \oplus (\{0d\} \cdot s_{2,c}) \oplus (\{09\} \cdot s_{3,c}) \\ (\{09\} \cdot s_{0,c}) \oplus (\{0e\} \cdot s_{1,c}) \oplus (\{0b\} \cdot s_{2,c}) \oplus (\{0d\} \cdot s_{3,c}) \\ (\{0d\} \cdot s_{0,c}) \oplus (\{09\} \cdot s_{1,c}) \oplus (\{0e\} \cdot s_{2,c}) \oplus (\{0b\} \cdot s_{3,c}) \\ (\{0b\} \cdot s_{0,c}) \oplus (\{0d\} \cdot s_{1,c}) \oplus (\{09\} \cdot s_{2,c}) \oplus (\{0e\} \cdot s_{3,c}) \end{bmatrix}.\end{aligned}$$

4.4.4. *AddRoundKey().* Finally, we need to inverse *AddRoundKey()*, and since we are performing arithmetic in GF(2^8), *AddRoundKey()* is the inverse of itself and this is same as Section 4.3.4.

4.5. **Key Schedule.** This section specifies how to derive Round Keys from the cipher key with the help of the key schedule, which consists of the Key Expansion and the Round Key selection [4].

Depending upon the type of AES used, the key length could be 128, 192, or 256 bits, and the the number of rounds in algorithm could be 10, 12, or 14. The input length is 128 bits regardless of the key size. Following our convention in Section 4.3, we will denote number of words in a key as *Nk*, number of words in block (input) as *Nb*, and number of rounds as *Nr*.

4.5.1. *Key Expansion.* The key expansion uses the initial *Nk* words in the key to generate a total of *Nb(Nr + 1)* words. As an example, if we use AES-128 (10 rounds) for 128-bit (4 words) of input, the key expansion produces $4 \times (10+1) = 44$ words. The first 4 words will be used in the very beginning and each of 10 rounds will use 4 words, totalling to 44. Each word in the sequence will be denoted as $[w_i]$ with $0 \leq i < Nb(Nr+1)$.

The first *Nk* words of the expanded key are the same as the cipher key, but then every following word $w[i]$ is equal to the XOR of the previous word $w[i-1]$ and the word *Nk* positions earlier, $w[i - Nk]$ [16]. For words in positions that are multiple of *Nk*, a transformation is applied to $w[i-1]$ prior to XOR, followed by an XOR with a round constant $Rcon[i]$ [16]. The transformation consists of two functions *RotWord()* and *SubWord()*.

RotWord() simply performs a cyclic permutation to the words by transforming the word $[a_0, a_1, a_2, a_3]$ into $[a_1, a_2, a_3, a_0]$. *SubWord()* applies S-box (1) to each of the bytes in the word to produce an output word. $Rcon[i]$ contains the values given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$ with i starting from 1. Note that $x \in \mathrm{GF}(2^8)$ is byte $\{02\}$ and the multiplication by x is explained in Theorem 4.3.

The Key Exapnsion is a bit different for AES-256 (*Nk = 8*). For this case, if $i-4$ is a multiple of *Nk*, then *SubWord()* is applied to $w[i-1]$ before XOR [16].

4.5.2. *Round Key Selection.* After we have the expanded key, the specific round key for round i is the set of words between $w[Nb \cdot i]$ to $w[Nb(i+1)]$ [4]. Figure 1 shows an example of key expansion and round key selection for $N_b = 4$ and $N_k = 6$.

5. Elliptic Curve Cryptography

In this final chapter, we will explore the application of finite field arithmetic in public key cryptosystems, specifically looking at cryptography via elliptic curves. Elliptic curve cryptography is one of the popular public key cryptosystems used mainly in digital signatures, key exchanges, and encryption. In addition, other public cryptosystems like Diffie-Hellman Key Exchange, Elgamal, Massey-Omura, and Digital Signatures all have equivalent

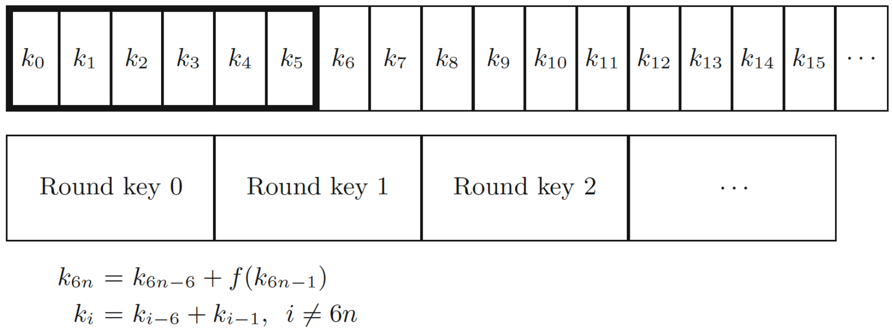

k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10} k_{11} k_{12} k_{13} k_{14} k_{15} $\cdots$
Round key 0
Round key 1
Round key 2
$\cdots$
$k_{6n} = k_{6n-6} + f(k_{6n-1})$
$k_i = k_{i-6} + k_{i-1},\ i \neq 6n$

Figure 1. Key Expansion and Round Key Selection for $N_b = 4$ and $N_k = 6$. The function f includes operation consists of *RotWord()* and *SubWord()* (Figure 3.10 of [11]).

analogues using elliptic curves over finite fields. Elliptic curve cryptosystems are more extensive, faster, and secure. As an example, while regular Diffie-Hellman Key exchange is limited to two parties, Elliptic Curve Diffie-Hellman can be extended to key exchange between three parties. Similarly, it has been found that a key size of 4096 bits for RSA gave the same security as 313 bits in an elliptic curve system [18].

This chapter will introduce elliptic curves, define the addition of points within elliptic curves, and construct elliptic curves over finite fields. With necessary background on elliptic curves over finite fields and the arithmetic on them, we will also discuss one of the popular applications of elliptic curves to cryptography: the Elliptic Curve Digital Signature Algorithm (ECDSA).

5.1. **Elliptic Curves.**

This section will focus on the mathematics behind the elliptic curves.

Definition 5.1 (Elliptic Curves; Section 6.1 of [10])**.** An **elliptic curve** E over the XY-plane is the set of solutions to a Weierstrass equation

$$E : Y^2 = X^3 + AX + B, \quad \text{where } X, Y, A, B \in \mathbb{R}.$$

together with an extra point $\mathcal{O}$, where the constants A and B must satisfy

$$4A^3 + 27B^2 \neq 0.$$

The quantity $\Delta_E = 4A^3 + 27B^2$ is called the discriminant of E and $\Delta_E \neq 0$ is equivalent to the condition that the cubic polynomial $X^3 + AX + B$ has three distinct roots. As per [10], having repeated roots doesn't work well with the addition of points on the curve. We will discuss more on the addition of points in Section 5.1.1.

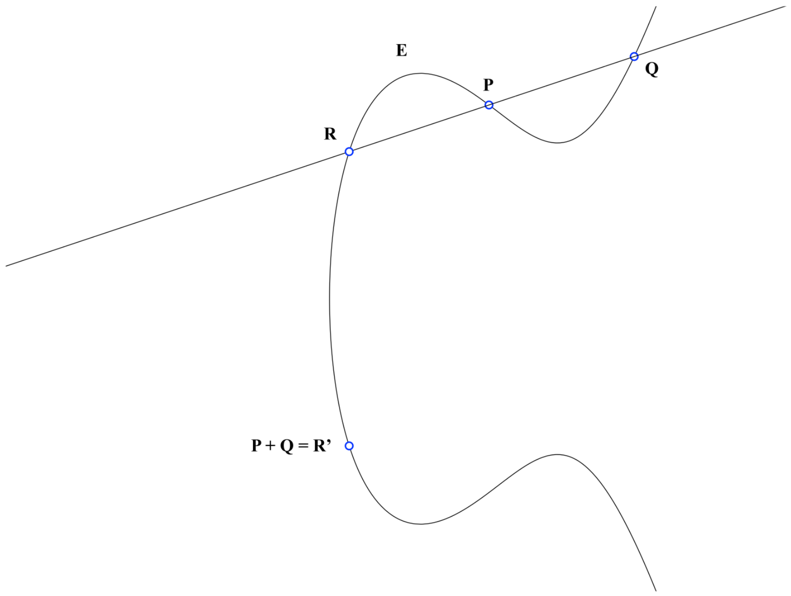


Figure 2. Animation showing addition of two points P and Q to give the sum R' on an elliptic curve E.
Note: Viewing animation requires Adobe Acrobat Reader.

The point $\mathcal{O}$ doesn't live in the XY-plane but is assumed to live at "infinity" in each vertical line. In fact, we will see that the point $\mathcal{O}$ serves as the additive identity or zero for addition.

Example 5.2. The curve $Y^2 = X^3 - 6X + 16$ is an elliptic curve as $4(-6)^3 + 27(16)^2 = 6048 \neq 0$. Some points that lie on the curve are $(0, 4)$, $(0, -4)$, $(3, 5)$, and $(3, -5)$.

5.1.1. *Addition on Elliptic curves.* We will now see how two points on an elliptic curve can be "added". Given two points P and Q on an elliptic curve E, we draw a line L passing through these points. The third point R where line L intersects the curve E is reflected over the X-axis to give the sum R'. That is, $P \oplus Q = R'$. Note that the same idea extends to adding a point to itself. As point Q tends to P, the line L becomes tangent to E. Another point where the tangent intersects the curve is reflected over the X-axis to give the result. Figure 2 represents the addition of points in an elliptic curve geometrically.

Throughout this section, we will represent the reflection of the point P on the X-axis by P'. Now consider adding P to P'. By definition of reflection, the line connecting P and P' is parallel to the Y-axis and does not intersect the curve anywhere else. That is, the sum is $\mathcal{O}$, which serves as the additive identity in elliptic curve addition. Therefore P and P' are additive inverses of each other. In short, $P \oplus P' = \mathcal{O}$ and we define $-P = P'$. In the next theorem, we will explore some of the properties of addition in elliptic curves.

Theorem 5.3 (Properties of Elliptic Curve Addition; Theorem 6.5 in [10])**.** *Let E be an elliptic curve. Then the addition law on E has the following properties:*

$$\begin{array}{lrl}
P \oplus Q = Q \oplus P & \textit{for all } P, Q \in E & \textit{[Commutative]} \\
P \oplus \mathcal{O} = \mathcal{O} \oplus P = P & \textit{for all } P \in E & \textit{[Identity]} \\
P \oplus (-P) = \mathcal{O} & \textit{for all } P \in E & \textit{[Existence of Inverse]} \\
(P \oplus Q) \oplus R = P \oplus (Q \oplus R) & \textit{for all } P, Q, R \in E & \textit{[Associative]}
\end{array}$$

While the structure of the proof is taken from Theorem 6.5 in [10], the exposition has been enhanced by the author.

Proof. Commutativity is straightforward as the line passing through P and Q is same as the the line passing through Q and P. For the identity, $P \oplus \mathcal{O}$ is geometrically a vertical line connecting $\mathcal{O}$ that lies on every vertical line and P. The third point that this line intersects the curve is P's reflection on the X-axis or P'. We need to reflect this on the X-axis to get the result, which is P. For the inverses, the same argument applies. The vertical line passing through a point and its reflection on the X-axis intersects the curve at point $\mathcal{O}$. Associativity is a bit challenging to prove in a few lines but can be verified with some algebraic calculations after we derive a formula for elliptic curve addition. □

Example 5.4. Given the point $P = (0, 4)$ and $Q = (3, 5)$ on the curve $Y^2 = X^3 - 6X + 16$, we will find $P \oplus Q$. First, the equation of the line passing through P and Q is given by:

$$y - 4 = \frac{5-4}{3-0}(x - 0)$$
$$y = \frac{1}{3}x + 4$$

Now substituting this line in the equation of the curve will allow us to find the third point of intersection.

$$\begin{aligned}
& \left(\frac{1}{3}X + 4\right)^2 = X^3 - 6X + 16 \\
\text{or,} \quad & \frac{1}{9}X^2 + \frac{8}{3}X + 16 = X^3 - 6X + 16 \\
\text{or,} \quad & X^3 - \frac{1}{9}X^2 - \frac{26}{3}X = 0 \\
\text{or,} \quad & X(X-3)\left(X + \frac{26}{9}\right) = 0
\end{aligned}$$

The X-coordinate of the third point is $-\frac{26}{9}$. Plugging this to the line or the curve gives us the Y-coordinate.

$$y = \left(\frac{1}{3}\right)\left(-\frac{26}{9}\right) + 4 = \frac{82}{27}$$

The third point is $R = \left(-\frac{26}{9}, \frac{82}{27}\right)$. Finally, the reflection of R in the X-axis is $R' = \left(-\frac{26}{9}, -\frac{82}{27}\right)$. Hence, $(0,4) \oplus (3,5) = \left(-\frac{26}{9}, -\frac{82}{27}\right)$.

This example can be generalized to derive an Elliptic Curve Addition Algorithm to add any two points on the elliptic curve.

Theorem 5.5 (Elliptic Curve Addition Algorithm; Theorem 6.6 in [10])**.** *Let $E : Y^2 = X^3 + AX + B$ be an elliptic curve, and let P_1 and P_2 be points on E.*

(1) *If $P_1 = \mathcal{O}$, then $P_1 + P_2 = P_2$.*
(2) *Otherwise, if $P_2 = \mathcal{O}$, then $P_1 + P_2 = P_1$.*
(3) *Otherwise, let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$.*
    (a) *If $x_1 = x_2$ and $y_1 = -y_2$, then $P_1 + P_2 = \mathcal{O}$.*
    (b) *Otherwise, $P_1 + P_2 = (x_3, y_3)$, where*

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\ \frac{3{x_1}^2 + A}{2y_1} & \text{if } P_1 = P_2. \end{cases}$$

The structure of the proof has been inspired from Theorem 6.6 in [10] with the concluding discussion on cases added by the author.

Proof. Proofs of statements (1), (2), and (3)(a) are trivial from Theorem 5.3 so we will focus on the proof of (3)(b).

In any case, λ represents the slope of the line joining P_1 and P_2. In the case when $P_1 \neq P_2$, the slope is simply given by $\frac{y_2 - y_1}{x_2 - x_1}$. When $P_1 = P_2$, this line is tangent to the elliptic curve at point P_1. To find the slope of the tangent, we take the derivative of the curve to find the slope.

$$\begin{aligned} Y^2 &= X^3 + AX + B \\ \frac{dY^2}{dX} &= \frac{d}{dX}\left(X^3 + AX + B\right) \\ 2Y\frac{dY}{dX} &= 3X^2 + A \\ \frac{dY}{dX} &= \frac{3X^2 + A}{2Y} \end{aligned}$$

Thus, at point $P_1(x_1, y_1)$, the slope of tangent is given by $\frac{3{x_1}^2 + A}{2y_1}$. Now the equation of the line is given by $y - y_1 = \lambda(x - x_1)$ or $y = \lambda(x - x_1) + y_1$.

We need to substitute this equation to the equation of the curve to find the point of intersection.

$$\begin{aligned} & Y^2 = X^3 + AX + B \\ \text{or,}\quad & (\lambda(x - x_1) + y_1)^2 = x^3 + Ax + B \\ \text{or,}\quad & \lambda^2(x - x_1)^2 + 2\lambda(x - x_1)y_1 + {y_1}^2 = x^3 + Ax + B \\ \text{or,}\quad & x^3 - \lambda^2 x^2 + (A - 2\lambda(y_1 - \lambda x_1))x + (B - (y_1 - \lambda x_1)^2) = 0. \end{aligned}$$

We know that the X-coordinates of the intersection points are x_1, x_2, and x_3 so the above equation must factor into:

$$\begin{aligned} & (x - x_1)(x - x_2)(x - x_3) = 0 \\ \text{or,}\quad & x^3 + x^2(-x_1 - x_2 - x_3) + x(x_1x_2 + x_2x_3 + x_1x_3) - x_1x_2x_3 = 0. \end{aligned}$$

The corresponding coefficients of polynomials must be equal. Looking at coefficients of x^2 gives us $-x_1 - x_2 - x_3 = -\lambda^2$. Hence, $x_3 = \lambda^2 - x_1 - x_2$. Substituting this into our line gives us the Y-coordinate of the point of intersection $y'_3 = \lambda(x_3 - x_1) + y_1$. Finally, reflecting this point on the X-axis gives us $y_3 = \lambda(x_1 - x_3) - y_1$. Note that the way the algorithm is structured, the λ cannot be undefined, as the denominator is never 0. If $P_1 \neq P_2$, then $x_2 \neq x_1$ and thus $\frac{y_2 - y_1}{x_2 - x_1} \neq \infty$. If $P_1 = P_2$ and $y_1 = 0$, then we would never reach case (3)(b), as $y_1 = -y_2 = 0$ and case (3)(a) would be used to compute the result as $\mathcal{O}$. □

Example 5.6. We will redo Example 5.4 to verify this algorithm. Our goal is to add $P_1(0, 4)$ and $P_2(3, 5)$ on the curve $Y^2 = X^3 - 6X + 16$. Recall that A is -6 and B is 16. Since none of the points are $\mathcal{O}$, we move to case (3) in the algorithm. Therefore, $x_1 = 0$, $y_1 = 4$, $x_2 = 3$ and $y_2 = 5$. Clearly, $x_1 \neq x_2$ so we will have to use (3)(b). As $P_1 \neq P_2$, we compute λ by $\frac{y_2 - y_1}{x_2 - x_1} = \frac{5-4}{3-0} = \frac{1}{3}$. Finally, we calculate x_3 and y_3 as

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 = \left(\frac{1}{3}\right)^2 - 0 - 3 = -\frac{26}{9}, \text{ and} \\ y_3 &= \lambda(x_1 - x_3) - y_1 = \frac{1}{3}\left(0 - \left(-\frac{26}{9}\right)\right) - 4 = -\frac{82}{27}. \end{aligned}$$

As expected, we get the same result obtained in Example 5.4:
$P_1 + P_2 = \left(-\frac{26}{9}, -\frac{82}{27}\right)$.

5.2. **Elliptic Curves Over Finite Fields.**

In the previous section, we looked at elliptic curves over the regular XY-plane where the points were real numbers. Recall that the set of real numbers is just an example of a field. In this section, we will extend the idea to elliptic curves over finite fields.

Definition 5.7 (Elliptic Curves Over Finite Fields; Section 6.2 of [10])**.** Let $p \geq 3$ be a prime. An elliptic curve over $\mathbb{F}_p$ is an equation of the form $E : Y^2 = X^3 + AX + B$ with $A, B \in \mathbb{F}_p$ satisfying $4A^3 + 27B^2 \neq 0$.
The set of points on E with coordinates in $\mathbb{F}_p$ is the set

$$E(\mathbb{F}_p) = \{(x, y) \mid x, y \in \mathbb{F}_p \text{ and } y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\}.$$

Example 5.8. Let us go back to our previous example of the curve $E : Y^2 = X^3 - 6X + 16$ and consider it over the field $\mathbb{F}_5$. Since $A, B \in \mathbb{F}_5$, the curve is $Y^2 = X^3 - 6X + 16 \bmod 5$ or simply $Y^2 = X^3 + 4X + 1$. Computing the discriminant gives us $4A^3 + 27B^2 = 4(4^3) + 27(1^2) = 256 + 27 = 283 = 3 \neq 0$. Thus, it is a valid elliptic curve for our purpose. We will now find all the points in this curve. Recall that $\mathbb{F}_5 = \{0, 1, 2, 3, 4\}$. For every $x \in \mathbb{F}_5$, we have to find y such that the equation is satisfied.

x	$x^3 + 4x + 1 = y^2$	y	Points
0	1	1 or 4	(0, 1) and (0, 4)
1	1	1 or 4	(1, 1) and (1, 4)
2	2	-	-
3	0	0	(3, 0)
4	1	1 or 4	(4, 1) and (4, 4)

Hence, $E(\mathbb{F}_5) = \{(0, 1), (0, 4), (1, 1), (1, 4), (3, 0), (4, 1), (4, 4), \mathcal{O}\}$.

Just as we added points in elliptic curves over the reals, similarly we can add points in elliptic curves over finite fields. However, a caveat to consider is that our previous geometric intuition of drawing a line, and reflecting the point of intersection over X-axis cannot be visualized anymore since we are no longer in the real XY-plane. Algebraically, the addition is exactly the same except the fact that all arithmetic happens within the field.

Theorem 5.9 (Elliptic Curve Addition over Finite Fields; Theorem 6.9 from [10])**.** *Let E be an elliptic curve over $\mathbb{F}_p$ and let P and Q be points in $E(\mathbb{F}_p)$.*

(1) *The Elliptic Curve Addition Algorithm (Theorem 5.5) applied to P and Q yields a point in $E(\mathbb{F}_p)$. That point is defined to be the sum $P \oplus Q$. Since Theorem 5.5 applied to a general field of characteristic $\neq 2$ involves arithmetic within the field, the sum $P \oplus Q$ is always defined and unique.*

(2) *This addition law on $E(\mathbb{F}_p)$ satisfies all of the properties listed in Theorem 5.3. That is, $E(\mathbb{F}_p)$ is a finite abelian group under addition.*

Instead of a rigorous proof, we will give an intuitive discussion on why this should sound plausible. The arguments to follow are mostly derived from Theorem 6.9 from [10]. Recall that the formula in Theorem 5.5 is derived by substituting the line passing through two points into the elliptic curve equation and solving its roots. Therefore the resulting point must lie on E and satisfies (1) though an additional argument might be needed to indicate why the cubic polynomial has a double root when $P = Q$. For (2), we have to consider Commutativity, Identity, Inverse, and Associativity. Commutativity is intuitive as the switching (x_1, y_1) with (x_2, y_2) has no impact on the result of the algorithm. Cases (1) and (2) of Theorem 5.5 show existence of identity and inverse follow from (3)(a). Associativity is tricky to prove but can be verified by using the addition algorithm formulas.

Example 5.10. We will continue with Example 5.8 where we were looking at curve $Y^2 = X^3 + 4X + 1$ over $\mathbb{F}_5$.
We found that $E(\mathbb{F}_5) = \{(0,1),(0,4),(1,1),(1,4),(3,0),(4,1),(4,4),\mathcal{O}\}$. We will form an addition table of $E(\mathbb{F}_5)$ to verify Theorem 5.9.

$+$	$\mathcal{O}$	$(0,1)$	$(0,4)$	$(1,1)$	$(1,4)$	$(3,0)$	$(4,1)$	$(4,4)$
$\mathcal{O}$	$\mathcal{O}$	$(0,1)$	$(0,4)$	$(1,1)$	$(1,4)$	$(3,0)$	$(4,1)$	$(4,4)$
$(0,1)$	$(0,1)$	$(4,1)$	$\mathcal{O}$	$(4,4)$	$(3,0)$	$(1,1)$	$(1,4)$	$(0,4)$
$(0,4)$	$(0,4)$	$\mathcal{O}$	$(4,4)$	$(3,0)$	$(4,1)$	$(1,4)$	$(0,1)$	$(1,1)$
$(1,1)$	$(1,1)$	$(4,4)$	$(3,0)$	$(4,1)$	$\mathcal{O}$	$(0,1)$	$(0,4)$	$(1,4)$
$(1,4)$	$(1,4)$	$(3,0)$	$(4,1)$	$\mathcal{O}$	$(4,4)$	$(0,4)$	$(1,1)$	$(0,1)$
$(3,0)$	$(3,0)$	$(1,1)$	$(1,4)$	$(0,1)$	$(0,4)$	$\mathcal{O}$	$(4,4)$	$(4,1)$
$(4,1)$	$(4,1)$	$(1,4)$	$(0,1)$	$(0,4)$	$(1,1)$	$(4,4)$	$(3,0)$	$\mathcal{O}$
$(4,4)$	$(4,4)$	$(0,4)$	$(1,1)$	$(1,4)$	$(0,1)$	$(4,1)$	$\mathcal{O}$	$(3,0)$

Clearly, closure exists as all the resulting points lie on the $E(\mathbb{F}_5)$. Commutativity and Identity is also clear. Every point has an unique inverse: $(0,1)$ and $(0,4)$, $(1,1)$ and $(1,4)$, $(4,1)$ and $(4,4)$ are inverses of each other. Note that $(3,0)$ is its own inverse. Associativity can also be checked by performing addition, choosing any three points on the curve.

As we saw from Theorem 5.9, $E(\mathbb{F}_p)$ is a finite abelian group under addition. As we will spend a little bit exploring this group, we will begin by recalling some of the group theory from algebra.

Definition 5.11 (Order of Finite Group; Appendix B of [18])**.** If G is a finite group, the **order** of G is the number of elements in G, denoted by $\#G$.

Example 5.12. Recall that, in Example 5.10 for the curve $Y^2 = X^3 + 4X + 1$ over $\mathbb{F}_5$, $E(\mathbb{F}_5) = \{(0,1),(0,4),(1,1),(1,4),(3,0),(4,1),(4,4),\mathcal{O}\}$. Since the number of elements in $E(\mathbb{F}_5)$ is 8, $\#E(\mathbb{F}_5) = 8$.

Now that we have addition, we can define multiplication by an integer in a similar fashion.

Definition 5.13 (Multiplication by an integer; Section 6.1 of [10])**.** Given a point P on an elliptic curve E over finite field $\mathbb{F}_p$ and an integer $n \in \mathbb{Z}$, multiplication of P by n is defined as the repeated addition of P to itself n

times. In other words,

$$nP = \underbrace{P \oplus P \oplus \cdots \oplus P}_{\text{n times}}.$$

Example 5.14. Continuing with our Example 5.10 for curve $Y^2 = X^3 + 4X + 1$ over $\mathbb{F}_5$, below is a table showing multiplication by some integers for point $P = (0, 1)$.

n	$P \oplus P \oplus \cdots \oplus P$ (n times)	nP
1	$(0,1)$	$1 \cdot (0,1) = (0,1)$
2	$(0,1) \oplus (0,1)$	$2 \cdot (0,1) = (4,1)$
3	$(0,1) \oplus (0,1) \oplus (0,1)$	$3 \cdot (0,1) = (1,4)$
4	$(0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1)$	$4 \cdot (0,1) = (3,0)$
5	$(0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1)$	$5 \cdot (0,1) = (1,1)$
6	$(0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1)$	$6 \cdot (0,1) = (4,4)$
7	$(0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1)$	$7 \cdot (0,1) = (0,4)$
8	$(0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1) \oplus (0,1)$	$8 \cdot (0,1) = \mathcal{O}$

Again, we recall some concepts such as order of points and Lagrange's Theorem from group theory below.

Definition 5.15 (Order of Points; Section 4.3.3 of [18], Section 6.8.1 of [10])**.** Let $P \in E(\mathbb{F}_p)$. The **order** of P, denoted by $|P|$ is the smallest positive integer k such that $kP = \mathcal{O}$. Furthermore, if $|P| = \#E(\mathbb{F}_p)$, P is said to be **generator** of $E(\mathbb{F}_q)$. We denote the set of points of order k by

$$E[k] = \{P \in E : kP = \mathcal{O}\}.$$

Theorem 5.16 (Corollary from Lagrange's Theorem; Theorem B.1 of [18])**.** *Let G be a finite group and $g \in G$. Then, the order of g divides the order of G.*

For readers interested in the proof, we direct them to Theorem B.1 of [18].

Example 5.17. From Example 5.14, since $8 \cdot (0,1) = \mathcal{O}$ and 8 is the smallest integer to satisfy the property, the order of point $P = (0, 1)$ in $E(\mathbb{F}_5)$ is 8. The table below shows the order of points in

$$E(\mathbb{F}_5) = \{(0,1), (0,4), (1,1), (1,4), (3,0), (4,1), (4,4), \mathcal{O}\}.$$

Point	$\mathcal{O}$	$(0,1)$	$(0,4)$	$(1,1)$	$(1,4)$	$(3,0)$	$(4,1)$	$(4,4)$
Order	1	8	8	8	8	2	4	4

The possible order of points are 1, 2, 4, and 8 all of which divide $\#E(\mathbb{F}_5) = 8$ thereby verifying Lagrange's Theorem. Since the order of points $(0, 1)$, $(0, 4)$, $(1, 1)$ and $(1, 4)$ are 8 which is equal to $\#E(\mathbb{F}_5)$, they are generators of $E(\mathbb{F}_5)$.

There is an important connection between the group formed by points in an elliptic curve over a finite field and groups $\mathbb{Z}_n$ that we are familiar with, as described by the following theorem.

Theorem 5.18 (Isomorphism of Group formed by points in Elliptic Curve; Theorem 4.1 of [18])**.** *Let E be an elliptic curve over the finite field $\mathbb{F}_p$. Then*

$$E(\mathbb{F}_p) \cong \mathbb{Z}_n \qquad \textit{or} \qquad \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$$

for some integer $n \geq 1$, or for some integers $n_1, n_2 \geq 1$ with n_1 dividing n_2.

The proof is beyond the scope of the thesis and we will redirect the interested readers to Theorem 4.1 of [18]. Instead, we will work with an example below.

Example 5.19. Let us attempt to find a group isomorphism for the previous example $E(\mathbb{F}_5)$. We know that $\#E(\mathbb{F}_5) = 8$, so the possible isomorphism according to Theorem 5.18 can be $\mathbb{Z}_8$ or $\mathbb{Z}_2 \oplus \mathbb{Z}_4$. The maximum order that an element in group $\mathbb{Z}_2 \oplus \mathbb{Z}_4$ can have is 4. However, in $E(\mathbb{F}_5)$, we have 4 elements of order 8. Therefore, $E(\mathbb{F}_5) \cong \mathbb{Z}_8$.

We saw that $E(\mathbb{F}_5)$ has 8 elements and is isomorphic to $\mathbb{Z}_8$. The number of elements in the group formed by points in an elliptic curve might seem random; remember the way we found all the points was by taking points in $\mathbb{F}_5 \times \mathbb{F}_5$ which satisfied the elliptic curve equation. Given a prime p, one might be curious about how many possible points there can be on the elliptic curve. It turns out that it depends upon the prime p. Hasse's Theorem shows us how.

Theorem 5.20 (Hasse's Theorem; Theorem 6.11 of [10])**.** *Let E be an elliptic curve over $\mathbb{F}_p$. Then*

$$\#E(\mathbb{F}_p) = p + 1 - t_p \qquad \textit{with } t_p \in \mathbb{Z} \textit{ satisfying } |t_p| \leq 2\sqrt{p}.$$

The proof of this theorem can be found in Section 4.2 of [18]. Note that the quantity $t_p = p + 1 - \#E(\mathbb{F}_p)$ has a special name and is called the **trace of Frobenius** for $E/\mathbb{F}_p$. The details can be found in Section 4.2 of [18].

Hasse's Theorem allows us to form a bound for $\#E(\mathbb{F}_p)$. That is,

$$\begin{aligned}
& \#E(\mathbb{F}_p) = p + 1 - t_p && \text{with } t_p \text{ satisfying } |t_p| \leq 2\sqrt{p}, \\
\text{or} \quad & t_p = p + 1 - \#E(\mathbb{F}_p) && \text{with } t_p \text{ satisfying } |t_p| \leq 2\sqrt{p}, \\
\text{or} \quad & |p + 1 - \#E(\mathbb{F}_p)| \leq 2\sqrt{p}. \\
\text{Either,} \quad & p + 1 - \#E(\mathbb{F}_p) \leq 2\sqrt{p} \quad \text{or} \quad -p - 1 + \#E(\mathbb{F}_p) \leq 2\sqrt{p} \\
\text{Hence,} \quad & -2\sqrt{p} + p + 1 \quad \leq \quad \#E(\mathbb{F}_p) \quad \leq \quad 2\sqrt{p} + p + 1.
\end{aligned}$$

Suppose we know that the order of an element $Q \in E(\mathbb{F}_p)$ is k. By Lagarange's Theorem (Theorem 5.16), we know that the order of $E(\mathbb{F}_p)$ must be a multiple of k. By using bounds of Hasse's Theorem, we can at least shortlist the possible orders of the group. However, note that Hasse's Theorem doesn't give an algorithm for exactly determining an order. Of course, we could use our naive approach like in Example 5.8 where we compute possible Y-coordinates for all possible X-coordinates in the curve. However, as we employ larger fields, this becomes very inefficient. Algorithms like Baby Step, Giant Step (Section 4.3.4 of [18]) can speed this calculation to around $4p^{\frac{1}{4}}$ steps. Even better, Schoof's Algorithm (Section 4.5 of [18]) can compute $\#E(\mathbb{F}_p)$ in $(\log p)^8$ bit operations.

Example 5.21. Let us verify that Hasse's Theorem is true for our previous example. We were working with curve $Y^2 = X^3+4X+1$ over $\mathbb{F}_5$. Our $p = 5$ so the bounds of $\#E(\mathbb{F}_p)$ are $-2\sqrt{5}+5+1 \leq \#E(\mathbb{F}_5) \leq 2\sqrt{5}+5+1$. Simplifying gives us $1 \leq \#E(\mathbb{F}_5) \leq 10$. We found in Example 5.12 that $\#E(\mathbb{F}_5) = 8$ which satisfies the bounds. Also, we know that $\#E(\mathbb{F}_p) = p + 1 - t_p$. As $\#E(\mathbb{F}_p) = 8$ and $p = 5$, we can infer that the trace of Frobenius $t_5 = -2$.

We noted that we could use elementary methods or Baby Step, Giant Step or Schoof's Algorithm to compute the order of the the group formed by an elliptic curve over a finite field $\mathbb{F}_p$. However, sometimes we may want to know the order of elliptic curves over larger finite fields $\mathbb{F}_{p^n}$ for some n. Once we have found $\#E(\mathbb{F}_p)$, the procedure of finding $\#E(\mathbb{F}_{p^n})$ becomes simple using the following theorem.

Theorem 5.22 (Determining Group Order for Large Fields; Theorem 4.12 of [18]). *Let E be an elliptic curve over $\mathbb{F}_{p^n}$ and $\#E(\mathbb{F}_p) = p+1-t_p$. Write $X^2 - t_pX + p = (X - \alpha)(X - \beta)$ where $\alpha, \beta \in \mathbb{C}$. Then,*

$$\#E(\mathbb{F}_{p^n}) = p^n + 1 - (\alpha^n + \beta^n)$$

for all $n \geq 1$.

As the mathematics behind the proof is beyond the scope the thesis, we omit the proof which can be found in Theorem 4.12 of [18]. One interesting observation is that $(\alpha^n + \beta^n)$ must be an integer as any group order is always a natural number. Since α and β are roots in $\mathbb{C}$, this is not trivial. For details, kindly refer to Lemma 4.13 of [18].

Example 5.23. We will expand Example 5.21 to compute the group order of points in $Y^2 = X^3 + 4X + 1$ over $\mathbb{F}_{5^2}$. Our $p = 5$, $n = 2$, and we calculated in Example 5.21 that $t_5 = -2$. Thus, our polynomial is $X^2 + 2X + 5$. We can factorize it to find $\alpha = (-1 - 2i)$ and $\beta = (-1 + 2i)$. Finally,

$$\#E(\mathbb{F}_{5^2}) = 5^2 + 1 - ((-1 - 2i)^2 + (-1 + 2i)^2) = 32.$$

5.2.1. *Elliptic Curves Over $\mathbb{F}_{2^k}$.* Note that in Section 5.2 we defined elliptic curves over $\mathbb{F}_p$ for $p \geq 3$. In this section, we will explore elliptic curves over $\mathbb{F}_2$ and more generally $\mathbb{F}_{2^k}$. Since computers use binary, elliptic curves over $\mathbb{F}_{2^k}$ tend to be very efficient in encryption and decryption of information.

In Section 5.2, we looked at a simplified version of an elliptic curve and noted that the discriminant was given by $\Delta_E = 4A^3 + 27B^2$. In fact, the correct discriminant for the elliptic curve described before is $\Delta_E = -16(4A^3 + 27B^2)$. Since we did not consider $\mathbb{F}_2$, just using $\Delta_E = 4A^3 + 27B^2$ sufficed our need when we attempted to check for singularity. However, to extend the idea to $\mathbb{F}_2$ and beyond, we will work with general elliptic curves defined below.

Definition 5.24 (Generalized Elliptic Curve; Section 6.7 of [10])**.** An elliptic curve E over the real XY-plane is the set of solutions to a generalized Weierstrass equation

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6,$$

together with an extra point $\mathcal{O}$. The coefficients $a_1, \ldots, a_6 \in \mathbb{R}$ are required to satisfy $\Delta \neq 0$, where the discriminant Δ is defined in terms of certain quantities b_2, b_4, b_6, b_8 as follows:

$$\begin{aligned}
b_2 &= a_1^2 + 4a_2\\
b_4 &= 2a_4 + a_1a_3\\
b_6 &= a_3^2 + 4a_6\\
b_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2{a_3}^2 - {a_4}^2\\
\Delta &= -{b_2}^2b_8 - 8{b_4}^3 - 27{b_6}^2 + 9b_2b_4b_6.
\end{aligned}$$

The geometric definition of the addition law on E is similar to our previous definition except that the old reflection step $(x, y) \to (x, -y)$ is replaced by a slightly more complicated step $(x, y) \to (x, -y - a_1x - a_3)$ [10]. The addition algorithm is also slightly more complicated but the proof has a similar structure.

Theorem 5.25 (General Elliptic Curve Addition Algorithm; Exercise 6.22 of [10])**.** *Let E be an elliptic curve over $\mathbb{F}_{p^n}$ given by a generalized Weierstrass equation*

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6.$$

Let P_1 and P_2 be points on E. The sum $P_1 + P_2$ is given by:

(1) *If $P_1 = \mathcal{O}$, then $P_1 \oplus P_2 = P_2$.*
(2) *Otherwise, if $P_2 = \mathcal{O}$, then $P_1 + P_2 = P_1$.*
(3) *Otherwise, let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$.*
   (a) *If $x_1 = x_2$ and $y_1 + y_2 + a_1x_2 + a_3 = 0$, then $P_1 + P_2 = \mathcal{O}$.*
   (b) *Otherwise, $P_1 + P_2 = (x_3, y_3)$ where,*

$$\begin{aligned}
x_3 &= \lambda^2 + a_1\lambda - a_2 - x_1 - x_2\\
y_3 &= -(\lambda + a_1)x_3 - \nu - a_3
\end{aligned}$$

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } x_1 \neq x_2\\ \frac{3{x_1}^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{if } x_1 = x_2\end{cases}$$

$$\nu = \begin{cases} \frac{y_1x_2 - y_2x_1}{x_2 - x_1} & \text{if } x_1 \neq x_2\\ \frac{-{x_1}^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3} & \text{if } x_1 = x_2.\end{cases}$$

The proof is skipped here but should follow similar structure as the proof of Theorem 5.5. This generalized algorithm can be used for addition of points in elliptic curve over any finite fields.

5.3. **Elliptic Curve Digital Signature Algorithm (ECDSA).** In this section, we will cover the Elliptic Curve Digital Signature Algorithm (ECDSA) in detail by first providing necessary background and finally concluding with an example used in verifying digital transactions.

5.3.1. *Discrete Logarithm Problem.* The heart of the ECDSA is the Elliptic Curve Discrete Logarithm Problem (ECDLP) which is known to be a very hard problem (see Section 2.6 of [10]). To understand it fully, let us first look at Euler's Totient function.

Definition 5.26 (Euler's totient function; Section 6.3 of [12])**.** Euler's totient function is the map $\phi : \mathbb{N} \to \mathbb{N}$ defined by $\phi(1) = 1$ and for $n > 1$, $\phi(n)$ is defined as

$$\phi(n) = \#\{1 \leq a < n \mid \gcd(a, n) = 1\}.$$

To put it into words, given a natural number m, $\phi(m)$ is the total number of integers less than m that are coprime to m.

Example 5.27. Consider $m = 15$. The set of numbers that are less than 15 and are co-prime to 15 is $\{1, 2, 4, 7, 8, 11, 13, 14\}$. Hence, $\phi(15) = 8$.

Corollary 5.28 (Euler's totient function for primes)**.** *For a prime p,*

$$\phi(p) = p - 1.$$

This should be intuitive from the definition of prime numbers. Prime numbers can only be divided without a remainder by 1 and itself. Therefore, all numbers less than p must be in the set resulting in $\phi(p) = p - 1$.

Example 5.29. For instance, let us work with $p = 7$. Our set is $\{1, 2, 3, 4, 5, 6\}$ and $\phi(7) = 6$.

Now, we will use the Euler's totient function to state Euler's Theorem which is one of the fundamental results in modular arithmetic.

Theorem 5.30 (Euler's Theorem; Theorem 6.19 of [12])**.** *Let a and n be integers such that $n > 0$ and $gcd(a, n) = 1$. Then,*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

For proof, refer to Theorem 6.19 from [12]. Here, we will verify with an example.

Example 5.31. Let us pick $a = 2$ and $n = 15$. We have $gcd(2, 15) = 1$. In Example 5.27, we saw that $\phi(15) = 8$. Now, $2^8 = 256 \equiv 1 \pmod{15}$. Note that $a^8 \equiv 1 \pmod{15}$ is true for any a satisfying $gcd(a, 15) = 1$.

From Euler's Theorem, while we can be certain that for $n > 0$ and $gcd(a, n) = 1$, there exists $\phi(n)$ that satisfies $a^{\phi(n)} \equiv 1$, it is not guaranteed that $\phi(n)$ is the smallest positive integer that satisfies this equation. This motivates a stronger version of the Euler's Theorem using Carmichael Function.

Definition 5.32 (Carmichael Function; Section 2 of [15])**.** Let $n = {p_1}^{e_1}{p_2}^{e_2} \cdots {p_r}^{e_r}$ be the prime factorization of $n \in \mathbb{Z}$ and let $\phi(p^e)$ be the Euler's Totient Function, the **carmichael function** $\lambda(n)$ is defined as:

$$\lambda(n) = \text{lcm}(\lambda({p_1}^{e_1}), \lambda(({p_2}^{e_2}), \cdots, \lambda({p_r}^{e_r})),$$

$$\lambda(p^e) = \begin{cases} \frac{1}{2}\phi(2^e) & \text{for } p = 2 \text{ and } e > 2, \\ \phi(p^e) & \text{for } p \text{ odd prime or } p = 2 \text{ and } e = 1 \text{ or } 2. \end{cases}$$

Example 5.33. Let us attempt to compute $\lambda(15)$. We need to first find the prime factorization of 15 which is $15 = 3 \cdot 5$. Now, $\lambda(15) = \text{lcm}(\lambda(3), \lambda(5))$.

$$\lambda(3) = \phi(3) = 2.$$

$$\lambda(5) = \phi(5) = 4.$$

Finally,

$$\lambda(15) = \text{lcm}(\lambda(3), \lambda(5)) = \text{lcm}(2, 4) = 4.$$

Theorem 5.34 (Euler's Theorem using Carmichael Function; Section 2 of [15])**.** *Let a and n be integers such that $n > 0$ and $gcd(a, n) = 1$. Then,*

$$a^{\lambda(n)} \equiv 1 \pmod{n}.$$

Furthermore, $\lambda(n)$ is the smallest positive integer that satisfies the above equation for every a.

For more details on the theorem, we refer readers to Section 2 of [15]. We will try to strengthen the claim made in Example 5.31.

Example 5.35. In Example 5.31, we had $a = 2$ and $n = 15$. In Example 5.33, we saw that $\lambda(15) = 4$. Now, $2^4 = 16 \equiv 1 \pmod{15}$. In fact, note that $a^4 \equiv 1 \pmod{15}$ is true for any a satisfying $gcd(a, 15) = 1$.

In the discrete logarithm problem, we will consider the special case where n is a prime p. As $\lambda(p)$ is the smallest such integer that $a^{\lambda(p)} \equiv 1 \pmod{p}$, we know from Definition 5.15 that a is the generator of the group formed by the nonzero element of $\mathbb{F}_p$. Therefore, for an integer $b \not\equiv 0 \pmod{p}$ there must exist an integer k such that $a^k \equiv b \pmod{p}$. As we know that such an integer a can exist, we can cover the classical discrete logarithm problem. More formally, such an integer a is also called a primitive root of prime p and is discussed in detail in Section 1.5 of [10]. Specifically, for the readers interested in a more formal approach, we refer them to Theorem 1.30 of [10].

Definition 5.36 (The Classical Discrete Logarithm Problem; Chapter 5 of [18])**.** Let p be a prime and let a, b be integers that are nonzero mod p. Suppose we know that there exists an integer k such that

$$a^k \equiv b \pmod{p}.$$

The **classical discrete logarithm problem** is to find k.

Example 5.37. Consider $p = 7$. From Example 5.29, we know $\phi(7) = 6$. It should not be difficult to see that $\phi(p) = \lambda(p)$ for any prime p and thus $\lambda(7) = 6$. We know there exists an element a satisfying $gcd(a, 7) = 1$ for which 6 is smallest integer such $a^6 \equiv 1 \pmod{7}$. For $p = 7$, such an a is 3. Let us pick $b = 5$. Now there must exist k such that $3^k \equiv 5 \pmod{7}$. By trial and error or techniques like the collision algorithm (see Section 2.7 of [10]), we can find that $k = 5$.

Let us take a moment to think about approaches to solve the discrete logarithm problem (DLP). The brute force method would be to multiply a to itself until we get the point b. For prime p, $|{\mathbb{F}_p}^{\times}| = p - 1$. Thus in the worst case, we would have to multiply a to itself $p - 1$ number of times. In terms of Big O notation, the complexity is $\mathcal{O}(p)$ but for large values of p, this still takes a significant amount of time. There are other algorithms that are a bit faster; for example, the index calculus method solves DLP in $\mathcal{O}(e^{c\sqrt{\log p \log\log p}})$. The details on the index calculus method can be found in Section 3.8 of [10].

We will now extend discrete logarithms to elliptic curves, which is known to be even harder.

Definition 5.38 (The Elliptic Curve Discrete Logarithm Problem; Section 6.3 of [10])**.** Let E be an elliptic curve over the finite field $\mathbb{F}_p$ and let P be a generator and Q be any point in $E(\mathbb{F}_p)$. The **Elliptic Curve Discrete Logarithm Problem (ECDLP)** is the problem of finding an integer n such that $Q = nP$. The integer n is also denoted as $n = \log_P(Q)$ and called the **elliptic discrete logarithm** of Q with respect to P.

As referenced in Section 6.3.2 in [10], the fastest known algorithm to solve ECDLP in $E(\mathbb{F}_p)$ takes approximately $\sqrt{p}$ steps. However, it is also important to note that there are some elliptic curves and primes for which the ECDLP is relatively easy. More details can be found in Section 6.9.1 of [10].

5.3.2. *Elliptic Curve Digital Signature Algorithm.* As previously stated in Section 3.2, one of the reasons public key cryptosystems were introduced was to solve the problem of digital signatures. Digital Signatures enable us to verify that the message came from the right sender and solves the problem of authentication. The Elliptic Curve Digital Signature Algorithm is one of the widely used asymmetric cryptosystems. Some examples include authentication of commands between AWS Key Management Service [1] and signatures for iMessages in iPhone [9].

This algorithm is taken from Section 6.6 from [18]. We consider our analogy from Section 3.2 where Alice wants to sign a document m and send it to Bob.

To specify an Elliptic Curve Digital Signature Algorithm, Alice does the following:

(1) She first chooses an elliptic curve E and a prime p to construct an elliptic curve over a finite field $\mathbb{F}_p$. The group order can be written as $\#E(\mathbb{F}_p) = fr$ where r is a large prime number and f is an small integer called cofactor (usually kept as 1, 2, or 4 to make the algorithm efficient). While there is a lot of flexibility in terms of choosing E, and p, specifications that can break down $\#E(\mathbb{F}_p)$ into a large prime are generally better for a more secure system. Recall that there are some curves and primes for which ECDSA can be easy, so it pays off to do some careful analysis of this step before proceeding further.
(2) Then, she chooses a base point $G = (x_G, y_G)$ in $E(\mathbb{F}_p)$ of order r.
Remark: We know that a base point G of order r exists. This is because $\#E(\mathbb{F}_p)$ is a finite abelian group and the converse of the Lagrange Theorem applies. The details can be found in Section 3.2 of [5]. Alternatively, in Theorem 5.18, we have seen that $E(\mathbb{F}_p) \cong \mathbb{Z}_n$ or $\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ and clearly, r divides $\#E(\mathbb{F}_p)$.
(3) Finally, she chooses a secret integer a and computes $Q = aG$.

Alice makes the following information public:

- The elliptic curve E
- The finite field $\mathbb{F}_p$
- The group order r
- Points G and Q

While f is not explicitly in the above list, it can be deduced easily from p and r using elementary methods, Hasse's theorem, or algorithms like Baby Step, Giant Step or Schoof's. The real secret is the a which is difficult to compute because of the Elliptic Curve Discrete Logarithm Problem.

To sign the message m, Alice does the following:

(1) First, she chooses a random integer k with $1 \leq k < r$ and computes $R = kG = (x, y)$.
(2) Then she computes $s = k^{-1}(m + ax) \pmod{r}$.

The signed document that is passed to Bob is (m, R, s).

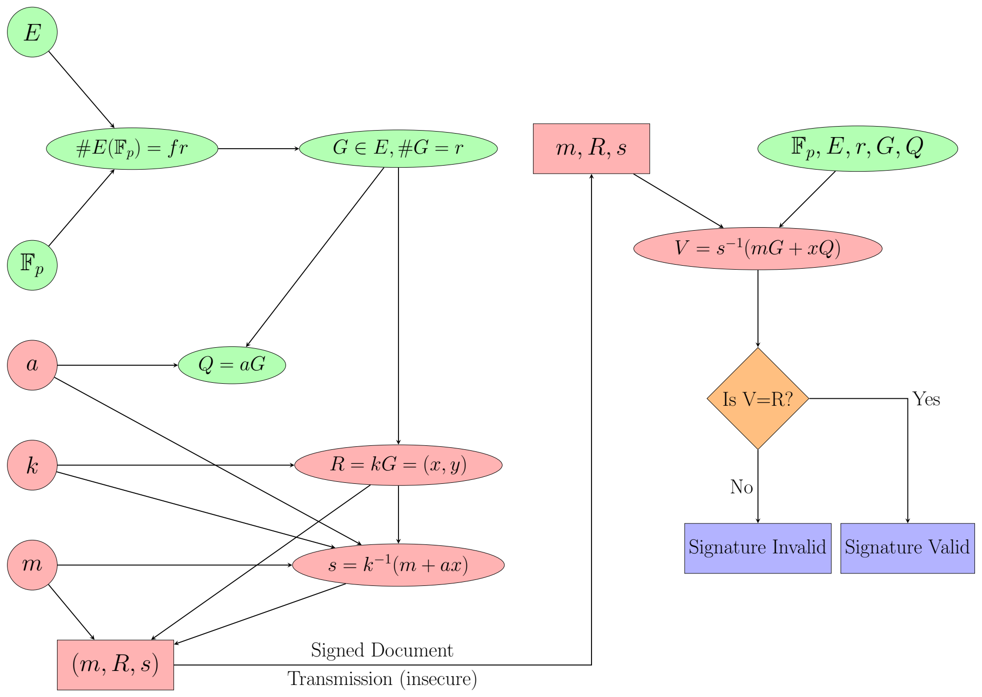
Finally, to verify the signature, Bob does the following.

(1) He computes $u_1 = s^{-1}m \pmod{r}$ and $u_2 = s^{-1}x \pmod{r}$.
(2) He also computes $V = u_1G \oplus u_2Q$.
(3) The signature is declared valid if $V = R$.

If the document is signed correctly, this is why verification should work:

$$\begin{aligned} V &= u_1G \oplus u_2Q \\ &= s^{-1}mG \oplus s^{-1}xQ \\ &= s^{-1}mG \oplus s^{-1}xaG \\ &= Gs^{-1}(m+xa) \\ &= Gs^{-1}sk \quad [\text{Note: } s = k^{-1}(m+ax)(\text{mod r})] \\ &= Gk \\ &= R. \end{aligned}$$

The flowchart below summarizes the Elliptic Curve Digital Signature Algorithm. The public information is marked with green while the private information is marked with red.



Example 5.39. We will construct a simple example of the Elliptic Curve Digital Signature Algorithm. In our previous example, we used the curve $Y^2 = X^3 + 4X + 1$ over $\mathbb{F}_5$ and calculated $\#E(\mathbb{F}_5) = 8$. Since $8 = 2^3$ is a product of the smallest prime, we will use a different field with a group order of points that is relatively large. Let us consider $Y^2 = X^3 + 4X + 1$ over $\mathbb{F}_7$. Following procedure in Example 5.8, we can find the points in the curve: $E(\mathbb{F}_7) = \{\mathcal{O}, (0,1), (0,6), (4,2), (4,5)\}$. We also find that $\#E(\mathbb{F}_7) = 5$ which is prime. In our case, $f = 1$ and $r = 5$. Below we will also construct an addition table to help in our calculations but note that this addition table is difficult to generate for cryptography in real world where large primes are used.

$+$	$\mathcal{O}$	$(0,1)$	$(0,6)$	$(4,2)$	$(4,5)$
$\mathcal{O}$	$\mathcal{O}$	$(0,1)$	$(0,6)$	$(4,2)$	$(4,5)$
$(0,1)$	$(0,1)$	$(4,5)$	$\mathcal{O}$	$(0,6)$	$(4,2)$
$(0,6)$	$(0,6)$	$\mathcal{O}$	$(4,2)$	$(4,5)$	$(0,1)$
$(4,2)$	$(4,2)$	$(0,6)$	$(4,5)$	$(0,1)$	$\mathcal{O}$
$(4,5)$	$(4,5)$	$(4,2)$	$(0,1)$	$\mathcal{O}$	$(0,6)$

Now suppose Alice chooses a base point $G = (0,1)$ on $E(\mathbb{F}_5)$. The order of $(0,1)$ is $r = 5$. Also suppose Alice chooses a secret integer $a = 3$. Now, she computes $Q = 3 \cdot (0,1) = (4,2)$.

Following information is made public:

- The curve $Y^2 = X^3 + 4X + 1$.
- The field $\mathbb{F}_7$.
- The group order $r = 5$.
- Points $(0,1)$ and $(4,2)$.

Because of the discrete logarithm problem, it is assumed that finding a when given $(0,1)$ and $(4,2)$ is hard.

To sign the message $m = 9$, Alice does the following:

(1) For a random integer $1 \leq k < r = 5$, say she chooses $k = 2$.

$$R = 2 \cdot (0,1) = (4,5).$$

Thus, $x = 4, y = 5$

(2) Computes s as

$$\begin{aligned} s &= 2^{-1}(9 + 3 \cdot 4) \pmod 5 \\ &= 2^{-1} \cdot 1 \pmod 5 \\ &= 3 \end{aligned}$$

The signed document passed to Bob is

$$(m = 9, R = (4,5), s = 3).$$

Bob can verify the signature as follows:

(1) He computes:

$$\begin{aligned} u_1 &= 3^{-1} \cdot 9 \pmod 5 \\ &= 2 \cdot 9 \pmod 5 \\ &= 3 \\ u_2 &= 3^{-1} \cdot 4 \pmod 5 \\ &= 2 \cdot 4 \pmod 5 \\ &= 3 \end{aligned}$$

(2) Now he computes V as:

$$\begin{aligned} V &= u_1 G + u_2 Q \\ &= 3 \cdot (0,1) + 3 \cdot (4,2) \\ &= (4,2) + (0,6) \\ &= (4,5) \end{aligned}$$

(3) Clearly $V = R = (4,5)$ and the signature is declared valid.

The example above was fairly simple as we used a small prime p. In order to leave the readers with a taste of how this is actually applied in real world, we will look at an example below.

5.3.3. *Example: secp256k1.* One fascinating application of the Elliptic Curve Digital Signature Algorithm, as mentioned in Section 8.8 of [10], is implementing blind digital signatures where the document to be signed is first concealed and then signed, yet the signature can be verified against the original unblinded document. Blind digital signatures can be implemented for digital cash whose payments are verifiable but untraceable as explained in [2]. An example of a popular digital cash mechanism is Bitcoin which uses a 256-bit elliptic curve domain parameters over $\mathbb{F}_p$ where $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$. From [13], we specify this elliptic curve domain parameter called $secp256k1$.

The elliptic curve is given by $E : y^2 = x^3 + 7$.

It might help to know actually how large the prime p is. We use [8] to calculate $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ as

$$\begin{aligned} p =& 115792089237316195423570985008687907853 \\ & 269984665640564039457584007908834671663 \end{aligned}$$

We will look at the base point G in its compressed form below. By compressed form, we mean the X-coordinate of G. Note that this specification lists the coordinates in hexadecimal number system.

$$\begin{aligned} G = 02 \quad & 79BE667E \quad F9DCBBAC \quad 55A06295 \quad CE870B07 \\ & 029BFCDB \quad 2DCE28D9 \quad 59F2815B \quad 16F81798 \end{aligned}$$

The above hexadecimal is equal to the decimal number below. For all large hexadecimal to decimal conversions, we use the calculator specified in [8].

$$\begin{aligned} x_G =& 55066263022277343669578718895168534326250 \\ & 603453777594175500187360389116729240 \end{aligned}$$

We could have substituted x_G into the curve to find y_G, but the specification conveniently also provides the base point G in uncompressed form, in other words, both X and Y-coordinates.

$$\begin{aligned} G = 04 \quad & 79BE667E \quad F9DCBBAC \quad 55A06295 \quad CE870B07 \\ & 029BFCDB \quad 2DCE28D9 \quad 59F2815B \quad 16F81798 \\ & 483ADA77 \quad 26A3C465 \quad 5DA4FBFC \quad 0E1108A8 \\ & FD17B448 \quad A6855419 \quad 9C47D08F \quad FB10D4B8 \end{aligned}$$

In decimal, the Y-coordinate translates to

$$\begin{aligned} y_G =& 32670510020758816978083085130507043184 4 \\ & 71273380659243275938904335757337482424. \end{aligned}$$

The order r of G is:

$$\begin{aligned} r =& FFFFFFFF \quad FFFFFFFF \quad FFFFFFFF \quad FFFFFFFE \\ & BAAEDCE6 \quad AF48A03B \quad BFD25E8C \quad D0364141 \end{aligned}$$

In decimal, this translates to

$$\begin{aligned} r =& 115792089237316195423570985008687907852 \\ & 837564279074904382605163141518161494337. \end{aligned}$$

The cofactor $f = 1$.

As we see above, the prime p used is roughly 2^{256}. This suggests that even using the fastest algorithm that we currently have will take 2^{128} steps to solve this digital signature scheme. Recall that this was similar for AES-128 as well. There are 2^{128} possible keys and if someone wanted to brute force the key, they would have to perform at least 2^{128} operations. According to [17], the fastest supercomputer can perform 1 Exaflop (2^{60}) operations in a second. For 2^{128} steps, that translates to 2^{68} seconds. This is equivalent to approximately 9 trillion years!

6. Conclusion

To sum up, finite fields are abstract mathematical structures following certain properties that allows us to nicely perform arithmetic within them. Cryptography, on the other hand, is a very applied methodology to share information over an insecure channel while ensuring confidentiality and authentication. Based on the whether same keys are used for encryption and decryption, cryptography is classified into private-key (symmetric) and public-key (asymmetric) cryptosystems. As an example for private-key cryptography, we studied Advanced Encryption Standard (AES) which used arithmetic within finite field GF(2^8). In the last chapter, we studied the mathematics behind elliptic curves and used elliptic curves over finite fields to set up Elliptic Curve Digital Signature Algorithm (ECDSA), a widely used public-key cryptosystem for verifying the source of information. We conclude that finite field arithmetic has remarkable applications in both private and public key cryptosystems and serve practical needs of information security in modern world.

Acknowledgements

I would like to begin by thanking my parents, Dr. Bishnu Prasad Sharma and Satya Shree Sharma, as well as my sister Anushka who have supported me throughout the journey. I am very grateful to my advisors Prof. Elizabeth Milićević (Department of Mathematics and Statistics) and Prof. Steven Lindell (Department of Computer Science) who helped me turn an idea to a complete joint thesis with their continued guidance and feedback. I would also like to thank Prof. Yuxin Zhou (Department of Computer Science) for his time in attending my thesis meetings. Special thanks to Prof. Nimisha Ladva (Department of Writing Program) and friend Li Fan (Math@WritingCenter) for carefully proofreading the final document!

References

[1] Amazon Web Services. “AWS Key Management Service: Cryptographic Details” (Dec. 2020). https://docs.aws.amazon.com/pdfs/kms/latest/cryptographic-details/kms-crypto-details.pdf.

[2] David Chaum. “Blind Signatures for Untraceable Payments” (1983). Ed. by David Chaum, Ronald L. Rivest, and Alan T. Sherman, pp. 199–203.

[3] Katriel Cohn-Gordon et al. *A Formal Security Analysis of the Signal Messaging Protocol*. Cryptology ePrint Archive, Paper 2016/1013. https://eprint.iacr.org/2016/1013. 2016.

[4] Joa Daor, Joan Daemen, and Vincent Rijmen. “AES proposal: rijndael” (Oct. 1999).

[5] Richard M. Foote David S. Dummit. *Abstract Algebra*. 3rd ed. Wiley, 2004.

[6] W. Diffie and M. Hellman. “New directions in cryptography”. *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[7] Joseph A. Gallian. *Contemporary Abstract Algebra*. 10. Chapman Hall/CRC, 2021.

[8] Taylor Hornby. *Online big number calculator*. https://defuse.ca/big-number-calculator.htm.

[9] Apple Inc. “Apple Platform Security” (May 2022). https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf.

[10] Joseph H. Silverman (auth.) Jeffrey Hoffstein Jill Pipher. *An Introduction to Mathematical Cryptography*. 2nd ed. Undergraduate Texts in Mathematics. Springer-Verlag New York, 2014.

[11] Vincent Rijmen Joan Daemen. *The Design Of Rijndael: The Advanced Encryption Standard (AES)*. 2nd Edition. Information Security And Cryptography. Springer, 2020.

[12] Thomas W. Judson. *Abstract Algebra: Theory and Applications (The Prindle, Weber & Schmidt Series in Advanced Mathematics)*. Prindle Weber & Schmidt, 2022.

[13] Certicom Research. “Standards for efficient cryptography, SEC 2: Recommended Elliptic Curve Domain Parameters” (Jan. 2010). Version 2.0.

[14] R. L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342.

[15] Günter Pilz (auth.) Rudolf Lidl. *Applied Abstract Algebra*. 1st ed. Undergraduate Texts in Mathematics. Springer-Verlag New York, 1984.

[16] National Institute of Standards and Technology. “Advanced Encryption Standard”. *NIST FIPS PUB 197* (2001).

[17] Erich Strohmaier et al. *ORNL’s Frontier First to Break the Exaflop Ceiling*. `https://www.top500.org/news/ornls-frontier-first-to-break-the-exaflop-ceiling/`. May 2022.

[18] Lawrence C. Washington. *Elliptic curves: number theory and cryptography*. 2nd ed. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2008.