Midwestern State University

CMPS 4663: Introduction to Cryptography

Course Notes

Professors: Pat and Terry

Fall 2020

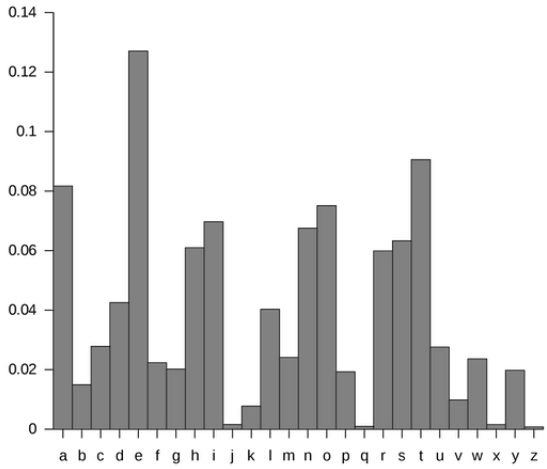
Last Updated: September 28, 2020

Contents

1 Frequency Analysis

1.1 Overview

Frequency analysis put simply is "counting letters". Every language has specific letters that occur more often than others in typical correspondence. These frequencies may vary depending on the context of the communication. English tends to use E, T, A and O the most while using Z, Q, X and J the least. Likewise, TH, ER, ON, and AN are the most common pairs of letters (termed bigrams or digraphs), and SS, EE, TT, and FF are the most common repeats. Having this kind of knowledge allows someone to break a ciphertext if these frequency patterns are preserved somehow.



Frequency Histogram

1.2 Python example

```
import sys
import os
import requests

alphabet = [chr(x+97) for x in range(26)]

class Frequency():
    def __init__(self):
        self.text = ""
        self.freq = {}
        self.sort_freq = None
        self.total = 0

        for l in alphabet:
            self.freq[l] = 0

    def count(self,text):
        for l in text:
            l = l.lower()
```

```
            if l in alphabet:
                self.freq[l] += 1
                self.total += 1

        # https://realpython.com/python-lambda/
        self.sort_freq = sorted(self.freq.items(), key=lambda x: x[1], reverse=True)

    def print(self):
        for i in range(5):
            for j in range(5):
                if j > 0:
                    print(" , ",end="")
                pct = (int(self.sort_freq[i*5+j][1])/self.total)*100
                pct = "{:5.2f}".format(pct)
                print(f"{self.sort_freq[i*5+j][0]}:{pct}%",end=" ")
            print("")

if __name__=='__main__':
    url = "https://www.gutenberg.org/files/2701/2701-0.txt"
    print("Downloading book ...")
    f = requests.get(url)
    text = f.text
    print("Calculating frequency...")
    F = Frequency()
    F.count(text)
    F.print()
```

Below is an example frequency table output from the code snippet (but formatted for this document). Notice the **z** is missing. It's the letter that occurs the least, in this instance, and left off the table as the 26th letter.

e	12.30%	t	9.26%	a	8.16%	o	7.29%	n	6.88%
i	6.87%	s	6.71%	h	6.57%	r	5.52%	l	4.47%
d	4.00%	u	2.80%	m	2.44%	c	2.40%	w	2.32%
g	2.19%	f	2.19%	p	1.84%	y	1.78%	b	1.77%
v	0.90%	k	0.85%	q	0.16%	j	0.12%	x	0.11%

Frequency Table Example

2 Cryptanalysis Hints

2.1 Introduction

Use these hints that the experts use along with a frequency analysis to decipher ciphertext.

As well as the analysis of letter frequencies, other patterns can also be detected that may help to decipher a piece of ciphertext.

The following text explains some of the clues that can be used to deduce a word or a letter in a piece of ciphertext. If you scroll further down the page, you will see a list of tables that explain letter frequencies and patterns in the English language.

2.2 Identify Common Pairs of Letters

If the ciphertext appears to encode a message in English, but the plaintext does not reveal itself immediately, which is often the case, then focus on pairs of repeated letters. In English the most common repeated letters are ss, ee, tt, ff, ll, mm and oo. If the ciphertext contains any repeated characters, you can assume that they represent one of these.

2.3 Identify the Smallest Words First

If the ciphertext contains spaces between words, then try to identify words containing just one, two or three letters. The only one-letter words in English are a and I. The most common two-letter words are of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am. The most common three-letter words are the and and.

2.4 Taylor Made Frequency Tables

If possible, tailor the table of frequencies to the message you are trying to decipher. E.g., military messages tend to omit pronouns and articles, and the loss of words such as I, he, a and they will reduce the frequency of some of the commonest letters. If you know you are tackling a military message, you should use a frequency table generated from other military messages.

2.5 Play The Guessing Game

This can be one of the most useful skills for a cryptanalyst to employ - the ability to identify words, or even entire phrases, based on experience or sheer guesswork. Al-Khalil, an early Arabian cryptanalyst, demonstrated this talent when he cracked a Greek ciphertext. He guessed that the ciphertext began with the greeting 'In the name of God'. Having established that these letters corresponded to a specific section of ciphertext, he could use them as a crowbar to prise open the rest of the ciphertext. This is known as a crib.

2.6 Cryptanalysis Help Table

Letter and word frequencies have been analysed in a number of different languages. A few of the most commonly used ones are listed below, and may help you to decipher your secret messages…

Order Of Frequency Of Single Letters	E T A O I N S H R D L U
Order Of Frequency Of Digraphs	th er on an re he in ed nd ha at en es of or nt ea ti to it st io le is ou ar as de rt ve
Order Of Frequency Of Trigraphs	the and tha ent ion tio for nde has nce edt tis oft sth men
Order Of Frequency Of Most Common Doubles	ss ee tt ff ll mm oo
Order Of Frequency Of Initial Letters	T O A W B C D S F M R H I Y E G L N P U J K
Order Of Frequency Of Final Letters	E S T D N R Y F L O G H A K M P U W
One-Letter Words	a, I
Most Frequent Two-Letter Words	of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am
Most Frequent Three-Letter Words	the, and, for, are, but, not, you, all, any, can, had, her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way, who, boy, did, its, let, put, say, she, too, use
Most Frequent Four-Letter Words	that, with, have, this, will, your, from, they, know, want, been, good, much, some, time

Cryptanalysis Hint Summary[keating]

3 Playfair Cipher

3.1 Overview

The Playfair cipher is a crypto system that encodes two letters at a time (called digraphs). This system was invented around 1854 by Sir Charles Wheatstone. He named the system after a friend, the Baron Playfair, of St Andrews, who lobbies the government for its use. This system was used during World War II and it was used by the British forces in the Boer War.

The playfair cipher uses a 5x5 matrix to encode messages. It populates the matrix with the letters of the alphabet, and since the English language has 26 letters, the letters **I** and **J** are considered the same letter. How the letters are placed in the 5x5 matrix is up to the sender, but it needs to be recreated by the receiver to decrypt the message, so typically a keyword or key-phrase would be used to construct the **playfair square**. Using the keyword or key-phrase method (referred to as ***key*** from now on), you start to fill in the 5x5 matrix with your ***key*** making sure you do not repeat any letters that may be duplicated in your ***key***. As an example if ***key***=**mathematics** it would become: **matheics**. We would place the unique letters from the ***key*** in the beginning of the *playfair square* and the remainder of the matrix gets filled in using the rest of the unused letters in the alphabet.

Playfair example:

M	A	T	H	E
I/J	C	S	R	U
L	W	O	D	B
F	G	K	N	P
Q	X	V	Y	Z

Playfair Square using the phrase: **Mathematics rules the world**

3.2 Preparing the Plaintext

Just like most ancient ciphers we would normally pre-process the text and remove any characters not A-Z. For readability we will leave in spaces, but we also need perform some additional tasks:

1. Separate any double letters with an X.
2. Arrange the message into digraphs (groups of two letters)
3. Finally, if the message has an odd number of letters add an **X** at the end.

Original message:

WHERE IS THE BEEF

Separating double letters:

WHERE IS THE BE**X**EF

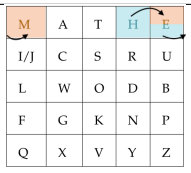
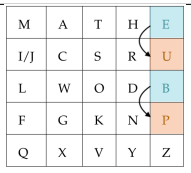
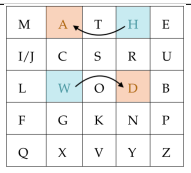
Arranging and fixing odd message length:

WH ER EI ST HE BE XE F**X**

3.3 Rules for Encoding the Message

1. If the letters of a diagraph lie on the same row of the Playfair square, we replace each letter by the letter to its immediate right, wrapping around to the first column is necessary.
2. If the letters of a diagraph lie on the same column of the Playfair square, we replace each letter by the letter directly below, wrapping up the first row if necessary.
3. If the letters are in different rows and columns of the square, this forms the corners of a square inside the Playfair square. We encode each letter with the other corner on the same row.

Examples of Each Rule

Rule 1	Rule 2	Rule 3
		
$H \Rightarrow E\,, E \Rightarrow M$	$B \Rightarrow E\ P \Rightarrow U$	$W \Rightarrow H\ D \Rightarrow A$

Encoding Each Digraph

PlainText:	WH	ER	EI	ST	HE	BE	XE	FX
	↓	↓	↓	↓	↓	↓	↓	↓
CipherText:	DA	HU	MU	OS	EM	PU	ZA	GQ

Resulting Ciphertext

DAHUMUOSEMPUZAGQ

3.4 Rules for Encoding the Message

Apply the same rules for encoding a message, but in reverse.

4 Vigenère cipher

The Vigenère Cipher is a polyalphabetic substitution cipher. The method was originally described by Giovan Battista Bellaso in his 1553 book La cifra del. Sig. Giovan Battista Bellaso; however, the scheme was later incorrectly attributed to Blaise de Vigenère in the 19th century, and is now widely known as the Vigenère cipher.

Blaise de Vigenère actually invented the stronger Autokey cipher in 1586.

The Vigenère Cipher was considered "le chiffre ind hiffrable" (French for the unbreakable cipher) for 300 years, until in 1863 Friedrich Kasiski published a successful attack on the Vigenère cipher. Charles Babbage had, however, already developed the same test in 1854. Gilbert Vernam worked on the Vigenère cipher in the early 1900s, and his work eventually led to the one-time pad, which is a provably unbreakable cipher.

The Algorithm

The algorithm can be stated as follows:

1. Choose a keyword or phrase of any length, and let us call it the ***key*** (*If the key is the same length as the message (plaintext) or longer then you have a one time pad!*).
2. Using an index, $i = 0...n - 1$, where n is the size of the message.
3. Choose the first letter in the plaintext (p_i) and the first letter in the key (k_i).
4. Then use p_i and k_i as indexes, where p_i is a row index and k_i is a column index.
5. Lookup the encrypted letter e_i in our Vigenère Tableau, by identifying the letter e_i at the intersection of Vigenère$[p_i][k_i]$.
6. Continue encrypting the plaintext by increasing the index i from $0...n$ and using p_i, k_i as indexes to find e_i until the entire message is encrypted.
7. If the key is not as long as the message, simply use $i \ mod \ key.length()$ to continue back at the beginning of the key and repeat the letters as needed.

There is an example on the next page.

4.1 Encoding a Message

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenère Tableau

For this example example we will use:

key: "FORTIFICATION"

plaintext: "DEFENDTHEEASTWALLOFTHECASTLE".

- Write out your keyword as many times as you need to match it with the length of the plaintext. Our message is of length 28 and our keyword is length 13, so we print the keyword twice plus the first two letters of the keyword to make 28:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
F O R T I F I C A T I O N F O R T I F I C A T I O N F O
```

- Next we place our message underneath the extended keyword:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
F O R T I F I C A T I O N F O R T I F I C A T I O N F O
D E F E N D T H E E A S T W A L L O F T H E C A S T L E
```

- Now we use an index, $i = 0...n-1$, where $n = 28$ (the size of the message).
- We find our row and column indexes: $p_i = \boldsymbol{D}$ and $k_i = \boldsymbol{F}$
- Find the letter at the intersections of Vigenère[**D**][**F**] which gives us the letter: ***I***
- Continue this process until the entire message is encrypted.
- Below is the result:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
F O R T I F I C A T I O N F O R T I F I C A T I O N F O
D E F E N D T H E E A S T W A L L O F T H E C A S T L E
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
I S W X V I B J E X I G G B O C E W K B J E V I G G Q S
```

4.2 Using Python

```
1  plaintext   = "DEFENDTHEEASTWALLOFTHECASTLE"
2  key         = "FORTIFICATION"
3  ciphertext  = ""
4
5  i = 0
6  for letter in plaintext:
7          a = ord(letter)-97                  # turn ascii into value 0-25
8          b = ord(key[i])-97                  # do same for key
9
10         ciphertext += chr(((a+b)%26)+97)    # add two together modded by 26
11                                             # and add 97 to turn it back into a letter
12
13         i = (i + 1) % len(key)              # move index over mod by length of key
```

Encrypting Message Using Python

To decrypt the ciphertext, do the same thing encrypt, however instead of adding ***a+b*** on line 10l, you subtract ***a-b***.

5 ADFGVX cipher

The ADFGX cipher was a field cipher used by the German Army during World War I. It is closely related to the ADFGVX cipher. ADFGX is a fractionating transposition cipher which combined a modified Polybius square with a single columnar transposition. The cipher is named after the five possible letters used in the ciphertext: A, D, F, G and X. These letters were chosen deliberately because they sound very different from each other when transmitted via morse code. The intention was to reduce the possibility of operator error.

5.1 Building The Polybius Square

The first step in encoding a message with ADFGX is by creating the modified **Polybius square**.

You first choose a "keyword" and write it as if it were in a 5x5 matrix (we'll call this keyword$_1$) and for our example we will use **superbad**.

s	u	p	e	r
b	a	d		

We then fill in the remaining letters in the alphabet (except 'j') to complete the 5x5 square..

s	u	p	e	r
b	a	d	c	f
g	h	i	k	l
m	n	o	q	t
v	w	x	y	z

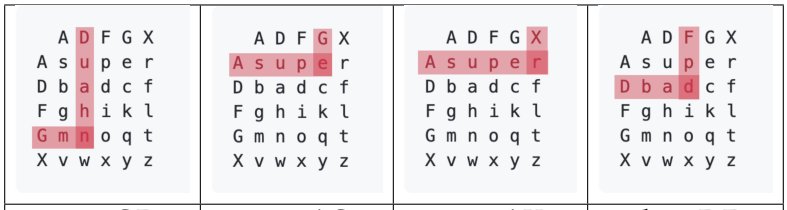
Once we have the "key square" built, we can then add the corresponding "keys" (the letters **ADFGX** for which the cipher is named) at the beginning and top of every row and column. We now have a complete modified **polybius** square. By using a "keyword" to build the square, we can avoid transmitting the entire square to the receiver, and the message receiver can simply build it themselves using their knowledge of keyword$_1$.

	A	**D**	**F**	**G**	**X**
A	s	u	p	e	r
D	b	a	d	c	f
F	g	h	i	k	l
G	m	n	o	q	t
X	v	w	x	y	z

5.2 Using the Polybius Square

The **polybius** square works by locating the letter in the matrix, then pulling out the letter at the beginning of the row, and the letter at the top of the column.

For example, if you wanted to encode **nerd**:



$n \rightarrow GD$	$e \rightarrow AG$	$r \rightarrow AX$	$d \rightarrow DF$
$nerd \rightarrow GDAGAXDF$			

5.3 The Algorithm

5.3.1 Part 1: Polybius Square

We know how to build our *polybius* square and use it to encode a word by turning each letter into a digraph (essentially doubling the length of our plaintext). This doesn't describe the whole process. After building and using the polybius square, there are a few more steps we must apply to fully encrypt the plaintext.

To apply the remaining steps, let us use a slightly bigger example. Actually it's just a single word **discombobulate** (which means *"to confuse someone"*) but it will have enough text to apply the fractionating process (described below). Using our previous steps, this word would get encrypted to:

DF FF AA DG GF GA DA GF DA AD FX DD GX AG

We keep the spaces between each digraph just for ease of reading.

5.3.2 Part 2: Fractionating (aka columnar transposition)

1. Choose yet another keyword (that does not have duplicate letters) (we'll call this keyword$_2$).
2. Write your encoded message below keyword$_2$ as if each letter of keyword$_2$ is a column header.
3. Add the message to the matrix in a row-wise fashion, meaning the keyword ***BUG*** would have a message like ***AAADDDFFFXX*** loaded like the following:

B	**U**	**G**
A	A	A
D	D	D
F	F	F
X	X	

Using the message encrypted from discombobulate, here are two different length keyword$_2$'s loaded.

6 letter key							5 letter key				
H	**I**	**J**	**A**	**C**	**K**		**Q**	**U**	**A**	**R**	**K**
D	F	F	F	A	A		D	F	F	F	A
D	G	G	F	G	A		A	D	G	G	F
D	A	G	F	D	A		G	A	D	A	G
A	D	F	X	D	D		F	D	A	A	D
G	X	A	G				F	X	D	D	G
							X	A	G		

The next step is to perform a columnar transposition. Sort the code word alphabetically, moving the columns as you go. Note that the letter pairs that make up each letter get split apart during this step, this is called ***fractionating***.

6 letter key							5 letter key				
A	**C**	**H**	**I**	**J**	**K**		**A**	**K**	**Q**	**R**	**U**
F	A	D	F	F	A		F	A	D	F	F
F	G	D	G	G	A		G	F	A	G	D
F	D	D	A	G	A		D	G	G	A	A
X	D	A	D	F	D		A	D	F	A	D
G		G	X	A			D	G	F	D	X
							G		X		A

Read the final ciphertext off in columns to get the message that will be sent (spaces kept for readability):

Hijack: FF FX GA GD DD DD AG FG AD XF GG FA AA AD
Quark: FD GA DG AF GD GD AG FF XF GA AD FD AD XA

5.4 Decrypting

To decrypt the message sent using our ***ADFGX*** cipher, we "simply" reverse the steps. Of course the receiver needs keyword$_1$ so they can build the modified Polybius square. This would be used to reverse the lookup process, using the digraphs to find the decrypted letter. However, before that can happen, they need keyword$_2$ to reverse the ***fractionating*** process. Doing all of this using pencil and paper (as it was designed) is much easier. However, programming this is a little more involved.

1. Calculate the number of rows needed in the matrix based on length of keyword$_2$ and the length of the message.

2. Since not all messages are evenly divisible by a chosen keyword, figuring out the number of columns that will end up short is necessary:

```
rows       = math.ceil( float(message_length) / float(len(keyword2) )
short_cols = len(keyword2) - ( message_length % len(keyword2) )
```

3. The number of short columns, s would be assigned to the last s letters in keyword$_2$ (before alphabetizing).
4. Alphabetize the keyword$_2$ (keeping track of which letters have short columns).
5. Load the encrypted message into the matrix in a **column-wise** fashion, filling each column with the specified number of letters (based on short and long columns).
6. Reverse the **columnar transposition** process, by sorting the columns back to their original locations based on the keyword.
7. Reading the letters off in a **row-wise** fashion from the matrix, use each **digraph** to lookup the decrypted letter in the **polybius square** built using keyword$_1$

6 Number Theory stuff

Theorem 6.1. *Given a, b that are both integers then there exists an integer s, t such $a \cdot s + b \cdot t = gcd(a, b)$*

Definition. Let a, b be integers. We say a divides b and write $a|b$ if there exists an integer t such that $b = a \cdot t$.

1. For every $a \neq 0$, $a|0$ and $a|a$https://www.overleaf.com/project/5f4992f5f9a1660001233b80
2. If $a|b$ and $b|c$, then $a|c$
3. If $a|b$ and $a|c$ then $a|b \cdot s + c \cdot t$ for every integers s, t

Definition (Prime)**.** A number $p > 1$ that is divisible only by 1 and itself is called **prime** If n is not prime, we call it **composite**

Theorem 6.2 (Prime Number Thm)**.** *Let $\pi(n)$ denote the number of primes less than n, Then $\pi(n) \approx \frac{n}{\ln}$*

Every integer can be written as the product of primes, unique up to order.

6.1 Canonical prime factorization

$n = p_1^{e^1} p_2^{e^2} p_n^{e^n}$

6.2 Multiplicative Inverse

The sum, difference and product of two integers is an integer. We will take this as an axiom. However the quotient of two integers may not be an integer. We shall start with this operation. The following are two definitions for divisiblity. We will use both of them.

Definition. Let a, b be integers. We say a divides b and write $a|b$ if b/a is an integer.

Definition. Let a, b be integers. We say a divides b and write $a|b$ if there exists an integer t such that $b = at$.

Example: $3|6$ since $6 = 3 \cdot 2$
$7|0$ since $0 = 7 \cdot 0$
$8 \nmid 4$
The first theorem that we present can be proven using either definition, we will present both proofs here for comparison.

Theorem 6.3. *Given integers a, b and c, if $a|b$ and $b|c$, then $a|c$.*

Proof using first definition. By definition of $a|b$, we have that b/a and c/b are both integers. Thus the product

$$\frac{b}{a} \cdot \frac{c}{b} = \frac{c}{a}.$$

This means that $a|c$. □

Proof using second definition. Since $a|b$ there exists an integer k such that $b = ak$. Likewise there exists an integer t such that $c = bt$. Then

$$c = bt = (ak)t = a(kt).$$

Therefore $a|c$. □

Theorem 6.4. *Given integers a, b and c, if $a|b$ and $a|c$, then $a|bx + cy$ for any integers x and y.*

Proof. Since $a|b$, there exists an integer t such that $b = at$. Likewise there exists an integer s such that $c = as$. Then

$$bx + cy = (at)x + (as)y = a(tx + sy).$$

This means that $a|bx + cy$ □

Definition. Let a and b be integers, we say that d is the greatest common divisor (gcd) in case d is the largest of all the integers dividing both a and b, we write $d = (a, b)$

Example: Let $a = 30$ and $b = 42$
The divisors of 30 are $\pm 1, \pm 2, \pm 3, \pm 5, \pm 6, \pm 10, \pm 15, \pm 30$.
The divisors of 42 are $\pm 1, \pm 2, \pm 3, \pm 6, \pm 7, \pm 14, \pm 21, \pm 42$.
The common divisors of 30 and 42 are $\pm 1, \pm 2, \pm 3, \pm 6$
Therefore, $(30, 42) = 6$.
Finding the greatest common divisor by listing all the divisors is certainly one way to accomplish this task. However it is not very efficient. We will demonstrate an far more efficient method later.

Definition. Let a and b be integers with $(a, b) = 1$. In this case we say that a and b are relatively prime.

Definition. Let a and b be integers, we say that m is the least common multiple in case m is the smallest of all the positive integers that are multiples of both a and b, we write $m = [a, b]$

Example: Compute $[15, 10]$ from the definition.
Postive multiples of 15: $15, 30, 45, 60, 75, 90, 105, 120, \ldots$
Positive multiples of 10: $10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, \ldots$
Common multiples of 15 and 10: $30, 60, 90, 120, \ldots$
Thus, we have $[15, 10] = 30$
The next theorem that we shall present you have used ever since grade school when you started learning long division. Although you were probably not presented it in this way.

Theorem 6.5. *(The Division Algorithm) Suppose a and b are integers, $a > 0$. Then there exist unique integers q and r, $0 \leq r < a$, such that $b = aq + r$.*

Proof. First we will show that q and r exist. Suppose that the real number b/a is $q + \epsilon$, where q is an integer and $0 \leq \epsilon < 1$. Then

$$b = a(q + \epsilon) = aq + a\epsilon.$$

From $b = aq + a\epsilon$ if follows that $a\epsilon$ is an integer and $0 \leq a\epsilon < a$. Set $a\epsilon = r$. Thus $b = aq + r$, and $0 \leq r < a$.
Next we show q and r are unique. Suppose

$$b = aq + r, 0 \leq r < a,$$

$$b = aq' + r', 0 \leq r' < a.$$

Subtracting the two equations yields

$$r' - r = a(q - q').$$

Note that $-a < -r \leq 0$ and $0 \leq r' < a$, combining these two inequalities yields $-a < r' - r < a$. If we divide this inequality by a yields

$$-1 < \frac{r' - r}{a} < 1.$$

Since $(r' - r)/a$ is the integer $q - q'$, we have $q - q' = 0$. This implies $q = q'$ and $r = r'$. □

There is an alternate form of the Division Algorithm which we can implement later. The only difference is on the restrictions for r.

Theorem 6.6. *(Fast Division Algorithm) Suppose a and b are integers, $a > 0$. Then there exist unique integers q and r, such that $b = aq + r$, with $|r| \leq \frac{|a|}{2}$.*

Hello agian
At this point we have the definition of the greatest common divisor, however finding the gcd using the list of divisors is very inefficient. To discover a more efficient method we need to explore some properties of the gcd. We start with a useful proposition.

Theorem 6.7. *For any integers $a > 0$, b, c, and k. If $a = bk + c$, then $(a, b) = (b, c)$.*

Proof. Suppose the d divides both a and b. Since $c = a - bk$, it follows that d divides c. Thus d divides both b and c. Conversely, if d divides both b and c then d divides a. Therefore, the common divisors of a and b are the same as the common divisors of b and c. As the gcd of two numbers is the the greatest of their common divisors, $(a, b) = (b, c)$. □

Now we will apply this theorem to the division algorithm. With that added constraints on the remainder given to us in the division algorithm we will get a smaller pair of numbers to compute the gcd.
Example: Suppose we wanted to compute $(451, 143)$. Using the division algorithm we have

$$451 = 3(143) + 22$$

so $(451, 143) = (143, 22)$. Now apply the division algorithm again.

$$143 = 6(22) + 11$$

So $(143, 22) = (22, 11)$. At this point it is clear that the greatest common divisor is 11. However let us continue the process. Apply the division algorithm again.

$$22 = 2(11) + 0$$

So $(22, 11) = (11, 0)$. Since any positive number divides 0, we have $(451, 143) = 11$.
This process is known as the Euclidean Algorithm. The following theorem spells it out.

Theorem 6.8. *(Euclidean Algorithm) Let a and b be integers, $a > 0$. Apply the division algorithm repeatly as follows:*

$$\begin{array}{cc} b = aq_1 + r_1 & 0 < r_1 < a \\ a = r_1q_2 + r_2 & 0 < r_2 < r_1 \\ r_1 = r_2q_3 + r_3 & 0 < r_3 < r_2 \\ \vdots & \vdots \\ r_{n-2} = r_{n-1}q_n + r_n & 0 < r_n < r_{n-1} \\ r_{n-1} = r_nq_{n+1} & \end{array}$$

Let r_n be the last nonzero remainder. Then $(a, b) = r_n$

Recall that we did mention an alternate division algorithm where $0 \leq r < a$ is replaced by the constraint that $|r| \leq \frac{|a|}{2}$. If we apply this version in the Euclidean Algorithm we have what is called as the Fast Euclidean Algorithm.
One use of the Euclidean Algorithm is to write (a, b) in the form $ax + by$. In other words, the Euclidean Algorithm can be used to write (a, b) as a linear combination of a and b. The process is to write out the steps of the algorithm then go backwards.
Example: First compute $(306, 252)$. Use the Euclidean algorithm after each step solve for the remainder.

$$\begin{array}{ccc} 306 = 1(252) + 54 & \rightarrow & 54 = 306 - 1(252) \\ 252 = 4(54) + 36 & \rightarrow & 36 = 252 - 4(54) \\ 54 = 1(36) + 18 & \rightarrow & 18 = 54 - 1(36) \\ 36 + 2(18) + 0 & & \end{array}$$

Now starting with the equation with the last nonzero remainder back substitute using the equations on the right to accomplish the linear combination.

$$\begin{array}{rcll} 18 & = & 54 - 1(36) & \\ & = & 54 - 1(252 - 4(54)) & = 5(54) - 1(252) \\ & = & 5(306 - 1(252)) - 1(252) & = 5(306) - 6(252) \end{array}$$

Thus we have written 18 as a linear combination of 306 and 252.

The following theorem summarizes what we have just done.

Theorem 6.9. *(GCD as a linear combination)*
Leta and b be integers (not both 0). Then (a, b) is the smallest positive value of $ax + by$, where x and y are integers. That is, (a, b) is the least positive integer that is a linear combination of a and b.

Proof. Let d be the least positive integer that is a linear combination of a and b. There is a least such positive integer, by the well-ordering property. We write

$$d = ma + nb,$$

where m and n are integers. We will show that $d|a$ and $d|b$.
By the division algorithm, we have

$$a = dq + r, 0 \leq r < d.$$

We see that

$$r = a - dq = a - q(ma + nb) = (1 - qm)a - qnb.$$

This shows that the integer r is a linear combination of a and$b$. Since $0 \leq r < d$, and d is the least positive linear combination of a and b, we conclude that $r = 0$, and hence $d|a$. In a similar manner, we can show that $d|b$.
We have shown that d, the least positive integer that is a linear combination of a and b, is a common divisor of a and b. It remains to be shown that it is the greatest common divisor. To show this, all we need show is that any common divisor c of a and b must divide d, since any proper positive divisor of d is less that d. Since$d = ma + nb$, if $c|a$ and $c|b$, Theorem 0.4 yields $c|d$, so that $d \geq c$. This concludes the proof. □

7 When is $ax + by = c$ solvable

We have seen previously that we can find integers x and y such that $ax + by = (a, b)$. What if c is a multiple of (a, b)? If $ax + by = (a, b)$ and $c = k(a, b)$, then $a(kx) + b(ky) = c$. Conversely, if $ax + by = c$, then (a, b) must divide c.

Theorem 7.1. *Given integers a, b, and c with a and b not both 0, there exist integers x and y such that $ax + by = c$ if and only if $(a, b)|c$.*

Theorem 7.2. *Let a and b be integers. There exist integers x andy such that $ax + by = 1$ if and only if $(a, b) = 1$.*

Theorem 7.3. *If a, b, and c are integers such that $(a, b) = 1$ and $a|bc$, then $a|c$.*

Proof. Let a and b be integers with $(a, b) = 1$. Then there exists integers x and y such that $ax + by = 1$. Multiplying this equation by c yields $c = acx + bcy$. Since $a|bc$ there exists an integer k such that $bc = ak$. Substituting this into the previous equation, we have

$$c = acx + bcy = acx + aky = a(cx + ky).$$

Thus $a|c$. □

Theorem 7.4. *Let a and b be integers with $(a, b) = d$. Then $(a/d, b/d) = 1$*

Proof. Let a and b be integers with $(a, b) = d$. We will show that a/d and b/d have no common positive divisors other that 1. Assume e is a positive integer such that $e|(a/d)$ and $e|(b/d)$. Then there exist integers k and l such that $a/d = ke$ and $b/d = le$. Multiplying both these equations by d yields $a = dek$ and $b = del$. Thus, de is a common divisor of a and b. Since $d = (a, b)$ we have $de \leq d$, so e must be 1. Consequently, $(a/d, b/d) = 1$. □

8 Congruence

Let $m > 0$ we say that a is congruent to b modulo m and write

$$a \equiv b \pmod{m}$$

in case $m|b-a$. Such a statement is called a congruence and m is called the modulus.
Notice that $a \equiv b \pmod{m}$ if and only if $a = b + km$, for some integer k.
One of the benefits of this definition is writing $m|b-a$ in the form $a \equiv b \pmod{m}$ is because it looks like an equation. In fact congruences share many properties with equations.
We can perform algebra in many similar ways.
If $a \equiv a' \pmod{m}$ and $b \equiv b' \pmod{m}$, then

$$a + b \equiv a' + b' \pmod{m}$$

and

$$ab \equiv a'b' \pmod{m}$$

Proof. By assumption we have $a' = a + km$ and $b' = b + lm$ for some integers k and l. It follows that

$$a' + b' = a + b + (k + l)m,$$

and

$$a'b' = ab + (kb + la + klm)m,$$

thus we have $a + b \equiv a' + b' \pmod{m}$ and $ab \equiv a'b' \pmod{m}$. □

Example: Suppose we wanted to "solve" the congruence

$$2x + 3 \equiv 4 \pmod{5}.$$

First, we add -3 to both sides of the congruence, this yields

$$2x \equiv 1 \pmod{5}.$$

Multiply both sides by 3, the congruence becomes

$$6x \equiv 3 \pmod{5}.$$

Note that $6 \equiv 1 \pmod{5}$, therefore we have

$$x \equiv 3 \pmod{5}.$$

The congruence relation $\equiv \pmod{m}$ is an equivalence relation on $\mathbb{Z}$. That is, the congruence relation satisfies the following.

1. (Reflexivity) $a \equiv a \pmod{m}$ for all a
2. (Symmetry) If $a \equiv b \pmod{m}$, then $b \equiv a \pmod{m}$
3. (Transitivity) If $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$ then $a \equiv c \pmod{m}$

Proof. Since $m|a-a=0$, we have $a \equiv a \pmod{m}$.
If $a \equiv b \pmod{m}$, then $a-b=mk$ for some integer k. Hence $b-a=m(-k)$ and so $b \equiv a \pmod{m}$
If $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$, then $a-b=mk$ and $b-c=ml$ for some integers k and l. Adding the equations, we obtain $(a-b)+(b-c)=mk+ml$. Thus $a-c=m(k+l)$ and so $a \equiv c \pmod{m}$ □

Recall that an equivalence relation on a set partitions the set into equivalence classes. Every element of the set belong to a single equivalence class, and the equivalence class of an element consists of all the elements of the set equivalent to the given element.
We call the equivalence classes formed by the relation $\equiv \pmod{m}$ residue classes modulo m. We will denote the residue class of a as $[a]$, or maybe when there is no confusion simply as a. We denote the set of residue classes modulo m as $\mathbb{Z}_m$. Note that all members of $[a]$ are of the form $a+km$, where $k \in \mathbb{Z}$
Example: Let $m=3$. The equivalence class of 0 consists of numbers a satisfying $a \equiv 0 \pmod{3}$. Thus the equivalence class of 0 is the set of numbers divisible by 3. So,

$$[0] = \{\dots, -6, -3, 0, 3, 6, \dots\}$$

The equivalence class of 1 consists of numbers a satisfying $a \equiv 1 \pmod{3}$. So these numbers have the form $1+3k$.

$$[1] = \{\dots, -5, -2, 1, 4, 7, \dots\}$$

The equivalence class of 2 consists of numbers a satisfying $a \equiv 2 \pmod{3}$. So these numbers have the form $2+3k$.

$$[2] = \{\dots, -4, -1, 2, 5, 8, \dots\}$$

Every integer integer is in one of these classes, so $\mathbb{Z}_3 = \{[0], [1], [2]\}$.
We can generalize this process by making use of the division algorithm. For any integer a, we can find an integer r such that $0 \leq r < m$ and $a \equiv r \pmod{m}$. In addition if $0 \leq a < b < m$, then $b-a$ is not divisible by m and so $[a] \neq [b]$. Combining these two observations we see that $\mathbb{Z}_m = \{[0], [1], \dots, [m-1]\}$.
Any element of an equivalence class is called a representative of the class. We have just noted that there is a unique nonnegative representative less than m. These leads us to the following definition. If $m > 0$ and r is the remainder when the division algorithm is used to divide b by m, then r is called the least residue of b modulo m.
In performing calculations with residue classes it is often more convenient to work with least residues.
A complete residue system modulo m is a collection of integers containing exactly one representative of each residue class.
Example: Two complete residue systems modulo 5 are $\{0, 1, 2, 3, 4\}$ and $\{10, 6, -3, 23, -1\}$
Now that we have the definitions in order let us return to calculations and our bag of algebra tricks.
Example: Find the least residue of $33 \cdot 26^2$ modulo 31.
Note that $33 \equiv 2 \pmod{31}$ and $26 \equiv -5 \pmod{31}$ so

$$33 \cdot 26^2 \equiv 2(-5^2) = 50 \equiv 19 \pmod{31}$$

since $0 \leq 19 < 31$ we have our least residue.
Example: Find the least residue or $3(53)+27^2$ modulo 7.

Note $53 \equiv 4 \pmod 7$ and $27 \equiv -1 \pmod 7$ so

$$3(53) + 27^2 \equiv 3(4) + (-1)^2 = 13 \equiv 6 \pmod 7.$$

One thing to not is that it is true that both sides of a congruence can be multiplied by the same number, division is another story. Note

$$2 \cdot 1 \equiv 2 \cdot 3 \pmod 4$$

but

$$1 \not\equiv 3 \pmod 4.$$

The following theorems will address this issue.
(The cancellation theorem). If $a, b > 0$, x, and x' are integers such that $(a, b) = 1$, then $ax \equiv ax' \pmod b$ implies $x \equiv x' \pmod b$.
This theorem is a special case of the following more general theorem that we will prove.
If $a, b > 0$, d, x, and x' are integers such that $(a, b) = d$, then

$$ax \equiv ax' \pmod b$$

if and only if

$$x \equiv x' \pmod{b/d}.$$

Proof. If $ax \equiv ax' \pmod b$, then b divides $a(x' - x)$. Say $a(x' - x) = bk$. Dividing through by d yields $(a/d)(x' - x) = (b/d)k$. Since $(a/d, b/d) = 1$ we have b/d divides $x' - x$. Thus $x \equiv x' \pmod{b/d}$.
Conversely, if $x \equiv x' \pmod{b/d}$, then $x' - x = (b/d)k$ for some k. Multiplying through by a yields $ax' - ax = b(a/d)k$, which shows that $ax \equiv ax' \pmod b$. □

Here is the divisiblity properties that you may have learn in secondary school in modular terminology.

Theorem 8.1. *(digit theorem) Let $n > 0$ have the decimal representation $a_k a_{k-1} \cdots a_0$. Then*

1. $n \equiv a_0 \pmod 2$
2. $n \equiv a_0 \pmod 5$
3. $n \equiv a_k + \cdots + a_0 \pmod 3$
4. $n \equiv a_k + \cdots + a_0 \pmod 9$
5. $n \equiv a_0 - a_1 + a_2 - \cdots \pmod{11}$

9 Primes

A positive integer is said to be prime in case it has exactly two positive divisors. A positive integer having more than two positive divisors is said to be composite.
In class we went over the Sieve of Eratosthenes. Here we will present a formal statement of the algorithm.
(Sieve of Eratosthenes) Given n, then prime numbers up to n are identified.

1. Let n be a positive integer; set $S = \{2, 3, \ldots, n\}$.
2. For $2 \leq i \leq \lfloor \sqrt{n} \rfloor$, do
   If i has not been crossed out in S, then cross out all proper multiples of i in S.
3. The elements of S that are not crossed out are the prime numbers up to n.

In performing this algorithm we discovered the following theorem.
If n is a composite number, then n has a prime factor less than or equal to $\sqrt{n}$.

Proof. First we will prove the statement that every number greater than 1 has a prime divisor. To accomplish this we will use the strong form of induction, with our base case $n = 2$. Since 2 is a prime divisor of itself, our statement holds. Now we suppose that we know the numbers $2, 3, 4, \ldots, n-1$ all have a prime divisor for some $n > 2$. We must show that n has a prime divisor. If n is itself prime, then n is a prime divisor. If n is not prime, we may write $n = ab$ where $1 < a, b < n$. We now apply our inductive hypothesis to a and find a prime divisor of a, which is necessarily also a prime divisor of n.
Now we show that every composite number n has a prime divisor less than or equal to $\sqrt{n}$. Let p be the smallest prime divisor of n, and suppose by way of contradiction the $p > \sqrt{n}$. If follows that

$$\frac{n}{p} < \frac{n}{\sqrt{n}} = \sqrt{n}.$$

Since n is composite, we know that $n \neq p$ and so $\frac{n}{p}$ is an integer greater than 1. Let q be a prime divisor of this number. We now have

$$q \leq \frac{n}{p} < \sqrt{n} < p,$$

which contradicts our assumption that p was the smallest prime divisor of n. □

We now will move on to the uniqueness of factorization of integers. First we will present an important observation.
Every integer $n > 1$ is either prime or can be written as a product of prime factors.

Proof. We will preceed with strong induction on n. The statement holds for $n = 2$, since 2 is a prime number. Now suppose that the result holds for numbers $2, 3, \ldots, n-1$. Consider the number n. If n is prime, then there is nothing to prove. If n is composite, say, $n = ab$, with $1 < a, b < n$, then we may apply our inductive hypothesis to wrtie a and b as products of primes. Combining these two products together, we obtain a factorization of n □

Every integer $n > 1$ is either prime or can be written as a product of prime factors. Furthermore, such a factorization is unique except for rearrangement of factors.

Proof. Suppose the integer $n > 1$ has two factorizations into primes, say

$$n = p_1 p_2 \dots p_s = q_1 q_2 \dots q_t.$$

Since p_1 divides the right side of the above equation it must divide the left side. Thus p_1 divides q_1 or $q_2 q_3 \dots q_t$. If p_1 divides q_1 then clearly $p_1 = q_1$, if not then p_1 must divide $q_2 q_3 \dots q_t$. Repeat this process we see that $p_1 = q_i$ for some i. Divide the above equation by $p_1 = q_i$ and repeat the process with the remaining p_i. This process will terminate and we see that $s = t$ and the factorization is unique up to order. □

The above theorem allows us to view each integer as the product of primes and allows us the write each integer is the following form $n = p_1^{k_1} p_2^{k_2} \dots p_m^{k_m}$ where the p_i are distinct primes that divide n. This is some times call the canonical prime factorization.
Thinking about an integer in terms of its prime factorization provides an entry to many number-theoretic problems. In addition it might even allow us to calculate things much more efficiently. The next theorem is an example of that.
Suppose a and b are positive integers. Let the distinct primes dividing a or b (or both) be $p_1, p_2, p_3, \dots p_n$. Suppose

$$a = p_1^{j_1} p_2^{j_2} \dots p_n^{j_n}, \quad b = p_1^{k_1} p_2^{k_2} \dots p_n^{k_n}$$

where some of the exponents may be 0. Let m_i be the smaller and M_i the larger of j_i and k_i. Then

1. $a|b$ if and only if $j_i \leq k_i$ for each i.
2. $(a, b) = p_1^{m_1} p_2^{m_2} \dots p_n^{m_n}$.
3. $[a, b] = p_1^{M_1} p_2^{M_2} \dots p_n^{M_n}$.

What does the set of primes look like? How are they distributed? There are several questions that one might ask. The most fundamental fact about the set of primes is proved simply and elegantly in Euclid's "The Elements".
There are infinitely many primes

Proof. Assume to the contrary, that is, there are only a finite number of primes. Hence we can list them, say, $p_1, p_2, p_3, \dots p_n$. Let $N = p_1 p_2 \cdots p_n + 1$. Let q be any prime factor of N. Note that $q \neq p_i$ for any $i \in \{1, 2, \dots, n\}$ because $q|N$ and $q \not| 1$. Hence, q is not in the list. This is a contradiction. Thus there is infinitely many primes. □

Now that we have that established, there is a surprising fact about composite numbers.
Given any $n > 1$, there exist n consecutive composite numbers.

Proof. Let $a = (n+1)! + 2$. Then each of the numbers $a, a+1, \dots, a+(n-1)$ is composite, because, for each $k = 0, 1, \dots, n-1$, it follows that $k+2$ divides $(n+1)! + 2 + k = a + k$. □

There are many unanswered questions concerning primes, the following two conjectures are some famous ones.
(Twin Primes) There are infinitely many primes p for which $p + 2$ is also prime. The pair $p, p+2$ are known as twin primes.
(Goldbach, 1742) Every even number greater than 2 is the sum of two primes.
Here are a couple of theorems about primes that we will not prove. However feel free to use them if the need arises.

(Dirichlet) If $(a, b) = 1$, then there are infinitely many primes of the form $a + bn$.
Suppose we want to know how many primes there are less than or equal to a given integer n. We will denote this by $\pi(n)$. For example $\pi(6) = 3$ and $\pi(7) = 4$. How does the function behave in general? The following theorem states that a good approximation in the function $\frac{x}{\log x}$.
(Prime number theorem)

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\log x} = 1$$

10 Solving Linear Congruence

To "solve" the equation $x^2 + x - 2 = 0$ we mean to find all the values of x that make the equation true. We will consider the similar problem where the equality is replaced by $\equiv$. In general we will restrict our attention to congruences of the form

$$F(x) \equiv 0 \pmod{b},$$

where $F(x)$ is a polynomial with integer coefficients.
Suppose x_1 satisfies the congruence

$$x^2 + x - 2 \equiv 0 \pmod{7},$$

and suppose that $x_1 \equiv x_2 \pmod{7}$. Then we have

$$x_2^2 + x_2 - 2 \equiv x_1^2 + x_1 - 2 \equiv 0 \pmod{7},$$

and so x_2 is also a solution to the congruence. Thus if we find one solution to a congruence we immediately have infinitely many solutions. Namely, all integers congruent to x modulo 7.
Let x_1, x_2 and $b > 0$ be integers with $x_1 \equiv x_2 \pmod{b}$, and let F be a polynomial with integer coefficients. Then x_1 is a solution to $F(x) \equiv 0 \pmod{b}$ if and only if x_2 is also a solution.
For completeness sake recall the definitions of a congruence class modulo b and a complete residue system modulo b.
We call $x_1, x_2, \ldots, x_k$ a complete solution to the congruence in case it is the set of all solutions in some complete residue system modulo b. It is the least complete solution in case it is the set of all solutions among $0, 1, \ldots, b-1$.
Consider the congruence $4x - 18 \equiv 0 \pmod{6}$. Among $1, 2, 3, 4, 5, 6$ exactly $3, 6$ satisfy the congruence. Thus $3, 6$ is a complete solution. Another complete solution $x = -3, 0$ or $x = 9, 72$. The least complete solution is $x = 0, 3$.
To solve a linear congruence $ax \equiv c \pmod{b}$, we first note that by the definition of congruence we have that $b|c - ax$ or equivalently we have $c - ax = by$ for some integer y. Recall that we have already solved linear diophantine equations of the form $ax + by = c$, so we will use that process now.

1. $ax + by = c$ has a solution if and only if $(a, b)|c$
2. A solution can be found using the Euclidean Algorithm.
3. Id x_0, y_0 are particular solutions, then all solutions are given by $x = x_0 + \frac{bt}{d}$ and $y = y_0 - \frac{at}{d}$ where t runs through the integers and $d = (a, b)$.

Example: Solve the congruence $147x \equiv 77 \pmod{161}$. First we use the euclidean algorithm.

$$161 = 1 \cdot 147 + 14$$

$$147 = 10 \cdot 14 + 7$$

$$14 = 2 \cdot 7 + 0$$

Back substitute to find the linear combination.

$$7 = 147 - 10 \cdot 14 = 147 - 10(161 - 147) = 11 \cdot 147 - 10 \cdot 161$$

Multiplying both sides of the equation by 11 we have

$$77 = 121 \cdot 147 - 110 \cdot 161$$

thus a particular solution is $x_0 = 121$. Our general solution is $x = 121 + \frac{161t}{7} = 121 + 23t$, where t is any integer.
At this point we need to determine how we find a complete solution or a least complete solution from the above general solution. The next Lemma and theorem will answer this question. First we will present a useful lemma that we will need to prove the theorem.
If a, b, c and m are integers such that $m > 0$, $d = (c, m)$, and $ac \equiv bc \pmod{m}$, then $a \equiv b \pmod{\frac{m}{d}}$

Proof. If $ac \equiv bc \pmod{m}$, we know that $m|(ac - bc) = c(a - b)$. Hence, there is an integer k with $c(a - b) = km$. By dividing both sides by d, we have $(\frac{c}{d}(a - b) = k(\frac{m}{d})$. Because $(\frac{m}{d}, \frac{c}{d}) = 1$, it follows that $\frac{m}{d}|(a - b)$, hence $a \equiv b \pmod{\frac{m}{d}}$ □

Let a, b and m be integers such that $m > 0$ and $(a, m) = d$. If $d \nmid b$, then $ax \equiv b \pmod{m}$ has no solutions. If $d|b$, then $ax \equiv b \pmod{m}$ has exactly d incongruent solutions modulo m.

Proof. Note $ax \equiv b \pmod{m}$ is equivalent to the linear diophantine in two variables $ax - my = b$. The integer x is a solution of $ax \equiv b \pmod{m}$ if and only if there is an integer y such that $ax - my = b$, we have seen if $d \nmid b$, there is no solution, whereas if $d|b$, $ax - my = b$ has infinitely many solutions, given by $x = x_0 + (\frac{m}{d})t$, $y = y_0 - (\frac{a}{d})t$ where x_0 and y_0 is a particular solution. Note $x = x_0 + (\frac{m}{d})t$ are the solutions to the linear congruence. To determine how many incongruent solutions there are, we find the condition that describes when two of the solutions $x_1 = x_0 + (\frac{m}{d})t_1$ and $x_2 = x_0 + (\frac{m}{d})t_2$ are congruent modulo m.

$$x_0 + \left(\frac{m}{d}\right) t_1 \equiv x_0 + \left(\frac{m}{d}\right) t_2 \pmod{m}$$

$$\left(\frac{m}{d}\right) t_1 \equiv \left(\frac{m}{d}\right) t_2 \pmod{m}$$

Now $(m, \frac{m}{d}) = \frac{m}{d}$ since $(\frac{m}{d})|m$, so by the previous lemma

$$t_1 \equiv t_2 \pmod{d}.$$

This shows that a complete set of incongruent solutions is obtained by taking $x = x_0 + (\frac{m}{d})t$, where t ranges through a complete system of residues modulo d, one such set $x = x_0 + (\frac{m}{d})t$, where $t = 0, 1, 2, \ldots d - 1$. □

Now let us return back to our example

$$147x \equiv 77 \pmod{161},$$

for which we have found a particular solution $x = 121$. We also found that $(a, b) = (147, 161) = 7$. Then a complete solution to the congruence is given by

$$x = 121 + \frac{161t}{7} = 121 + 23t,$$

where t runs through any complete residue system modulo 7. Letting $t = 0, 1, 2, 3, 4, 5, 6$ yields the complete solution

$$x = 121, 144, 167, 190, 213, 236, 259.$$

A least complete solutions can be obtained by subtracting 161 from any of the representatives that are greater than 161, so a least complete solution $x = 121, 144, 6, 29, 52, 75, 98$.

11 Chinese Remainder Theorem

Suppose a and b relatively prime integers. We can choose x so that ax is in any congruence class we want modulo b. We can choose y so that by is in any congruence class we want modulo a. We will use this symmetry to produce so useful results.
Changing notation le b_1 and b_2 be relatively prime integers. We can find x_1 such that

$$b_2 x_1 \equiv 1 \pmod{b_1}$$

then $b_2 x_1 c_1 \equiv c_1 \pmod{b_1}$ where c_1 is completely arbitrary.
Likewise, we can find x_2 such that

$$b_1 x_2 \equiv 1 \pmod{b_2}$$

then $b_1 x_2 c_2 \equiv c_2 \pmod{b_2}$ where c_2 is completely arbitrary.
The important implication of the above congruence is taht we can make the sum $z = b_2 x_1 c_1 + b_1 x_2 c_2$ simultaneously congruent to whatever we want modulo both b_1 and b_2.
For

$$z = b_2 x_1 c_1 + b_1 x_2 c_2 \equiv b_2 x_1 c_1 + 0 \equiv c_1 \pmod{b_1}$$

and

$$z = b_2 x_1 c_1 + b_1 x_2 c_2 \equiv 0 + b_1 x_2 c_2 \equiv c_2 \pmod{b_2}.$$

The ability to find an integer z such that

$$z \equiv c_1 \pmod{b_1} \text{ and } z \equiv c_2 \pmod{b_2}$$

where c_1 and c_2 are arbitrary, has many applications.
(Chinese Remainder Theorem) Let $m_1, m_2, \ldots m_r$ be pairwise relatively prime positive integers. Then the system of congruences

$$\begin{aligned} x_1 &\equiv a_1 \pmod{m_1} \\ x_2 &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x_r &\equiv a_r \pmod{m_r} \end{aligned}$$

has a unique solution modulo $M = m_1 m_2 \ldots m_r$.

Proof. First, we construct a simultaneous solution to the system of congruences. Let $M_k = M/m_k$, we know $(M_k, m_k) = 1$ because $(m_j, m_k) = 1$ whenever $j \neq k$. Hence we can find an inverse y_k of M_k modulo m_k so that $M_k y_k \equiv 1 \pmod{m_k}$. Now form the sum

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \cdots + a_r M_r y_r.$$

The integer x is a simultaneous solution of the r congruences. To demonstrate this, we must show that $x \equiv a_k \pmod{m_k}$ for $k = 1, 2, \ldots, r$. Since $m_k | M_j$ whenever $j \neq k$, we have $M_j \equiv 0 \pmod{m_k}$. Therefore, in the sum for x, all the terms except the k^{th} term are congruent to $0 \pmod{m_k}$. Hence $x \equiv a_k M_k y_k \equiv a_k \pmod{m_k}$, since $M_k y_k \equiv 1 \pmod{m_k}$. Now we need to show that any two solutions are congruent modulo M. Let x_0 and x_1 both be simultaneous solutions to the system of r congruences. Then, for each k, $x_0 \equiv x_1 \equiv a_k \pmod{m_k}$, so that $m_k | (x_0 - x_1)$. We see $M | (x_0 - x_1)$. †Therefore, $x_0 \equiv x_1 \pmod{M}$.

□

†This comes from the theorem: If a, b, c are integers, then $[a, b] | c$ if and only if $a | c$ and $b | c$.
Example: Solve the system of congruences.

$$x \equiv 1 \pmod{3}$$

$$x \equiv 2 \pmod{5}$$

$$x \equiv 3 \pmod{7}$$

To solve this we will follow the proof of the previous theorem. We have $M = 3 \cdot 5 \cdot 7 = 105$, $M_1 = 105/3 = 35$, $M_2 = 105/5 = 21$, and $M_3 = 105/7 = 15$.
To determine y_1, we solve $35y_1 \equiv 1 \pmod{3}$ which yields $y_1 \equiv 2 \pmod{3}$.
To determine y_2, we solve $21y_2 \equiv 1 \pmod{5}$ which yields $y_2 \equiv 1 \pmod{5}$.
To determine y_3, we solve $15y_3 \equiv 1 \pmod{7}$ which yields $y_3 \equiv 1 \pmod{7}$.

Hence, $x \equiv 1 \cdot 35 \cdot 2 + 2 \cdot 21 \cdot 1 + 3 \cdot 15 \cdot 1 \equiv 157 \equiv 52 \pmod{105}$
The following example give a recursive way of solving a CRT problem.
Example: Solve the system of congruences.

$$x \equiv 1 \pmod{5}$$

$$x \equiv 2 \pmod{6}$$

$$x \equiv 3 \pmod{7}$$

Start with the congruence with the highest modulus (although this is not necessary it does help when solving the linear congruence with decreasing moduli)
Note $x \equiv 3 \pmod{7}$ implies that $x = 7t + 3$, for some integer t.
Insert this into the second congruence.

$$7t + 3 \equiv 2 \pmod{6}$$

$$7t \equiv -1 \pmod{6}$$

$$t \equiv 5 \pmod{6}$$

thus $t = 6u + 5$ for some integer u.
Hence $x = 7t + 3 = 7(6u + 5) + 3 = 42u + 35 + 3 = 42u + 38$.

Take this and insert it into the first congruence,

$$42u + 38 \equiv 1 \pmod 5$$

$$42u \equiv -37 \pmod 5$$

$$2u \equiv 3 \pmod 5$$

$$u \equiv 4 \pmod 5$$

thus $u = 5v + 4$ for some integer v
Hence $x = 42u + 38 = 42(5v + 4) + 38 = 210v + 168 + 38 = 210v + 206$. So $x \equiv 206 \pmod{210}$

12 Math Review for Exam 1

Here are some definitions that you should know.

Definition. Let a, b be integers, not both zero. The greatest common divisor of a and b is the largest positive divisor of both a and b, we denote this by $\gcd(a, b)$,

Definition. If the $\gcd(a, b) = 1$, then a and b are relatively prime.

Definition. Let a, b, n be integers with $n \neq 0$. We say a is congruent to b modulo n, denoted $a \equiv b \pmod{n}$ if $n|a - b$.

Definition. Let $\phi(n)$ be the number of integers $1 \leq a \leq n$ such that $\gcd(a, n) = 1$. The function ϕ is called Euler's ϕ-function.

Recall: $\phi(n) = n \prod_{p|n}(1 - \frac{1}{p})$
Example: Suppose $n = 30 = 2 \cdot 3 \cdot 5$ then $\phi(30) = 30(1 - \frac{1}{2})(1 - \frac{1}{3})(1 - \frac{1}{5}) = 8$
According to the above formula, it is worth noting that if $n = pq$ where p and q are prime then $\phi(n) = (p - 1)(q - 1)$. Also if $n = p^s$ then $\phi(n) = p^s - p^{s-1}$.

Theorem 12.1. *(Euclidean Algorithm) Let a and b be integers, $a > 0$. Apply the division algorithm repeatly as follows:*

$$\begin{array}{cc} b = aq_1 + r_1 & 0 < r_1 < a \\ a = r_1q_2 + r_2 & 0 < r_2 < r_1 \\ r_1 = r_2q_3 + r_3 & 0 < r_3 < r_2 \\ \vdots & \vdots \\ r_{n-2} = r_{n-1}q_n + r_n & 0 < r_n < r_{n-1} \\ r_{n-1} = r_nq_{n+1} & \end{array}$$

Let r_n be the last nonzero remainder. Then $(a, b) = r_n$

Method of successive squaring.

Suppose you want to calculate $2^{1}234 \pmod{789}$. To do this you are going to repeated squaring.

$$2^2 \equiv 4 \pmod{789}$$
$$2^4 \equiv 4^2 \equiv 16 \pmod{789}$$
$$2^8 \equiv 16^2 \equiv 256 \pmod{789}$$
$$2^{16} \equiv 256^2 \equiv 49 \pmod{789}$$
$$2^{32} \equiv 46^2 \equiv 34 \pmod{789}$$
$$2^{64} \equiv 34^2 \equiv 367 \pmod{789}$$
$$2^{128} \equiv 367^2 \equiv 559 \pmod{789}$$
$$2^{256} \equiv 559^2 \equiv 37 \pmod{789}$$
$$2^{512} \equiv 37^2 \equiv 580 \pmod{789}$$
$$2^{1024} \equiv 580^2 \equiv 286 \pmod{789}$$

Note that $1234 = 1024 + 128 + 64 + 16 + 2$, so we have

$$2^{1234} \equiv 286 \cdot 559 \cdot 367 \cdot 49 \cdot 4 \equiv 481 \pmod{789}$$

Theorem 12.2. *(Fermat's Little Theorem) If p is a prime and p does not divide a, then $a^{p-1} \equiv 1 \pmod{p}$.*

Theorem 12.3. *(Euler's Theorem) If $gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.*

Possible types of questions for Exam 1

1. Use the Euclidean algorithm to calculate the gcd of two integer.
2. Use the Euclidean algorithm to verify that two integers are relatively prime.
3. Calculate the Euler ϕ-function with various integers.
4. Given integers a, b and $n \neq 0$, find integers x and y such that $ax + by = \gcd(a, b)$. To do this use the Euclidean algorithm to find the gcd. Then back solve.
5. Given integers a, n with $\gcd(a, n) = 1$ find $a^{-1} \pmod{n}$. To accomplish this calculate the gcd and then back solve and x is the multiplicative inverse.
6. Solve a linear congruence $ax + b \equiv \pmod{m}$.
7. Calculate $a^n \pmod{m}$ using successive squaring and/or Fermat's little theorem and/or Euler's theorem.

13 Vocabulary

- **Access control**
  A method of restricting access to resources, allowing only privileged entities access.
- **AES (Advanced Encryption Standard)**
  NIST approved standards, usually used for the next 20 to 30 years.
- **Affine Cypher**
  The affine is a type of monoalphabetic substitution cipher, where each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. The formula used means that each letter encrypts to one other letter, and back again, meaning the cipher is essentially a standard substitution cipher with a rule governing which letter goes to which. As such, it has the weaknesses of all substitution ciphers. Each letter is enciphered with the function (ax + b) mod 26, where b is the magnitude of the shift.
- **Algorithm (encryption)**
  A set of mathematical rules (logic) used in the processes of encryption and decryption.
- **Algorithm (hash)**
  A set of mathematical rules (logic) used in the processes of message digest creation and key/signature generation.
- **Anonymity**
  Of unknown or undeclared origin or authorship, concealing an entity's identification.
- **Asymmetric keys**
  A separate but integrated user key-pair, comprised of one public key and one private key. Each key is one way, meaning that a key used to encrypt information can not be used to decrypt the same data.
- **Authentication**
  To prove genuine by corroboration of the identity of an entity.
- **Block cipher**
  A symmetric cipher operating on blocks of plain text and cipher text, usually 64 bits.
- **Blowfish**
  A 64-bit block symmetric cipher consisting of key expansion and data encryption. A fast, simple, and compact algorithm in the public domain written by Bruce Schneier.
- **Cipher Text**
  The result of manipulating either characters or bits via substitution, transposition, or both.
- **Clear Text** (Plain Text)
  Usually refers to data that is transmitted or stored unencrypted (' in clear ')
- **Coprime Integers**
  In number theory, two integers a and b are said to be relatively prime, mutually prime,[1] or coprime (also written co-prime) if the only positive integer (factor) that divides both of them is 1. Consequently, any prime number that divides one of a or b does not divide the other. This is equivalent to their greatest common divisor (gcd) being 1.

- **Credentials**
  Something that provides a basis for credit or confidence.

- **Cryptanalysis**
  The art or science of transferring cipher text into plain text without initial knowledge of the key used to encrypt the plain text.

- **Cryptographic hash function**
  A cryptographic hash function (CHF) is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit array of a fixed size (the "hash value", "hash", or "message digest"). It is a one-way function, that is, a function which is practically infeasible to invert. Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes. Cryptographic hash functions are a basic tool of modern cryptography.

- **Cryptography**
  The art and science of creating messages that have some combination of being private, signed, unmodified with non-repudiation.

- **Cryptosystem**
  A system comprised of cryptographic algorithms, all possible plain text, cipher text, and keys.

- **Data integrity**
  A method of ensuring information has not been altered by unauthorized or unknown means.

- **Decryption**
  The process of turning cipher text back into plain text. DES (Data Encryption Standard) A 64-bit block cipher, symmetric algorithm also known as Data Encryption Algorithm (DEA) by ANSI and DEA-1 by ISO. Widely used for over 20 years, adopted in 1976 as FIPS 46.

- **Dictionary Attack**
  A calculated brute force attack to reveal a password by trying obvious and logical combinations of words.

- **Diffie-Hellman**
  The first public key algorithm, invented in 1976, using discrete logarithms in a finite field.

- **Discrete logarithm**
  The underlying mathematical problem used in/by asymmetric algorithms, like Diffie-Hellman and Elliptic Curve. It is the inverse problem of modular exponentiation, which is a one-way function.

- **Digital signature**
  An electronic identification of a person or thing created by using a public key algorithm. Intended to verify to a recipient the integrity of data and identity of the sender of the data.

- **Encryption**
  The process of disguising a message in such a way as to hide its substance.

- **Entropy**
  A mathematical measurement of the amount of uncertainty or randomness.

- **Fingerprint**
  A unique identifier for a key that is obtained by hashing specific portions of the key data.

- **Hash function**
  A one-way hash function—a function that produces a message digest that cannot be reversed to produced the original.

- **Integrity**
  Assurance that data is not modified (by unauthorized persons) during storage or transmittal.

- **Key**
  A means of gaining or preventing access, possession, or control represented by any one of a large number of values.

- **Key exchange**
  A scheme for two or more nodes to transfer a secret session key across an unsecured channel.

- **Key length**
  The number of bits representing the key size; the longer the key, the stronger it is.

- **Key management**
  The process and procedure for safely storing and distributing accurate cryptographic keys; the overall process of generating and distributing cryptographic key to authorized recipients in a secure manner.

- **Key splitting**
  A process for dividing portions of a single key between multiple parties, none having the ability to reconstruct the whole key.

- **MD5 (Message Digest 5)**
  Improved, more complex version of MD4, but still a 128-bit, one-way hash function.

- **Message digest**
  A number that is derived from a message. Change a single character in the message and the message will have a different message digest.

- **One-time pad**
  A large non-repeating set of truly random key letters used for encryption, considered the only perfect encryption scheme, invented by Major J. Mauborgne and G. Vernam in 1917.

- **One-way hash**
  A function of a variable string to create a fixed length value representing the original pre-image, also called message digest, fingerprint, message integrity check

- **Passphrase**
  An easy-to-remember phrase used for better security than a single password; key crunching converts it into a random key.

- **Password**
  A sequence of characters or a word that a subject submits to a system for purposes of authentication, validation, or verification.

- **Perfect forward secrecy**
  A cryptosystem in which the cipher text yields no possible information about the plain text, except possibly the length.

- **Pretty Good Privacy (PGP)**
  An application and protocol (RFC 1991) for secure e-mail and file encryption developed by Phil R. Zimmermann. Originally published as Freeware, the source code has always been available for public scrutiny. PGP uses a variety of algorithms, like IDEA, RSA, DSA, MD5, SHA-1 for providing encryption, authentication, message integrity, and key management. PGP is based on the "Web-of-Trust" model and has worldwide deployment.

- **Plain Text** (Clear Text)
  The human readable data or message before it is encrypted.

- **Polyalphabetic cipher**
  A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The Vigenère cipher is probably the best-known example of a polyalphabetic cipher, though it is a simplified special case. The Enigma machine is more complex but is still fundamentally a polyalphabetic substitution cipher.

- **Pseudo-Random Number**
  A number that results from applying randomizing algorithms to input derived from the computing environment, for example, mouse coordinates.

- **Private Key**
  The privately held "secret" component of an integrated asymmetric key pair, often referred to as the decryption key.

- **Public Key**// The publicly available component of an integrated asymmetric key pair often referred to as the encryption key.

- **Random Number**
  An important aspect to many cryptosystems, and a necessary element in generating a unique key(s) that are unpredictable to an adversary. True random numbers are usually derived from analog sources, and usually involve the use of special hardware.

- **Rijndael**
  A block cipher designed by Joan Daemen and Vincent Rijmen, chosen as the new Advanced Encryption Standard (AES). It is considered to be both faster and smaller than its competitors. The key size and block size can be 128-bit, 192-bit, or 256-bit in size and either can be increased by increments of 32 bits.

- **ROT-13 (Rotation Cipher)**
  A simple substitution (Caesar) cipher, rotating each 26 letters 13 places.

- **RSA**
  Short for RSA Data Security, Inc.; or referring to the principals - Ron Rivest, Adi Shamir, and Len Adleman; or referring to the algorithm they invented. The RSA algorithm is used in public key cryptography and is based on the fact that it is easy to multiply two large prime numbers together, but hard to factor them out of the product.

- **Salt**
  A random string that is concatenated with passwords (or random numbers) before being operated on by a one-way function. This concatenation effectively lengthens and obscures the password, making the cipher text less susceptible to dictionary attacks.

- **Secret key**
  Either the “private key” in public key (asymmetric) algorithms or the “session key” in symmetric algorithms.

- **Session key**
  The secret (symmetric) key used to encrypt each set of data on a transaction basis. A different session key is used for each communication session.

- **SHA-1 (Secure Hash Algorithm)**
  The 1994 revision to SHA, developed by NIST, (FIPS 180-1) used with DSS produces a 160-bit hash, similar to MD4, which is very popular and is widely implemented.

- **Single Sign-on**
  One log-on provides access to all resources of the network.

- **SSL (Secure Socket Layer)**
  Developed by Netscape to provide security and privacy over the Internet. Supports server and client authentication and maintains the security and integrity of the transmission channel. Operates at the transport layer and mimics the “sockets library,” allowing it to be application independent. Encrypts the entire communication channel and does not support digital signatures at the message level.

- **Stream cipher**
  A class of symmetric key encryption where transformation can be changed for each symbol of plain text being encrypted, useful for equipment with little memory to buffer data.

- **Substitution cipher**
  In cryptography, a substitution cipher is a method of encrypting by which units of plaintext are replaced with ciphertext, according to a fixed system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing the inverse substitution.[**wiki:subcipher**]

- **Symmetric algorithm**
  Also known as conventional, secret key, and single key algorithms; the encryption and decryption key are either the same or can be calculated from one another. Two sub-categories exist: Block and Stream.

- **Timestamping**
  Recording the time of creation or existence of information.

- **Transposition Cipher**
  The plain text remains the same but the order of the characters is transposed.

- **Trust**
  A firm belief or confidence in the honesty, integrity, justice, and/or reliability of a person, company, or other entity.

- **Twofish**
  A new 256-bit block cipher, symmetric algorithm. Twofish was one of five algorithms that the U.S. National Institute of Standards and Technology (NIST) considered for the Advanced Encryption Standard (AES).
- **Validation**
  A means to provide timeliness of authorization to use or manipulate information or resources.
- **Verification**
  To authenticate, confirm, or establish accuracy.
- **XOR**
  Exclusive-or operation; a mathematical way to represent differences.