EDIN01: Cryptography - Reference Sheet

Karl Hallsby

Last Edited: December 16, 2019

Contents

List of Theorems

1 Cryptography Introduction

Defn 1 (Cryptographic Primitive)**.** A *cryptographic primitive* is an algorithm with basic cryptographic properties. These are solutions to different problems where cryptography is required.

Defn 2 (Cryptographic Protocol)**.** A *cryptographic protocol* involves the back-and-forth communication among two or more parties.

Remark 2.1 (Bob and Alice). Typically, the parties are named Bob and Alice. These are arbitrary names, but these are the most commonly used ones.

There are have been several Cryptographic Protocols.

1. *Symmetric-key cryptography* - Methods in which both the sender and receiver share the same key
   (a) Block ciphers
   (b) Stream ciphers
   (c) MAC algorithms
2. *Public-key cryptography*: 2 different, but mathematically related keys are used. A public key and a private key.
   (a) The public key cannot decrypt something that was encrypted with the private key.
   (b) The public key can be shared freely, because the private key cannot be generated from the public key.
3. *Cryptographic hash functions* are a related and important class of cryptographic algorithms.
   (a) This is a keyless Cryptographic Primitive.
   (b) Takes an arbitrary length input and produces a fixed-length output.
   (c) The mapping between the input and output is such that the output cannot generate the input, therefore making it cryptographic.

1.1 Historical Cryptography

Defn 3 (Cryptology)**.** *Cryptology* was the science of secret writing.

Defn 4 (Cryptography)**.** *Cryptography* dealt with the development of systems for secret writing.

Defn 5 (Cryptanalysis)**.** *Cryptanalysis* was the analysis of existing cryptographic systems to break them.

Just to give a super quick background on how we've gotten to where we are today when it comes to cryptography.

1.1.1 Monoalphabetic Ciphers

Defn 6 (Monoalphabetic Cipher)**.** In a *monoalphabetic cipher* a single letter is replaced by the cipher's mapping. Since the cipher can do this to arbitrary letters, this could continue indefinitely for any single letter.

These were some of the first ciphers developed by Man. These include simple substitute ciphers, and letter shifting ciphers. However, these can be broken with *frequency analysis*.

1.1.2 Polyalphabetic Ciphers

Defn 7 (Polyalphabetic Cipher)**.** In a *polyalphabetic cipher* multiple letters are replaced by the cipher's mapping. Additionally, since the cipher can output multiple letters, the ciphered letters could be run through the cipher again.

These were developed in response to Monoalphabetic Ciphers being broken. However, these can also be broken, with *extended frequency analysis*.

Eventually, it was realized that the secrecy of the cipher is not sensible/possible. This leads us to the conclusion that **any cryptographic scheme should remain secure even if the adversary understands the cipher algorithm itself**.

1.1.3 Cryptographic Keys

The use of keys as ciphers is a slightly more modern occurrence.

Defn 8 (Kerckhoff's Principle)**.** *Kerckhoff's Principle* states that the security of the key used should alone be sufficient for a good cipher to maintain confidentiality under an attack. Essentially, the security of the key used should be sufficient such that the cipher can be maintained confidently while under attack.

However, only since the mid-1970s, has public key cryptography has been possible.

Computers can efficiently encrypt, given the following constraints:

Symmetric-Key Cryptography	Public-Key Cryptography
Block ciphers	Public-Key encryption
Stream ciphers	Digital Signature Schemes
Cryptographic Hash Functions	Key exchange protocols
	Electronic Cash/Cryptocurrency
	Interactive Proof Systems

Table 1.1: Uses of Key-Based Cryptography

1. Some modern techniques can only keep the keys secret if certain mathematical problems are Intractable.
   (a) Integer factorization
   (b) Discrete logarithm problems
2. However, there are no absolute proofs that a cryptographic technique is secure.

Defn 9 (Intractable)**.** An *intractable* problem is one in which there are no **efficient** algorithms to solve them.

2 Number Theory

Before we can start with any of the deeper cryptography stuff, we need to start with some basic number theory.

Defn 10 (Number Theory)**.** *Number theory* is a branch of pure mathematics devoted primarily to the study of the integers and integer-valued functions. Number theorists study prime numbers as well as the properties of objects made out of integers (for example, rational numbers) or defined as generalizations of the integers (for example, algebraic integers).

Defn 11 (Divides)**.** For $a, b \in \mathbb{Z}$, we say that a *divides* b (written $a \mid b$) if there exists an integer c such that $b = ac$.
Properties:
(i) $a \mid a$
(ii) If $a \mid b$ and $b \mid c$, then $a \mid c$.
(iii) If $a \mid b$ and $a \mid c$, then $a \mid (bx + cy)$ for any $x, y \in \mathbb{Z}$.
(iv) If $a \mid b$ and $b \mid a$, then $a = \pm b$.

2.1 Integer Long Division

For $a, b \in \mathbb{Z}$, with $b \geq 1$. Then an ordinary long division of a by b, i.e. $a \div b$ yields two integers q and r such that

$$a = qb + r, \text{where } 0 \leq r < b \tag{2.1}$$

q and r are called the Quotient and Remainder, respectively, and are **unique**.

Defn 12 (Quotient)**.** The *quotient*, q, of a divided by b is denoted $a \operatorname{div} b$.

Defn 13 (Remainder)**.** The *remainder*, r, of a divided by b is denoted $a \bmod b$.

Example 2.1: Integer Long Division.

If $a = 53$ and $b = 9$, what is $a \bmod b$?

$$53 = q9 + r$$
$$q = 5$$
$$r = 8$$

Thus, $53 \bmod 9 = 8$.

2.2 Greatest Common Divisor

Defn 14 (Common Divisor)**.** An integer c is a *common divisor* of a and b if $c \mid a$ and $c \mid b$.

Defn 15 (Greatest Common Divisor)**.** A non-negative integer d is called the *greatest common divisor* (GCD) of integers a and b if:

1. d is a Common Divisor of a and b.
2. For every other common divisor c it holds that $c \mid d$.

The greatest common divisor is denoted

$$\gcd(a, b) \tag{2.2}$$

$\gcd(a, b)$ is the **largest positive** integer dividing both a and b (except for $\gcd(0, 0) = 0$).

Remark 15.1. If $a, b \in \mathbb{Z}^+$, then $\text{lcm}(a, b) \cdot \gcd(a, b) = a \cdot b$

Example 2.2: Greatest Common Divisor.

What is the $\gcd(18, 24)$?

Common Divisors $= \{\pm 1, \pm 2, \pm 4, \pm 6\}$.
Since we can only allow positive integers,

$$\gcd(18, 24) = +6$$

2.3 Least Common Multiple

Defn 16 (Least Common Multiple)**.** A non-negative integer d is called the *least common multiple* (LCM) of integers a and b if:

1. $a \mid d$ and $b \mid d$
2. For every integer c such that $a \mid c$ and $b \mid c$, we have $d \mid c$.

The least common multiple is denoted

$$\text{lcm}(a, b) \tag{2.3}$$

$\text{lcm}(a, b)$ is the **smallest positive** integer divisible by both a and b.

Remark 16.1. If $a, b \in \mathbb{Z}^+$, then $\text{lcm}(a, b) \cdot \gcd(a, b) = a \cdot b$

2.4 Primality

Defn 17 (Relatively Prime)**.** a, b are called *relatively prime* if $\gcd(a, b) = 1$.

Defn 18 (Prime)**.** An integer $p \geq 2$ is called *prime* if its only positive divisors are 1 and p. Otherwise, p is called a *Composite*.

Defn 19 (Composite)**.** An integer $p \geq 2$ is called *composite* if it has more positive divisors than just 1 and p. Otherwise, p is called a *Prime*.

2.4.1 Number of Primes

The number of primes $\leq x$ is denoted

$$\pi(x) \tag{2.4}$$

1. There are infinitely many primes
2. $\lim_{x \to \infty} \frac{\pi(x)}{\frac{x}{\ln(x)}} = 1$
3. For $x \geq 17$, $\frac{x}{\ln(x)} < \pi(x) < \frac{1.25506x}{\ln(x)}$

2.5 Unique Factorization

Theorem 2.1 (Unique Factorization Theorem)**.** *Every integer $n \geq 2$ can be written as a product of prime powers,*

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$$

where $p_1, p_2, \ldots p_k$ are distinct primes and $e_1, e_2, \ldots e_k$ are positive integers. Furthermore, the factorization is unique up to rearrangement of the factors.

2.5.1 Greatest Common Divisor and Least Common Multiple with Unique Factors

If $a = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ and $b = p_1^{f_1} p_2^{f_2} \cdots p_k^{e_k}$, where $e_i, f_i, i = 1, 2, \ldots k$ are non-negative integers, then

$$\gcd(a,b) = p_1^{\min(e_1,f_1)} p_2^{\min(e_2,f_2)} \cdots p_k^{\min(e_k,f_k)} \tag{2.5}$$

and

$$\operatorname{lcm}(a,b) = p_1^{\max(e_1,f_1)} p_2^{\max(e_2,f_2)} \cdots p_k^{\max(e_k,f_k)} \tag{2.6}$$

2.6 Euler Phi Function

Defn 20 (Euler Phi Function)**.** For $n \geq 1$, let $\phi(n)$ denote the number of integers in the interval $[1, n]$, which are Relatively Prime to n. This function is called the *Euler Phi Function.*

$$\phi(n) = \left(p_1^{e_1} - p_1^{e_1-1}\right)\left(p_2^{e_2} - p_2^{e_2-1}\right) \cdots \left(p_k^{e_k} - p_k^{e_k-1}\right) \tag{2.7}$$

Remark 20.1. The Euler Phi Function is closely related to Set Order.

Theorem 2.2 (Euler Phi Function)**.** *There are a few properties of the Euler Phi Function that we will treat as true because of this theorem.*

(i) *If p is a Prime, then $\phi(p) = p - 1$.*
(ii) *If $\gcd(a,b) = 1$, then $\phi(ab) = \phi(a)\phi(b)$.*
(iii) *If $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, then*

$$\phi(n) = \left(p_1^{e_1} - p_1^{e_1-1}\right)\left(p_2^{e_2} - p_2^{e_2-1}\right) \cdots \left(p_k^{e_k} - p_k^{e_k-1}\right)$$

Example 2.3: Euler Phi Function. Exercise 1, Question 1.2a

Find the value of $\phi(36)$?

First, we note that 36 is not a Prime number, thus we need to find a set of primes that are equal to 36. The divisors of 36 are

$$\{\pm 1, \pm 2, \pm 3, \pm 4, \pm 9, \pm 12, \pm 18, \pm 36\}$$

And all possible prime numbers present as divisors of 36 are

$$\{2, 3\}$$

So, there must be some combination of $2^x \cdot 3^y$ that yields 36. In fact, $2^2 \cdot 3^2 = 4 \cdot 9 = 36$. Since 36 can be broken up as a product of 2 Prime numbers raised to some power, we can use Property **(iii)** of the Euler Phi Function to simplify this. But first, we need to separate the two values from each other using Property **(ii)** of the Euler Phi Function to apply Property **(iii)**.
We need to check $\gcd\left(2^2, 3^2\right) = 1$.

$$\begin{aligned} \gcd\left(2^2, 3^2\right) &= \gcd(4, 9) \\ 9 &= a4 + b \\ &= 2 \cdot 4 + 1 \\ 4 &= a \cdot 1 + b \\ &= 4 \cdot 1 + 0 \end{aligned}$$

So, $\gcd\left(2^2, 3^2\right) = 1$, so we can use Property **(ii)**.

$$\phi\left(2^2 \cdot 3^2\right) = \phi\left(2^2\right)\phi\left(3^2\right)$$

And now we can apply Property **(iii)** to find our answer.

$$\begin{aligned} \phi\left(2^2\right)\phi\left(3^2\right) &= \left(2^2 - 2^{2-1}\right)\left(3^2 - 3^{2-1}\right) \\ &= (4-2)(9-3) \\ &= (2)(6) \\ &= 12 \end{aligned}$$

Thus, $\phi(36) = 12$.

Lemma 2.2.1 (Computing the Greatest Common Divisor)**.** *If a and b are positive integers where $a > b$, then*

$$\gcd(a, b) = \gcd(b, a \bmod b) \tag{2.8}$$

Remark. This can be repeated to efficiently calculate the $\gcd(a, b)$. This is called the Euclidean Algorithm.

Defn 21 (Euclidean Algorithm)**.** The *euclidean algorithm* is a way to efficiently calculate the $\gcd(a, b)$.

1. Set $r_0 \leftarrow a, r_1 \leftarrow b, i \leftarrow 1$.
2. While $r_i \neq 0$ do:
   (a) Set $r_{i+1} \leftarrow r_{i-1} \bmod r_i, i \leftarrow i + 1$
3. Return r_i

Example 2.4: Euclidean Algorithm. Exercise 1, Question 1.1a

Find the Greatest Common Divisor of 222 and 1870?

$$\begin{aligned}
1870 &= a222 + b \\
&= 8 \cdot 222 + 94 \\
222 &= a94 + b \\
&= 2 \cdot 94 + 34 \\
94 &= a34 + b \\
&= 2 \cdot 34 + 26 \\
34 &= a26 + b \\
&= 1 \cdot 26 + 8 \\
26 &= a8 + b \\
&= 3 \cdot 8 + 2 \\
8 &= a2 + b \\
&= 4 \cdot 2 + 0
\end{aligned}$$

Thus, since $4 \cdot 2 = 8$, 2 is the Greatest Common Divisor of 222 and 1870.

Theorem 2.3. *There exist integers x, y such that $\gcd(a, b)$ can be written as*

$$\gcd(a, b) = ax + by \tag{2.9}$$

Proof.

$$\begin{aligned}
\gcd(a, b) &= r_i \\
&= r_{i-2} - q_{i-1} r_{i-1} \\
&= r_{i-2} - q_{i-1}(r_{i-3} - q_{i-2} r_{i-2}) \\
&\vdots \\
&= r_0 x + r_1 y \\
&= ax + by
\end{aligned}$$

for some integers $x, y \in \mathbb{Z}$. ■

This means that the Euclidean Algorithm can be extended to return the values of x and y from Equation (2.9).

Defn 22 (Extended Euclidean Algorithm)**.** The *extended euclidean algorithm* is a way to efficiently calculate the linear pair of integers $(x, y \in \mathbb{Z})$ that satisfy Equation (2.9).

$$\gcd(a, b) = ax + by$$

1. If $b = 0$, then return $a, x \leftarrow 1, y \leftarrow 0$.

2. Set $x_2 \leftarrow 1, x_2 \leftarrow 0, y_2 \leftarrow 0, y_1 \leftarrow 1$
3. While $b > 0$ do:
   (a) $q \leftarrow a \operatorname{div} b, r \leftarrow a - qb, x \leftarrow x_2 - qx_1, y \leftarrow y_2 - qy_1$
   (b) $a \leftarrow b, b \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x, y_2 \leftarrow y_1, y_1 \leftarrow y$.
4. Set $d \leftarrow a, x \leftarrow x_2, y \leftarrow y_2$ and return d, x, y.

Example 2.5: Extended Euclidean Algorithm. Exercise 1, Question 1.1b

Find the integers x and y such that $\gcd(222, 1870) = 222x + 1870y$?

From Example 2.4, we know $\gcd(222, 1870) = 2$, so we can plug that in. We now know that

$$2 = 222x + 1870y$$

Now, we essentially run the Euclidean Algorithm backwards.

$$\begin{aligned}
2 &= 26 - 3 \cdot 8 \\
&= 26 - 3(34 - 1 \cdot 26) = 26 - 3 \cdot 34 + 3 \cdot 26 \\
&= 4 \cdot 26 - 3 \cdot 34 \\
&= 4(94 - 2 \cdot 34) - 3 \cdot 34 = 4 \cdot 94 - 8 \cdot 34 - 3 \cdot 34 \\
&= 4 \cdot 94 - 11 \cdot 34 \\
&= 4 \cdot 94 - 11(222 - 2 \cdot 94) = 4 \cdot 94 - 11 \cdot 222 + 22 \cdot 94 \\
&= 26 \cdot 94 - 22 \cdot 222 \\
&= 26(1870 - 8 \cdot 222) - 22 \cdot 222 = 26 \cdot 1870 - 208 \cdot 222 - 11 \cdot 222 \\
&= -219 \cdot 222 + 26 \cdot 1870
\end{aligned}$$

Now we need to check the solution we might have found

$$\begin{aligned}
-219 \cdot 222 + 26 \cdot 1870 &= 2 \\
-48618 + 48620 &= 2 \\
2 &= 2
\end{aligned}$$

Thus,

$$\begin{aligned}
x &= -219 \\
y &= 26
\end{aligned}$$

2.7 The Integers modulo n

Let n be a positive integer.

Defn 23 (Congruence)**.** If a and b are integers, then a *is said to be congruent to* b *modulo* n, which is written as

$$a \equiv b \pmod{n} \tag{2.10}$$

If n divides $(a - b)$, i.e. $n \,|\, (a - b)$, then we call n the *modulus* of the congruence.

Theorem 2.4. *For* $a, a_1, b, b_1, c \in \mathbb{Z}$*, we have*

(i) $a \equiv b \pmod{n}$ if and only if a *and* b *leave the same Remainder when divided by* n*.*
(ii) $a \equiv a \pmod{n}$
(iii) *If* $a \equiv b \pmod{n}$*, then* $b \equiv a \pmod{n}$
(iv) *If* $a \equiv b \pmod{n}$ *adn* $b \equiv c \pmod{n}$*, then* $a \equiv c \pmod{n}$
(v) *If* $a \equiv a_1 \pmod{n}$ *and* $b \equiv b_1 \pmod{n}$*, then* $a + b = a_1 + b_1 \pmod{n}$ *and* $ab = a_1 b_1 \pmod{n}$*.*

Properties ***(ii)*** *to* ***(iv)*** *are called* reflexivity*,* symmetry*, and* transitivity*, respectively.*

Example 2.6: Integers modulo n. Exercise 1, Question 1.2b

Write all the units (Invertible Element) in $\mathbb{Z}_{36}$?

First, we start by constructing our set of integers modulo n.

$$\mathbb{Z}_{36} = \big\{[0], [1], [2], [3], [4], \ldots, [35]\big\}$$

Since we are only worried about the units of $\mathbb{Z}_{36}$, we need to find the integers that satisfy Equation (4.3). This is done by finding an a value that has a Multiplicative Inverse, which requires that Equation (3.1) be true, namely

$$\gcd(a, n) = 1$$

This leaves us with

$$\mathbb{Z}_{36} = \big\{[1], [5], [7], [11], [13], [17], [19], [23], [25], [29], [31], [35]\big\}$$

which is the solution.

2.8 Equivalence Classes

Defn 24 (Equivalence Class)**.** Congruence modulo n partitions $\mathbb{Z}$ into n sets, called *equivalence class*es, where each integer belongs to exactly one equivalence class.

For example, these are all congruent to each other modulo n:

$$[0] = \{\ldots, -2n, -n, 0, n, 2n, \ldots\} \tag{2.11a}$$

$$[1] = \{\ldots - 2n + 1, -n + 1, 1, n + 1, 2n + 1 \ldots\} \tag{2.11b}$$

$$[r] = \mathbb{Z}_r = (x \bmod r) + n\,\mathbb{Z} \tag{2.11c}$$

Since all elements in an equivalent class have the same Remainder, r, we use r as a *represenatative* for the equivalence class.

Remark 24.1. In this case, the representatives of the equivalence classes shown in Equations (2.11a) to (2.11b) are 0 and 1, respectively, and consist of all integers that are mod 0 or mod 1, respectively.

3 Number Theory on Sets

While this section is not technically different than Section 2, it is worth it to split these up, since Section 2 does not deal with sets. However, using what we learned in Section 2, Number Theory, it is natural to extend these to sets of numbers.

3.1 $\mathbb{Z}_n$

Defn 25 ($\mathbb{Z}_n$)**.** The Integers modulo n, denoted $\mathbb{Z}_n$, is the set of (Equivalence Classes of) integers $\{[0], [1], \ldots, [n-1]\}$. Addition, subtraction, and multiplication are all performed with modulo n. Examples 3.1 to 3.3 demonstrate this.

Example 3.1: Addition on Integers mod n.

When dealing with the set of integers $\mathbb{Z}_{15}$, what is the sum of 5 and 9?

$$5 \bmod 15 + 9 \bmod 15 = 11 \bmod 15$$

Thus, the answer is 11.

Example 3.2: Subtraction on Integers mod n.

When dealing with the set of integers $\mathbb{Z}_{15}$, what is 5 minus 9?

$$\begin{aligned}5 \bmod 15 - 9 \bmod 15 &= 5 \bmod 15 + (-9 \bmod 15) \\ &= 5 \bmod 15 + \underbrace{-9 \bmod 15}_{-9+15=6} \\ &= 5 \bmod 15 + 6 \bmod 15 \\ &= 11 \bmod 15\end{aligned}$$

Thus, the answer is, again, 11.

Example 3.3: Multiplication on Integers mod n.

When dealing with the set of integers $\mathbb{Z}_{15}$, what is the product of 5 and 9?

$$\begin{aligned}5 \bmod 15 \cdot 9 \bmod 15 &= 45 \bmod 15 \\ &= 0\end{aligned}$$

Thus, the answer is 0, because $45 = 3 \cdot 15$.

3.2 Inverse in $\mathbb{Z}_n$

Addition, subtraction, and multiplication can be performed trivially in $\mathbb{Z}_n$, as shown in Examples 3.1 to 3.3. However, the concept of division is a little bit more difficult.

Defn 26 (Multiplicative Inverse)**.** Let $a \in \mathbb{Z}_n$. The *multiplicative inverse* of a is an integer $x \in \mathbb{Z}_n$, such that $ax = 1$. If such an integer, x, exists, then a is said to be *invertible* and x is called the inverse of a, denoted as a^{-1}.

Defn 27 (Division in $\mathbb{Z}_n$)**.** *Division* of a by an element $b \in \mathbb{Z}_n$ (written a/b) is defined as ab^{-1}, and is only defined if b has a Multiplicative Inverse.

Defn 28 (Invertible)**.** Let $a \in \mathbb{Z}_n$. Then a is *invertible* if and only iff

$$\gcd(a, n) = 1 \tag{3.1}$$

Proof. Assume that $\gcd(a, n) = 1$. We know that $1 = \gcd(a, n) = xa + yn$ for some $x, y \in \mathbb{Z}$. Since yn is a multiple of n, namely $yn \bmod n = 0$, it is removed from the equation. Then $x \bmod n$ is an inverse to a.

Now assume $\gcd(a, n) > 1$. If a has an inverse x, then $ax = 1 \bmod n$, which means $1 = ax + ny$ for some $x, y \in \mathbb{Z}$, directly contradicting the assumption that $\gcd(a, n) = 1$. ■

The two possible cases of division, i.e. possible and impossible, are shown in Examples 3.4 to 3.5.

Example 3.4: Possible Division on Integers mod n.

When dealing with the set of integers $\mathbb{Z}_{15}$, what is the result from the division of 5 by 11?

$$5 \bmod 15 \div 11 \bmod 15 = 5 \cdot 11^{-1}$$

Now we need to find the Multiplicative Inverse of 1.

$$11^{-1} = \gcd(11, 15)$$

We can compute the Greatest Common Divisor efficiently with the Euclidean Algorithm.

$$\begin{aligned}15 &= 1 \cdot 11 + 4 \\ 11 &= 2 \cdot 4 + 3 \\ 4 &= 1 \cdot 3 + 1 \\ 3 &= 3 \cdot 1 + 0\end{aligned}$$

Thus, the Euclidean Algorithm gives us $\gcd(11, 15) = 1$. Since $\gcd(11, 15) = 1 = 1$, 11 **does** have a Multiplicative Inverse, making the division possible. Now we need to run through the Extended Euclidean Algorithm, to find the values $x, y \in \mathbb{Z}$.

$$\begin{aligned} 1 &= 4 - 1 \cdot 3 \\ &= 4 - 1 \cdot (11 - 2 \cdot 4) = 3 \cdot 4 - 1 \cdot 11 \\ &= 3(15 - 1 \cdot 11) - 1 \cdot 11 \\ &= 3 \cdot 15 - 3 \cdot 11 - 1 \cdot 11 \\ &= 3 \cdot 15 - 4 \cdot 11 \end{aligned}$$

Thus,

$$\begin{aligned} x &= -4 \\ y &= 3 \end{aligned}$$

Now we know

$$(11 \bmod 15)^{-1} = -4 \bmod 15$$

Since -4 is not part of $\mathbb{Z}_{15}$, we need to find the additive inverse. $-4 + 15 = 11$. Thus,

$$(11 \bmod 15)^{-1} = 11 \bmod 15$$

Now, we perform a simple substitution.

$$\begin{aligned} 5 \bmod 15 / 11 \bmod 15 &= 5 \bmod 15 \cdot (11 \bmod 15)^{-1} \\ &= 5 \bmod 15 \cdot 11 \bmod 15 \\ &= 55 \bmod 15 \\ &= 10 \end{aligned}$$

So, the result of the division of 5 by 11 is 10.

Example 3.5: Impossible Division on Integers mod n.

When dealing with the set of integers $\mathbb{Z}_{15}$, what is the result from the division of 5 by 9?

$$5 \bmod 15 \div 9 \bmod 15 = 5 \cdot 9^{-1}$$

Now we need to find the Multiplicative Inverse of 9.

$$9^{-1} = \gcd(9, 15)$$

We can compute the Greatest Common Divisor efficiently with the Euclidean Algorithm.

$$\begin{aligned} 15 &= 1 \cdot 9 + 6 \\ 9 &= 1 \cdot 6 + 3 \\ 3 &= 1 \cdot 3 + 0 \end{aligned}$$

Thus, the Euclidean Algorithm gives us $\gcd(9, 15) = 3$. Since $\gcd(9, 15) = 3 \neq 1$, 9 does **not** have a Multiplicative Inverse, making the division impossible.

3.3 Chinese Remainder Theorem

Theorem 3.1 (Chinese Remainder Theorem)**.** *Let the integers $n_1, n_2, \ldots, n_k$ be pairwise Relatively Prime. Then the system of Congruences*

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

has a unique solution modulo $n = n_1 n_2 \cdots n_k$.

Defn 29 (Gauss's Algorithm)**.** The solution x to the system of Congruences promised by the Chinese Remainder Theorem can be calculated as

$$x = \left(\sum_{i=1}^{k} a_i N_i M_i \right) \bmod n \tag{3.2}$$

where $N_i = \frac{n}{n_i}$ and $M_i = N_i^{-1} = \left(\frac{n}{n_i}\right)^{-1} \bmod n_i$ (M_i is the Multiplicative Inverse of $N_i \bmod n_i$).

Defn 30 (Chinese Remainder Theorem)**.** The *Chinese Remainder Theorem* allows us to change the way we represent elements of our set, $\mathbb{Z}_n$.

The integers modulo n, $\mathbb{Z}_n$, where $n = n_1 n_2$ and $\gcd(n_1, n_2) = 1$. An element $a \in \mathbb{Z}_n$ has a unique representation: $(a \bmod n_1, a \bmod n_2)$. We can denote this mapping by $\gamma : \mathbb{Z}_n \to \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$.

(i) $\gamma(a) = \gamma(b)$ if and only if $a = b$.
(ii) For all $(a_1, a_2) \in \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, there exists an a such that $\gamma(a) = (a_1, a_2)$.
(iii) $\gamma(a + b) = \gamma(a) + \gamma(b)$
(iv) $\gamma(ab) = \gamma(a)\gamma(b)$

These properties (Properties **(i)** to **(iv)**) make γ an *Isomorphism*.

Remark 30.1. In the case of large integers for cryptography, knowing just one part of the number can ehlp get the other part. However, if the number is very large, 2048 bits for instance, these calculations start becoming Intractable.

Example 3.6: Chinese Remainder Theorem Mapping.

Find the mapping of 7 when in $\mathbb{Z}_{15}$?

We need to find a prime factorization for 15.

$$15 = 3^1 \cdot 5^1$$

So, the 2 modulos we will use are 3 and 5. Additionally, since 7 is an element in $\mathbb{Z}_{15}$ we can simply say,

$$7 \Leftrightarrow (7 \bmod 3,\ 7 \bmod 5) = (1, 2)$$

Example 3.7: Reverse Chinese Remainder Theorem Mapping 1.

Which element of $\mathbb{Z}_{168}$ corresponds to the pair $([7], [5]) \in \mathbb{Z}_8 \times \mathbb{Z}_{21}$?

If we apply the Extended Euclidean Algorithm to 8 and 21, we get $1 = 8 \cdot 8 + (-3) \cdot 21$. Note that $8 \cdot 8 = 64$ is congruent to 0 mod 8 and 1 mod 21, and $(-3) \cdot 21 = -63$ is congruent to 1 mod 8 and 0 mod 21. Therefore

$$\begin{aligned} 5 \cdot 64 + 7 \cdot (-63) &\equiv 5 \cdot 0 + 7 \cdot 1 = 7 (\mathrm{mod} 8) \\ 5 \cdot 64 + 7 \cdot (-63) &\equiv 5 \cdot 1 + 7 \cdot 0 = 5 (\mathrm{mod} 21) \end{aligned}$$

so,

$$5 \cdot 64 + 7 \cdot (-63) = -121 \equiv 47 (\mathrm{mod} 168)$$

works, that is, $[47] \Leftrightarrow ([7], [5])$.

Example 3.8: Reverse Chinese Remainder Theorem Mapping 2.

Given elements $z_2 \in \mathbb{Z}_2$ and $z_5 \in \mathbb{Z}_5$, find the corresponding element $z_{10} \in \mathbb{Z}_{10}$?

Because the $\gcd(2,5) = 1$, we can also use the Extended Euclidean Algorithm. It returns $1 = 1 \cdot 5 - 2 \cdot 2$. Now we need to match the terms up with modulo factors that do not reduce them, i.e. any element z_5 should not be matched with 5, and likewise with z_2 and 2.
Thus,

$$z_{10} = (z_2 \cdot 5 - z_5 \cdot 2) \bmod 10$$

3.4 Multiplicative Groups, $\mathbb{Z}_n^*$

Defn 31 (Multiplicative Group, $\mathbb{Z}_n^*$)**.** Define the *multiplicative group* of $\mathbb{Z}_n$, denoted $\mathbb{Z}_n^*$ as the set of all elements in $\mathbb{Z}_n$ with Multiplicative Inverses.

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \,|\, \gcd(a,b) = 1\} \tag{3.3}$$

Defn 32 (Set Order)**.** The *order a set*, for example, $\mathbb{Z}_n^*$, is the number of elements in $\mathbb{Z}_n^*$ ($|\mathbb{Z}_n^*|$). From the definition of the Euler Phi Function

$$|\mathbb{Z}_n^*| = \phi(n) \tag{3.4}$$

Remark 32.1 (Closed Under Multiplication)*.* Since the produce of two elements with Multiplicative Inverses is another element with a Multiplicative Inverse, we say that $|\mathbb{Z}_n^*|$ is *closed under multiplication.*

Defn 33 (Element Order)**.** The *order of an element* $a \in \mathbb{Z}_n^*$, denoted $\text{ord}(a)$ is defined as the least positive integer t ($t \in \mathbb{Z}$) such that

$$a^t \bmod n = 1 \tag{3.5}$$

Lemma 3.1.1 (Element Order)**.** *Let* $a \in \mathbb{Z}_n^*$ *. If* a^s *for some* s*, then* $\text{ord}(a) \,|\, s$*. In particular,* $\text{ord}(a) \,|\, \phi(n)$ *must be true.*

Example 3.9: Element Order. Exercise 1, Problem 1.6b

Find the ord(5) in $\mathbb{Z}_8^*$?

We need to solve

$$a^t \bmod n = 1$$

where $a = 5$ and $n = 8$.

$$5^t \bmod 8 = 1$$

Now we test values for t until we satisfy the equation.

$$t = 1 \rightarrow 5^1 \bmod 8 = 5 \bmod 8 = 5 \neq 1$$
$$t = 2 \rightarrow 5^2 \bmod 8 = 25 \bmod 8 = 1 = 1$$

Since $t = 2$ satisfies our equation, the ord(5) = 2.

Element Order. Let $t = \text{ord}(a)$. By long division, $s = qt + r$, where $r < t$. Then $a^s = a^{qt+r} = a^{qt}a^r$ and since $a^t = 1$, from Equation (3.5), we have $a^s = a^r$ and $a^r = 1$. This reduction is shown below:

$$\begin{aligned} a^s &= a^{qt+r} \\ &= a^{qt}a^r \\ &= \left(a^t\right)^q a^r \\ &= (1)^q a^r \\ &= 1^q a^r \\ &= 1a^r \\ &= a^r \end{aligned}$$

But, $r < t$, so we must have $r = 0$, and so $\text{ord}(a) \,|\, s$. ■

3.5 Euler's Theorem

Theorem 3.2 (Euler's Theorem)**.** *If $a \in \mathbb{Z}_n^*$, then*

$$a^{\phi(n)} \equiv 1 \pmod{n} \tag{3.6}$$

Euler's Theorem. Let $\mathbb{Z}_n^* = \{a_1, a_2, \ldots, a_{\phi(n)}\}$. Looking at the set of elements $A = \{aa_1, aa_2, \ldots, aa_{\phi(n)}\}$, it is easy to see that $A = a\,\mathbb{Z}_n^*$. So we have 2 ways of writing the product of all of the elements, i.e.

$$\prod_{i=1}^{\phi(n)} aa_i = \prod_{i=1}^{\phi(n)} a_i$$

This leads to

$$\prod_{i=1}^{\phi(n)} a = a^{\phi(n)} = 1$$

which is the same as what we said in Equation (3.6). ■

3.6 Fermat's Little Theorem

Theorem 3.3 (Fermat's Little Theorem)**.** *Let p be a Prime number. If $\gcd(a, p) = 1$, then*

$$a^{p-1} \equiv 1 \pmod{p} \tag{3.7}$$

Remark. This ties in with Euler's Theorem, because working in $\mathbb{Z}_n$, all exponents can be reduced by modulo $\phi(n)$.

3.7 Generators

Defn 34 (Generator)**.** Let $a \in \mathbb{Z}_n^*$. If the Element Order of a is equal to the Euler Phi Function, i.e. $\text{ord}(a) = \phi(n)$, then a is said to be a *generator* (or a *primitive element*) of $\mathbb{Z}_n^*$.

$$\text{ord}(a) = \phi(n) \tag{3.8a}$$

$$\text{ord}(a) = |\mathbb{Z}_n^*| \tag{3.8b}$$

Furthermore, if $\mathbb{Z}_n^*$ has a generator, then $\mathbb{Z}_n^*$ is said to be*Cyclic*.

Remark 34.1. It is clear that if $a \in \mathbb{Z}_n^*$ is a Generator, then every element in $\mathbb{Z}_n^*$ can be expressed as a^i for some integer i ($i \in \mathbb{Z}$). So, we can write

$$\mathbb{Z}_n^* = \{a^i | 0 \leq i \leq \phi(n) - 1\} \tag{3.9}$$

Example 3.10: Cyclic Group. Exercise 1, Problem 1.6c

Is $\mathbb{Z}_8^*$ a Cyclic Group?

We need to find an a that satisfies $\text{ord}(a) = \phi(n)$. In this case $n = 8$. First, we will calculate $\phi(n)$.

$$\phi(n) = \phi(8)$$

We need to find a Prime factorization of 8.

$$\begin{aligned}\phi(8) &= \phi\left(2^3\right) \\ &= \left(2^3 - 2^{3-1}\right) \\ &= \left(2^3 - 2^2\right) \\ &= (8 - 4) \\ &= 4\end{aligned}$$

So, $\phi(8) = 4$.
Now we need to solve

$$\text{ord}(a) = 4$$

All of the terms in $\mathbb{Z}_8^*$ must be Invertible, so they **must** satisfy $\gcd(z, 8) = 1$, $z \in \mathbb{Z}_8$. All of the terms in $\mathbb{Z}_8$ are:

$$\mathbb{Z}_8 = \Big\{[0], [1], [2], [3], [4], [5], [6], [7]\Big\}$$

Now we check the Greatest Common Divisors for $z \in \mathbb{Z}_8$.

$$\begin{aligned} \gcd(0,8) &= 8 & \gcd(4,8) &= 2 \\ \gcd(1,8) &= 1 & \gcd(5,8) &= 1 \\ \gcd(2,8) &= 2 & \gcd(6,8) &= 2 \\ \gcd(3,8) &= 1 & \gcd(7,8) &= 1 \end{aligned}$$

So,

$$\mathbb{Z}_8^* = \Big\{[1],[3],[5],[7]\Big\}$$

Now we test $\operatorname{ord} a = 4$, $a \in \mathbb{Z}_8^*$.

$$\begin{aligned} \operatorname{ord}(1) &= 1 \\ \operatorname{ord}(3) = 3^t \bmod 8 = 1 \to t = 2 &\to \operatorname{ord}(3) = 2 \\ \operatorname{ord}(5) &= 2 \\ \operatorname{ord}(7) = 7^t \bmod 8 = 1 \to t = 2 &\to \operatorname{ord}(7) = 2 \end{aligned}$$

Since no element from $\operatorname{ord}(\mathbb{Z}_8^*) = \phi(8) = 4$, $\mathbb{Z}_8^*$ is **NOT** Cyclic.

3.8 Quadratic Residues

Defn 35 (Quadratic Residue)**.** An element $a \in \mathbb{Z}_n^*$ is said to be a *quadratic residue* modulo n (or a *square*) if there exists an $x \in \mathbb{Z}_n^*$ such that $x^2 = a$.

$$a \in \mathbb{Z}_n \, \exists x \in \mathbb{Z}_n^* \;\; x^2 = a \pmod{n} \tag{3.10a}$$

$$a \in \mathbb{Z}_n \, \exists x \in \mathbb{Z}_n^* \;\; a \equiv x^2 \pmod{n} \tag{3.10b}$$

Remark 35.1 (Square Root). If $x^2 = a$, then x is called the *square root* of $a \bmod n$.

Otherwise, a is called a *Quadratic Non-Residue modulo n.*

Defn 36 (Quadratic Non-Residue)**.** An element $a \in \mathbb{Z}_n^*$ is said to be a *quadratic non-residue* modulo n if there does not exist an $x \in \mathbb{Z}_n^*$ such that $x^2 = a$.

$$a \in \mathbb{Z}_n \nexists x \in \mathbb{Z}_n^* \;\; x^2 = a \pmod{n}$$

Otherwise, a is called a *Quadratic Residue modulo n.*

4 Abstract Algebra

In the beginning of this course, we covered some basic abstract algebra. These include:

- Aspects covering integers and calculus modulo n
- Covering a few examples of algebraic structures
- Some basic concepts from abstract algebra, which provide a more general treatement of algebraic structures
- In cryptography, we are generally interested in *finite* sets

4.1 Groups

Defn 37 (Binary Operation)**.** A *binary operation*, denoted $*$ on a set S, is a mapping from $S * S$ to S.

Remark 37.1. Note that $*$ is **NOT** multiplication nor any kind of convolution. It is just a placeholder for some Binary Operation that can be done.

Defn 38 (Group)**.** A *group* is a set G and a Binary Operation, $*$, on G denoted as $(G, *)$, which satisfies the following properties:

(i) $a * (b * c) = (a * b) * c$ for all $a, b, c, \in G$ (Associativity)..

(ii) There is a special element $1 \in G$ such that $a * 1 = 1 * a = a$ for all $a \in G$ (Identity).

$$\exists 1 \in G \forall a \in G \; (a * 1 = 1 * a = a)$$

(iii) For each $a \in G$ there is an element $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = 1$ (Inverse).

We call 1 the *identity element* as defined in Property **(ii)**, and call a^{-1} the *inverse* of a. Furthermore,

(iv) If $a * b = b * a$ for all $a, b \in G$ (Abelian/Commutativity).

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

Example 4.1: Show Set Is Group. Exercise 1, Problem 1.8a

Let S be the set of binary triples, $S = \{(s_0, s_1, s_2) \,|s_i \in \mathbb{Z}_2\}$. Let the operation be bitwise addition.

Remark. In this case, "bitwise addition" means element-wise addition, i.e. each element of each triple gets added together. Additionally, all additions are done in modulo 2.

I will define the bitwise addition Binary Operation with the symbol .+, like how MATLAB or GNU Octave define it. We will need to prove that Properties **(i)** to **(ii)** are true. Property **(iv)** is *not* necessary to show that a set is a Group. Property **(iv)** only shows that the Group is Abelian
First, the elements present in the set S:

$$S = \Big\{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\Big\}$$

To prove that S is a group with an element-wise addition Binary Operation, I will arbitrarily select

$$a = (0,1,0)$$
$$b = (0,1,1)$$
$$c = (1,1,0)$$

Starting with Property **(i)**:

$$a \,.+ (b \,.+ c) = (a \,.+ b) \,.+ c$$
$$(0,1,0) \,.+ \big((0,1,1) \,.+ (1,1,0)\big) = \big((0,1,0) \,.+ (0,1,1)\big) \,.+ (1,1,0)$$
$$(0,1,0) \,.+ \big((1,0,1)\big) = \big((0,0,1)\big) \,.+ (1,1,0)$$
$$(1,1,1) = (1,1,1)$$

Thus, Property **(i)** is satisfied.
Now, Property **(ii)**

$$a \,.+ 1 = a$$
$$(0,1,0) \,.+ 1 = (0,1,0)$$
$$1 = (0,1,0) \,.+ \Big(-(0,1,0)\Big)$$

It is important to remamber that these additions are done in the modulo 2 domain, so addition and subtraction yield the same results, making negatives irrelevant.

$$1 = (0,1,0) \,.+ (0,1,0)$$
$$1 = (0,0,0)$$

Thus, Property **(ii)** is satisfied, and $1 = (0,0,0)$.
Lastly, we need to prove Property **(iii)**.

$$a \,.+ a^{-1} = 1$$
$$(0,1,0) \,.+ a^{-1} = (0,0,0)$$
$$a^{-1} = (0,0,0) \,.+ \Big(-(0,1,0)\Big)$$

Like before, addition and subtraction are irrelevant here.

$$a^{-1} = (0,0,0) \,.+ (0,1,0)$$
$$a^{-1} = (0,1,0)$$

Thus, Property **(iii)** is satisfied, where each element a is its own inverse.
Since Properties **(i)** to **(iii)** are satisfied, S **is** a Group with the bitwise addition Binary Operation.

Defn 39 (Abelian)**.** An *abelian* group, also called a *commutative Group*, is a group in which the result of applying the group operation to two group elements does not depend on the order in which they are written. That is, these are the groups that obey the axiom of commutativity.

4.1.1 Examples of Groups

- $\mathbb{Z}$ with the addition Binary Operation, denoted $(\mathbb{Z}, +)$ is a Group.
- Finite Groups are $(\mathbb{Z}_n, +)$ and $(\mathbb{Z}_n^*, \cdot)$, where $\cdot$ denotes multiplication modulo n
- **NOTE:** $(\mathbb{Z}_n, \cdot)$ is **NOT** a Group, nor is $(\mathbb{Z}, \cdot)$.

4.1.2 Definitions for Groups

Defn 40 (Subgroup)**.** A non-empty subset H of a Group G is called a *subgroup* of G, if H itself is a Group with respect to the operation of G.

Example 4.2: Find Subgroups. Exercise 1, Problem 1.10

Find all Subgroups in the multiplicative group $\mathbb{Z}_{19}^*$ (under the multiplication Binary Operation)?

The goal here is to find a Generator that can be raised to some set of exponents that can generate Subgroups of $\mathbb{Z}_{19}^*$. The first thing to note is that 19 is a Prime, so

$$|\mathbb{Z}_{19}^*| = 18 = \phi(n)$$

We can also apply Lagrange's Theorem to restrict the number of Subgroups possible. Because $\mathbb{Z}_{19}^*$ is a finite Group, and we are considering G to be a Subgroup of $\mathbb{Z}_{19}^*$, we apply Lagrange's Theorem. This means $|G| \,||\mathbb{Z}_{19}^*|$ must be true. Thus,

$$|G| = \{1, 2, 3, 6, 9, 18\}$$

Now, we can find a Generator for the Group $\mathbb{Z}_{19}^*$. This means, we need to find some element a that raised to some power t, modulo 19 will yield 1. We know that t **must** equal 18, because $\phi(n) = 18$, and we want to find a Generator. So, we solve for a.

$$a^{18} \bmod 18 = 1$$

In this case, $a = 2$, making 2 a Generator of $\mathbb{Z}_{19}^*$, or $< 2 >$.
Now, all possible Subgroups of $\mathbb{Z}_{19}^*$ are:

$$G_1 = \left\{ \left[2^{18}\right] \right\}$$
$$G_2 = \left\{ \left[2^{9}\right], \left[2^{18}\right] \right\}$$
$$G_3 = \left\{ \left[2^{6}\right], \left[2^{12}\right], \left[2^{18}\right] \right\}$$
$$G_6 = \left\{ \left[2^{3}\right], \left[2^{6}\right], \left[2^{9}\right], \left[2^{12}\right], \left[2^{15}\right], \left[2^{18}\right] \right\}$$
$$G_9 = \left\{ \left[2^{2}\right], \left[2^{4}\right], \left[2^{6}\right], \left[2^{8}\right], \left[2^{10}\right], \left[2^{12}\right], \left[2^{14}\right], \left[2^{16}\right], \left[2^{18}\right] \right\}$$
$$G_{18} = \left\{ \left[2^{0}\right], \left[2^{1}\right], \ldots, \left[2^{18}\right] \right\}$$

Defn 41 (Cyclic)**.** G is *cyclic* if there is an element $a \in G$ such that each $b \in G$ can be written as a^i for some integer i. The element a is called a *Generator* of G.

Example 4.3: Cyclic.

Find a, the Cyclic term in the Group $\mathbb{Z}_{15}$?

$$\begin{aligned}\mathbb{Z}_{15} &= \{3, 6, 9, 12, 0, \ldots\} \\ &= \{3, 3+3, 3+3+3, 3+3+3+3, 5 \cdot 3, \ldots\}\end{aligned}$$

Thus, $< a >= 3 \rightarrow < 3 >$.

Defn 42 (Element Order)**.** The *order of an element* a, denoted $\text{ord}(a)$ is defined as the least positive integer t ($t \in \mathbb{Z}$) such that

$$a^t = 1 \tag{4.1}$$

If such an integer t does not exist, then the order of a is defined to be ∞.

Remark 42.1. **NOTE:**

- The a^t part does not mean exponentiation, but rather repeated uses of the Binary Operation used to create the Group.
- The 1 is not a value 1, but the Identity of the Group, as defined by Property **(ii)**.

Example 4.4: Order of Element in Group. Exercise 1, Problem 1.8c

Given the element $(1, 1, 1,)$ from the group of bit triples, $S = \{(s_0, s_1, s_2) \,|s_i \in \mathbb{Z}_2\}$, using an bitwise addition Binary Operation, what is the Element Order of $(1, 1, 1)$? The identity element of this Group is $1 = (0, 0, 0)$.

It is important to remember the note in Remark 42.1, especially for this problem.
Since the given Binary Operation was "bitwise", i.e. element-wise, I will define the operation to be $.+$. In this case, the Element Order of any element in this Group will be the number of element-wise additions that must occur to get the identity element, 1.
For this problem, since we are working in the modulo 2 domain, we have some basic facts:

$$\begin{aligned}(0+0) \bmod 2 &= 0 \\ (1+0) \bmod 2 &= 1 \\ (0+1) \bmod 2 &= 1 \\ (1+1) \bmod 2 &= 0\end{aligned}$$

And for subtraction:

$$\begin{aligned}(0-0) \bmod 2 &= 0 \\ (1-0) \bmod 2 &= 1 \\ (0-1) \bmod 2 &= -1 \bmod 2 = 1 \\ (1-1) \bmod 2 &= 0\end{aligned}$$

We start by constructing the equation needed to solve this problem.

$$(1, 1, 1). + a = (0, 0, 0)$$

Now we can move the $(1, 1, 1)$ term over to the left, and since we are working in the modulo 2 domain, the "negative" that would be introduced by normal subtraction (negative addition) is irrelevant. Thus, we end up with

$$\begin{aligned}a &= (0, 0, 0). + (1, 1, 1) \\ &= (1, 1, 1)\end{aligned}$$

This means that if t from Equation (4.1) is 2, we get the identity element 1. So, our answer is $t = 2$, thus $\text{ord}\big[(1, 1, 1)\big] = 2$.

Lemma 4.0.1. *Let $a \in G$ be an element of finite Element Order t. Then the set of all powers of a forms a Cyclic Subgroup of G, denoted by $< a >$. Furthermore, the Element Order of $< a >$ is t.*

Defn 43 (Left Coset)**.** Let H be a Subgroup in G and pick an element $a \in G$. A set of elements of the form

$$aH = \{ah | h \in H\} \tag{4.2}$$

is called the *left coset* of H. If G is commutative, it is simply called a *coset*.

The set consisting of all such left cosets is written G/H. Note that H itself is a left coset. Furthermore, every left coset contains the same number of elements (the order of H) and every element is contained in exactly one left coset.

So, the elements of G are partitioned into $|G|/|H|$ different cosets, each containing $|H|$ elements.

4.2 Properties of Groups

(i) Suppose $a^n = 1$ for some $n > 0$. We must have $ElementOrder(a) \mid n$.

1. Write $n = k \cdot \text{ord}(a) + r$, where $0 \leq r \leq \text{ord}(a)$.
2. Then, $1 = a^n = a^{k \cdot \text{ord}(a) + r} = a^r$ and $r = 0$.

(ii) There is a k such that $a^k = 1$ for all $a \in G$.

1. If G is a finite Group, all elements must have finite Element Order.
2. Choose k as the product of the Element Order of all different elements in G.
3. Then, $a^k = 1$ for all $a \in G$

4.2.1 Lagrange's Theorem

Theorem 4.1 (Lagrange's Theorem)**.** *If G is a finite group and H is a Subgroup of G, then $|H|$ divides $|G|$. In particular if $a \in G$, then the order of a divides $|G|$.*

Remark. If $|G|$ is a Prime number, then the order of an element a is either 1 or $|G|$. In particular, if $|G|$ is a Prime number, then G must be Cyclic.

4.3 Rings

Defn 44 (Ring)**.** A *ring* (with unity) consists of a set R with two Binary Operations. A ring is denoted as $(R, +, *)$., where $+$ and $*$ are **not** addition and multiplication respectively, but placeholders for the two Binary Operations. A ring must also satisfy the following conditions:

(i) $(R, +)$ is an Abelian Group with an identity element denoted 0.

(ii) $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all $a, b, c \in R$ (Associativity).

$$\forall a, b, c \in R \, a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

(iii) There is a multiplicative identity, denoted 1, with the multiplicative identity not being equal to the identity element of the underlying Abelian Group ($1 \neq 0$), such that $1 \cdot a = a \cdot 1 = a$ for all $a \in R$ (Multiplicative Identity).

(iv) $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$ for all $a, b, c \in R$ (Distributive).

$$\forall a, b, c \in R \, a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$
$$\forall a, b, c \in R \, (b + c) \cdot a = (b \cdot a) + (c \cdot a)$$

A *commutative ring* is a ring where additionally

(v) $a \cdot b = b \cdot a$ for all $a, b \in R$ (Commutativity)

$$\forall a, b \in R \, a \cdot b = b \cdot a$$

Remark 44.1 (Ring Multiplicative Inverses). Note that we haven't said anything about multiplicative inverses yet.

Defn 45 (Invertible Element)**.** An element $a \in R$ is called an *invertible element* or a *unit* if there is an element $b \in R$ such that

$$a \cdot b = b \cdot a = 1 \tag{4.3}$$

Remark 45.1. The set of Invertible Elements in a Ring R forms a Group under multiplication. For example, the Group of Invertible Elements of the Ring $\mathbb{Z}_n$ is $\mathbb{Z}_n^*$.

Remark 45.2 (Multiplicative Inverse). The Multiplicative Inverse of an element $a \in R$ is denoted by a^{-1}, assuming it exists. The division expression a/b should then be interpreted as $a \cdot b^{-1}$.

4.3.1 Examples of Rings

- A commutative ring is $(\mathbb{Z}, +, \cdot)$, where $+$ and $\cdot$ are the susual operations of addition and multiplication.
- Finite Ring: $\mathbb{Z}_n$ with addition and multiplication modulo n
  - The additive inverse of $a \in R$ is denoted $-a$. So the subtraction expression $a - b$ should be interpreted as $a + (-b)$.
  - The multiplication of $a \cdot b$ is equivalently written ab.
  - Similarly $a^2 = aa = a \cdot a$.

4.4 Fields

Defn 46 (Field)**.** A *field* is a commutative Ring where all nonzero elements from the underlying set have Multiplicative Inverses.

Example 4.5: Prove Set is Not Field. Exercise 1, Problem 1.11

Prove that $\mathbb{Z}_4$ is not a Field?

We begin by checking that all elements from the underlying set, $\mathbb{Z}_4$, have Multiplicative Inverses, for all the **non-zero** elements.

$$\mathbb{Z}_4 = \Big\{[0],[1],[2],[3]\Big\}$$

First, we check 1.

$$1^{-1} = \gcd(1,4) = 1a + 4b$$

$$\begin{aligned} \gcd(1,4) \Rightarrow 4 &= q1 + r \\ &= 4 \cdot 1 + 0 \\ \gcd(1,4) &= 1 \end{aligned}$$

$$\begin{aligned} 1 &= 1(-3) + 4(1) \\ a &= -3 \\ b &= 1 \end{aligned}$$

Since $a = -3$, we need to find the additive inverse:

$$\begin{aligned} -3 \bmod 4 &= (-3+4) \bmod 4 \\ &= 1 \bmod 4 \\ &= 1 \end{aligned}$$

So, 1 has a Multiplicative Inverse.
Now we check 2.

$$2^{-1} = \gcd(2,4) = 2a + 4b$$

$$\begin{aligned} \gcd(2,4) \Rightarrow 4 &= q2 + r \\ &= 2 \cdot 2 + 0 \\ \gcd(2,4) &= 2 \end{aligned}$$

Since $\gcd(2,4) \neq 0$, 2 is **not** invertible, meaning it has no Multiplicative Inverses.
$\mathbb{Z}_4$ is NOT a Field because 2 is **not** invertible.

Defn 47 (Characteristic)**.** The *characteristic* of a field is the smallest integer $m > 0$ such that

$$\overbrace{1 + 1 + \cdots + 1}^{m} = 0 \tag{4.4}$$

If no such integer m exists, the characteristic is defined to be 0.

Defn 48 (Finite Field)**.** A Field is *finite* only if Theorem 4.2 is satisfied.

Theorem 4.2 (Finite Field)**.** *$\mathbb{Z}_n$ is a Field if and only if n is a Prime number. If n is a Prime, the Characteristic of $\mathbb{Z}_n$ is n.*

Defn 49 (Subfield)**.** A subset F of a Field E is called a *subfield* of E if F is itself a Field with respect to the operations of E.

Remark 49.1 (Extension Field). Likewise, we say E is an *extension field* of F.

Defn 50 (Isomorphism)**.** Two Fields are *isomorphic* if they are structurally the same, but elements hvae a different representation. For example, for a Prime, p, $\mathbb{Z}_p$ is a Field or Set Order p. So we associate the Finite Field $\mathbb{F}_{p^w}$ith $\mathbb{Z}_p$ and its representation.

4.4.1 Examples of Fields

- The rational Numbers: $\mathbb{Q}$
- The Real Numbers: $\mathbb{R}$
- The Complex Numbers: $\mathbb{C}$
- Finite Field:
  - $\mathbb{Z}_p$, where p is Prime

4.5 Polynomial Rings

Polynomial Rings are getting their own subsection separate from other Rings, because they are actually a more general case of what we have learned already.

Defn 51 (Polynomial)**.** A *polynomial* in the indeterminate x over the Ring R is an expression of the form

$$f(x) = a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0 \tag{4.5}$$

where each $a_i \in R$, $a_n \neq 0$, and $n \geq 0$.

Remark 51.1. **Remember that even integers can be polynomials!** This means that everything we have learned about Rings already is actually a special case of Polynomial Rings.

$$1 = 0x^m + 0x^{m-1} + \cdots + 0x + 1$$

Defn 52 (Degree)**.** We say that $f(x)$ has *degree* n, denoted

$$\deg f(x) = n \tag{4.6}$$

Remark 52.1. We also allow $f(x)$ to be the Polynomial with all coefficients being zero, in which case, the Degree is defined to be $-\infty$.

Defn 53 (Monic)**.** A Polynomial is said to be *monic* if the leading coefficient is equal to 1.

$$a_n = 1 \tag{4.7}$$

Defn 54 (Polynomial Ring)**.** Let R be a commutative Ring (i.e. Property **(v)** is fulfilled). Then the *polynomial ring*, denoted by $R[x]$ is the ring formed by the set of all Polynomials in the indeterminate x having coefficients from R. The operations are addition and multiplication of Polynomials, with the coefficient arithmetic performed in R.

Example 4.6: Polynomial Ring. Lecture 2

Find the Polynomial Ring formed when the underlying Ring is $\mathbb{Z}_2$?

$$\mathbb{Z}_2 \underset{R[x]}{\Longrightarrow} \mathbb{Z}_2[x] = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x, x^2+x+1, \ldots\}$$

If you consider the $F[x]$, where F denotes an arbitrary Field. $F[x]$ has many properties in common with integers.

Defn 55 (Irreducible)**.** A polynomial $f(x) \in F[x]$, of Degree $d \geq 1$ is *irreducible* if $f(x)$ cannot be written as a product of 2 polynomials, $g(x), h(x) \in F[x]$, where the $\deg g(x)$ and $\deg h(x)$ are both less than d.

Remark 55.1 (Sum to Zero). It seems that Irreducible Polynomials, $f(x)$, where $f(x) \in \mathbb{F}_{p^q}$ must sum to a non-zero value for all values of x from the underlying set of integers, $\mathbb{Z}_p$.

Remark 55.2 (Relation Between Irreducible Polynomials and Prime Numbers). Irreducible polynomials are the Polynomial Ring counterpart of Prime numbers.

Example 4.7: Irreducible Polynomial. Exercise 1, Problem 1.14a

Is $f(x) = x^4 + x + 1$ in $\mathbb{Z}_2$ an Irreducible Polynomial and/or a Primitive Element?

We start by checking the Sum to Zero conditions.

$$f(0) = 0^4 + 0 + 1 = 1$$
$$f(1) = 1^4 + 1 + 1 = 1$$

The Sum to Zero conditions for this Polynomial.
Because our Polynomial is of Degree 4, we can factor it 2 ways:

1.

$$\left(x^2 + y\right)\left(x^2 + z\right) = x^4 + yx^2 + zx^2 + yz$$
$$x^4 + yx^2 + zx^2 + yz = x^4 + x + 1$$

This cannot work, because there are no x^2 terms in the potential factorization.

2.

$$(x + y)\left(x^3 + z\right) = x^4 + yx^3 + zx + yz$$
$$x^4 + yx^3 + zx + yz = x^4 + x + 1$$

This also cannot work, because the coefficient's terms do not line up ($y \neq 0$ because $yz = 1$).

Thus, $f(x)$ is an Irreducible.
Now we need to check if $f(x)$ is a Primitive Element.

$$\alpha^0 = 1$$
$$\alpha^1 = \alpha$$
$$\alpha^2 = \alpha^2$$
$$\alpha^3 = \alpha^3$$
$$\alpha^4 = \alpha + 1$$
$$\alpha^5 = \alpha^2 + \alpha$$
$$\alpha^6 = \alpha^3 + \alpha^2$$
$$\alpha^7 = \alpha^4 + \alpha^3 = \alpha^3 + \alpha + 1$$
$$\alpha^8 = \alpha^4 + \alpha^2 + \alpha = \alpha^2 + \alpha + \alpha + 1 = \alpha^2 + 1$$
$$\alpha^9 = \alpha^3 + \alpha$$
$$\alpha^{10} = \alpha^4 + \alpha^2 = \alpha^2 + \alpha + 1$$
$$\alpha^{11} = \alpha^3 + \alpha^2 + \alpha$$
$$\alpha^{12} = \alpha^4 + \alpha^3 + \alpha^2 = \alpha^3 + \alpha^2 + \alpha + 1 = \alpha^3 + \alpha^2 + \alpha + 1$$
$$\alpha^{13} = \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha + 1 + \alpha + 1 = \alpha^3 + \alpha^2 + 1$$
$$\alpha^{14} = \alpha^4 + \alpha^3 + \alpha = \alpha + 1 + \alpha^3 + \alpha = \alpha^3 + 1$$
$$\alpha^{15} = \alpha^4 + \alpha = \alpha + 1 + \alpha = 1$$

Because $\alpha^4 = \alpha + 1$ can generate all Polynomials in $\mathbb{F}_{2^4}$, $f(x)$ is also Primitive Element.
Thus, the Polynomial, $f(x)$ is both Irreducible and a Primitive Element.

Example 4.8: Reducible Polynomial. Exercise 1, Problem 1.14c

Is $f(x) = x^4 + x + 1$ in $\mathbb{Z}_2$ an Irreducible Polynomial and/or a Primitive Element?

Just by inspection, Sum to Zero holds up.
Now we have to check if the Polynomial is reducible. I will arbitrarily choose to start with a factorization of

$$\left(x^2 + c\right)\left(x^2 + c\right) = x^4 + 2cx^2 + c^2$$
$$x^4 + 2cx^2 + c^2 = x^4 + x^2 + 1$$
$$x^4 + c^2 = x^4 + x^2 + 1$$
$$c^2 = x^2 + 0x + 1$$

Because of the modulo addition we are doing here, $2 \bmod 2 = 0$, eases the factorization.

$$c^2 = x^2 + 2x + 1$$
$$c^2 = (x + 1)(x + 1)$$
$$c = x + 1$$

So, $f(x)$ is reducible.
We can check if $f(x)$ is a Irreducible, though we don't have to.

$$\alpha^4 = \alpha^2 + 1$$

$$\begin{aligned}
\alpha^0 &= 1 \\
\alpha^1 &= \alpha \\
\alpha^2 &= \alpha^2 \\
\alpha^3 &= \alpha^3 \\
\alpha^4 &= \alpha^2 + 1 \\
\alpha^5 &= \alpha^3 + \alpha \\
\alpha^6 &= \alpha^4 + \alpha^2 + \alpha^2 + 1 + \alpha^2 = 1
\end{aligned}$$

Thus, $f(x)$ is reducible and NOT a Primitive Element.

Example 4.9: Reducible Polynomial By Sum. Exercise 1, Problem 1.14d

Is $f(x) = x^4 + x + 1$ in $\mathbb{Z}_3$ an Irreducible Polynomial and/or a Primitive Element?

We can start by looking at Remark 55.1.

$$\begin{aligned}
f(x = 0) &= 0^4 + 0 + 1 = 1 \bmod 3 = 1 \\
f(x = 1) &= 1^4 + 1 + 1 = 3 \bmod 3 = 0 \\
f(x = 2) &= 2^4 + 2 + 1 = 19 \bmod 3 = 1
\end{aligned}$$

Since there is a case where $f(x) = 0$ in the set $\mathbb{Z}_3$, $f(x)$ is reducible. We can make the table of α:

$$\alpha^4 = \alpha + 1$$

$$\begin{aligned}
\alpha^0 &= 1 & \alpha^8 &= \alpha^4 + \alpha^2 + \alpha = \alpha + 1 + \alpha^2 + \alpha = \alpha^2 + 1 \\
\alpha^1 &= \alpha & \alpha^9 &= \alpha^3 + \alpha \\
\alpha^2 &= \alpha^2 & \alpha^{10} &= \alpha^4 + \alpha^2 = \alpha^2 + \alpha + 1 \\
\alpha^3 &= \alpha^3 & \alpha^{11} &= \alpha^3 + \alpha^2 + \alpha \\
\alpha^4 &= \alpha + 1 & \alpha^{12} &= \alpha^4 + \alpha^3 + \alpha^2 = \alpha^3 + \alpha^2 + \alpha + 1 \\
\alpha^5 &= \alpha^2 + \alpha & \alpha^{13} &= \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha + \alpha + \alpha^3 + \alpha^2 + 1 = \alpha^3 + \alpha^2 + 1 \\
\alpha^6 &= \alpha^3 + \alpha^2 & \alpha^{14} &= \alpha^4 + \alpha^3 + \alpha = \alpha + 1 + \alpha^3 + \alpha = \alpha^3 + 1 \\
\alpha^7 &= \alpha^4 + \alpha^3 = \alpha^3 + \alpha + 1 & \alpha^{15} &= \alpha^4 + \alpha = \alpha + 1 + \alpha = 1
\end{aligned}$$

Thus, $f(x)$ is reducible and MAY be a Primitive Element.

4.5.1 Long Division of Polynomials

Similarly as for integers (Section 2.1), we have a division algorithm for polynomials.

Defn 56 (Polynomial Long Division)**.** If $a(x), b(x) \in F[x]$, with $b(x) \neq 0$, then there are polynomials $q(x), r(x) \in F[x]$ such that

$$a(x) = q(x)b(x) + r(x) \tag{4.8}$$

where

- $\deg r(x) < \deg b(x)$.

- $q(x)$ and $r(x)$ are unique.
- $q(x)$ is referred to as the Polynomial Quotient
- $r(x)$ is referred to as the Polynomial Remainder

Defn 57 (Polynomial Quotient)**.** The *Polynomial quotient*, $q(x)$, of $a(x)$ divided by $b(x)$ ($a(x) \div b(x)$, $a(x)/b(x)$) is denoted $a(x) \operatorname{div} b(x)$.

Defn 58 (Polynomial Remainder)**.** The *Polynomial remainder*, $r(x)$, of $a(x)$ divided by $b(x)$ ($a(x) \div b(x)$, $a(x)/b(x)$) is denoted $a(x) \bmod b(x)$.

Example 4.10: Long Division of Polynomials. Lecture 3

Calculate $\left(x^3 + 11x^2 + x + 7\right) \bmod \left(x^2 + x + 3\right)$ while working in the Field $Z_{13}[x]$?

First thing to note is that $Z_{13}[x]$ is a Finite Field because 13 is a Prime. Now, we perform long division:

$$\begin{array}{r|l} & x + 10 \\ \hline x^2 + x + 3 & x^3 + 11x^2 + x + 7 \\ & -x^3 - x^2 - 3x \\ \hline & 10x^2 - 2x + 7 \\ & -10x^2 - 10x - 30 \\ \hline & -12x - 23 \end{array}$$

Since our Polynomial Remainder is $(-12x - 23) \bmod 13$, we perform some reduction. Thus, the Polynomial Remainder is:

$$-12x \bmod 13 = 1x \bmod 13$$
$$-23 \bmod 13 = -10 \bmod 13 = 3 \bmod 13$$

So, the reduced Polynomial Quotient and Polynomial Remainder are:

$$q(x) = x + 10$$
$$r(x) = x + 3$$

4.5.2 Properties of Polynomial Rings

Defn 59 (Divide)**.** If $g(x), h(x) \in F[x]$, then $h(x)$ is said to *divide* $g(x)$, written

$$h(x) \,|\, g(x) \text{if } g(x) \bmod h(x) = 0 \tag{4.9}$$

Defn 60 (Congruent)**.** Let $g(x), h(x) \in F[x]$. Then, $g(x)$ is said to be *congruent* to $h(x) \bmod f(x)$ if $f(x) \,|\, \big(g(x) - h(x)\big)$. We denote this

$$g(x) \equiv h(x) \pmod{f(x)} \tag{4.10}$$

(i) $g(x) \equiv h(x) \pmod{f(x)}$ if and only if $g(x)$ and $h(x)$ leave the same remainder when divided by $f(x)$.
(ii) $g(x) \equiv g(x) \pmod{f(x)}$.
(iii) If $g(x) \equiv h(x) \pmod{f(x)}$, the $h(x) \equiv g(x) \pmod{f(x)}$.
(iv) If $g(x) \equiv h(x) \pmod{f(x)}$ and $h(x) \equiv s(x) \pmod{f(x)}$, then $g(x) \equiv s(x) \pmod{f(x)}$.
(v) If $g(x) \equiv g_1(x) \pmod{f(x)}$ and $h(x) \equiv h_1(x) \pmod{f(x)}$, then:

- $g(x) + h(x) \equiv g_1(x) + h_1(x) \pmod{f(x)}$
- $g(x)h(x) \equiv g_1(x)h_1(x) \pmod{f(x)}$

We can divide $F[x]$ into sets called Equivalence Class, where each Equivalence Class contains all Polynomials that leaves a certain Polynomial Remainder when divided by $f(x)$.

Defn 61 (Equivalence Class)**.** By $F[x]/f(x)$, we denote the set of *equivalence class*es of Polynomials in $F[x]$ of degree less than $\deg f(x0$. The addition and multiplication operations are performed $\bmod f(x)$.

Defn 62 (Representative)**.** Since the Polynomial Remainder, $r(x)$ itself is a Polynomial in the Equivalence Class we use it as a *representative* of the Equivalence Class.

Defn 63 (Commutative Ring)**.** A commutative Ring for Polynomials is defined as

$$F[x]/f(x) \tag{4.11}$$

Remark 63.1. Note that this is the inverse condition of when a Polynomial Ring is a Field.

Defn 64 (Field)**.** If $f(x) \in F[x]$ is Irreducible, then $F[x]/f(x)$ is a Field.

Remark 64.1. Note that this is the inverse condition of when a Polynomial Ring is a Commutative Ring

Defn 65 (Finite Field)**.** A *finite Field* is a Field which contains a finite number of elements, i.e. the Set Order of the Field is not ∞.

(i) If F is a Finite Field, then the Set Order of F is p^m for some Prime p and integer $m \geq 1$.
(ii) For every Prime power order p^m, there is a unique (up to Isomorphism) Finite Field of Set Order p^m. This Field is denoted by $\mathbb{F}_{p^m}$ or $GF\left(p^m\right)$.
(iii) If $\mathbb{F}_q$ is a Finite Field of Set Order $q = p^m$, i.e. $\mathbb{F}_{p^m}$, the the *Characteristic* of $\mathbb{F}_q$ is p. Furthermore, $F[q]$ contains a copy of $\mathbb{Z}_p$ as a *Subfield*.
(iv) Let $\mathbb{F}_q$ be a Finite Field of Set Order $q = p^m$, i.e. $\mathbb{F}_{p^m}$. Then every Subfield of $\mathbb{F}_q$ has Set Order p^n for some positive integer n where $n \,|\, m$. Conversely, if $n \,|\, m$, then there is exactly one Subfield of $\mathbb{F}_q$ of Set Order p^n.
(v) An element $a \in \mathbb{F}_q$ is in the Subfield $\mathbb{F}_{p^n}$ if and only if $a^{p^n-1} = 1$.

Defn 66 (Multiplicative Group)**.** The non-zero elements of $\mathbb{F}_{q^a}$ll have inverses and thus, they form a Group under multiplication. This Group is called the *multiplicative group* of $\mathbb{F}_{q^a}$nd denoted by $\mathbb{F}_q^*$. It can be shown that $\mathbb{F}_q^*$ is a Cyclic Group (of Set Order $q-1$). Especially, this means that $a^{q-1} = 1$ for all $a \in \mathbb{F}_q$.

Defn 67 (Primitive Element)**.** A Generator of the Cyclic Group $\mathbb{F}_q^*$ is called a *primitive element*.

4.5.3 Extension of Greatest Common Divisor, Euclidean Algorithm, and Extended Euclidean Algorithm

Defn 68 (Greatest Common Divisor)**.** Let $g(x), h(x) \in \mathbb{Z}_p[x]$, where not both are zero. Then the *greatest common divisor*, *GCD*, of $g(x)$ and $h(x)$, denoted $\gcd\big(g(x), h(x)\big)$, is the Monic of greatest Degree in $Z_p[x]$ which Divides both $g(x)$ and $h(x)$.

Remark 68.1. By definition $\gcd(0,0) = 0$.

Theorem 4.3 (Unique Factorization of Polynomials)**.** *Every non-zero polynomial $f(x) \in \mathbb{Z}_p[x]$ has a factorization*

$$f(x) = af_1(x)^{e_1} f_2(x)^{e_2} \cdots f_k(x)^{e_k} \tag{4.12}$$

where each $f_i(x)$ is a distinct Monic Irreducible Polynomial in $Z_p[x]$, the e_i are positive integers, and $a \in \mathbb{Z}_p$, where p is for Primes. The factorization is unique up to the rearrangement of factors.

Defn 69 (Polynomial Euclidean Algorithm)**.** Takes 2 non-negative Polynomials $a(x), b(x) \in \mathbb{F}_q[x]$. Returns the $\gcd\big(a(x), b(x)\big)$

1. Set $r_0(x) \leftarrow a(x)$, $r_1(x) \leftarrow b(x)$, $i \leftarrow 1$.
2. While $r_i(x) \neq 0$ do:
   (a) Set $r_{i+1}(x) \leftarrow r_{i-1}(x) \bmod r_i(x)$, $i \leftarrow i+1$
3. Return the value $r_i(x)$.

This is demonstrated in Example 4.11

Example 4.11: Polynomial Euclidean Algorithm. Lecture 3

Given $f(x) = x^3 + x$, where $f(x) \in Z_2[x]$, find the Greatest Common Divisor of $f(x)$ given that $f(x)g(x) = 1 \bmod \left(x^4 + x + 1\right)$. Assume $f(x)g(x)$ is Irreducible.

Considering that $Z_2[x]/f(x)$ where $f(x) \in Z_2[x]$ and $f(x)$ is Irreducible, this is a Polynomial Ring. Since the order of the Polynomial Ring is a Prime power, this is a Finite Field. Because this is a Finite Field, $\mathbb{F}_{2^4}[] = 2^4 = 16$ elements. Now, we calculate the Polynomial Euclidean Algorithm.

$$\gcd\Big(\left(x^4 + x + 1\right), \left(x^3 + x\right)\Big)$$
$$x^4 + x + 1 = q(x)\left(x^3 + x\right) + r(x)$$

We can perform Polynomial Long Division to find $q(x)$ and $r(x)$ for this iteration.

$$\begin{array}{r|l} x^3+x & \begin{array}{r} \quad\quad\quad\quad\quad x \\ \hline x^4 \quad\quad\quad +x+1 \\ -x^4-x^2 \\ \hline -x^2+x+1 \end{array} \end{array}$$

So,

$$\begin{aligned} q(x) &= x \\ r(x) &= \left(-x^2+x+1\right) \bmod 2 \end{aligned}$$

We need to correct the coefficients in $r(x)$ with modulo 2 (where the coefficients come from in the first place).

$$\begin{aligned} r(x) &= \left(-x^2+x+1\right) \bmod 2 \\ &= x^2+x+1 \\ x^4+x+1 &= x\left(x^3+x\right)+\left(x^2+x+1\right) \end{aligned}$$

Now, the next iteration of Polynomial Euclidean Algorithm.

$$x^3+x = q(x)\left(x^2+x+1\right)+r(x)$$

$$\begin{array}{r|l} x^2+x+1 & \begin{array}{r} \quad\quad\quad\quad\quad x-1 \\ \hline x^3 \quad\quad\quad +x \\ -x^3-x^2-x \\ \hline -x^2 \\ x^2+x+1 \\ \hline x+1 \end{array} \end{array}$$

$$\begin{aligned} q(x) &= x-1 \\ r(x) &= x+1 \\ x^3+x &= (x-1)\left(x^2+x+1\right)+(x+1) \end{aligned}$$

Another iteration:

$$x^2+x+1 = q(x)(x+1)+r(x)$$

$$\begin{array}{r|l} x+1 & \begin{array}{r} \quad\quad\quad x \\ \hline x^2+x+1 \\ -x^2-x \\ \hline 1 \end{array} \end{array}$$

$$\begin{aligned} q(x) &= x \\ r(x) &= 1 \\ x^2+x+1 &= x(x+1)+1 \end{aligned}$$

Another iteration (the last):

$$x+1 = q(x)\cdot 1+r(x)$$

$$\begin{aligned} q(x) &= x+1 \\ r(x) &= 0 \\ x+1 &= (x+1)\cdot 1+0 \end{aligned}$$

Thus, the $\gcd\left(x^4+x+1, x^3+x\right) = 1$.

Defn 70 (Extended Euclidean Algorithm)**.** let $a(x)$ and $b(x)$ be two non-negative polynomials in $F_q[x]$. Then, there exists

polynomials $s(x), t(x)$ such that $\gcd\big(a(x), b(x)\big)$ can be written as

$$\gcd\big(a(x), b(x)\big) = a(x)s(x) + b(x)t(x) \tag{4.13}$$

This is demonstrated in Example 4.12

Example 4.12: Polynomial Extended Euclidean Algorithm. Lecture 3

Given $f(x) = x^3 + x$, where $f(x) \in Z_2[x]$, find $g(x)$ of $f(x)$ given that $f(x)g(x) = 1 \bmod \big(x^4 + x + 1\big)$. Assume $f(x)g(x)$ is Irreducible.

Since we know that the $\gcd\left(x^4 + x + 1, x^3 + x\right) = 1$, from Example 4.11, we can solve Equation (4.13).

$$\begin{aligned}
1 &= \big(x^2 + x + 1\big) - x(x + 1) \\
&= \big(x^2 + x + 1\big) - x\Big(\big(x^3 + x\big) - (x - 1)\left(x^2 + x + 1\right)\Big) = \big(x^2 + x + 1\big) - x\left(x^3 + x\right) - x(x - 1)\left(x^2 + x + 1\right) \\
&= \big(1 - x(x - 1)\big)\left(x^2 + x + 1\right) - x\left(x^3 + x\right) \\
&= \big(1 - x(x - 1)\big)\Big(\big(x^4 + x + 1\big) - x\left(x^3 + x\right)\Big) - x\left(x^3 + x\right) \\
&= \big(1 - x(x - 1)\big)\left(x^4 + x + 1\right) - x\big(1 - x(x - 1)\big)\left(x^3 + x\right) - x\left(x^3 + x\right) \\
&= \big[1 - x(x - 1)\big]\left(x^4 + x + 1\right) - \Big[x\big(1 - x(x - 1)\big) - x\Big]\left(x^3 + x\right)
\end{aligned}$$

$$\begin{aligned}
s(x) &= 1 - x(x - 1) = -x^2 - x - 1 \\
t(x) &= x\big(1 - x(x - 1)\big) - x = -x^3 + x^2
\end{aligned}$$

Now, we reduce the coefficients with respect to modulo 2.

$$\begin{aligned}
s(x) &= x^2 + x + 1 \\
t(x) &= x^3 + x^2
\end{aligned}$$

So, we end up with:

$$\begin{aligned}
1 \bmod \big(x^4 + x + 1\big) &= \big(x^2 + x + 1\big)\left(x^4 + x + 1\right) + \left(x^3 + x^2\right)\left(x^3 + x\right) \\
&= \Big(\big(x^2 + x + 1\big)\left(x^4 + x + 1\right) + \left(x^3 + x^2\right)\left(x^3 + x\right)\Big) \bmod \big(x^4 + x + 1\big) \\
&= 0 + \left(x^3 + x^2\right)\left(x^3 + x\right) \\
f(x) &= \left(x^3 + x\right) \\
&= \left(x^3 + x^2\right) f(x)
\end{aligned}$$

Thus, $g(x) = x^3 + x^2$.

Defn 71 (Polynomial Basis Representation)**.** The most common representation of element of a Finite Field $\mathbb{F}_{q^{\prime}}$ where $q = p^m$, p is a Prime, and is a *polynomial basis representation.*

Theorem 4.4. *Let $f(x) \in \mathbb{Z}_p[x]$ be an Irreducible of Degree m. Then $\mathbb{Z}_p[x]/f(x)$ is a Finite Field of Set Order p^m. The elements are all Polynomials of Degree less than m. Addition and multiplication of elements is performed modulo $f(x)$.*

Lemma 4.4.1. *For each $m \geq 1$, there exists a Monic that is also an Irreducible of Degree m over $\mathbb{Z}_p$.*

5 Classical Cryptography

Defn 72 (Cryptosystem)**.** A *cryptosystem* five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of possible *Plaintext*, producing a message, $\mathbf{m}$.
2. $\mathcal{C}$ is a finite set of possible *Ciphertext*

3. $\mathcal{K}$, the *Keyspace*, is a finite set of all possible keys
4. For each $K \in \mathcal{K}$, there is an *Encryption Rule* $e_K \in \mathcal{E}$ and a corresponding *Decryption Rule* $d_K \in \mathcal{D}$. Each $e_K : \mathcal{P} \to \mathcal{C}$ and $d_K : \mathcal{C} \to \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every Plaintext element $x \in \mathcal{P}$.

This can be illustrated with the Shannon Model for Symmetric Encryption.

Figure 5.1: Shannon Model for Symmetric Encryption

Defn 73 (Symmetric Encryption)**.** A *symmetric encryption* system has Alice and Bob both share the **same** key. We make the assumption that Alice can share the key with Bob in a way such that Eve cannot use the key. Figure 5.1 gives a visualization of this type of system.

A convention is to give the various parties in play names:

- Alice
- Bob
- Caesar
- Eve (The enemy)

Defn 74 (Plaintext)**.** *Plaintext* is the information that Alice wants to send to Bob, and is denoted $\mathcal{P}$ This information can be in any arbitrary format, we do not care, however the elements in the message are drawn from the Alphabet. However, Alice does not want Eve to be able to understand what she sends to Bob.

The message to be sent is:

$$\mathbf{m} \in \mathcal{P} = (m_1, m_2, \ldots, m_n) \, \forall m \in \mathcal{A} \tag{5.1}$$

Defn 75 (Alphabet)**.** Let $\mathcal{A}$ be a finite set, which is called the *alphabet*. We often use the English letters as the alphabet, we number them $\mathtt{a} = 0, \mathtt{b} = 1, \ldots, \mathtt{z} = 25$. This given $\mathcal{A} = \mathbb{Z}_{26}$ and $|\mathcal{A}| = 26$.

a	0.0804	j	0.0016	s	0.0654
b	0.0154	k	0.0067	t	0.0925
c	0.0306	l	0.0414	u	0.0271
d	0.0399	m	0.0253	v	0.0099
e	**0.1251**	n	0.0709	w	0.0192
f	0.0230	o	0.0760	x	0.0019
g	0.0196	p	0.0200	y	0.0173
h	0.0549	q	0.0011	z	0.0009
i	0.0726	r	0.0612		

Table 5.1: Frequency of Letters in the English Alphabet

Defn 76 (Ciphertext)**.** A *ciphertext* is a piece of Plaintext information that has been run through an element of Encryption Rule set.

A message that has been transformed with an Encryption Rule is a cipher message.

$$\mathbf{c} = e_K(\mathbf{m}) \tag{5.2}$$

Defn 77 (Keyspace)**.** *Keyspace*, denoted $\mathcal{K}$. Each *key* in the keyspace describes a certain function

$$e_K : \mathcal{P}^n \to \mathcal{C}^{n'} \tag{5.3}$$

Defn 78 (Encryption Rule)**.** The *encryption rule* is an element e_K from the set of all encryption rules, $\mathcal{E}$.

$$e_K : \mathcal{P} \to \mathcal{C} \text{ where } e_K \in \mathcal{E} \tag{5.4}$$

Namely, the encryption rule element is used to map the Plaintext pieces of information that Alice wants to send to a corresponding Ciphertext that she can send to Bob.

Remark 78.1 (Invertible). Each Encryption Rule **must** be *invertible*, thus allowing decryption of the ciphertext.

Defn 79 (Decryption Rule)**.** The *decryption rule* is an element d_K from the set of all decryption rules, $\mathcal{D}$.

$$d_K : \mathcal{C} \to \mathcal{P} \text{ where } d_K \in \mathcal{D} \tag{5.5}$$

Namely, the decryption rule element is used to map the Ciphertext pieces of information that Alice sent to a corresponding Plaintext that Bob can use.

Remark 79.1.

$$pd_K(e_K(\mathbf{m})) = \mathbf{m} \; \forall \, \mathbf{m} \in \mathcal{P} \tag{5.6}$$

5.1 Types of Attacks

There are 2 main types of attacks:

1. Ciphertext-only: A ciphertext-only attack is an attack in which the enemy has access to the ciphertext $\mathcal{C}$ and tries to recover the plaintext $\mathcal{P}$ or the key K.
2. Known Plaintext: A known-plaintext attack is an attack in which the enemy has access to not only the ciphertext, $\mathcal{C}$, but may also have some or all of the $\mathcal{P}$, and is trying to recover the key K.

5.2 The Caesar Cipher

The Caesar Cipher is a Monoalphabetic Cipher in which each letter is shifted 3 steps.

This can be generalized to shift not just 3, but K positions, where $K \in \{0, 1, \ldots, 25\}$ is the secret key. In this case, the Keyspace, $\mathcal{K}$, is the set of shifts in position possible.

$$K \in \mathcal{K},\ \mathcal{K} = \{0, 1, \ldots, 25\}$$

To encrypt a message, each individual character is encrypted according to Equation (5.7).

$$e_K(m) = m + k \bmod 26 \tag{5.7}$$

The Decryption Rule for The Caesar Cipher is used to decrypt a message encrypted according to Equation (5.7).

$$d_K(c) = c - k \tag{5.8}$$

5.2.1 Cryptanalysis of The Caesar Cipher

The Caesar Cipher can be broken with an *exhaustive key search.*

Example 5.1: Cryptanalysis of The Caesar Cipher. Lecture 4

Given the input string `wklvlvdphvvdjhwrbrx`, decrypt and find the message. The message was encrypted using The Caesar Cipher

K	Plaintext Equivalent
0	`wklvlvdphvvdjhwrbrx`
1	`vjkukucoguucigvqaqw`
2	`uijtjtbnfttbhfupzpv`
3	`thisisamessagetoyou`
4	`sghrhrzldrrzfdsnxnt`
5	`rfgqgqykcqqyecrmwms`
⋮	⋮
25	`xlmwmweqiwwekixscsy`

So, the key used in this Caesar Cipher was $K = 3$, and the message was `thisisamessagetoyou`.

5.3 The Simple Substitution Cipher

The Simple Substitution Cipher operates on the individual characters in $\mathcal{P}$. However, the key, K is now an arbitrary permutation of the characters.

The set of all permutations on $\mathbb{Z}_{26}$ is written as

$$\mathcal{E} = \{\pi_1, \pi_2, \ldots, \pi_{|\mathcal{K}|}\}$$

The keyspace, $\mathcal{K}$ for this cipher is:

$$\mathcal{K} = \{0, 1, 2, \ldots, 26! - 1\}$$
$$|\mathcal{K}| = 26!$$

The Encryption Rule for this cipher is:

$$e_K(m) = \pi_K(m) \tag{5.9}$$

The Decryption Rule for this cipher's output is:

$$d_K(c) = \pi_k^{-1}(c) \tag{5.10}$$

5.3.1 Cryptanalysis of The Simple Substitution Cipher

We could attemp to perform an exhaustive key search, but because the keyspace is all integers from 0 to $26! - 1$, which is incredibly large, it makes more sense to exploit the statistical nature of the plaintext source.

We could try matching the most common letters present in the Ciphertext against the most common letter in the Plaintext Alphabet, using Table 5.1. However, these 1-grams are only good for the most common letters in the English Alphabet.

If we take use n-grams we will have better luck, because there is a dependence between consecutive letters.

5.4 Polyalphabetic Ciphers

Polyalphabetic ciphers operate differently on different portions of the Plaintext.

The simplist Polyalphabetic Cipher is a number of Monoalphabetic Ciphers with different keys are used sequentially, the cyclically repeated. The Vigenére Cipher is one example of this kind of cipher.

5.4.1 The Vigenére Cipher

Cyclically uses t Caesar ciphers, where t is the period of the cipher.

The Encryption Rule maps the Plaintext message $\mathbf{m} = m_1, m_2, \ldots$ to the Ciphertext $\mathbf{c} = c_1, c_2, \ldots$ through

$$\mathbf{c} = e_{\mathbf{K}}(m_1, m_2, \ldots, m_t), e_{\mathbf{K}}(m_{t+1}, m_{t+2}, \ldots, m_{t+t}), \ldots \tag{5.11}$$

where

$$e_{\mathbf{K}}(m_1, m_2, \ldots, m_t) = (m_1 + k_1, m_2 + k_2, \ldots, m_t + k_t)$$

and $\mathbf{K}$ consists of t characters $\mathbf{K} = (k_1, k_2, \ldots, k_t)$.

Remark. The key, $\mathbf{K}$ is often chosen to be a word, called a keyword.

Example 5.2: Encryption Using The Vigenere Cipher. Lecture 4

The Plaintext `youmustvisitmetonight` is to be encrypted using a Vigenére Cipher, with a period 4, where $\mathbf{K} =$ `lucy`.

y	o	u	m	u	s	t	v	i	s	i	t	m	e	t	o	⋯
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	⋯
l	u	c	y	l	u	c	y	l	u	c	y	l	u	c	y	⋯
j	i	w	k	f	m	v	t	t	m	k	r	x	y	v	m	⋯

The resulting Ciphertext is `jiwkfmvttmkrxyvm`.

5.4.1.1 Cryptanalysis of The Vigenére Cipher

There are 2 steps to breaking The Vigenére Cipher:

1. Determine the period of the cipher, t. Generally, Kasiski's Method is used.
2. Reconstruct the t different The Simple Substitution Cipher Alphabets used.

Defn 80 (Kasiski's Method)**.** *Kasiski's Method* relies on the observation that repeated portions of Plaintext input will yield a repeating Ciphertext output, with some cyclic distance, t between each occurrence. This distance will be a multiple of the keyword length.

To solve this, you calculate the Greatest Common Divisor of these distances.

Now we move onto the second part of breaking The Vigenére Cipher. There are 2 main ways to do this.

1. Split the Ciphertext characters into t different multisets, where each character should have been shifted according to a Simple Substitution Cipher. The key of each Monoalphabetic Cipher can be determined with the statistics in Table 5.1 and some trial-and-error.
2. Measure of Roughness and Index of Coincidence

Defn 81 (Measure of Roughness)**.**

$$\begin{aligned} \mathrm{MR} &= \sum_{i=a\Rightarrow 1}^{z\Rightarrow 26} \left(p_i - \frac{1}{26}\right)^2 \\ &= \sum_{i=a\Rightarrow 1}^{z\Rightarrow 26} (p_i)^2 - \frac{1}{26} \end{aligned} \tag{5.12}$$

Defn 82 (Index of Coincidence)**.** The *index of coincidence* uses values found in the Measure of Roughness.

$$\mathrm{IC} = \frac{\sum_{i=a\Rightarrow 1}^{z\Rightarrow 26} f_i\,(f_1 - 1)}{n(n-1)} \tag{5.13}$$

However, it does *not* return the period of The Vigenére Cipher.

5.4.2 The Vernam Cipher (One-Time Pad)

This is a special case of the The Vigenére Cipher, where the period of the key, **K** is chosen to be the same length as the output ciphertext, **C**. More concretely, for a ciphertext, **C** of length n, pick a Vigenére Cipher key with a period t. Where,

$$n = t$$

5.4.2.1 Cryptanalysis of The Vernam Cipher (One-Time Pad) The Vernam Cipher (One-Time Pad) is an example of Perfect Secrecy (unbreakable) if the the elements of the key **K** are chosen uniformly at random among all possible $\mathbb{Z}_{26}^n$ possible values.

5.4.3 Transposition Cipher (Permutation Cipher)

This cipher reorders all the letters in a block of length t. The Encryption Rule's key, K that permutes all the characters in a word block. Equation (5.14) is an example of how the Encryption Rule is constructed.

$$\begin{aligned} \pi_e(2,5,1,3,4) \Longrightarrow \pi(1) &= 2 \\ \pi(2) &= 5 \\ \pi(3) &= 1 \\ \pi(4) &= 3 \\ \pi(5) &= 4 \end{aligned} \tag{5.14}$$

The Decryption Rule for Equation (5.14) is given in Equation (5.15).

$$\begin{aligned} \pi_d(4,3,1,5,2) \Longrightarrow \pi(1) &= 4 \\ \pi(2) &= 3 \\ \pi(3) &= 1 \\ \pi(4) &= 5 \\ \pi(5) &= 2 \end{aligned} \tag{5.15}$$

Remark 82.1. The number on the **right-hand side** is assigned to the position present on the **left-hand side**.

Example 5.3: Transposition Cipher. Lecture 5

Given the plaintext message **M** = findi, and the Encryption Rule shown in Equation (5.14), what is the resulting ciphertext message?

$$\begin{aligned} \pi(1) &= 2 \\ \pi(2) &= 5 \\ \pi(3) &= 1 \\ \pi(4) &= 3 \\ \pi(5) &= 4 \end{aligned}$$

So, each character in the input "findi" is moved around to result in "iifnd".

5.4.4 The Hill Cipher

The Hill Cipher acts on t-grams from $\mathbb{Z}_{26}$ through a key that is an Invertible $t \times t$ matrix.

6 Information Theory

Information theory is heavily influenced by probability and its theories. For **much** greater detail on some of the concepts presented here, refer to the Math 374 - Probability and Statistics document.

Defn 83 (Random Experiment)**.** A *random experiment* is an experiment whose outcome varies in an unpredictable fashion when performed under the same conditions.

6.1 Sample Space

Defn 84 (Sample Space)**.** The *sample space* is the set of **all** possible outcomes, the Elementary Events, denoted

$$\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\} \tag{6.1}$$

Defn 85 (Elementary Event)**.** The possible outcomes in a Sample Space are called *elementary event*s. In Equation (6.1), they are denoted

$$\omega_i \tag{6.2}$$

Defn 86 (Event)**.** An *event* is any subset of Ω.

$$E \subset \Omega \tag{6.3}$$

Defn 87 (Probability)**.** The *probability* of an Event E is given by

$$\mathrm{P}(E) = \sum_{\omega \in E} \mathrm{P}(\omega) \tag{6.4}$$

Defn 88 (Random Variable)**.** A *random variable* X is a function that assigns a real number $X(\zeta)$ to each outcome ζ in the Sample Space of the Random Experiment.

6.2 Discrete Random Variables

Defn 89 (Discrete Random Variable)**.** A *discrete random variable* is a Random Variable that assumes values from a finite set, $\mathcal{X}$ It is a mapping

$$X(\omega) : \Omega \mapsto \mathcal{X} \tag{6.5}$$

Defn 90 (Probability Distribution)**.** A Discrete Random Variable has a *probability distribution* $P(X = x)$

$$\mathrm{P}(X = x) = \sum_{\omega : X(\omega) = x} \mathrm{P}(x) \tag{6.6}$$

If $E \subset \mathcal{X}$, then $X \in E$ is an Event and

$$\mathrm{P}(X \in E) = \sum_{x \in E} \mathrm{P}(x) \tag{6.7}$$

(i) A pair of Random Variables defined on the same Sample Space can be considered as a single Random Variable, say $Z = (X, Y)$.
(ii) Z takes values in $\mathcal{X} \times \mathcal{Y}$, with $Z(\omega) = \big(X(\omega), Y(\omega)\big)$
(iii) $\mathrm{P}(Z) = \mathrm{P}(X, Y)$
(iv) Consider the joint Event G as both Events E and F occurring. Then G corresopnds to the Event $G = E \cap F$.
(v) The probability of the joint event $\mathrm{P}(G) = \mathrm{P}(E, F) = \mathrm{P}(E \cap F)$

Defn 91 (Expected Value/Mean of Single Discrete Random Variable)**.** The *expected value* or *mean* of a single discrete random variable X is defined by

$$m_X = \mathbb{E}[X] = \sum_{x \in S_X} x \cdot p_X(x) \tag{6.8}$$

Remark 91.1. If X is countably infinite, you will have an infinite series that exists only if

$$\sum_{s \in S_X} |x| \cdot p_X(x) \tag{6.9}$$

is absolutely convergent.

6.2.1 Independent Discrete Random Variables

Defn 92 (Independent)**.** 2 Events, E and F, are said to be *independent* if

$$\mathrm{P}(E, F) = \mathrm{P}(E)\,\mathrm{P}(F) \tag{6.10}$$

2 Random Variables, X and Y, are said to be *independent Random Variables* if

$$\mathrm{P}(X = x, Y = y) = \mathrm{P}(X = x)\,\mathrm{P}(Y = y), \forall x \in \mathcal{X}, y \in \mathcal{Y} \tag{6.11}$$

6.2.2 Conditional Probability of Discrete Random Variables

Defn 93 (Conditional Probability)**.** The *conditional probability* $\mathrm{P}(E \mid F)$ (Read as "the probability of E given F") is defined as

$$\mathrm{P}(E \mid F) = \frac{\mathrm{P}(E, F)}{\mathrm{P}(F)} \text{ where } \mathrm{P}(F) \neq 0 \tag{6.12}$$

For Random Variables, a similar definition is used.

$$\mathrm{P}(X = x \mid Y = y) = \frac{\mathrm{P}(X = x, Y = y)}{\mathrm{P}(Y = y)} \tag{6.13}$$

6.3 Entropy

Defn 94 (Entropy)**.** The *entropy* $\mathrm{H}(X)$ of a Discrete Random Variable X is defined as

$$\mathrm{H}(X) = -\sum_{x \in \mathcal{X}} \mathrm{P}(x) \log_2\big(\mathrm{P}(x)\big) \tag{6.14}$$

Remark 94.1 (Probability is 0). By convention,

$$0 \log_2(0) = 0 \tag{6.15}$$

This can be justified by saying $\lim_{x \to 0} x \log_2(x) = 0$

Remark 94.2 (Entropy Bits). Since this logarithm is expressed in base-2, the output has the units of "bits". **These are not the same as bits in a computer**. They are just a unit, if the natural logarithm ln were used, the unit would be "nats".

Equation (6.14) can be expressed using the Expected Value/Mean of Single Discrete Random Variable.

$$\mathrm{H}(X) = \mathbb{E}\left[-\log_2\big(\mathrm{P}(X)\big)\right] \tag{6.16}$$

The Entropy of multiple Discrete Random Variables can be expressed as

$$\mathrm{H}(X, Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \mathrm{P}(x, y) \log_2\big(\mathrm{P}(x, y)\big) \tag{6.17}$$

- The actual values of X are not used or needed in the calculation of Entropy, only the Probability Distribution
- The Entropy of X can be calculated even if X does not take on numerical values.
- An interpretation of Entropy can be seen as *uncertainty* about the outcome of the Random Variable.
- If X takes one value with $\mathrm{P}(X = x) = 1$, and all other values have $\mathrm{P}(X \neq x) = 0$, then the Entropy is 0 bits.
- If the probabilities are evenly distributed among the possible outcomes, then the Entropy is $\log_2$(Num Outcomes).

6.3.1 Properties of Entropy

(i) If X is a Discrete Random Variable taking values in the set $\mathcal{X} = \{x_1, x_2, \ldots, x_{|\mathcal{X}|}\}$, then

$$0 \leq \mathrm{H}(X) \leq \log_2|\mathcal{X}| \tag{6.18}$$

(ii) $\mathrm{H}(X) = 0$ if and only if $\mathrm{P}(x) = 1$ **for some** $x \in \mathcal{X}$.

(iii) $\mathrm{H}(X) = \log_2|\mathcal{X}|$ if and only if $\mathrm{P}(x) = \frac{1}{|\mathcal{X}|}$ **for all** $x \in \mathcal{X}$.

(iv) If there are multiple Discrete Random Variables, the Entropy of their combination can be broken up

$$\mathrm{H}(X_1 X_2 \ldots X_n) = \mathrm{H}(X_1) + \mathrm{H}(X_2 \mid X_1) + \mathrm{H}(X_3 \mid X_1 X_2) + \cdots + \mathrm{H}(X_n \mid X_1 X_2 \ldots X_{n-1}) \tag{6.19}$$

(v) Property **(iv)** leads to the very useful equation

$$\mathrm{H}(XY) = \mathrm{H}(X) + \mathrm{H}(Y \mid X) = \mathrm{H}(Y) + \mathrm{H}(X \mid Y) \tag{6.20}$$

(vi) Due to Properties of Conditional Entropy in Paragraph 6.3.2.1, if and only if X and Y are Independent,

$$\mathrm{H}(XY) = \mathrm{H}(X) + \mathrm{H}(Y) \tag{6.21}$$

Defn 95 (Entropy of Binary Discrete Random Variable)**.** In the special case, when $|\mathcal{X}| = 2$, there is a special notation

$$\mathrm{h}(p) = -p \log_2(p) - (1 - p) \log_2(1 - p) \tag{6.22}$$

6.3.2 Conditional Entropy

Defn 96 (Conditional Entropy)**.** The *conditional Entropy*, $\mathrm{H}(X \,|\, Y)$ (Read as "the entropy of X given Y occurs") is defined as

$$\mathrm{H}(X \,|\, Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \mathrm{P}(x, y) \log_2\big(\mathrm{P}(x \,|\, y)\big) \tag{6.23}$$

Equation (6.23) can be expressed with Expected Value/Mean of Single Discrete Random Variable.

$$\mathrm{H}(X \,|\, Y) = \mathbb{E}\left[-\log_2\big(\mathrm{P}(X \,|\, Y)\big)\right] \tag{6.24}$$

If $\mathrm{P}(y) \neq 0$, we can introduce the notation

$$\mathrm{H}(X \,|\, Y) = \sum_{y \in \mathcal{Y}} \mathrm{P}(y)\, \mathrm{H}(X \,|\, Y = y) \tag{6.25}$$

6.3.2.1 Properties of Conditional Entropy

(i) If X and Y are Discrete Random Variables taking values from the sets $\mathcal{X} = \{x_1, x_2, \ldots, x_{|\mathcal{X}|}\}$ and $\mathcal{Y} = \{y_1, y_2, \ldots, y_{|\mathcal{Y}|}\}$, respectively, then

$$0 \leq \mathrm{H}(X \,|\, Y) \leq \log_2 |\mathcal{X}| \tag{6.26}$$

(ii) $\mathrm{H}(X \,|\, Y) = 0$ if and only if, **for every** y, $\mathrm{P}(x \,|\, Y) = 1$ **for some** $x \in \mathcal{X}$.
(iii) $\mathrm{H}(x \,|\, Y) = \log_2 |\mathcal{X}|$ if and only if **for every** y, $\mathrm{P}(x \,|\, y = \frac{1}{|\mathcal{X}|}$ **for all** $x \in \mathcal{X}$.
(iv) Similar properties hold for $\mathrm{H}(X \,|\, Y = y)$
(v) The Entropy of a Random Variable, X, can never increase by knowledge of the outcome of another Random Variable.

$$\mathrm{H}(X \,|\, Y) \leq \mathrm{H}(X) \tag{6.27}$$

6.4 Relative Entropy

Defn 97 (Relative Entropy)**.** The *relative entropy* between 2 Probability Distributions of 2 Discrete Random Variables, $P(X)$ and $Q(X)$ is defined as

$$\mathrm{D}\big(P(X) \,\|\, Q(X)\big) = \sum_{x \in \mathcal{X}} P(x) \log_2 \left(\frac{P(x)}{Q(x)}\right) \tag{6.28}$$

Equation (6.28) can also be written using Expected Value/Mean of Single Discrete Random Variable.

$$\mathrm{D}\big(P(X) \,\|\, Q(X)\big) = \mathbb{E}_P\left[\log_2 \left(\frac{P(x)}{Q(x)}\right)\right] \tag{6.29}$$

Remark 97.1 (Distance)*.* The Relative Entropy is a measure of "distance" between 2 Probability Distributions. It can be thought of as the inefficiency of assuming the Probability Distribution $Q(x)$ when $P(x)$ is actually correct.

6.4.1 Conditional Relative Entropy

Defn 98 (Conditional Relative Entropy)**.** *Conditional relative entropy* is defined similarly to the previous cases on conditionals.

$$\mathrm{D}\big(\mathrm{P}(X \,|\, Z) \,\|\, \mathrm{P}(X \,|\, Z)\big) = \sum_{z \in \mathcal{Z}} \mathrm{P}(z) \sum_{x \in \mathcal{X}} \mathrm{P}(x \,|\, z) \log_2 \left(\frac{P(x \,|\, z)}{Q(x \,|\, z)}\right) \tag{6.30}$$

6.5 Mutual Information

Defn 99 (Mutual Information)**.** The *mutual information* $\mathrm{I}(X;Y)$ (Read "the mutual information between X and Y") measures the amount of information (in bits) we receive about the Random Variable X when observing the outcome of the Random Variable Y. It is defined as

$$\mathrm{I}(X;Y) = \mathrm{D}\big(\mathrm{P}(X,Y) \,\|\, \mathrm{P}(X)\,\mathrm{P}(Y)\big) \tag{6.31}$$

Equation (6.31) can be equivalently rewritten using using only summations.

$$\mathrm{I}(X;Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \mathrm{P}(x, y) \log_2 \left(\frac{\mathrm{P}(x, y)}{\mathrm{P}(x)\,\mathrm{P}(y)}\right) \tag{6.32}$$

6.5.1 Properties of Mutual Information

(i) $\mathrm{I}(X;Y) = \mathrm{I}(Y;X)$
(ii) $\mathrm{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X \mid Y)$
(iii) $\mathrm{I}(X;Y) = \mathrm{H}(Y) - \mathrm{H}(Y \mid X)$
(iv) $\mathrm{I}(X;Y) = \mathrm{H}(X) + \mathrm{H}(Y) - \mathrm{H}(X,Y)$
(v) $\mathrm{I}(X;X) = \mathrm{H}(X)$

6.5.2 Conditional Mutual Information

Defn 100 (Conditional Mutual Information)**.** The *conditional mutual information* is defined similarly as the other conditional cases.

$$\mathrm{I}(X;Y \mid Z) = \mathrm{D}\big(\mathrm{P}(X,Y \mid Z) \,\|\, \mathrm{P}(X \mid Z)\,\mathrm{P}(Y \mid Z)\big) \tag{6.33}$$

where $\mathrm{D}\big(\mathrm{P}(X \mid Z) \,\|\, \mathrm{P}(Y \mid Z)\big)$ is the Conditional Relative Entropy.

7 Shannon's Theory of Secrecy

Defn 101 (Shannon's Theory of Secrecy)**.** *Shannon's Theory of Secrecy* was developed by Claude Shannon. It is an attack/defense model for a Symmetric Encryption system.

This model operates under the assumptions that there is Unconditional Security and only Ciphertext-Only Attacks.

Remark 101.1 (Flaws of this Model's Assumptions)*.* These assumptions are very strong. In reality, these assumptions would correspond to the **absolute best-case scenario**.

There is a set of Encryption Rules, one for each k, which maps Plaintext letters in a message $\mathbf{m} = m_1, m_2, \ldots, m_i \in \mathcal{P}$ to Ciphertext letters $\mathbf{c} = c_1, c_2, \ldots, c_i \in \mathcal{C}$.

7.1 Attack and Security Assumptions

There are several possible assumptions for Eve's attack on Alice's message.

1. *Ciphertext-Only Attack*: Eve has only the ciphertext C, and wants to get the key K or the plaintext message M.
2. *Known-Plaintext Attack*: Eve has both the ciphertext C and the plaintext message M, and wants the key K.
3. *Chosen-Plaintext Attack*: Eve knows M and and can arbitrarily choose M to get C back. She wants to get the key K.
4. *Chosen-Ciphertext Attack*: Eve has the Decryption Rule, d_K and can feed in any arbitrary C. She is attempting to find the key K.
5. *Related-Key Attack*: The message M is encrypted with a key K similar to another.
6. *Side Channel Attack*: Eve is attacking the implementation of the Encryption Rule, e_K, by observing some other outputs.
    - The power consumed by the algorithm
    - The network communcations occurring

Defn 102 (Ciphertext-Only Attack)**.** In a *ciphertext-only attack*, Eve has only the ciphertext C, and wants to get the key K or the plaintext message M.

Defn 103 (Known-Plaintext Attack)**.** In a *known-plaintext attack*, Eve has both the ciphertext C and the plaintext message M, and wants the key K.

Defn 104 (Chosen-Plaintext Attack)**.** In a *chosen-plaintext attack*, Eve knows M and and can arbitrarily choose M to get C back. She wants to get the key K.

Defn 105 (Chosen-Ciphertext Attack)**.** In a *chosen-ciphertext attack*, Eve has the Decryption Rule, d_K and can feed in any arbitrary C. She is attempting to find the key K.

Remark 105.1 (Lunchtime Attack)*.* Sometimes the Chosen-Ciphertext Attack is called a *lunchtime attack*, because an arbitrary Plaintext can be fed through the Encryption Rule quickly. This means the output can be found quickly, for example, while a coworker is having lunch.

Defn 106 (Related-Key Attack)**.** In a *related-key attack*, the message M is encrypted with a key K similar to another, already broken key.

Defn 107 (Side Channel Attack)**.** In a *side channel attack*, Eve is attacking the implementation of the Encryption Rule, e_K, by observing some other outputs.

- The power consumed by the algorithm
- The network communcations occurring

7.2 Security Scenarios

There are several scenarios we have for our Encryption Rule and Decryption Rule. In the order of **strongest security** to **weakest security**:

1. *Unconditional Security*: A Cryptographic Primitive is unconditionally secure if it cannot be broken even if Eve has infinite computational power.
   - This means she can also perform an exhautive key search (Brute Force)
2. *Computational Security*: A Cryptographic Primitive is computationally secure if the best algorithm to break the key requires at least T operations, where T is a very large number.
3. *Provable Security*: A Cryptographic Primitive is provably secure if the key can be reduced to a well known and well-studied problem.
   - Semiprime Integer Factorization is an example a key that is provably secure.
4. *Heuristic Security*: If there is no known method of breaking the Cryptographic Primitive, but the security cannot be proven in any sense.

Defn 108 (Unconditional Security)**.** A Cryptographic Primitive has *unconditional security*, is *unconditionally secure*, if it cannot be broken even if Eve has infinite computational power.

- This means she can also perform an exhautive key search (Brute Force)

Defn 109 (Computational Security)**.** A Cryptographic Primitive has *computational security*, is *computationally secure*, if the best algorithm to break the key requires at least T operations, where T is a very large number.

Defn 110 (Provable Security)**.** A Cryptographic Primitive has *provable security*, is *provably secure*, if the key can be reduced to a well known and well-studied problem.

- Semiprime Integer Factorization is an example a key that is provably secure.

Defn 111 (Heuristic Security)**.** A Cryptographic Primitive has *heuristic security*, is *heuristically secure*, if there is no known method of breaking the Cryptographic Primitive, but the security cannot be proven in any sense.

Defn 112 (Perfect Secrecy)**.** A cryptosystem has *perfect secrecy* if Equation (7.1) is true.

$$\mathrm{I}(M;C) = 0 = H(M) - H(M \,|\, C) \tag{7.1}$$

7.3 Shannon's Theory with Discrete Random Variables

Shannon's Theory of Secrecy can be modelled with 3 random variables:

$$\begin{aligned} &\mathbf{M} = (M_1, M_2, \ldots, M_N) \text{ and } \mathrm{P}(\mathbf{M}) \\ &\mathbf{C} = (C_1, C_2, \ldots, C_N) \text{ and } \mathrm{P}(\mathbf{C}) \\ &K \in \mathcal{K} \text{ and } \mathrm{P}(K) \end{aligned} \tag{7.2}$$

Defn 113 (Key Entropy)**.** The *key Entropy* is the uncertainty Eve faces regarding an unknown key that she has no experience with.

$$\mathrm{H}(K) = -\sum_{k \in \mathcal{K}} \mathrm{P}(k) \log_2\big(\mathrm{P}(k)\big) \tag{7.3}$$

Remark 113.1 (Key Entropy Upper Bound)*.*

$$\mathrm{H}(K) \leq \log_2 |\mathcal{K}| \tag{7.4}$$

Defn 114 (Message Entropy)**.** The *message Entropy* is the uncertainty regarding the transmitted message.

$$\mathrm{H}(\mathbf{M}) = -\sum_{\mathbf{m} \in \mathcal{P}^N} \mathrm{P}(\mathbf{m}) \log_2\big(\mathrm{P}(\mathbf{m})\big) \tag{7.5}$$

Remark 114.1 (Message Entropy Upper Bound)*.*

$$\mathrm{H}(\mathbf{M}) \leq \log_2 |\mathcal{P}|^N \tag{7.6}$$

Defn 115 (Key Equivocation)**.** *Key equivocation* is the case when Eve uses her knowledge of the Ciphertext to help break the key.

$$\mathrm{H}(K \,|\, \mathbf{C}) = -\sum_{k \in \mathcal{K}, \mathbf{c} \in \mathcal{C}^N} \mathrm{P}(k, \mathbf{c}) \log_2\big(\mathrm{P}(k \,|\, \mathbf{c})\big) \tag{7.7}$$

Remark 115.1 (Key Equivocation Upper Bound). The uncertainty of the key can never increase by observing $\mathbf{C}$. Otherwise, why even encrypt the Plaintext?

$$\mathrm{H}(K \mid \mathbf{C}) \leq \mathrm{H}(K) \tag{7.8}$$

Defn 116 (Message Equivocation)**.** *Message equivocation* is the case when Eve uses her knowledge of the Ciphertext to help break the Plaintext.

$$\mathrm{H}(\mathbf{M} \mid \mathbf{C}) = -\sum_{\mathbf{m} \in \mathcal{P}^N, \mathbf{c} \in \mathcal{C}^N} \mathrm{P}(\mathbf{m}, \mathbf{c}) \log_2\big(\mathrm{P}(\mathbf{m} \mid \mathbf{c})\big) \tag{7.9}$$

Remark 116.1 (Message Equivocation Upper Bound). The uncertainty of the message can never increase by observing $\mathbf{C}$. Otherwise, why even encrypt the Plaintext?

$$\mathrm{H}(\mathbf{M} \mid \mathbf{C}) \leq \mathrm{H}(\mathbf{M}) \tag{7.10}$$

Theorem 7.1 (Message Equivocation Bounded by Key Equivocation)**.** *For an encryption scheme, we have*

$$\mathrm{H}(\mathbf{M} \mid \mathbf{C}) \leq \mathrm{H}(K \mid \mathbf{C}) \tag{7.11}$$

Message Equivocation Bounded by Key Equivocation. The expression $\mathrm{H}(K, \mathbf{M} \mid \mathbf{C})$ can be written

$$\mathrm{H}(K, \mathbf{M} \mid \mathbf{C}) = \mathrm{H}(K \mid \mathbf{C}) + \mathrm{H}(\mathbf{M} \mid \mathbf{C}, K)$$

When the key and ciphertext are given, the plaintext can be uniquely determined, so

$$\mathrm{H}(\mathbf{M} \mid \mathbf{C}, K) = 0$$

Since $\mathit{Entropy}(\mathbf{M} \mid \mathbf{C}) \leq \mathrm{H}(K, \mathbf{M} \mid \mathbf{C})$, we get

$$\mathrm{H}(\mathbf{M} \mid \mathbf{C}) \leq \mathrm{H}(K \mid \mathbf{C})$$

■

Defn 117 (Alphabet Size)**.** The *alphabet size*, denoted L, is

$$|\mathcal{P}| = |\mathcal{C}| = L \tag{7.12}$$

Defn 118 (Rate of the Alphabet)**.** The *rate of the alphabet* is the **maximum Entropy** possible.

$$H_0 = \log_2(L) \tag{7.13}$$

Defn 119 (Entropy Per Alphabet Symbol)**.** The actual Entropy of the message source **per alphabet symbol** is denoted $H(M)$ and is defined as

$$\mathrm{H}(M) = \frac{\mathrm{H}(\mathbf{M})}{N} \tag{7.14}$$

for a message $\mathbf{M} = (M_1, M_2, \ldots, M_N)$ of length N.

Remark 119.1 (Cases of Entropy Per Alphabet Symbol). There are 2 cases:

1. A memoryless source
2. A non-memoryless source

Defn 120 (Redundancy)**.** The *redundancy* of a source, denoted D is

$$D = H_0 - \mathrm{H}(M) \tag{7.15}$$

A higher redundancy means there is more regularity in the message, which makes it easier to break.

Remark 120.1 (Benefits for Eve). Redundancy helps Eve! There are 2 ways to reduce redundancy in digital systems:

1. Compress the message before encryption, because regularities in the message are "removed" by compression.
2. Channel coding (the addition of parity bits for correct/detection of errors) should be done after encryption, because their regularity before will increase redundancy, hurting the key's strength.

Theorem 7.2 (Shannon's Theory of Secrecy)**.**

$$\mathrm{H}(K \mid \mathbf{C}) \geq \mathrm{H}(K) - ND \tag{7.16}$$

Shannon's Theory of Secrecy. Since $\mathrm{H}(K, \mathbf{M}, \mathbf{C}) = \mathrm{H}(K, \mathbf{M}) = \mathrm{H}(K, \mathbf{C})$ (assuming unique encryption/decryption, and non-probabilistic encryption).

Starting with $\mathrm{H}(K, \mathbf{M})$, we have the relation

$$\mathrm{H}(K, \mathbf{M}) = \mathrm{H}(K) + \mathrm{H}(\mathbf{M})$$

which can be simplified to

$$\mathrm{H}(K, \mathbf{M}) = \mathrm{H}(K) + N\,\mathrm{H}(M)$$

On the other side, $\mathrm{H}(K, \mathbf{C})$, we have the relation

$$\mathrm{H}(K, \mathbf{C}) = \mathrm{H}(\mathbf{C}) + \mathrm{H}(K \,|\, \mathbf{C}) \leq NH_0 + \mathrm{H}(K \,|\, \mathbf{C})$$

Putting these 2 together gives us

$$\mathrm{H}(K \,|\, \mathbf{C}) \geq N\,\mathrm{H}(M) + \mathrm{H}(K) - NH_0 = \mathrm{H}(K) - ND$$

■

Defn 121 (Unicity Distance)**.** The *unicity distance*, denoted N_0 is defined as

$$N_0 = \frac{\mathrm{H}(K)}{D} \tag{7.17}$$

The unicity distance, practically, gives the number of symbols in the Ciphertext we need to observe to go from having multiple possible keys to a single possible key.

Theorem 7.3 (Drawback of Perfect Secrecy)**.** *For a cryptosystem with Perfect Secrecy, we have*

$$\mathrm{H}(\mathbf{M}) \leq \mathrm{H}(K)$$

Drawback of Perfect Secrecy. We know

$$\mathrm{H}(\mathbf{M} \,|\, \mathbf{C}) \leq \mathrm{H}(K \,|\, \mathbf{C}) \leq \mathrm{H}(K)$$

But the definition of Perfect Secrecy (Definition 112) implies

$$\mathrm{H}(\mathbf{M} \,|\, \mathbf{C}) = \mathrm{H}(\mathbf{M})$$

So, Theorem 7.3 follows.

$$\mathrm{H}(\mathbf{M}) \leq \mathrm{H}(K \,|\, \mathbf{C}) \leq \mathrm{H}(K)$$

■

8 Stream Ciphers — LFSR Sequences

There are 2 main types of Symmetric Encryption algorithms:

1. Block Ciphers
2. Stream Ciphers

Defn 122 (Block Cipher)**.** *Block cipher*s encrypt a block of symbols of a fixed size from the Plaintext message using an fixed-size encryption transformation.

Defn 123 (Stream Cipher)**.** *Stream cipher*s encrypt individual characters of the Plaintext using an encryption transofrmation that varies with time. The 2 elements that make up a stream cipher are **memory** and a **combinatorial function**.

1. Memory
   - Linear Feedback Shift Registers
   - Tables (Arrays)
2. Combinatorial Function
   - Nonlinear Boolean functions, S-boxes
   - XOR, Modular addition, cyclic rotations, multiplication

Defn 124 (Keystream)**.** The *keystream* is the output of the generator, which gets used on the Plaintext message to form the Ciphertext

$$\mathbf{z} = z_1, z_2, \ldots \tag{8.1}$$

Remark 124.1 (Attacks). For a synchronous Stream Cipher, a Known-Plaintext Attack, Chosen-Plaintext Attack, or Chosen-Ciphertext Attack is equivalent to having access to the Keystream.

Remark 124.2 (Secret Key). When the Keystream first starts, because it is a synchronous system, it needs to have a starting value with which to generate subsequent values. This first value may be fixed, and is the secret key, K, in this case.

8.1 Assumptions

We assume that

- The output Keystream of length N is known to Eve.

8.2 Attacks

An attack is considered successful only if the complexity of performing it is considerably lower than the exhaustive keysearch of complexity 2^k.

The attack types that are of interest are:

- *Key Recovery Attack*: Eve tries to recover the Secret Key, K.
- *Distinguishing Attack*: Eve tries to determine whether a given sequence $\hat{\mathbf{z}} = z_1, z_2, \ldots, z_N$ is likely to have been generated from the considered Stream Cipher, or if it's a truly random sequence.

Defn 125 (Key Recovery Attack)**.** In a *key recovery attack*, Eve tries to recover the Secret Key, K.

Defn 126 (Distinguishing Attack)**.** In a *distinguishing attack*, Eve tries to determine whether a given sequence $\hat{\mathbf{z}} = z_1, z_2, \ldots, z_N$ is likely to have been generated from the considered Stream Cipher, or if it's a truly random sequence.

Let $D(\mathbf{z})$ be an algorithm that takes in a sequence $\mathbf{z}$ of length N, and outputs either "X" or "RANDOM". With probability $\frac{1}{2}$, the sequence $\mathbf{z}$ is produced by a generator X otherwise, it is a random sequence. The probability that $D(\mathbf{z})$ correctly determines the origin of $\mathbf{z}$ is written $\mathrm{P}\big(D(\mathbf{z})\big) = \frac{1}{2} + \epsilon$. If ϵ is not very close to zero, then $D(\mathbf{z})$ is a *distinguisher* for the generator X, because it can more often than not distinguish that the sequence is nonrandom.

Remark 126.1. A Distinguishing Attack is a much weaker attack than a Key Recovery Attack.

8.3 Linear Feedback Shift Registers

Defn 127 (Linear Feedback Shift Register)**.** A *linear feedback shift register* is a data storage element. A register has L delay (storage) elements, each of which is capable of storing one element from the field $\mathbb{F}_q$, and a clock signal. When the clock signal is applied, the register's delay elements are shifted one step, and the value shifted into the beginning is calculated as a linear function of the contents of the register.

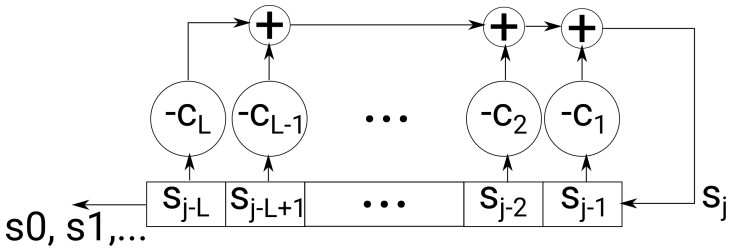


Figure 8.1: Linear Feedback Shift Register

Defn 128 (Shift Register Equation)**.** The *shift register equation* is a way of describing the coefficients, $c_1, c_2, \ldots, c_L \in \mathbb{F}_q$, and their recurrence relation

$$s_j = -c_1 s_{j-1} - c_2 s_{j-2} - \cdots - c_L s_{j-L} \tag{8.2}$$

for $j = L, L+1, \ldots$.

If $c_0 = 1$, we can write

$$\sum_{i=0}^{L} c_i s_{j-i} = 0, \text{ for } j = L, L+1, \ldots \tag{8.3}$$

Remark 128.1 (Initial State). The first L symbols, $s_0, s_1, \ldots, s_{L-1}$ form the *initial state*.

Remark 128.2 (Fibonacci Implementation). The Linear Feedback Shift Register setup shown in Figure 8.1 is implemented in a *fibonacci*-style.

Defn 129 (Connection Polynomial)**.** The coefficients, $c_0, c_1, \ldots, c_L$ are summarized in the *connection polynomial*.

$$C(D) = 1 + c_1 D + c_2 D^2 + \cdots + c_L D^L \tag{8.4}$$

Remark 129.1 (Alternate Form). We can write the Connection Polynomial in a slightly different form, to denote both the Connection Polynomial and the length.

$$\langle C(D), L \rangle \tag{8.5}$$

Defn 130 (D-Transform)**.** The *D-Transform* is actually just the $\mathcal{Z}$-Transform, with a different indeterminate variable. The D-transform of a sequence $\mathbf{s} = s_0, s_1, \ldots$ is defined as

$$S(D) = s_0 + s_1 D + s_2 D^2 \tag{8.6}$$

if $s_i \in \mathbb{F}_q$.

The indeterminate D is the "delay", and the exponent indicates the number of delays.

Remark 130.1 (Our Assumptions)*.* For this course, we assume that $s_i = 0$ for $i < 0$, i.e. the signal is causal. The set of all such sequences having the form

$$f(D) = \sum_{i=0}^{\infty} f_i D^i$$

where $f_i \in \mathbb{F}_q$ is denoted $\mathbb{F}_q[[\mathrm{D}]]$, and is called the *ring of formal power series.*

Theorem 8.1. *The set of sequences generated by the Linear Feedback Shift Register with Connection Polynomial $C(D)$ is the set of sequences that have the D-Transform*

$$S(D) = \frac{P(D)}{C(D)} \tag{8.7}$$

where $P(D)$ is an arbitrary Polynomial of Degree at most $L-1$,

$$P(D) = p_0 + p_1 D + \cdots + p_{L-1} D^{L-1} \tag{8.8}$$

The relation between the initial state of the Linear Feedback Shift Register and the $P(D)$ Polynomial is given by the linear relation.

$$\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{L-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ c_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{L-1} & c_{L-2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{L-1} \end{pmatrix}$$

8.4 Linear Feedback Shift Registers Sequences and Extension Fields

Let $\pi(x)$ be an Irreducible Polynomial over $\mathbb{F}_q$ and assume its coefficients are

$$\pi(x) = x^L + c_1 x^{L-1} + \cdots + c_L \tag{8.9}$$

This means that $\pi(x)$ is the *reciprocal* Polynomial of the Connection Polynomial $C(D)$.

We can construct an Extension Field $\mathbb{F}_{q^L}$ through $\pi(\alpha) = 0$. The term β from $\mathbb{F}_q$ can be expressed in a Polynomial basis as

$$\beta = \beta_0 + \beta_1 \alpha^1 + b_2 \alpha^2 + \cdots + \beta_{L-1} + \alpha^{L-1} \tag{8.10}$$

where $\beta_0, \beta_1, \ldots, \beta_{L-1} \in \mathbb{F}_q$.

If we multiply Equation (8.10) by α, we get

$$\alpha\beta = \beta_0\alpha + \beta_1\alpha^2 + \cdots + \beta_{L-1}\alpha^L$$

Now, we can reduce the α^L according to the $\pi(\alpha) = 0$ relation we found earlier. This gives us

$$\alpha\beta = -c_L\beta_{L-1} + (\beta_0 - c_{L-1}\beta_{L-1})\,\alpha + \cdots + (\beta_{L-2} - c_1\beta_{L-1})\,\alpha^{L-1} \tag{8.11}$$

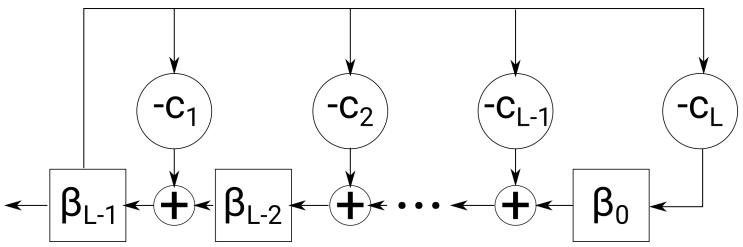


Figure 8.2: Linear Feedback Shift Register in Galois Form

We can check that this form, the *Galois Form* of this Linear Feedback Shift Register is the same as the other.

$$s_j = -c_1 s_{j-1} - c_2 s_{j-2} - \cdots - c_L s_{j-L}$$

when $j \geq L$. Also,

$$p_0 = s_0, p_1 = s_1 + c_1 s_0, \ldots$$

where $p_0, p_1, \ldots, p_{L-1}$ is the initial state of the Linear Feedback Shift Register in its Galois Form.

The set of Linear Feedback Shift Register sequences, when $C(D)$ is Irreducible, is exactly the set of sequences possible to produce by the implmentation of multiplication of an element β by the fixed element α in $\mathbb{F}_{q^L}$.

For a specific sequence specified as $S(D) = \frac{P(D)}{C(D)}$, the initial state if the first L symbols, whereas the same sequence is produced in Figure 8.2 if the initial state is $p_0, p_1, \ldots, p_{L-1}$.

8.4.1 Properties of Linear Feedback Shift Registers

(i) A sequence $\mathbf{s} = \ldots, s_0, s_1, \ldots$ is called *periodic* if there is a positive integer T such that $s_i = s_{i+T}\ \forall i \geq 0$.
(ii) The *period* is the least such positive integer T for which $s_i = s_{i+T}\ \forall i \geq 0$
(iii) The Linear Feedback Shift Register state runs through different values. The initial state will appear again after visiting a number of states. If $\deg\big(C(D)\big) = L$, the period of a sequence is the same as the number of different states visited, before returning to the initial state.
(iv) If $C(D)$ is Irreducible, the state corresponds to an element in $\mathbb{F}_{q^L}$, call it β.
(v) The sequence of diferent states that we are entering is

$$\beta, \alpha\beta, \alpha^2\beta, \ldots, \alpha^{T-1}\beta, \big(\alpha^T\beta = \beta\big)$$

where T is the Element Order of α

Defn 131 (m-Sequence)**.** An *m-sequence* is one where α is a Primitive Element. This means we go through all $q^L - 1$ states, where $\text{ord}(\alpha) = q^L - 1$. These will only appear if and only if the Polynomial $\pi(x)$ is a Primitive Element. A $\pi(x)$ being Irreducible is not enough, it **must** also be Primitive Element.

A periodic and causal sequence, we denoted $[s_0, s_1, \ldots, s_{T-1}]^\infty$. This is equivalent to

$$s_0, s_1, \ldots, s_{T-1}, s_0, s_1, \ldots, s_{T-1}, s_0, \ldots$$

where $s_i \in \mathbb{F}_q, i = 0, 1, \ldots, T-1$

Defn 132 (Polynomial Period)**.** The *period of a Polynomial* $C(D)$ is the least positive integer T such that $C(D) \mid \big(1 - D^T\big)$.
This is calculated by dividing 1 by $C(D)$ and continuing until you receive a remainder of the form $1 \cdot D^N$. The remainder **MUST** have the coefficient of 1. And the value of $T = N$.

Example 8.1: Polynomial Period. Lecture 9, Example 4.6

Calculate the period of the Polynomial $C(D) = 3 + D^2$ in $\mathbb{F}_5$?

Start by dividing 1 by $C(D)$.

$$\frac{1}{3 + D^2} = 2 + D^2 + 3D^4 + 4D^6$$

with a remainder of $1D^8$.
Thus, the period of $C(D) = 3 + D^2$ is $T = 8$. We can check this by making sure the remainder of the division is 0, with

$$\begin{array}{r} \frac{\big(1 + -1D^8\big)}{\frac{1}{3+1D^2}} \\ \big(3 + 1D^2\big)\overline{\big)\ \big(1 + -1D^8\big)} \\ \underline{-\big(1 + -1D^8\big)} \\ 0 \end{array}$$

and it is, so we found the right answer.

Theorem 8.2. *If* $\gcd\big(C(D), P(D)\big) = 1$*, then the Connection Polynomial* $C(D)$ *and the sequence* $\mathbf{s}$ *with D-Transform*

$$S(D) = \frac{P(D)}{C(D)}$$

have the same period, the period of $\mathbf{s}$ *is the same as the Polynomial Period* $C(D)$*.*

- This $C(D)$ gives the shortest Linear Feedback Shift Register generating $\mathbf{s}$.
- Any other $C(D)$ generating $\mathbf{s}$ must be a multiple of $C(D)$.

Theorem 8.3. *If the 2 sequences* $\mathbf{s}_A$ *and* $\mathbf{s}_B$*, with period* T_A *and* T_B *have D-Transforms*

$$S_A(D) = \frac{P_A(D)}{C_A(D)} \qquad\qquad S_B(D) = \frac{P_B(D)}{C_B(D)}$$

then the sum of the sequences $\mathbf{s} = \mathbf{s}_A = \mathbf{s}_B$ *has a D-Transform*

$$S(D) = S_A(D) + S_B(D) \tag{8.12}$$

and the shared period is

$$\operatorname{lcm}(T_A, T_B) \tag{8.13}$$

This all happens under the assumptions:

- $\gcd\big(P_A(D), C_A(D)\big) = 1$
- $\gcd\big(P_B(D), C_B(D)\big) = 1$
- $\gcd\big(C_A(D), C_B(D)\big) = 1$

Example 8.2: Shortest LFSR Sequence. Lecture 9, Example 4.7

Given the sequence $[1, 0, 1, 0, 0, 1, 1]^\infty$ in $\mathbb{F}_2$, what is the shortest Linear Feedback Shift Register that can produce this sequence?

Start by finding the general case of this sequence, in a D-Transformed state.

$$S(D) = \frac{1 + D^2 + D^5 + D^6}{1 + D^7}$$

The numerator is drawn from the given sequence. The denominator is found from the length of the period. However, this is not the shortest Linear Feedback Shift Register possible to generate this sequence.
If we find a common factor, we can simplify this Linear Feedback Shift Register.

$$\gcd(1 + D^2 + D^5 + D^6, 1 + D^7)$$

We solve this with the Polynomial Euclidean Algorithm.

$$\begin{aligned} 1 + D^7 &= q(D)\left(1 + D^2 + D^5 + D^6\right) + r(D) \\ &= (D+1)\left(1 + D^2 + D^5 + D^6\right) + \left(D^5 + D^3 + D^2 + D\right) \end{aligned}$$

Since the remainder is not 0, we continue reducing.

$$\begin{aligned} \left(1 + D^2 + D^5 + D^6\right) &= q(D)\left(D^5 + D^3 + D^2 + D\right) + r(D) \\ &= (D+1)\left(D^5 + D^3 + D^2 + D\right) + \left(D^4 + D^2 + D + 1\right) \\ \left(D^5 + D^3 + D^2 + D\right) &= q(D)\left(D^4 + D^2 + D + 1\right) + r(D) \\ &= (D)\left(D^4 + D^2 + D + 1\right) + 0 \end{aligned}$$

So, the $\gcd(1 + D^2 + D^5 + D^6, 1 + D^7) = D^4 + D^2 + D + 1$.

Now we factor that value out of the $S(D)$ fraction, and reduce it out.

$$\begin{aligned} S(D) &= \frac{\left(D^4+D^2+D+1\right)\left(D^2+D+1\right)}{\left(D^4+D^2+D+1\right)\left(D^3+D+1\right)} \\ &= \frac{\left(D^2+D+1\right)}{\left(D^3+D+1\right)} \end{aligned}$$

Thus, the shortest Linear Feedback Shift Register for the given sequence has a Connection Polynomial of $C(D) = D^3+D+1$, with an initial state D-Transform equation of $P(D) = D^2+D+1$.

8.5 Cycle Sets

Defn 133 (Cycle Set)**.** The *cycle set* for $C(D)$ is the number of cycles of a given length. This is assuming $L = \deg\big(C(D)\big)$. It is written

$$n_1\left(T_1\right) \oplus n_1\left(T_1\right) \oplus \cdots \tag{8.14}$$

For example, $1(1) \oplus 3(5)$ means there is one cycle of length one, and three cycles of length 5.

Remark 133.1 (Combining). You can combine cycle with the same period by simple addition.

$$n_1(T) \oplus n_2(T) = \left(n_1+n_2\right)(T) \tag{8.15}$$

8.5.1 Properties of Cycle Sets

(i) If $C(D)$ is a Primitive Element of Degree L over $\mathbb{F}_q$, then the Cycle Set is

$$1(1) \oplus 1\left(q^L-1\right) \tag{8.16}$$

(ii) If $C(D)$ is an Irreducible Polynomial, then the Cycle Set is

$$1(1) \oplus \frac{\left(q^L-1\right)}{T}(T) \tag{8.17}$$

where T is the Polynomial Period of $C(D)$, or the Element Order of α when $\pi(\alpha) = 0$.

Theorem 8.4 (Connection Polynomial Multiple)**.** *If $C(D) = C_1(D)^n$, then the Cycle Set of $C(D)$ is*

$$1(1) \oplus \frac{\left(q^{L_1}-1\right)}{T_1}\left(T_1\right) \oplus \frac{q^{L_1}\left(q^{L_1}-1\right)}{T_2}\left(T_2\right) \oplus \cdots \oplus \frac{q^{(n-1)L_1}\left(q^{L_1}-1\right)}{T_n}\left(T_n\right) \tag{8.18}$$

where $\deg\big(C(D)\big) = L$ and T_j is the Polynomial Period of $C_1(D)^j$.

Theorem 8.5 (Irreducible Connection Polynomial Multiple)**.** *if $C_1(D)$ is Irreducible with period T_1, then the Polynomial Period of $C_1(D)^j$ is*

$$T_j = p^m T_1 \tag{8.19}$$

where p is the Characteristic of the Field and m is the integer satisfying

$$p^{m-1} < j \leq p^m \tag{8.20}$$

Example 8.3: Cycle Set of Connection Polynomial. Lecture 9, Example 4.9

Compute the Cycle Set for the Connection Polynomial $C(D) = \left(1+D+D^2\right)^3$ in $\mathbb{F}_2$?

The Connection Polynomial here is actually in the form $C(D) = C_1(D)^3$. In this case, $C_1(D) = 1+D+D^2$, which is both Irreducible and Primitive Element.
We can find the Polynomial Period in 3 ways.

1. Because it's $C_1(D)$ is a primitive Polynomial, $T_1 = q^L - 1 = 3$.
2. Could perform a Polynomial Period, which yields the same thing.
3. Because it's $C_1(D)$ is a primitive Polynomial, $\pi(\alpha) = 0$, and the ord $\left(\alpha^2\right) = 3$.

Now, T_2 must be calculated with $T_2 = 2^{m_2} \cdot 3$, where $2^{m_2-1} < 2 \leq 2^{m_2}$. $m_2 = 1$, plugging that back in gives $T_2 = 2$.
Doing the same for T_3 yields $T_3 = 12$.
Now we combine these into a single line.

$$1(1) \oplus \frac{\left(2^2-1\right)}{3}(3) \oplus \frac{2^2\left(2^2-1\right)}{3}(6) \oplus \frac{2^6\left(2^2-1\right)}{3}(12)$$
$$1(1) \oplus 1(3) \oplus 2(6) \oplus 4(12)$$

Theorem 8.6 (Arbitrary Connection Polynomial)**.** *For a Connection Polynomial $C(D)$ factoring like*

$$C(D) = C_1(D)^{n_1} C_2(D)^{n_2} \cdots C_m(D)^{n_m}$$

$C_i(D)$ is Irreducible, has a Cycle Set $S_1 \times S_2 \times \cdots S_m$, where S_i is the Cycle Set for $C_i^{e_i}$, and

$$n_1\left(T_1\right) \times n_2\left(T_2\right) = n_1 n_2 \gcd(T_1, T_2)\big(\operatorname{lcm}(T_1, T_2)\big) \tag{8.21}$$

and the distributive law holds for $\times$ and $\oplus$.

8.6 Decimation

An m-Sequence $\mathbf{s} = s_0, s_1, s_2, \ldots$.

- Define the sequence $\mathbf{s}'$ obtained through decimation by k, defined by the sequence

$$\mathbf{s}' = s_0, s_k, s_{2k}, \ldots$$

- $\mathbf{s}$ corresponds to a multiplication of β by the fixed element α. Thus, $\mathbf{s}'$ corresponds to multiplication β by the fixed element α^k, which means the cycle of different states corresponds to

$$\beta, \alpha^k\beta, \alpha^{2k}\beta, \ldots, \alpha^{(T-1)k}\beta, \left(\alpha^{Tk}\beta = \beta\right)$$

- The period of $\mathbf{s}'$ is $\operatorname{ord}\left(\alpha^k\right) = \dfrac{q^L-1}{\gcd\left(q^L-1, k\right)}$.

8.7 Statistical Properties

The importance of Linear Feedback Shift Register sequences and m-Sequences are due to their pseudorandomness properties. $\mathbf{s} = s_0, s_1, \ldots$ is an m-Sequence, and an n-gram is a subsequence of length n,

$$\left(s_t, s_{t+1}, \ldots, s_{t+n-1}\right)$$

for $t = 0, 1, \ldots$

Theorem 8.7. *Among the $q^L - 1$ L-grams that can be constructed for $t = 0, 1, \ldots, q^L - 2$, ever nonzero vector appears exactly once.*

8.7.1 Autocorrelation Function

Defn 134 (Autocorrelation Function)**.** The *autocorrelation function* has:

- 2 binary sequences, $\mathbf{x}, \mathbf{y}$ of the same length n.
- The autocorrelation $C(\mathbf{x}, \mathbf{y})$ between the 2 sequences is defined as the number of positions of agreements minus the number of disagreements.
- The autocorrelation function $C(\tau)$ is defined to be the correlation between a sequence $\mathbf{x}$ and its τth cyclic shift, i.e.,

$$C(\tau) = \sum_{i=1}^{n} (-1)^{x_i + x_{i+\tau}} \tag{8.22}$$

where subscripts are taken modulo n and addition in the exponent is $\mod 2$ addition.

Theorem 8.8 (Autocorrelation of m-Sequence)**.** *If $\mathbf{s}$ is an m-sequence of length $2^L - 1$, then*

$$C(\tau) = \begin{cases} 2^L - 1 & \text{if } \tau \equiv 0 \bmod n \\ -1 & \text{otherwise} \end{cases} \tag{8.23}$$

9 Block Ciphers

Defn 135 (Block Cipher)**.** A *block cipher* encrypts a fixed-length block of plaintext bits x to a fixed-length block of ciphertext y. This transformation is controlled by the secret key K, and is written $E_K(x) = y$.

The secret key defines a fixed mapping of the plaintext block x to the ciphertext block y. This can sometimes make block ciphers a form of The Simple Substitution Cipher.

Block ciphers are usually implemented with several Round Functions, based on the Round Key.

Defn 136 (Round Function)**.** Block Ciphers are usually implemented as iterated ciphers, where a simple encryption function is iteratively applied for N rounds. This is called a *round function*. It is commonly denoted

$$h(w_{i-1}, k_i) \tag{9.1}$$

where

- i is the current round (current iteration).
- w is the input plaintext block.
- k is the Round Key on the ith iteration.
- h is the Round Function.

These functions must also be invertible, namely,

$$h^{-1}\big(h(w_{i-1}, k_i), k_i\big) = w_{i-1} \tag{9.2}$$

These functions must be efficient to compute, and be efficient to compute the inverse round function.

Defn 137 (Round Key)**.** The *round key* is derived from the key K. The way in which the round key is derived from K is called the *key schedule*.

If we want to mathematically illustrate the implementation of a Block Cipher's encryption with a iterative cipher, it is defined as:

$$\begin{aligned} w_0 &= x \\ w_1 &= h(w_0, k_1) \\ w_2 &= h(w_1, k_2) \\ &\vdots \\ w_{N-1} &= h(w_{N-2}, k_{N-1}) \\ w_N &= h(w_{N-1}, k_N) \end{aligned}$$

- x is the plaintext block.
- w_i are intermediate values in the implementation of the iteration.
- w_N is the final output from the cipher.
- $h(w_{i-1}, k_i)$ denotes the round function.
- k_i is the round key used in the ith round.

For the same iterative cipher, the decryption function $D_K(y)$ is just the application of the Round Keys in reverse order.

$$\begin{aligned} w_N &= y \\ w_{N-1} &= h^{-1}(w_N, k_N) \\ w_{N-2} &= h^{-1}(w_{N-1}, k_{N-2}) \\ &\vdots \\ w_1 &= h^{-1}(w_2, k_2) \\ w_0 &= h^{-1}(w_1, k_1) = x \end{aligned}$$

9.1 Examples of Iterated Ciphers

9.1.1 Feistel Ciphers/DES

Defn 138 (Feistel Cipher)**.** A *Feistel cipher* is a type of iterated cipher that is easy to implement in both hardware and software. However, it is not safe to use anymore.

An example of these is the DES encryption algorithm.

The block being run through the Round Function is split in half; a left and right half, denoted

$$w_i = (L_i, R_i) \tag{9.3}$$

The Round Function $h(L_{i-1}, R_{i-1}, k_i)$ is implemented as

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, k_i) \end{aligned} \tag{9.4}$$

where $f(R_{i-1}, k_i)$ can be any function.

The decryption function for the Feistel cipher is implemented as

$$\begin{aligned} L_{i-1} &= R_i \oplus f(R_{i-1}, k_i) \\ R_{i-1} &= L_i \end{aligned} \tag{9.5}$$

and (L_0, R_0) gives back x.

9.1.2 SP Network

Defn 139 (SP Network)**.** An *SP network* is a iterated Block Cipher that consists of a mix between substitutions (S) (The Simple Substitution Ciphers) and permutations (P) or linear transformations (29s). The Round Key is added to the input to the Round Function in a simple way ($\oplus$).

An example of these is the AES (Advanced Encryption Scheme) cipher.

Defn 140 (S-Box)**.** An *S-Box* is the substitution portion of an SP Network. They are usually taken over a small alphabet, and only have 4, 6, or 8 bits as input.

9.2 Modes of Operation

Defn 141 (Mode of Operation)**.** A *mode of operation* describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block. This is helpful because, by themselves, Block Ciphers can only encrypt/decrypt one block. To accomplish this, the plaintext is split up into blocks, and each block is encrypted.

$$x_1, x_2, \ldots, x_N, x_{N+1}, \ldots$$

Let $X_1 = (x_1, x_2, \ldots, x_N)$ be the first block with length N. Let $X_2 = (x_{N+1}, x_{N+2}, \ldots, x_{2N})$ be the second block, also with length N. The encryption is then done blockwise

Most modes require a unique binary sequence, often called an Initialization Vector (IV), for each encryption operation.

There are 3 modes of operation that are discussed in this course:

1. Electronic Codebook (ECB) Mode
2. Cipher Block Chaining (CBC) Mode
3. Counter Mode

Defn 142 (Initialization Vector)**.** An *initialization vector* (IV) is a unique binary sequence that is used for each encryption operation. The IV has to be non-repeating and, for some modes, random as well.

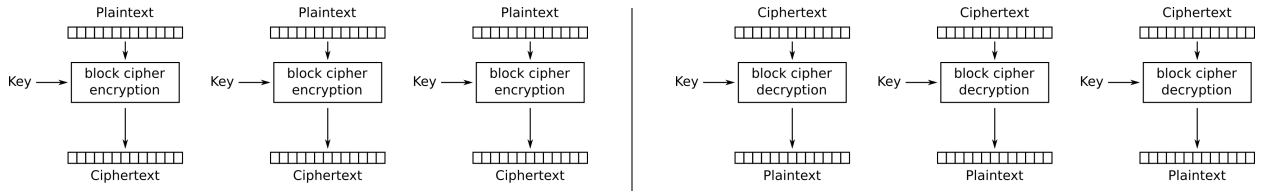
9.2.1 Electronic Codebook (ECB) Mode

$$C_I = E_K(X_i) \;\; i = 1, 2, \ldots \tag{9.6}$$

The steps required to operate a block cipher in ECB mode are illustrated in Figures 9.1a to 9.1b.

9.2.1.1 Problems with Electronic Codebook (ECB) Mode If 2 plaintext blocks are the same, then the 2 corresponding ciphertext blocks are *also* the same. For example,

Notice how one could still make out the outline of the image in Figure 9.2b. This is undesirable, because some information is still leaked to the attacker, even after encryption. Obviously, we need something better than ECB Mode.



(a) Electronic Codebook (ECB) Mode Encryption Steps

(b) Electronic Codebook (ECB) Mode Decryption Steps

Figure 9.1: Electronic Codebook (ECB) Mode Steps

(a) Original Plaintext

(b) ECB Mode Encrypted Plaintext

Figure 9.2: Problem with ECB Mode

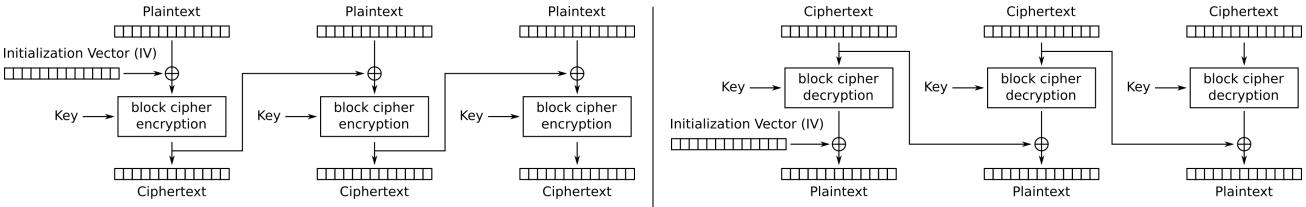
9.2.2 Cipher Block Chaining (CBC) Mode

Defn 143 (Cipher Block Chaining Mode)**.** *Cipher Block Chaining Mode* (*CBC Mode*) follows a different rule.

$$Y_i = E_K(Y_{i-1} \oplus X_i) \tag{9.7}$$

where Y_0 is the Initialization Vector.

The steps required to operate a block cipher in CBC mode are illustrated in Figures 9.3a to 9.3b.



(a) Cipher Block Chaining Mode Encryption Steps

(b) Cipher Block Chaining Mode Decryption Steps

Figure 9.3: Cipher Block Chaining Mode Steps

9.2.3 Counter Mode

Defn 144 (Counter Mode)**.** *Counter Mode* is another Mode of Operation that actually turns our Block Cipher into a Stream Cipher. In this Mode of Operation, the *count* of which block we are working with is encrypted. Then, the encrypted count is XORed with the plaintext message, to produce our ciphertext message.

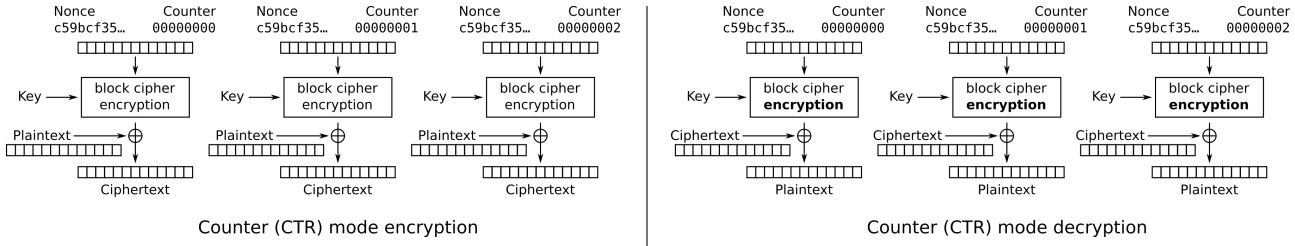
The keystream sequence is given by

$$E_K(0), E_K(1), E_K(2), \ldots \tag{9.8}$$

The steps required to operate a block cipher in counter mode are illustrated in Figures 9.4a to 9.4b.

9.3 Advanced Encryption Scheme

Defn 145 (Advanced Encryption Scheme)**.** The *Advanced Encryption Scheme* (*AES*) cipher is a SP Network Block Cipher. It was originally designed to be fast in both hardware and software. Today, it is faster to perform these calculations in software because CPU architectures support special instructions just for these calculations.



(a) Counter Mode Encryption Steps

(b) Counter Mode Decryption Steps

Figure 9.4: Cipher Block Chaining Mode Steps

In AES, the block size is fixed to 128 bits. The key size is variable; either: 128 bits, 192 bits, or 256 bits. These different key sizes correspond to the the number of rounds that happen (10, 12, or 14, respectively).

AES operates on binary matrices. They are 2×2 matrices where each element is a byte (8 bits). AES has several steps involved in it.

1. The first round is always a round of `AddRoundKey`.
2. Then, all subsequent rounds perform *one* of these actions
   (a) `AddRoundKey`
   (b) `SubBytes`
   (c) `ShiftRows`
   (d) `MixColumns`
   - This is *not* run on the last round.

AES has been broken in practice, but not in a practical sense. The best known attacks were on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys. **Side-Channel attacks are still possible (cache-timing attacks, differential fault analysis, etc.)**.

Remark 145.1 (Other Cryptographic Primitive's Reliance on Advanced Encryption Scheme). Many other Cryptographic Primitives rely on the security of Advanced Encryption Scheme. This means that these derived Cryptographic Primitives are as secure as Advanced Encryption Scheme.

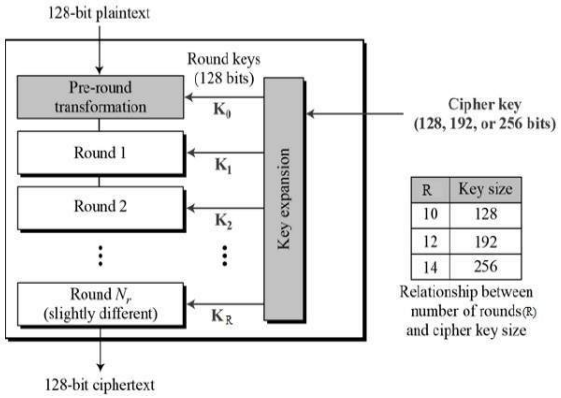


Figure 9.5: Advanced Encryption Scheme Structure

9.3.1 `AddRoundKey`

In the `AddRoundKey` step, each byte of the block is combined with the Round Key using bitwise XOR. Figure 9.6 illustrates this step.

9.3.2 `SubBytes`

The `SubBytes` is a non-linear substitution step where each byte is replaced with another according to a lookup table. Figure 9.7 illustrates this step.

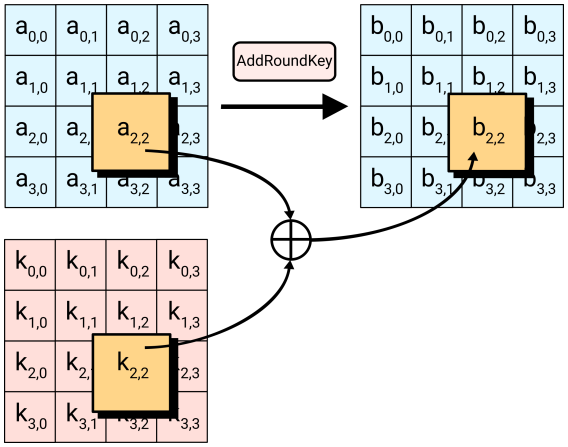


Figure 9.6: Advanced Encryption Scheme `AddRoundKey`

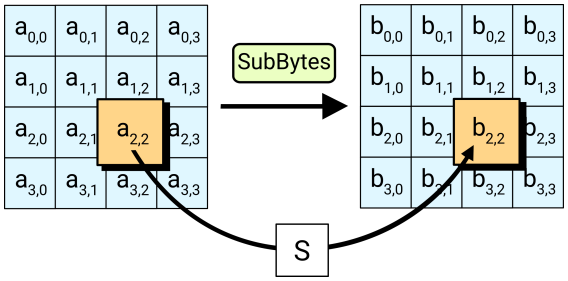


Figure 9.7: Advanced Encryption Scheme `SubBytes`

9.3.3 `ShiftRows`

The `ShiftRows` step is a transposition step where each row of the state is shifted cyclically a certain number of steps. Figure 9.8 illustrates this step.

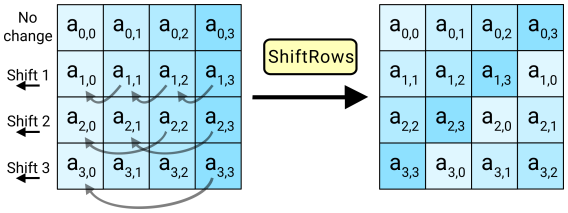


Figure 9.8: Advanced Encryption Scheme `ShiftRows`

9.3.4 `MixColumns`

The `MixColumns` step is a mixing operation which operates on the columns of the block, combining and evenly distributing the 4 bytes in each column. Figure 9.9 illustrates this step.

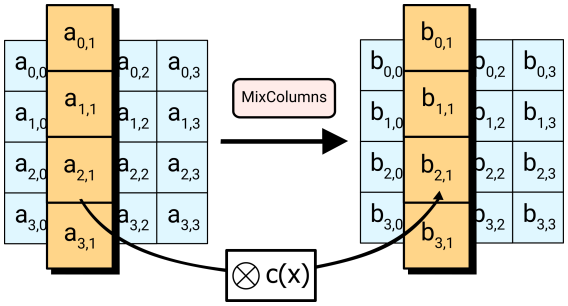


Figure 9.9: Advanced Encryption Scheme `MixColumns`

10 Public-Key Encryption

Defn 146 (Public-Key Encryption Scheme)**.** A public-key encryption scheme is a set of encryption transformations $\{E_e : e \in \mathcal{K}\}$ and a set of decryption transformations $\{D_d : d \in \mathcal{K}\}$. For each $e \in \mathcal{K}$ there is a corresponding $d \in \mathcal{K}$ such that $D_d\big(E_e(M)\big) = M, \forall M$.

Furthermore, after choosing such a pair (e, d), the *public key* e (or the *public parameter*) is made public, while the associated *secret key* d is kept secret. For the scheme to be secure, it must be computationally infeasible to compute d and/or $E_e^{-1}(C)$, knowing the public value e. These types of schemes are built with One-Way Functions and/or Trapdoor One-Way Functions.

This means that the encryption key can be public, allowing anyone to send an encrypted message to the receiver. Then, only the receiver can decrypt the message, because the decryption key is kept secret.

Remark 146.1 (Construction). These are constructed through multiple Trapdoor One-Way Functions.

Defn 147 (One-Way Function)**.** An informal definition of a *one-way function* $f(x)$ is a function from a set $\mathcal{X}$ to a set $\mathcal{Y}$ such that $f(x)$ is easy to compute for all $x \in \mathcal{X}$, but for "essentially all" elements $y \in \mathcal{Y}$ it is "computationally infeasible" to find any $x \in \mathcal{X}$ such that $f(x) = y$.

Remark 147.1 ("Essentially All"). There are some special values where One-Way Functions to not behave normally, in that the function becomes computationally feasible to solve.

Defn 148 (Trapdoor One-Way Function)**.** A *trapdoor one-way function* $f(x)$ is a One-Way Function $f : \mathcal{X} \mapsto \mathcal{Y}$ such that if one knows some specific information T, called the *trapdoor information*, then $f(x)$ is computationally easy to invert f, i.e., for any $y \in \mathcal{Y}$ it is easy to find a $x \in \mathcal{X}$ such that $f(x) = y$. For anyone without knowledge of the trapdoor information T, $f(x)$ is a One-Way Function.

10.1 RSA Public-Key Encryption Scheme

Defn 149 (RSA Public-Key Encryption Scheme)**.** The *RSA public-key encryption scheme* is defined as follows. Let $n = pq$, where p and q are two large Primes. Let $\mathcal{M} = \mathcal{C} = \mathbb{Z}_n$. Pick a number e that is Relatively Prime to $\phi(n)$ (the Set Order of $\mathbb{Z}_n$) and calculate a number d such that $ed = 1 \bmod \phi(n)$. The *public key* is the two numbers (n, e) and the public encryption transformation $E(M)$ is

$$E(M) = M^e \bmod n \tag{10.1}$$

and the associated decryption transformation is

$$D(C) = C^d \bmod n \tag{10.2}$$

RSA Public-Key Encryption Scheme. To verify that the RSA Public-Key Encryption Scheme returns the plaintext after it was encrypted by the public key, we first assume that n, e, and d are all properly defined. By extension, this means that $E(M)$ and $D(C)$ are also well-defined.

We start be substituting the ciphertext, C in Equation (10.2) with the definition of the cipher text.

$$\begin{aligned} D(C) &= C^d \bmod n \\ &= (M^e)^d \bmod n \\ &= M^{ed} \bmod n \end{aligned}$$

Now, we note that $ed = 1 \bmod \phi(n)$, which means we can write

$$ed = 1 + t\phi(n)$$

for some integer t.

So, we continue, and use our two equations together.

$$\begin{aligned} D(C) &= M^{ed} \bmod n \\ &= M^{1+\phi(n)} \bmod n \\ &= M \cdot M^{\phi(n)} \bmod n \end{aligned}$$

From Euler's formula, we know $x^{\phi(n)} = 1$ for any $x \in \mathbb{Z}_n^*$ (From Definition 26 of Multiplicative Inverse). We also assume that M is invertible, otherwise the entire scheme falls apart, since a non-invertible message means it cannot be decrypted once encrypted.

$$\begin{aligned} D(C) &= M \cdot M^{\phi(n)} \bmod n \\ &= (M \cdot 1) \bmod n \\ &= M \bmod n \end{aligned}$$

■

Example 10.1: RSA Public-Key Encryption Scheme. Lecture 12, Example 3

Let $p = 47$ and $q = 167$. Then, $n = pq = 7849$. Compute $\phi(n)$,

$$\begin{aligned}\phi(n) &= \phi(7849)\\ &= \phi(47 \cdot 167)\\ &= (47-1)(167-1)\\ &= 7636\end{aligned}$$

Now, we choose a value of e such that e and $\phi(n)$ are Relatively Prime, $\gcd(e, \phi(n)) = 1$. We choose $e = 25$. (This is just one correct e. There are many more.) Since we chose e where $\gcd(e, \phi(n)) = 1$, e is a Multiplicative Inverse in $\mathbb{Z}_{\phi(n)} = \mathbb{Z}_{7636}$.
We can use the Euclidean Algorithm and Bezout's lemma to find the inverse.

$$d = 2749$$

Thus,

$$25 \cdot 2749 = 1 \bmod 7636$$

Since this is true, we publish our public key, $(n, e) = (7636, 25)$.

Now, to send a message, Alice uses Equation (10.1). Her message has $M = 2728$.

$$\begin{aligned}C &= M^e \bmod n\\ &= 2728^{25} \bmod 7849\\ &= 2401 \bmod 7849\end{aligned}$$

When Bob receives this ciphertext, he can decrypt it using Equation (10.2).

$$\begin{aligned}M &= C^d \bmod n\\ &= 2401^{2749} \bmod 7849\\ &= 2728 \bmod 7849\end{aligned}$$

Thus, Bob gets Alice's original message, and he is likely the only one who was able to decrypt it.

10.1.1 Security of the RSA Public-Key Encryption Scheme

The RSA Public-Key Encryption Scheme relies on the factorization problem. Namely, that for large semi-prime numbers, it is difficult to find its constiuent Prime factors. However, this **does not** mean that breaking RSA is equivalent to solving a factorization problem. It is currently unknown if there is a way to break RSA without factoring the integer n.

10.1.2 Implementation of the RSA Public-Key Encryption Scheme

In order to compute $M^e \bmod n$ we need $L = \lceil \log_2 n \rceil$ bits to store a value from $\mathbb{Z}_n$.

To easily compute the exponentiation of M with e, we apply the *square and multiply algorithm.*

Defn 150 (Square and Multiply Algorithm)**.** The *square and multiply algorithm* is a way to efficiently compute exponents. If the expression in question is M^e, we first rewrite e as a sum of powers of 2.

$$e = e_0 + e_1 \cdot 2 + e_2 \cdot 2^2 + \cdots + e_{L-1} w^{L-1}$$

where $e_i \in \mathbb{Z}_2$ and $0 \leq i \leq L-1$.

Next, $L-1$ squarings of M occurs.

$$M^2, \left(M^4 = \left(M^2\right)^2\right), \left(M^4 = \left(M^4\right)^2\right), \ldots$$

in $\mathbb{Z}_n$ (i.e. reducing modulo n every time).

Lastly, we perform at most $L-1$ multiplications by computing M^e as

$$M^e = M^{e_0} \cdot \left(M^2\right)^{e_1} \cdot \left(M^4\right)^{e_2} \cdots \left(M^{2^{L-1}}\right)^{e_{L-1}}$$

Example 10.2: Square and Multiply Algorithm. Lecture 14, Example 1

Compute 2728^{25} in $\mathbb{Z}_{7849}$ using the Square and Multiply Algorithm?

First, rewrite 25 as a result of products of 2.

$$\begin{aligned} 25 &= 16 + 8 + 1 \\ &= 1(2^4) + 1(2^3) + 0(2^2) + 1(2^1) + 1(2^0) \end{aligned}$$

This results in

$$2728^{25} = 2728^1 \cdot 2728^8 \cdot 2728^{16}$$

Now we perform the squarings required until we get to 2728^{16}.

$$\begin{aligned} 2728^2 &= 7441984 \bmod 7849 = 1132 \\ 2728^4 &= \left(2728^2\right)^2 = 1132^2 \bmod 7849 = 2037 \\ 2728^8 &= \left(2728^4\right)^2 = 2037^2 \bmod 7849 = 5097 \\ 2728^{16} &= \left(2728^8\right)^2 = 5097^2 \bmod 7849 = 7068 \end{aligned}$$

Now substituting our values in and multiplying through (ensuring we reduce modulo n at the end).

$$\begin{aligned} 2728^{25} &= 2728^{16+8+1} \\ &= (2728 \cdot 5097 \cdot 7068) \bmod 7849 \\ &= 2401 \end{aligned}$$

Our answer is $2728^{25} \bmod 7849 = 2401$.

10.2 Primality Testing

How do we check whether a given number m is a Prime number?

The naïve approach said to perform trial divisions such that $x \mid m$ for all integers x where $2 \leq x \leq \sqrt{m}$. The problem with this is that if m is a 1024-bit number, for instance, then $\sqrt{m}$ is 512-bits, and performing 2^{512} tests is not possible.

So, we have probabilistic theorems for determining primality of numbers.

10.2.1 Probabilistic Algorithms for Testing Primality

10.2.1.1 Fermat's Little Theorem

Theorem 10.1 (Fermat's Little Theorem)**.** *Fermat's Little Theorem states that*

$$a^{m-1} = 1 \bmod m \tag{10.3}$$

if m is a Prime and a such that $1 \leq a \leq m-1$.

*When m is not a Prime, we cannot know what we will receive when computing a^{m-1}. However, if $a^{m-1} \neq 1 \bmod m$, we know **for certain** that m is **not** a Prime number.*

Remark (Error Probability of Fermat's Little Theorem). The error probability after repeating the test k times would be less than $\frac{1}{2^k}$.

Defn 151 (Pseudo-Prime)**.** A composite m such that $a^{m-1} = 1 \bmod m$ is said to be a *pseudo-prime* to the base a.

Defn 152 (Carmichael Number)**.** A number is called a *Carmichael number* if a composite m is such that it is a Pseudo-Prime for every base a with $\gcd(a, m) = 1$.

Remark 152.1 (Smallest Carmichael Number). The smallest Carmichael Number is 561.

Theorem 10.2 (Miller-Rabin Test)**.** *The Miller-Rabin test states that we choose a random integer as base, a. Then write $m - 1 = 2^b \cdot q$, where q is odd.*

*If $a^q = 1 \bmod m$ or $a^{2^c \cdot q} = -1 \bmod m$ for any $c < b$ then return "m probably Prime" and go back and choose a new base. Otherwise, return "m is definitely **not** Prime".*

One can prove that the probability that m is probably Prime with to the below equation.

$$\mathrm{P}\left(\text{"}m \text{ probably prime"} \mid \text{"}m \text{ not prime"}\right) < \frac{1}{4} \tag{10.4}$$

10.3 Factoring

All Public-Key Encryption Schemes used today rely on the intractability of factoring large semi-primes.

10.3.1 Pollard's $(p-1)$-Method

Theorem 10.3 (Pollard's $(p-1)$-Method)**.** *Let $n = pq$ and let $p-1$ factor as*

$$p - 1 = q_1 q_2 \cdots q_k$$

where each q_i is a Prime power.

There is a condition where each $q_i < B$ where B is a predetermined "bound". If this condition is valid, then we have

$$(p-1) \mid B!$$

We compute $a = 2^{B!} \bmod n$. Now let $a' = 2^{B!} \bmod p$. Since $p \mid n$, we must have $a \bmod p = a'$.

Fermat's Little Theorem states that

$$2^{p-1} = 1 \bmod p$$

and since $(p-1) \mid B!$, we also have $a' = 1 \bmod p$, which leads to $a = 1 \bmod p$.

So, $p \mid (a-1)$ and since $p \mid n$, we also have $p = \gcd(a-1, n)$.

The algorithm stands as:

1. Compute $a^{2B!} \bmod n$.
2. Compute the unknown prime p as $p = \gcd(a-1, n)$.

An important conclusion to draw here is that the Primes p and q must be chosen such that $p-1$ and $q-1$ each contains a large Prime in their factorization. The usual approach is to generate a random Prime p_1 and test whether $p = 2p_1 + 1$ is a prime number. If so, we choose p.

10.3.2 Other Factoring Methods

There are many modern factoring methods that are more efficient than Pollard's $(p-1)$-Method.

1. Quadratic Sieve
2. Number Field Sieve
3. Elliptic Curve Factorization
4. Pollard's Rho Algorithm Method
5. etc.

10.3.3 Computational Complexity of Factoring

We often use the function to model complexity of algorithms with sub-exponential behavior.

$$L_N(\alpha, \beta) = e^{\left(\beta + O(1)(\log(N))^{\alpha} \log(\log(N))^{1-\alpha}\right)} \tag{10.5}$$

Number field sieves are currently the most successful method for factoring semi-Prime numbers with more than 100 decimal digits. It can factor numbers of the size 2^{512} Its complexity behavior is sub-exponential, with $L_N(\frac{1}{3}, 1.923)$.

10.4 Uses of Public-Key Cryptography

There are 3 basic Cryptographic Primitives that Public-Key Encryption Schemes are used for.

1. Public-Key Encryption Scheme: A messis encrypted with a recipient's public key and cannot be decrypted by anyone except the recipient possessing the private key.
2. Digital Signatures: A message signed with a sender's private key can be verified by anyone who has access to the sender's public key, thereby proving that the correct original sender signed it and that the message has not been tampered with during message transmission (authenticity and nonrepudiation).
3. Key Exchange: A cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key.

The major problem with using a public key is linking it an entity or principal. The most common solution to this problem is the use of a Digital Certificate.

Defn 153 (Digital Certificate)**.** A *digital certificate* is a "stamp of approval" that the public key belongs to who the principal says it belongs to. There is a third-party called a Certificate Authority that handles the signing of entity's/principal's public keys to ensure they are authentic.

A digital certificate has the form of

(Alice, ..., Alice's public key, CA's signature)

Defn 154 (Certificate Authority)**.** A *certificate authority* or (*CA*) is a third-party that vouches for the validity of the public keys in use by an entity/principal.

10.4.1 Digital Certificates and Certificate Authorities

A Certificate Authority-based system works by establishing a "web of trust".

1. All users have a trusted copy of the public key of the **Certificate Authority**. For example, these are embedded in your web browser.
2. The Certificate Authority signs data strings containing the following information.

   (Alice, ..., Alice's public key, CA's signature)

   This data string and the associated signature is called a Digital Certificate. The Certificate Authority only signs the data if it believes the public key really belongs to Alice.
3. When Alice sends her public key, contained in the digital certificate, you now trust that the public key is really Alice's, since you trust the Certificate Authority and you checked their signature too.

10.5 The Discrete Logarithm Problem

Defn 155 (Discrete Logarithm Problem)**.** Let $(G, *)$ be an Abelian Group. The *discrete logarithm problem* states that given $g, h \in G$, find an x (if it exists) such that $g^x = h$.

The difficulty of this problem depends on the group G chosen.

- Very Easy: Polynomial Time Algorithm. $(\mathbb{Z}_n, +)$
- Hard: Sub-Exponential Time Algorithm. $(\mathbb{F}_{p^*})$
- Very Hard: Exponential Time Algorithm. Elliptic Curve Groups (Not discussed in this course).

10.6 Key Exchange

Defn 156 (Key Exchange)**.** *Key exchange* is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key.

10.6.1 Diffie Hellman Key Exchange

Defn 157 (Diffie Hellman Key Exchange)**.** The *Diffie Hellman Key Exchange* is a type of Key Exchange that allows 2 parties to agree to a secret key over an insecure channel without having met before.

Let $G = \mathbb{F}_{p^*}$ and $g \in \mathbb{F}_{p^*}$. The basic message flows for this protocol is:

1. Alice selects a secret a, and Bob selects a secret b.
2. Alice sends $A = g^a \bmod p$ to Bob.
3. Bob sends $B = g^b \bmod p$ to Alice.
4. Alice computes $K_A = B^a \bmod p = g^{b^a} = g^{ba}$.
5. Bob computes $K_B = A^b \bmod p = g^{a^b} = g^{ab}$.
6. If both $K_A = K_B$, then the secret key has been created for Alice and Bob.

Remark 157.1 (Flaws in Diffie Hellman Key Exchange). While this would work for most cases, the entire Diffie Hellman Key Exchange protocol is vulnerable to a *Man-in-the-Middle attack*.

Example 10.3: Diffie Hellman Key Exchange. Lecture 15, Example 1

Given the comain parameters $p = 2147482659$, $g = 2$ and the secret keys chosen by Alice $a = 12345$ and Bob $b = 654323$, compute their shared key?

Alice	Bob
$a = 12345$	$b = 654323$
$A = g^a = 2^{12345} \bmod 2147482659 = 428647416$	$B = g^b = 2^{654323} \bmod 2147482659 = 450904856$
$K_A = B^a \bmod 2147482659$	$K_B = A^b \bmod 2147482659$
$= 450904856^{12345} \bmod 2147482659$	$= 428647416^{654323} \bmod 2147482659$
$= 1333327162$	$= 1333327162$

Thus, Alice and Bob have managed to agree on a secret key of $K = 1333327162$.

11 Hash Functions

Defn 158 (Hash Function)**.** A cryptographic *hash function* h is a function which takes **arbitrary** length bit strings as input and produces a **fixed length** bit string as output, the *hash value*. There are a few requirements for hash functions:

1. It must be a One-Way Function
2. Preimage Resistant
3. Second Preimage Resistant
4. Collision Resistant

Defn 159 (Preimage Resistant)**.** A Hash Function given an output hash value of n bits, is considered *preimage resistant* if the time complexity for finding the input plaintext is $O(2^n)$.

Remark 159.1 (Assumption of Preimage Resistance)*.* Assuming a Hash Function is Preimage Resistant for almost every element of the range of h is a weaker assumption than assuming it is either Second Preimage Resistant or Collision Resistant.

Defn 160 (Second Preimage Resistant)**.** A Hash Function that is *second preimage resistant* is one where given one message, it is difficult to find another message with the same hash value.

Remark 160.1 (Assumption of Second Preimage Resistance)*.* Assuming a Hash Function is Second Preimage Resistant is a weaker assumption than assuming it is Collision Resistant.

Defn 161 (Collision Resistant)**.** A Hash Function is *collision resistant* if it is hard to find 2 plaintext messages with the exact same hash value.

Remark 161.1 (Assumption of Collision Resistance)*.* Assuming a Hash Function is Collision Resistant is the strongest assumption we can make about a well-crafted Hash Function.

11.1 Usages of Hash Functions

There are several reasons to use Hash Functions.

- *Commitment to messages* by disclosing the hash of a message, then later showing the message. This allows the hash to be checked. However, this requires a few different things.
  - If the Hash Function is collision-resistant, you cannot cheat by substituting your original message for another.
- *Verify integrity* of downloaded files
  - Torrents use this to make sure you download the right contents. If something goes wrong, only the right small chunk can be redownloaded.
- *Digital signatures* for things that require confirmation of action from the user/requester.
- *SSL/TLS* for integrity protection
- *Storing passwords* in operating systems (`/etc/shadow` on *nix machines) and websites.
  - You can add salt to change the hash around, usually for webserver passwords. These password databases store the userid, the salt added to the password, and the hashed password. At login time, the password and random number are hashed together again, and compared.

11.2 Merkle-Damgård Construction

Defn 162 (Merkle-Damgård Construction)**.** Since Hash Functions have an essentially infinite domain, due to their arbitrary length input, designing a Hash Function can be quite diffcult. However, if we break the input down into blocks, we can use a *compression function* to map bits from input length s into output hash values of lengths n. These compression functions can be chained together to produce a function that acts on an infinite domain. The chaining method most frequently used by Hash Functions is the *Merkle-Damgård Construction.*

Theorem 11.1. *If the compression function f is Collision Resistant, then the Hash Function h is also Collision Resistant.*

Defn 163 (Length Strengthening)**.** The input message is preprocessed by first padding with zero bits to obtain a message which has length a multiple of l bits. Then a final block of l bits is added which encodes the original length of the unpadded message in bits. The construction is limited to hashing messages with length less than $2l$ bits.

11.3 SHA-1

The internal state of the algorithm is a set of 5 32-bit values.

$$(H_1, H_2, H_3, H_4, H_5)$$

and 4 round constants

$$y_1, y_2, y_3, y_4$$

Length Strengthening is used, but is slightly modified in the SHA-1 algorithm. First, a one bit is appended to the plaintext message, to signal its end. Then zeros are padded until the length of the plaintext message is a multiple of 512-bits. Lastly, the number of bits of the message is appended as a separate, final, block.

Algorithm 11.1: SHA-1 Overview

Input : $(H_1, H_2, H_3, H_4, H_5)$ and (y_1, y_2, y_3, y_4)
Output: Concatenation of $(H_1, H_2, H_3, H_4, H_5)$

1 $(A, B, C, D, E) = (H_1, H_2, H_3, H_4, H_5)$
2 **for** $j = 16$ **to** 79 **do**
3 $\quad X_j = \big((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \lll 1\big)$
4 Execute Round 1 (Algorithm 11.2)
5 Execute Round 2 (Algorithm 11.3)
6 Execute Round 3 (Algorithm 11.4)
7 Execute Round 4 (Algorithm 11.5)

Each of the rounds in the SHA-1 algorithm is slightly different. First, different functions are applied to B, C, D and the round constant is also used in the corresponding round. The functions are defined as:

$$\begin{aligned} \wedge &= \text{AND} \\ \vee &= \text{OR} \\ f(u,v,w) &= (y \wedge v) \vee ((\neg u) \wedge w) \\ g(u,v,w) &= (u \wedge v) \vee (u \wedge w) \vee (v \wedge w) \\ h(u,v,w) &= y \oplus v \oplus w \end{aligned}$$

Algorithm 11.2: SHA-1 Round 1

for $j = 0$ **to** 19 **do**
  $t = (A \lll 5) + f(B, C, D) + E + X_j + y_1$
  $(A, B, C, D, E) = (t, A, B \lll 30, C, D)$

Algorithm 11.3: SHA-1 Round 2

for $j = 20$ **to** 39 **do**
  $t = (A \lll 5) + h(B, C, D) + E + X_j + y_2$
  $(A, B, C, D, E) = (t, A, B \lll 30, C, D)$

Algorithm 11.4: SHA-1 Round 3

for $j = 40$ **to** 59 **do**
  $t = (A \lll 5) + g(B, C, D) + E + X_j + y_3$
  $(A, B, C, D, E) = (t, A, B \lll 30, C, D)$

Algorithm 11.5: SHA-1 Round 4

for $j = 60$ **to** 79 **do**
  $t = (A \lll 5) + h(B, C, D) + E + X_j + y_4$
  $(A, B, C, D, E) = (t, A, B \lll 30, C, D)$

11.4 Security Status of Various Hash Functions

In practice, MD5 and SHA-1 are the most common, but are also broken.

- MD5 is broken in practice
- SHA-1 is broken in theory, no efficient implementation has been developed yet.

There are 2 new versions of the SHA family that have not been broken yet.

1. SHA-2
2. SHA-3
   - Output of an NIST competition in 2012.

11.5 SHA-3

Defn 164 (SHA-3)**.** *SHA-3* uses a sponge construction, where the message blocks are XORed together into the initial bits of the state.

11.6 Message Authentication Codes

Defn 165 (Message Authentication Code)**.** A *message authentication code* is a Hash Function that uses a key.

Remark 165.1 (Relation to Block Ciphers). Many Block Ciphers provide a Mode of Operation to work with Message Authentication Codes.

There are 2 major designs:

1. HMAC (Based on Hash Function)
2. CBC-MAC (Based on Block Cipher in CBC-Mode)

There are 2 naïve implementations of a Message Authentication Codes.

1. $$\mathrm{MAC}_k(m) = h(m \| k)$$
2. $$\mathrm{MAC}_k(m) = h(m \| k)$$

11.6.1 HMAC

HMAC is a MAC based on a Hash Function.

$$\text{HMAC}_k(m) = h((k \oplus) \| h((k \oplus ipad) \| m)) \tag{11.1}$$

- opad = `0x5c5c5c...`
- ipad = `0x363636...`

The implementation in Equation (11.1) was developd in 1996, and when used with MD5 and/or SHA-1, it is immune to previous attacks.

11.6.2 Usages of Message Authentication Codes

- Authenticate origin of messages
  - A symmetric key is shared between the sender and receiver.
  - Both the sender and receiver can create and verify a MAC.

12 Authentication

There are 3 ways we can confirm authenticity with authentication schemes:

1. Unconditionally Secure Authentication Codes
2. Message Authentication Code
3. Digital Signatures

12.1 Message Authentication Codes with Authentication

Message Authentication Codes can be used as an authentication technique that uses Cryptographic Primitives (Block Ciphers and Hash Functions) to provide authentication.

Remark. It is assumed that the sender and reciever both share the same key used for the Message Authentication Code.

12.1.1 Problems with Message Authentication Codes and Authentication

Message Authentication Codes do not protect against an unlimited enemy, i.e. an unlimited number of attackers or an attacker with infinite computing power. However, they are able to authenticate many messages without changing the key.

12.1.2 CBC-MAC for Authentication

CBC-MAC is a Mode of Operation for Block Ciphers to operate with Message Authentication Codes. A Block Cipher is secure for fixed-length messages, but insecure for variable-length messages. The problem is that the same key K cannot be reused anywhere.

12.2 Digital Signatures

Defn 166 (Digital Signature)**.** A *digitial signature* is a way to sign a message to ensure authenticity and that the sender cannot repudiate the message (say they didn't send it). A message signed with a sender's private key can be verified by anyone who has access to the sender's public key. Thereby proving that the correct original sender signed it and that the message has not been tampered with during message transmission (authenticity and nonrepudiation).

To generate the signature:

$$\text{Message} + \text{Alice's Private Key} = \text{Signature} \tag{12.1}$$

To validate Alice's message's signature, ensuring it is from her,

$$\text{Message} + \text{Signature} + \text{Alice's Private Key} = \text{YES/NO} \tag{12.2}$$

There are 3 important properties for security of a digital signature:

(i) *Message Integrity*: The message was not altered **DURING** transit.
(ii) *Message Origin*: The message **WAS** sent by Alice.
(iii) *Nonrepudiation*: Alice cannot claim she did not send **THE** message.

This is an asymmetric (Public-Key Encryption Scheme) solution. The advantages of this are:

- No need to distribute/establish a common secret key
- Nonrepudiation, if the receiver received an authentic message, the sender cannot deny having sending it

The disadvantages of this are:

- Signature schemes rely on the hardness of problems, like semiprime integer factoring.
- These signature schemes also rely on large numbers.
- Both of these factors slow down the process of this technique compared to others.

12.2.1 Security Assumptions for Digital Signatures

The 3 types of assumptions about the types of attacks that we can make are:

1. *Key-Only Attack*: Eve knows the public key and verification algorithm used.
2. *Known Message Attack*: Eve knows a list of signed messages.
3. *Chosen Message Attack*: Eve gets a list of messages of **her** choice signed.

The 3 different goals that an adversary could have are:

1. *Total Break*: The attacker attempts to find the secret signing key used to sign the message.
2. *Selective Forgery*: The attacker attempts to place a valid signature on a **given** message that has not been signed before.
3. *Existential Forgery*: The attacker tries to place a valid signature on **any** message that has not been signed before.

12.3 Authentication Codes

Defn 167 (Authentication Code)**.** An *authentication code* is used to check if the received message was sent by the claimed sender. They are also used to verify that the message was not modified (by a third-party, not random errors) during transmission.

The authentication code requires there be *secret keys* that are known to the sender and receiver, but not to the enemy.

There can be several models of authentication codes. In this course, we only look at The Unconditionally Secure Model. The formula defining this type of authentication code is given in Equation (12.3).

12.3.1 The Unconditionally Secure Model

An unconditionally secure solution is given below.

- The transmitted information, the *source message* (Plaintext) is denoted s and $s \in \mathcal{S}$.
- The source message is mapped to a *channel message* m where $m \in \mathcal{M}$.
- Using the secret key k where $k \in \mathcal{K}$.

$$f : \mathcal{S} \times \mathcal{K} \to \mathcal{M} : (s, k) \mapsto m \tag{12.3}$$

- An important property of f is that if $f(s, e) = m$ and $f(s', e) = m$, then $s = s'$ (Injective for each $k \in Keyspace$).
- Meaning the receiver must check whether a source message s even exists.
- If such an s exists, m is valid.
- Otherwise, the message m is not authentic.

The mapping f, along with $\mathcal{S}$, $\mathcal{M}$, and $\mathcal{K}$ define an Authentication Code (*A-Code*).

12.3.2 Attacks on Authentication Codes

There are only 2 reasonable attacks on Authentication Codes possible.

1. Impersonation Attacks
2. Substitution Attacks

Defn 168 (Probability of Deception)**.** The *probability of deception* is the probability that a non-authentic message is authenticated by the Authentication Code system as a valid message. It is based off probabilities of success of Impersonation Attacks and Substitution Attacks.

$$\mathrm{P}_D = \max(\mathrm{P}_i, \mathrm{P}_S) \tag{12.4}$$

Theorem 12.1 (Square Root Bound)**.** *By first multiplying the Simmons' Bounds on the probabilities of success of each attack, we can find their combined probability. Using Theorem 12.4 and Theorem 12.6 and multiplying:*

$$\mathrm{P}_I \, \mathrm{P}_S \geq 2^{-\mathrm{I}(M;K)-\mathrm{H}(K \mid M)} = 2^{-\mathrm{H}(K)}$$

From $\mathrm{H}(K) \leq \log_2|\mathcal{K}|$ *we can find the* square root bound *of the Probability of Deception* ***for any Authentication Code****.*

$$\mathrm{P}_D \geq \frac{1}{\sqrt{|\mathcal{K}|}} \tag{12.5}$$

Theorem 12.2 (Tightness of the Square Root Bound)**.** *The Square Root Bound can only be tight if*

$$|\mathcal{S}| \leq \sqrt{|\mathcal{K}|} + 1 \tag{12.6}$$

Meaning a large source size demands the key be twice as large. This is not very practical.

12.3.2.1 Impersonation Attacks

Defn 169 (Impersonation Attack)**.** An *impersonation attack* is a form of a Man-in-the-Middle attack where the attacker inserts a message m' into the channel, and hope that it is accepted as authentic.

The equation to determine the probability of success is shown below.

$$\mathrm{P}_I = \max_{m'} P(m' \text{ is valid}) \tag{12.7}$$

Theorem 12.3 (Bounds of Impersonation Attack Success)**.** *For any Authentication Code,*

$$\mathrm{P}_I \geq \frac{|\mathcal{S}|}{|\mathcal{M}|} \tag{12.8}$$

where $|\mathcal{M}| >> |\mathcal{S}|$.

Theorem 12.4 (Simmons' Bound for Impersonation Attack)**.** *For any Authentication Code,*

$$\mathrm{P}_I \geq 2^{-\mathrm{I}(M;E)} \tag{12.9}$$

Meaning, for good protection P_I *is small, we must give away a lot of information about they key.*

12.3.2.2 Substitution Attacks

Defn 170 (Substitution Attack)**.** A *substitution attack* is one in which an attacker observes a valid message sent by the sender, m, and replaces it with their message m', where $m \neq n'$. The attacker then hopes that the system recognizes m' as valid.

The equation to determine the probability of success is shown below.

$$\begin{aligned} \mathrm{P}_S &= \max_{\substack{m,m' \\ m \neq m'}} \mathrm{P}(m' \text{ is valid} \mid m \text{ is valid}) \\ &= \max_{\substack{m,m' \\ m \neq m'}} \frac{|\mathcal{K}(m) \cap \mathcal{K}(m')|}{|\mathcal{K}(m)|} \end{aligned} \tag{12.10}$$

which requires that they keys $k \in \mathcal{K}$ be uniformly distributed.

Example 12.1: Probability of Substitution Attack Success. 2010 Final Exam, Question 2B

In an authentication system, Alice would like to send the source state S given as $S = (s_1, s_2 2)$, where $s_i \in \mathbb{F}_3$, $i = 1, 2$. The key (encoding rule) E_k is given as $E_k = (e_1, e_2)$, where $e_1, e_2 \in \mathbb{F}_3$. The transmitted message M is a 3-tuple generated as $M = (s_1, s_2, t)$, where

$$t = e_1 + s_1 e_2 + s_2 e_2^2$$

.

Find the value of P_S? HINT:

$$\mathrm{P}_S = \max_{\substack{M,M' \\ M \neq M'}} \mathrm{P}(M' \text{ valid} \mid M \text{ valid})$$

To start with, we can expand the equation they give to us.

$$\begin{aligned} \mathrm{P}_S &= \max\left(\frac{|M' \cap M|}{|M|}\right) \\ &= \left(\frac{\text{Num keys for which } \begin{cases} t = k_1 + s_1 k_2 + s_2 k_2^2 \\ t' = k_1 + s_1' k_2 + s_2' k_2^2 \end{cases}}{\text{Num keys for which } t = k_1 + s_1 k_2 + s_2 k_2^2}\right) \end{aligned}$$

We can start by reducing the denominator. If we fix k_2, and since s_i are not variables but values, for any option of k_2, we have a system of linear equations that we can solve, which yields k_1. This means there are 3 possible options for k_2, and the options for k_1 become fixed once k_2 is chosen.

$$\mathrm{P}_S = \frac{\text{Num keys for which } \begin{cases} t = k_1 + s_1 k_2 + s_2 k_2^2 \\ t' = k_1 + s_1' k_2 + s_2' k_2^2 \end{cases}}{3}$$

We reduce the numerator now. To solve the system of linear equations, by performing $t' - t$, then we end up with

$$t' - t = k_1 + (s_1' - s_1)k_2 + (s_2' - s_2)k_2^2$$

Now, k_2 can be varied to anything other than 0, because if we choose $k_2 = 0$, then we lose our message. This means k_2 can take on 2 values, and once it takes on a value, k_1 becomes fixed.
Thus,

$$\mathrm{P}_S = \frac{2}{3}$$

Theorem 12.5 (Bounds of Substitution Attack Success)**.** *For any Authentication Code,*

$$\mathrm{P}_S \geq \frac{|\mathcal{S}| - 1}{|\mathcal{M}| - 1} \tag{12.11}$$

where $|\mathcal{M}| >> |\mathcal{S}|$.

Theorem 12.6 (Simmons' Bound for Substitution Attack)**.** *For any Authentication Code,*

$$\mathrm{P}_S \geq 2^{-\mathrm{H}(K \mid M)}, \text{ if } |\mathcal{S}| \geq 2 \tag{12.12}$$

12.3.3 Constructing Authentication Codes

Define $\mathcal{K}(m)$ as the set of keys for which a message m is valid.

$$\mathcal{K}(m) = \{k \in \mathcal{K}; existss \in \mathcal{S}, f(s, e) = m\} \tag{12.13}$$

12.4 Systematic Authentication Codes

Defn 171 (Systematic Authentication Code)**.** An Authentication Code for which the mapping function $f : \mathcal{S} \times \mathcal{K} \to \mathcal{M}$ can be written in the form

$$f : \mathcal{S} \times \mathcal{K} \to \mathcal{S} \times \mathcal{Z} : (s, k) \mapsto (s, z) \tag{12.14}$$

where $s \in \mathcal{S}$ and $z \in \mathcal{Z}$ is called a *systematic authentication code* (or *Cartesian authentication code*). z is the *tag* (or *authenticator*) and is taken from the tag alphabet $\mathcal{Z}$.

Theorem 12.7 (Bounds of Attacks on Systematic Authentication Codes)**.** *For any Systematic Authentication Code,*

$$\mathrm{P}_S \geq \mathrm{P}_I \tag{12.15}$$

12.4.1 Vector Space Construction

Let $|\mathcal{S}| = q^n$, $|\mathcal{Z}| = q^n$, and $|\mathcal{K}| = q^{2n}$. Decompose the keys as $k = (k_1, k_2)$, where $s, z, k_1, k_2 \in \mathbb{F}_{q^m}$ For the transmission of a source message s, generate a message $m = (s, z)$, where

$$z = k_1 + sk_2$$

12.4.2 Polynomial Evaluation Construction

Let $\mathcal{S} = \{\mathbf{s} = (s_1, s_2, \ldots, s_n); s_i \in \mathbb{F}_q\}$. Define the source message Polynomial to be

$$s(x) = s_1 x + s_2 x^2 + \cdots + s_n x^n$$

. Let $\mathcal{K} = \{k = (k_1, k_2); k_1, k_2 \in \mathbb{F}_q\}$ and $\mathcal{Z} = \mathbb{F}_q$. For the transmission of the source message $\mathbf{s}$, the sender sends $\mathbf{s}$ together with the tag

$$z = k_1 + s(k_2)$$

$s(k_2)$ is the application of the source message polynomial where x is replaced with k_2.

Theorem 12.8 (Systematic Authentication Code Parameters)**.** *This Polynomial Evaluation Construction given a Systematic Authentication Code has parameters*

$$\begin{aligned} |\mathcal{S}| &= q^n \\ |\mathcal{K}| &= q^2 \\ |\mathcal{Z}| &= q \\ \mathrm{P}_I &= \frac{1}{q} \\ \mathrm{P}_S &= \frac{k}{q} \end{aligned} \tag{12.16}$$

A Complex Numbers

Complex numbers are numbers that have both a real part and an imaginary part.

$$z = a \pm bi \tag{A.1}$$

where

$$i = \sqrt{-1} \tag{A.2}$$

Remark (i vs. j for Imaginary Numbers). Complex numbers are generally denoted with either i or j. Since this is an appendix section, I will denote complex numbers with i, to make it more general. However, electrical engineering regularly makes use of j as the imaginary value. This is because alternating current i is already taken, so j is used as the imaginary value instad.

$$Ae^{-ix} = A\left[\cos\left(x\right) + i\sin\left(x\right)\right] \tag{A.3}$$

A.1 Complex Conjugates

If we have a complex number as shown below,

$$z = a \pm bi$$

then, the conjugate is denoted and calculated as shown below.

$$\overline{z} = a \mp bi \tag{A.4}$$

Defn A.1.1 (Complex Conjugate)**.** The conjugate of a complex number is called its *complex conjugate*. The complex conjugate of a complex number is the number with an equal real part and an imaginary part equal in magnitude but opposite in sign.

The complex conjugate can also be denoted with an asterisk ($*$). This is generally done for complex functions, rather than single variables.

$$z^* = \overline{z} \tag{A.5}$$

A.1.1 Complex Conjugates of Exponentials

$$\overline{e^z} = e^{\overline{z}} \tag{A.6}$$

$$\overline{\log(z)} = \log(\overline{z}) \tag{A.7}$$

A.1.2 Complex Conjugates of Sinusoids

Since sinusoids can be represented by complex exponentials, as shown in Appendix B.2, we could calculate their complex conjugate.

$$\begin{aligned} \overline{\cos(x)} &= \cos(x) \\ &= \frac{1}{2}\left(e^{ix} + e^{-ix}\right) \end{aligned} \tag{A.8}$$

$$\begin{aligned} \overline{\sin(x)} &= \sin(x) \\ &= \frac{1}{2i}\left(e^{ix} - e^{-ix}\right) \end{aligned} \tag{A.9}$$

B Trigonometry

B.1 Trigonometric Formulas

$$\sin(\alpha) + \sin(\beta) = 2\sin\left(\frac{\alpha+\beta}{2}\right)\cos\left(\frac{\alpha-\beta}{2}\right) \tag{B.1}$$

$$\cos(\theta)\sin(\theta) = \frac{1}{2}\sin(2\theta) \tag{B.2}$$

B.2 Euler Equivalents of Trigonometric Functions

$$e^{\pm j\alpha} = \cos(\alpha) \pm j\sin(\alpha) \tag{B.3}$$

$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2} \tag{B.4}$$

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j} \tag{B.5}$$

$$\sinh(x) = \frac{e^{x} - e^{-x}}{2} \tag{B.6}$$

$$\cosh(x) = \frac{e^{x} + e^{-x}}{2} \tag{B.7}$$

B.3 Angle Sum and Difference Identities

$$\sin(\alpha \pm \beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta) \tag{B.8}$$

$$\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta) \tag{B.9}$$

B.4 Double-Angle Formulae

$$\sin(2\alpha) = 2\sin(\alpha)\cos(\alpha) \tag{B.10}$$

$$\cos(2\alpha) = \cos^2(\alpha) - \sin^2(\alpha) \tag{B.11}$$

B.5 Half-Angle Formulae

$$\sin\left(\frac{\alpha}{2}\right) = \sqrt{\frac{1-\cos(\alpha)}{2}} \tag{B.12}$$

$$\cos\left(\frac{\alpha}{2}\right) = \sqrt{\frac{1+\cos(\alpha)}{2}} \tag{B.13}$$

B.6 Exponent Reduction Formulae

$$\sin^2(\alpha) = \frac{1-\cos(2\alpha)}{2} \tag{B.14}$$

$$\cos^2(\alpha) = \frac{1+\cos(2\alpha)}{2} \tag{B.15}$$

B.7 Product-to-Sum Identities

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha-\beta) + \cos(\alpha+\beta) \tag{B.16}$$

$$2\sin(\alpha)\sin(\beta) = \cos(\alpha-\beta) - \cos(\alpha+\beta) \tag{B.17}$$

$$2\sin(\alpha)\cos(\beta) = \sin(\alpha+\beta) + \sin(\alpha-\beta) \tag{B.18}$$

$$2\cos(\alpha)\sin(\beta) = \sin(\alpha+\beta) - \sin(\alpha-\beta) \tag{B.19}$$

B.8 Sum-to-Product Identities

$$\sin(\alpha) \pm \sin(\beta) = 2\sin\left(\frac{\alpha \pm \beta}{2}\right)\cos\left(\frac{\alpha \mp \beta}{2}\right) \tag{B.20}$$

$$\cos(\alpha) + \cos(\beta) = 2\cos\left(\frac{\alpha + \beta}{2}\right)\cos\left(\frac{\alpha - \beta}{2}\right) \tag{B.21}$$

$$\cos(\alpha) - \cos(\beta) = -2\sin\left(\frac{\alpha + \beta}{2}\right)\sin\left(\frac{\alpha - \beta}{2}\right) \tag{B.22}$$

B.9 Pythagorean Theorem for Trig

$$\cos^2(\alpha) + \sin^2(\alpha) = 1^2 \tag{B.23}$$

B.10 Rectangular to Polar

$$a + jb = \sqrt{a^2 + b^2}e^{j\theta} = re^{j\theta} \tag{B.24}$$

$$\theta = \begin{cases} \arctan\left(\frac{b}{a}\right) & a > 0 \\ \pi - \arctan\left(\frac{b}{a}\right) & a < 0 \end{cases} \tag{B.25}$$

B.11 Polar to Rectangular

$$re^{j\theta} = r\cos(\theta) + jr\sin(\theta) \tag{B.26}$$

C Calculus

C.1 Fundamental Theorems of Calculus

Defn C.1.1 (First Fundamental Theorem of Calculus)**.** The *first fundamental theorem of calculus* states that, if f is continuous on the closed interval $[a, b]$ and F is the indefinite integral of f on $[a, b]$, then

$$\int_a^b f(x)\,dx = F(b) - F(a) \tag{C.1}$$

Defn C.1.2 (Second Fundamental Theorem of Calculus)**.** The *second fundamental theorem of calculus* holds for f a continuous function on an open interval I and a any point in I, and states that if F is defined by

$$F(x) = \int_a^x f(t)\,dt,$$

then

$$\begin{aligned} \frac{d}{dx}\int_a^x f(t)\,dt &= f(x) \\ F'(x) &= f(x) \end{aligned} \tag{C.2}$$

Defn C.1.3 (argmax)**.** The arguments to the *argmax* function are to be maximized by using their derivatives. You must take the derivative of the function, find critical points, then determine if that critical point is a global maxima. This is denoted as

$$\underset{x}{\operatorname{argmax}}$$

C.2 Rules of Calculus

C.2.1 Chain Rule

Defn C.2.1 (Chain Rule)**.** The *chain rule* is a way to differentiate a function that has 2 functions multiplied together.
If

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

then,

$$\begin{aligned} f'(x) &= g'(x) \cdot h(x) + g(x) \cdot h'(x) \\ \frac{df(x)}{dx} &= \frac{dg(x)}{dx} \cdot g(x) + g(x) \cdot \frac{dh(x)}{dx} \end{aligned} \tag{C.3}$$

D Laplace Transform

Defn D.0.1 (Laplace Transform)**.** The *Laplace transformation* operation is denoted as $\mathcal{L}\{x(t)\}$ and is defined as

$$X(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt \tag{D.1}$$