

Compositional Cryptology

Thesis

Presented to the Honors Committee of
McMurry University

In partial fulfillment of the requirements
for Undergraduate Honors in Math

By

Amy Bell
Abilene, TX
December 2005

Acknowledgements

I could not have completed this thesis without all the support of my professors, family, and friends. Dr. McCoun especially deserves many thanks for helping me to develop the idea of compositional cryptology and for all the countless hours spent discussing new ideas and ways to expand my thesis. Because of his persistence and dedication, I was able to learn and go deeper into the subject matter than I ever expected. My committee members, Dr. Rittenhouse and Dr. Thornburg were also extremely helpful in giving me great advice for presenting my thesis.

I also want to thank my family for always supporting me through everything. Without their love and encouragement I would never have been able to complete my thesis.

Thanks also should go to my wonderful roommates who helped to keep me motivated during the final stressful months of my thesis. I especially want to thank my fiancé, Gian Falco, who has always believed in me and given me so much love and support throughout my college career.

There are many more professors, coaches, and friends that I want to thank not only for encouraging me with my thesis, but also for helping me through all my pursuits at school. Thank you to all of my McMurry family!

Preface

The goal of this research was to gain a deeper understanding of some existing cryptosystems, to implement these cryptosystems in a computer programming language of my choice, and to discover whether the composition of cryptosystems leads to greater security.

The first chapter, Introduction to Cryptology, introduces several well-known cryptosystems, discusses their security, and describes how to measure the security of a cryptosystem.

The second chapter, Discoveries in Cryptology, includes discoveries I have made about specific cryptosystems, compositions of cryptosystems, and the security of these compositions. In completing this work, I have discovered that:

- There is an easy way to generate $n \times n$ invertible matrices over Z_{26} which leads to a practical method of implementing Hill's Cipher.
- The Affine Cipher discussed in undergraduate textbooks is a group under the operation of function composition, and the order of each Affine Cipher can be easily computed.
- Composition of cryptosystems can lead to either more or less security.

Finally, this research has resulted in programs that allow a user to make their own cryptosystem by creating finite compositions from a set of five different cryptosystems. Through this program I have been able to analyze the security and practicality of compositional cryptology.

Table of Contents

| | |
|---------------------------------------|----|
| Chapter 1: Introduction to Cryptology | 1 |
| Chapter 2: Discoveries in Cryptology | 12 |
| Appendix | 28 |
| References | 31 |

Chapter One: Introduction to Cryptology

What is Cryptology?

Krypto is a Greek word that means 'secret' or 'hidden'. Therefore, cryptology is the study of encrypting and decrypting secret messages. This study can be divided into cryptography and cryptanalysis.

Cryptography refers to designing cryptosystems that code and decode messages, whereas cryptanalysis is concerned with techniques used in the unauthorized decoding of messages.

Cryptology is useful in transferring messages that contain sensitive material, such as military orders during wartime or personal identification information over the Internet. However, because of clever cryptanalysts and the unauthorized decoding of secret messages, development of secure cryptosystems is vital.

History of Cryptology

Although there is evidence of cryptology as far back as 1900 BC, detailed examples are not found until 500 BC, when Hebrew scribes recorded the book of Jeremiah using a reversed-alphabet Substitution cipher called ATBASH. The most famous early use of cryptology is Julius Caesar using a Shift cipher of three letters [Dupuis]. Known as the Caesar cipher, it is still taught in many cryptology textbooks as a special case of a Shift cipher.

Classical cryptology can be defined as cryptosystems designed before the 1940s and the advent of the computer age. Classic ciphers differ from modern ciphers in that the underlying mathematics is usually much simpler. Before the development of computers, many classical ciphers were considered secure enough to use at least during some period in history. However, now computers can be programmed to carry out various attacks against classical ciphers that were before impossible. Cryptology has expanded and grown to a more complex study in an effort to develop ciphers secure enough to withstand attacks by computers.

Manipulating Letters

Although messages are understood through their alphabetic form, it is convenient to substitute numbers for letters to be able to manipulate and encode them. For instance, when using the English alphabet of A to Z, you can replace each letter with the numbers 0 through 25.

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

Table 1

Of course, when you take into consideration using lower case, upper case, punctuation, and other symbols, more numbers are required. When messages are converted into numbers it is possible to perform a variety of mathematical operations on the numbers and then convert the numbers back into letters, hence forming a secret, unintelligible message.

Uniformly Distributed Letter Cipher

Many ciphers have encryption/decryption functions that allow for easier enciphering and deciphering of the message. Other ciphers, like DES (Data Encryption Standard) make use of a sequence of encryption/decryption steps. The Uniformly Distributed Letter (UDL) cipher [Lewand, p.76-79] uses the relative frequencies of English letters to create an encryption relation rather than an encryption function. For instance, the letter *e* has a relative frequency of approximately 0.13, meaning that in typical English texts, *e* appears approximately 13% of the time. The rest of the letters in the English alphabet occur in typical English texts according to varying percentages [Lewand, p. 36].

| Letter | Relative Frequency | Letter | Relative Frequency |
|--------|--------------------|--------|--------------------|
| A | 8.167% | N | 6.749% |
| B | 1.492% | O | 7.507% |
| C | 2.782% | P | 1.929% |
| D | 4.253% | Q | 0.095% |
| E | 12.702% | R | 5.987% |
| F | 2.228% | S | 6.327% |
| G | 2.015% | T | 9.056% |
| H | 6.094% | U | 2.758% |
| I | 6.966% | V | 0.978% |
| J | 0.153% | W | 2.360% |
| K | 0.772% | X | 0.150% |
| L | 4.025% | Y | 1.974% |
| M | 2.406% | Z | 0.074% |

Table 2

Because *e* has a relative frequency of approximately 13%, the UDL cipher chooses 13 numbers between 00 and 99 inclusive that can be substituted for *e*. For example, the numbers substituted for *e* might come from the set, {08, 12, 20, 46, 47, 59, 64, 79, 81, 85, 90, 94, 97}. This continues until each English letter has been assigned its own set of unique number(s) between 00 and 99. The table below [Lewand, p. 78] can be used for encoding letters into numbers and decoding numbers back into letters.

| Letter | Set of Numbers | Letter | Set of Numbers |
|--------|--|--------|------------------------------------|
| A | 15,33,37, 55, 57, 72, 91, 96 | N | 06, 17, 22, 31, 49, 58 |
| B | 24 | O | 02, 10, 41, 51, 66, 75, 83 |
| C | 03, 39, 67 | P | 13,18 |
| D | 04, 43, 61, 88 | Q | 36 |
| E | 08, 12, 20, 46, 47, 59, 64, 79, 81, 85, 90, 94, 97 | R | 21, 25, 65, 68, 92, 95 |
| F | 40, 48 | S | 00, 28, 52, 63, 74, 78 |
| G | 29, 53 | T | 07, 19, 23, 35, 38, 54, 70, 84, 89 |
| H | 05, 16, 30, 42, 69, 99 | U | 09, 32 |
| I | 14, 45, 50, 60, 73, 82, 93 | V | 44 |
| J | 11 | W | 56, 80 |
| K | 77 | X | 86 |
| L | 01, 26, 71, 98 | Y | 62, 76 |
| M | 34, 87 | Z | 27 |

Table 3

Using Table 3 we could encrypt the word, *ebenezer*, to **08, 24, 12, 06, 20, 27, 46, 21**. Note that each time *e* is encoded the next letter in the set corresponding to *e* is used, and when the end of the set is reached, the numbers start over again. Another method of encoding *e* would be to randomly choose a number from the set, {08, 12, 20, 46, 47, 59, 64, 79, 81, 85, 90, 94, 97}, with each number being equally likely to be chosen. Because many of the letters correspond to more than one number, the encryption utilizes a relation rather than a function.

The UDL cipher is useful in evening out letter frequencies, which cryptanalysts sometimes use to break ciphers. In the enciphered message, each number between 00 and 99 will have approximately the same frequency and each number will have approximately the same relative frequency of 1%.

Substitution Cipher

Like its name suggests, the Substitution cipher encodes messages by substituting each letter for a unique letter. A key for the Substitution cipher is a one-to-one correspondence between the usual English alphabet and a rearrangement or permutation of it. The key for a Substitution cipher is often expressed in table form. For example, a permutation of the usual English alphabet might be obtained by reversing the order of the alphabet (Table 4), or by using a phrase like “I LOVE MCMURRY”, with spaces and duplicate letters deleted for the first part of the key, then appending the remaining unused letters of the English alphabet in alphabetical order for the last part of the key (Table 5).

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

Table 4

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

Table 5

Shift Cipher

The simplest classical cipher is the Shift cipher. Every cipher has a key that gives the receiver of the message the ability to decipher the code. For the Shift cipher, the key is a number between 1 and 25 (in the case of the alphabet used in Table 1). Zero is usually excluded because it would result in no shift of the resulting message. After converting the letters of the message into numbers using Table 1, the Shift cipher will add the key to each number (letter), thus shifting each letter a certain number of letters to the right with wrap-around occurring if a letter gets shifted beyond the end of the alphabet.

The encryption function for the Shift cipher is:

$$(1.001) \quad E_{key} = (P + key) \bmod 26 = C$$

C refers to the number of the encoded letter, P refers to the number of the unencoded letter, and mod refers to the modulus function that ‘wraps’ large numbers that extend beyond the end of the alphabet so that they correspond to a letter of the alphabet.

Note: I will frequently use the letters P and C to distinguish between plaintext numbers (original text) and cipher text numbers (encoded text).

As mentioned earlier, the Caesar cipher is a Shift cipher that uses the key of three. Using the Caesar cipher, the word, *running*, enciphers to UXQQLQJ, as indicated in Table 6 below.

| | | | | | | | |
|-------------|----|----|----|----|----|----|---|
| Plaintext | r | u | n | n | i | n | g |
| P | 17 | 20 | 13 | 13 | 8 | 13 | 6 |
| $C = P + 3$ | 20 | 23 | 16 | 16 | 11 | 16 | 9 |
| Ciphertext | U | X | Q | Q | L | Q | J |

Table 6

Today, the Caesar cipher can be easily broken. However, during Caesar’s time, few people were aware of cryptographic techniques, so this method was fairly secure.

The decryption function for the Shift cipher is:

$$(1.002) \quad D_{key} = (C - key) \bmod 26 = P$$

The ROT-13 cipher is a Shift cipher that is still used today. For the 26 letter English alphabet, the ROT-13 cipher is a Shift cipher with a key of 13. Other than using the key 0, this is the easiest key to use for simple encryption and decryption. Observe Table 5 below; each letter has its corresponding letter beneath or above it.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

Table 7

Some newsgroups use ROT-13 to encode offensive speech. That way, only those who wish to read an offensive word or phrase can use the ROT-13 decoder in their newsreader to quickly decode the message. Otherwise, it appears as nonsense, and those not wishing to read it can skip over it and have no idea what it says. The following text is included in the help file from a popular newsreader called NewsRover (<http://www.newsrover.com>).

ROT-13 encoding

Portions of some messages may be encrypted using "ROT-13" encryption. ROT-13 is a very mild form of encryption that just replaces each alphabetic character with another character located 13 characters further along in the ASCII character table. ROT-13 encryption is most often used to obscure portions of messages that may be offensive to some people. For example, some jokes found in rec.humor may be offensive, and the body of the joke is encoded using ROT-13.

If you encounter a message that has a portion encoded using ROT-13, the encoded portion will appear as gibberish, but you will see the normal punctuation characters and spaces between words.

To decode the ROT-13 portion of a message, first select (highlight) the encoded portion of the message, then right-click and select "ROT-13 translation" from the popup menu. The selected portion will be decoded and displayed in the message. Note: the decoding only affects the current display of the message; the message text stored on disk remains ROT-13 encoded.

Affine Cipher

Another classic cipher is the Affine cipher, which can be shown to be a special case of the Substitution cipher [Stinson, p.8]. Like the Shift and the Substitution ciphers, there is a one-to-one correspondence between plaintext letters and ciphertext letters (i.e. a mono-alphabetic cipher). The encryption function for the Affine cipher is:

$$(1.003) \quad E_{a,b} = (a * P + b) \% 26 \quad \text{where } a \in A \text{ and } b \in B$$

$A = \{1,3,5,7,9,11,15,17,19,21,23,25\}$, $B = \{0, \dots, 25\}$, and % indicates the use of modular arithmetic.

The decryption function for the Affine cipher is:

$$(1.004) \quad D_{a,b} = a^{-1} * (C - b) \% 26 \quad \text{where } a^{-1} \in A \text{ and } b \in B$$

Table 8 gives an example of what the letters of the alphabet would encrypt to using the key of (3,2).

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

Table 8

From this table you can notice the one-to-one correspondence between plaintext and ciphertext letters and also the difference from the Shift cipher. Observe that the plaintext letter, m, becomes the ciphertext letter, M, and the plaintext letter, z, becomes the ciphertext letter, Z, when a key of $(a,b) = (3,2)$ is used. This particular situation would never occur with the Shift cipher unless the key of zero was used, which would result in no change in any of the letters.

Hill Cipher

The Hill cipher is a poly-alphabetic cipher that uses matrices and modular arithmetic to encode plaintext messages. The key for this cipher consists of a square invertible matrix of any size. It could be a 2x2 or a 10x10; depending on how secure you would like the cipher to be. After converting the message to a numerical format, you perform matrix multiplication modulo 26 with your message and the key to obtain

the ciphertext. For example, consider the 2x2 key matrix of $\begin{bmatrix} 9 & 4 \\ 5 & 7 \end{bmatrix}$. To encode the word, *running*, it

would first be changed to the numerical word, (17, 20, 13, 13, 8, 13, 6) using Table 1. The matrix multiplication would involve multiplying the 2x2 key matrix by the column vector containing the first two numbers in the numerical word, i.e. 17 and 20. Following the rules of matrix multiplication, this operation would yield $y_1 = (9x_1 + 4x_2) \bmod 26 = (9*17 + 4*20) \bmod 26 = 233 \bmod 26 = 25$ and

$y_2 = (5x_1 + 7x_2) \bmod 26 = (5*17 + 7*20) \bmod 26 = 225 \bmod 26 = 17$. Therefore, the first two letters r and u would encode to Z and R. In general, the encryption rule for the Hill cipher with a 2x2 key (K) and a plaintext of length n , where n is a multiple of 2 is:

$$\begin{aligned}
 y_1 &= K_1x_1 + K_2x_2 \\
 y_2 &= K_3x_1 + K_4x_2 \\
 (1.005) \quad &: \\
 y_{n-1} &= K_1x_{n-1} + K_2x_n \\
 y_n &= K_3x_{n-1} + K_4x_n
 \end{aligned}$$

On the occasion that the length of the plaintext is not a multiple of the size of the matrix, random letters will be appended to the end of the plaintext until the desired length is reached. For instance, to completely encode the word, *running*, with a 2x2 matrix, a random letter must be added to the end of the message so that g can be encoded.

One interesting thing about Hill cipher relates to why it is a poly-alphabetic cipher. All the ciphers previously discussed had each plaintext letter encoding to a specific ciphertext letter. For instance, in the shift cipher with a key of 3 it is easy to see that the three n's in the word *running* are encoded into Q's (Table 6). That is a characteristic of a mono-alphabetic cipher. In a poly-alphabetic cipher, plaintext letters will not always encode to the same ciphertext letters [Garrett, p. 58]. To illustrate this point, when we encode the next two letters in the word *running*, n and n, we get $(9*13 + 4*13) \bmod 26 = 169 \bmod 26 = 13$ and $(5*13 + 7*13) \bmod 26 = 156 \bmod 26 = 0$. So, n and n, in this case, encode to N and A. Because the

same letters will sometimes encode into different letters, Hill cipher is more secure than the Shift cipher and the Affine cipher.

Vigenere Cipher

The Vigenere cipher is also poly-alphabetic, and therefore more secure than the UDL, Substitution, Shift, and Affine ciphers [Lewand, p.45]. It uses a key determined by the encoder, usually an unpredictable or random word or phrase, keeping in mind that an unauthorized cryptanalyst might attempt to break the code with a dictionary attack, using expected words as possible keys. Once the plaintext and the key are converted into numbers, you line up the two sets of numbers, repeating the key numbers as much as necessary and add the numbers modulo 26 to obtain the ciphertext. Table 9 below illustrates how the Vigenere cipher with the keyword, shoe, can be used to encrypt the word running.

| | | | | | | | |
|-----------------------------|--------------------------|--------------------------|--------------------------|---------------------------|--------------------------|---------------------------|---------------------------|
| Plaintext | r | u | n | n | i | n | g |
| <i>P</i> | 17 | 20 | 13 | 13 | 8 | 13 | 6 |
| Key | S | H | O | E | S | H | O |
| Key Number | 18 | 7 | 14 | 4 | 18 | 7 | 14 |
| $C = P + \text{Key Number}$ | $35 \text{ mod } 26 = 9$ | $27 \text{ mod } 26 = 1$ | $27 \text{ mod } 26 = 1$ | $17 \text{ mod } 26 = 17$ | $26 \text{ mod } 26 = 0$ | $20 \text{ mod } 26 = 20$ | $20 \text{ mod } 26 = 20$ |
| Ciphertext | J | B | B | R | A | U | U |

Table 9

Using the Vigenere cipher and the keyword, shoe, “running” is encoded into JBBRAUU. It is evident from this example that poly-alphabetic ciphers may result in the same plaintext letter being encoded into different ciphertext letters, and different plaintext letters might be encoded to the same ciphertext letters, which happened twice in this example.

RSA Cipher

This cipher is much more complicated than any others mentioned so far. Developed in 1978, it is named after the last name of its three inventors: Rivest, Shamir, and Adleman. It uses the concept of a public key, which makes the encryption key public while keeping the decryption key completely secret. One of RSA’s public keys is the product of two large, typically 100 digit, prime numbers, $n = pq$. The other public key, e , is a number chosen that is relatively prime to $\phi(n)$. $\phi(n)$ is known as Euler’s phi function and is equal to the number of nonnegative integers that are less than and relatively prime to n . The equation $\phi(n) = (p - 1)(q - 1)$ is derived from the fact that both p and q are prime. Then, the extended Euclidean algorithm is used to find d , the private decryption key, such that $ed \equiv 1 \text{ mod } \phi(n)$, that is, $d \equiv e^{-1} \text{ mod } \phi(n)$.

Because the factorization of large integers into prime numbers is extremely difficult, the RSA cipher can be very secure. The encryption and decryption rules of RSA are:

(1.006) $E_{n,e}(x) = x^e \text{ mod } n$

(1.007) $D_{n,d}(y) = y^d \text{ mod } n$

As a simple example, small primes can be used to illustrate how RSA works. By choosing $p = 17$ and $q = 43$, the product $n = 731$. From these variables, $\phi(n) = 16 * 42 = 672$. e can then be chosen as a number relatively prime to 672, in this case, 29 is relatively prime to 672 and can be chosen as e . To find the key for decryption, the multiplicative inverse of $e \bmod (p-1)(q-1) = e \bmod 672$ must be found. In this instance, by using the extended Euclidean algorithm, $d = e^{-1} \bmod 672 = 533$ [Barr, p. 288]. Therefore, $n = 731$ and $e = 29$ are published while keeping $p, q, \phi(n), d$ secret. Now that all the variables are found, they can be substituted into the equations to encode and decode.

Key Space

One measure of the security of a cipher is the size of its key space. For a particular cipher, the size of the key space refers to the total number of keys. Throughout this section k will refer to the set of possible keys and $|k|$ will refer to the cardinality of k or the size of the keyspace.

Combinations are used to find the size of the key space for the UDL cipher. Since the relative frequency of the letter, a, is approximately 8% in typical English text, we choose 8 numbers between 00 and 99 inclusive as ciphertext numbers corresponding to the letter, a. This can be done in $\binom{100}{8}$ ways. Since the relative frequency of the letter, b, is approximately 1% in typical English text, we choose 1 number from the remaining 92 numbers. This can be done in $\binom{92}{1}$ ways. This process continues to the letter z, in which case there will only be one number left to assign to the letter, z, this can be done in only 1 way which is also the combination, $\binom{1}{1}$. To calculate the size of the key space we use the multiplication rule. The size of the UDL key space is therefore,

$$\binom{100}{8} \binom{92}{1} \binom{91}{3} \binom{88}{4} \binom{84}{13} \binom{71}{2} \binom{69}{2} \binom{67}{6} \binom{61}{7} \binom{54}{1} \binom{53}{1} \binom{52}{4} \binom{48}{2} \binom{46}{6} \binom{40}{7} \binom{33}{2} \binom{31}{1} \binom{30}{6} \binom{24}{6} \binom{18}{9} \binom{11}{2} \binom{9}{2} \binom{7}{1} \binom{6}{2} \binom{4}{1} \binom{3}{2} \binom{1}{1}$$

which simplifies to $\frac{100!}{8!3!4!13!2!2!6!7!4!2!6!7!2!6!6!9!2!2!2!} = 3.39 * 10^{113}$

Each permutation (ordering) of the English alphabet is a possible key for the Substitution cipher. Determining the total number of permutations of the English alphabet is an application of the multiplication principle. Suppose you were going to put the English alphabet in the same or possibly a different order directly below the usual ordering of the English alphabet.

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
 - - - - -

The task of creating a permutation of the English alphabet can be accomplished by completing a sequence of tasks.

1. Select a letter from the alphabet to place in the blank below the letter A.
2. Select a letter from the alphabet other than the letter chosen in Task 1 to place in the blank below the letter B.
3. Select a letter from the alphabet other than the letters chosen in Task 1 and Task 2 to place in the blank below the letter C.
4. Select a letter from the alphabet other than the letters chosen in the previous tasks to place in the blank below the letter D.
- .
- .
- .
26. Select a letter from the alphabet other than the letters chosen in the previous tasks to place in the blank below the letter Z.

Note that Task 1 can be done in 26 ways, Task 2 in 25 ways, Task 3 in 24 ways, Task 4 in 23 ways, ..., and Task 26 in 1 way. According to the multiplication principle, the total number of ways of completing the original task of re-writing the English alphabet in the same or possibly different order is $26 * 25 * 24 * 23 * \dots * 2 * 1 = 26!$. Since there are 26! possible orderings of the English alphabet, one of which is the normal ordering, the cardinality (size) of the key space for the Substitution cipher is:

$$(1.008) \quad |k| = 26! - 1 \approx 4.03 * 10^{26}$$

The normal ordering is not counted in the size of the key space because it is the regular order of the English alphabet, and is therefore a worthless key for the Substitution cipher since it doesn't produce any effect.

In the case of the Shift cipher, when using a modulus of 26, the size of the key space is 25. A cryptanalyst can easily write a program that will quickly decrypt a Shift-enciphered message by using the decryption function twenty-five times with each of the possible keys. Although this would be arduous by hand, writing a program to accomplish this task is elementary. Now, if the alphabet included all ASCII characters, the key space for the Shift cipher would be larger, but still small enough to break with a computer program.

Because of the use of two parameters, a and b from equation (1.003), the key space of the Affine cipher is much larger than the key space of the Shift cipher. a can be any of the twelve numbers from the set A and b can be any of the 26 numbers in set B . The size of the key space would be the number of possible ordered pairs, (A, B) , omitting the ordered pair, $(1, 0)$, which would cause no effect on the plaintext. This leaves the Affine cipher with a key space size of $|k| = 26 * 12 - 1 = 311$ [Garrett, p. 14]. As with the Shift cipher, if a larger alphabet were being used, the key space of the Affine cipher would be much larger. In the general case, B would be the set of numbers representing the alphabet being used and the set A would contain the elements of B that are invertible under the modulus, $|B|$.

The size of the key space of the Vigenere cipher, with a key word length of n and an alphabet of 26, is 26^n . To increase the cardinality of the key space, it is possible to use a novel or a long document as the key. In this case, the key may be longer than the message itself, which would make it even harder for an unauthorized cryptanalyst to break.

For the RSA cipher, the size of the key space is infinite since it has been known since the time of Euclid that there are infinitely many primes. Factoring a large composite integer into primes can prove to be a very daunting task, and is what determines RSA's security. However, on May 9, 2005, RSA 200 (n contained 200 digits) was factored into p and q by a group from Germany [Weisstein]. Typically, p and q are 100-digit primes, and the Prime Number Theorem can be used to approximate the number of 100-digit primes. The Prime Number Theorem states that the number of primes less than an integer n is asymptotic to $\frac{n}{\ln(n)}$ [Caldwell, p. 4]. Therefore the number of 100-digit primes is approximately equal to

$$\frac{10^{100}}{100 \ln 10} - \frac{10^{99}}{99 \ln 10} \approx \frac{10^{98}}{\ln 10} - \frac{10^{97}}{\ln 10} = \frac{10^{98} - 10^{97}}{\ln 10} \approx 3.9 * 10^{97}$$

So, there are approximately $3.9 * 10^{97}$ 100 digit primes, which is quite a lot to choose from, and quite a lot to have to go through to factor $n=pq$.

Cryptanalysis

Cryptanalysis is a large study and practice that involves anything from complex math formulas to diligently substituting letters. The ciphers presented in this chapter have varying degrees of security and efficiency.

Security depends on how difficult the code is to break for an outsider. In the case of an outsider who does not possess the key, there are several different methods of attack which are described in many textbooks, but the following methods are defined [Stinson, p. 25]:

- Ciphertext-Only Attack - The opponent possesses a string of ciphertext, y .
- Known Plaintext Attack – The opponent possesses a string of plaintext, x , and the corresponding ciphertext, y .
- Chosen Plaintext Attack – The opponent has obtained temporary access to the encryption machinery. Hence he can choose a plaintext string, x , and construct the corresponding ciphertext string, y .
- Chosen Ciphertext Attack – The opponent has obtained temporary access to the decryption machinery. Hence he can choose a ciphertext string, y , and construct the corresponding plaintext, x .

For more complex poly-alphabetic ciphers, having access to a few characters and their keys may help a little, but there would still be a lot of work to be done. In addition to the above attacks, Paul Garret [Garret, p.14] describes a Brute Force Ciphertext Only Attack, which means that each possible decryption key is applied to the ciphertext until there is recognizable output that corresponds to the plaintext. In connection with the Brute Force Ciphertext Only Attack, a cryptanalyst may also use a dictionary attack that decodes while looking for familiar words or phrases. The Brute Force Attack is the most time consuming of all the attacks because of the large quantity of keys that some cipher schemes possess. Therein lies the importance of having a large key space, to make decoding by brute force too difficult and tedious to endeavor. Efficiency can be defined in many different ways; however, for the sake of this thesis, efficiency refers to how quickly authorized users can code and decode messages with the given keys. This may involve how quickly a computer program or a human mind could work to encode and decode. The trick to making a good cipher scheme is to make one that it is not difficult or time-consuming for authorized users to encode and decode, but is difficult for unauthorized users to encode and decode.

Unicity Distance

Another measure of a message's security is the unicity distance. According to Douglas Stinson, in his book, Cryptography: Theory and Practice, the unicity distance of a cryptosystem is defined to be the average amount of ciphertext required by an opponent to be able to uniquely compute the key, given enough computing time. It is calculated as the value of n , at which the expected number of spurious keys becomes zero. A spurious key is a key that when used with ciphertext, yields a reasonable plaintext, but is not the actual plaintext. For example, the shift-encrypted ciphertext WNAJW can be decoded into river or arena, depending on which of the 25 available keys are used. Observe all the possible combinations of a Shift cipher for WNAJW computed from a computer program:

```
>>> shift_d()
```

```
Enter the ciphertext to be decrypted: wnaajw
```

```
vmziv  shift applied to ciphertext = 1
```

```
ulyhu  shift applied to ciphertext = 2
```

```
tkxgt  shift applied to ciphertext = 3
```

```
sjwfs  shift applied to ciphertext = 4
```

```
river  shift applied to ciphertext = 5
```

qhudq shift applied to ciphertext = 6
 pgtcp shift applied to ciphertext = 7
 ofsbo shift applied to ciphertext = 8
 neran shift applied to ciphertext = 9
 mdqzm shift applied to ciphertext = 10
 lcpyl shift applied to ciphertext = 11
 kboxk shift applied to ciphertext = 12
 janwj shift applied to ciphertext = 13
 izmvi shift applied to ciphertext = 14
 hyluh shift applied to ciphertext = 15
 gxktg shift applied to ciphertext = 16
 fwjsf shift applied to ciphertext = 17
 evire shift applied to ciphertext = 18
 duhqd shift applied to ciphertext = 19
 ctgpc shift applied to ciphertext = 20
 bsfob shift applied to ciphertext = 21
arena shift applied to ciphertext = 22
 zqdmz shift applied to ciphertext = 23
 ypclly shift applied to ciphertext = 24
 xobkx shift applied to ciphertext = 25

Only one of the words was purposely encoded and the other coincidentally is an option for an unauthorized decoder.

The equation to determine the possible minimum number of spurious keys (s_n) is $s_n = \frac{|K|}{|P|^{nR_L}} - 1$ where n is the unicity distance (necessary length of the ciphertext), R_L is the redundancy (excess characters) of the language, $|K|$ is the size of the key space of the cryptosystem, and $|P|$ is the length of the alphabet in use. From this equation, you can determine the unicity distance by setting $s_n = 0$.

$$0 = \frac{|K|}{|P|^{nR_L}} - 1$$

$$1 = \frac{|K|}{|P|^{nR_L}}$$

$$\log_2 1 = \log_2 \frac{|K|}{|P|^{nR_L}}$$

$$0 = \log_2 |K| - \log_2 |P|^{nR_L}$$

$$\log_2 |P|^{nR_L} = \log_2 |K|$$

$$nR_L = \frac{\log_2 |K|}{\log_2 |P|}$$

$$n = \frac{\log_2 |K|}{R_L \log_2 |P|}$$

Therefore, the unicity distance is $\frac{\log_2 |K|}{R_L \log_2 |P|}$ [Stinson, 2.4].

For a substitution cipher with an alphabet of 26 and a redundancy of .75 (typical value of excess characters in the English language), the unicity distance would be $\frac{\log_2 26!}{.75 \log_2 26} \approx 25$. Therefore, with a ciphertext at least 25 characters long, there should only be one unique decryption possible. To use the knowledge of unicity distance as an advantage to the encoder, the message can be made to be shorter than the calculated unicity distance by omitting unnecessary letters or words. That way, there may be another unintentional message that could be decrypted by an unauthorized decoder, thus creating confusion as to which message is the true one.

Chapter Two: Discoveries in Cryptology

Composing Ciphers

An interesting concept that has been utilized in modern cryptology is having cipher schemes that involve more than one step. Rather than having a simple formula to encode and decode, there is a set procedure, or complex algorithm that must be used to complete the encryption/decryption process. The DES cipher is an excellent example of a multi-step cipher.

Modern cryptology also makes use of multi-round encryption, for example Triple DES. Multi-round ciphers apply the same cipher more than once in an effort to increase security. When a particular cipher, f , is composed with itself once, i.e. $f \circ f$, and then applied to some plaintext message, x , i.e. $(f \circ f)(x)$, it is said that we have performed a two-round cipher on the plaintext message, x . The notation, $f^2(x)$, is also a common way of representing, $(f \circ f)(x)$. After showing that the set of Affine Ciphers is a group under composition, it will be shown that an Affine Cipher can be composed with itself enough times to have no net effect on a plaintext message. For example, using theorems to be developed later in this thesis, it can be shown that a 12-round Affine Cipher with a key of $(a,b) = (15, 7)$ applied to some plaintext results in returning the original text.

Yet another kind of failure might occur when the same type of cipher is used more than once with a different key each time. For instance, if you encoded a message using a Shift cipher twice with different keys, every letter would still correspond to another letter a set number of letters away. The key for this cipher would be the addition of the two used keys. Therefore, it is pointless to double-encrypt a message with the Shift cipher because whoever tried to decode it would still break it the same way they would break a one-round Shift cipher encrypted message without any extra work.

Other ciphers do increase security when used twice in a row. For instance, in the case of the Vigenere cipher, if two or more key words were used to encrypt a message, the decoder would have to 'break' it multiple times to yield the plaintext message. However, if the key words had the same length, the net effect would be the same as just encrypting the message with one "word". Yet, when an unauthorized recipient of the message attempts to decode the message, the resulting key "word" from the multiple key words of equal length may not be an understandable word. It would still be possible to decrypt the two-round Vigenere cipher by using various methods that attack a Vigenere cipher; it would just be a little trickier with a non-word key word. If the key words were of different length, the message might be impossible to decrypt, unless the unauthorized decoder knew that the message was multiply encrypted using the Vigenere cipher.

The bulk of my research has involved gaining a deeper understanding of many of the classical ciphers, writing programs to perform various ciphers, and creating a program that combines or composes these ciphers. The latter required using several different cipher schemes one after the other in encoding a message. Instinctively, this will most likely improve the security of the message, but by how much and at what cost of efficiency I will investigate. Five of the cipher schemes mentioned earlier are those that I have chosen to compose together and analyze.

This research has resulted in software that combines the Shift, Affine, Hill, Vigenere, and RSA ciphers in any order desired, and using the cipher schemes as much (or not at all) as wanted. Although in these modern times there are many scholars and experienced cryptologists developing 'unbreakable' cipher systems, I feel that compositional ciphers can still be relevant in today's discoveries.

The rest of the material in this thesis concerns discoveries made regarding the Affine Cipher, a practical implementation of Hill's Cipher, and some results of composing various ciphers.

“Pastry Dough Mixing”

In Decrypted Secrets, by Friedrich Bauer, he states that composing two different encryption methods usually leads to a new encryption method, but it is usually closely related to its parent methods. As mentioned previously, combining two similar methods does not always prove to be fruitful. The trick in composing methods is to combine them in a way that leads to difficulty in decoding them by brute force or other attacks. Since a large part of decoding by brute force depends on recognizing actual words or common letter combinations, using two different methods that produce a good mix can prevent the use of recognizing words while decoding. For instance, if a message was encrypted by a Shift cipher followed by a Hill cipher, the message must first be correctly decoded as a Hill cipher and then as a Shift cipher. However, it would be rather difficult to determine whether or not the Hill cipher was correctly decoded because no words would be recognized. You could compare this brute force attack to assembling a giant jigsaw puzzle that is only one color.

Claude Shannon, a famous cryptologist, used the phrase ‘pastry dough mixing’ when discussing creditable cipher composition. When mixing pastry dough, you start with a lump of dough, roll it flat, fold it together, and roll it flat again. This is a good example of what cipher composition should produce. A confusing combination that is extremely tedious and almost impossible to sort out by brute force or other attacks. The ciphers should be non-commutable and completely independent of one another. A composed cipher is non-commutable if it yields different ciphertext when composed in a different order. If a composed cipher was commutable, it could be encrypted and decrypted in any order and result in the same ciphertext. Being non-commutable aids in the security of the message because it must be decrypted in a specific order of methods. Being completely independent refers to the ciphers performing very different operations on the plaintext, such as mixing the RSA cipher with a Vigenere cipher; both ciphers perform very different operations and are not related to each other. On the other hand, a Shift cipher and an Affine cipher are not completely independent because they are related to one another.

A large part of my preparation for this research has involved creating a computer program that composes different techniques to encrypt and decrypt messages. I set it up in such a way that the user can use as many different available techniques as many times as they want and in whatever order they want. From this, I am able to study the effects of composing ciphers. As I have mentioned earlier, some ciphers aren’t worth composing together multiple times. However, by using the idea of “pastry dough mixing”, it is possible to create a fairly secure cipher. The available techniques included in my program are the Shift cipher, the Affine cipher, the Hill cipher, the Vigenere cipher, and RSA.

The Shift cipher and the Affine cipher are both considered to be simple ciphers; they are mono-alphabetic and thus quite vulnerable to attack. The Hill cipher, Vigenere cipher, and RSA have varying degrees of security, but are all much more difficult to attack. When encoding a message, you must always assume that any unauthorized decoder would know how to perform cryptanalysis for any cipher that you use. Therefore it is important to keep that in mind when choosing the order in which to encrypt. For example, consider the composition of a Shift cipher, a Hill cipher, and then an Affine cipher. To decrypt this message correctly, it must first be decoded by the Affine, then by the Hill, and finally by the Shift because the composition is non-commutable.

The Affine Group

To be able to compose Affine ciphers together, it is an interesting exercise to determine if the Affine function produces a group. A group is defined to be a set with an operation that is associative, has an identity element that when multiplied by any element produces that same element, and an inverse for each element that when multiplied with that element produces the identity element [Pinter, p.25].

Definition 2.1

The Affine cipher as defined in elementary textbooks is

$$E_{a,b}(x) = (ax + b) \% 26$$

$$\text{where } a \in A = \{1,3,5,7,9,11,15,17,19,21,23,25\}$$

$$b \in B = \{0,1,2, \dots, 25\}$$

and $x \in B$

The proof of the theorem below uses theorems A1 and A3 in the appendix. I have also used that fact that $a \% 26 = a$ and $b \% 26 = b$ where $a \in A$ and $b \in B$.

Theorem 2.1

$$E_{a,b} \circ E_{c,d} = E_{ac,ad+b} = E_{(ac)\%26,(ad+b)\%26}$$

Proof:

$$\begin{aligned} & (E_{a,b} \circ E_{c,d})(x) \\ &= E_{a,b}(E_{c,d}(x)) \\ &= E_{a,b}((cx + d) \% 26) \\ &= (a((cx + d) \% 26) + b) \% 26 \\ &= (a \% 26 (cx + d) \% 26 + b \% 26) \% 26 \\ &= ((a \% 26 (cx + d) \% 26) \% 26 + b \% 26) \% 26 \\ &= ((a(cx + d)) \% 26 + b \% 26) \% 26 \\ &= ((a(cx) + ad) \% 26 + b \% 26) \% 26 \\ &= (((ac)x + ad) + b) \% 26 \\ &= ((ac)x + (ad + b)) \% 26 = E_{ac,ad+b}(x) \\ &= (((ac)x) \% 26 + (ad + b) \% 26) \% 26 \\ &= (((ac) \% 26 x \% 26) \% 26 + (ad + b) \% 26) \% 26 \\ &= (((ac) \% 26)x + (ad + b) \% 26) \% 26 \\ &= E_{(ac)\%26,(ad+b)\%26}(x) \end{aligned}$$

Definition 2.2

Let AF be the set of all Affine ciphers as defined in Definition 2.1. That is,

$$AF = \{E_{a,b} \mid a \in A, b \in B\}$$

$$\text{where } a \in A = \{1,3,5,7,9,11,15,17,19,21,23,25\}$$

$$b \in B = \{0,1,2, \dots, 25\}$$

To prove that the composition function is an operation on the set, AF , it is necessary to show that AF is closed under composition. Theorem 2.1 shows that for $E_{a,b} \in AF$ and $E_{c,d} \in AF$, then

$$E_{a,b} \circ E_{c,d} = E_{(ac)\%26,(ad+b)\%26}$$

In order for $E_{(ac)\%26,(ad+b)\%26}$ to be in AF it must be shown that $(ac) \% 26 \in A$ and $(ad + b) \% 26 \in B$ where A and B are the sets in Definition 2.1 and 2.2. All of the elements of A are relatively prime to 26. In other words, for each number in set A , the greatest common divisor of 26 and that number is 1. Therefore, since 26 is divisible by 2, all even numbers are not relatively prime to 26 because they are made up of factors that also make up 26. Out of all the odd numbers, the number 13 is the only one absent from the set because $13 * 2 = 26$. So, all the remaining numbers listed in set A share no common factors with 26. It is evident

that $(ad + b) \% 26 \in B$ and the following theorem shows that $(ac) \% 26 \in A$.

Theorem 2.2

Let $A = \{1,3,5,7,9,11,15,17,19,21,23,25\}$.

If $a \in A$ and $c \in A$, then $(ac) \% 26 \in A$.

Proof:

Obviously $(ac) \% 26$ is an integer between 0 and 25 inclusive. Because a and c are both odd numbers, the product of a and c is odd. According to the division algorithm [Pinter, p. 213], there exists unique integers q and r such that:

$$ac = q * 26 + r \text{ where } 0 \leq r < 26$$

$$r - ac = -q * 26$$

$$r \equiv (ac) \% 26$$

From the second equation above,

$$r = ac - 2(13q)$$

That is, r is the result of an odd integer minus an even integer, which is an odd integer. Therefore, r or $(ac) \% 26$ is an odd integer, and as stated above, $0 \leq r < 26$. If it can be shown that $13 \neq (ac) \% 26$, then the result will follow.

Assume $13 \equiv (ac) \% 26$. Then, $13 - ac = 26k$ for some integer, k . Thus $13 - 13(2k) = ac$ which implies $ac = 13(1 - 2k)$, that is, ac is a multiple of 13. But this is impossible since neither a nor c being from A have a prime factor of 13. Thus, our assumption is incorrect and $13 \neq (ac) \% 26$.

Theorem 2.3

The set of Affine ciphers, AF , with the operation of function composition is a group.

Proof:

$E_{1,0}$ is the identity element of AF under composition. (Theorem 2.4)

Each element of AF has an inverse with respect to composition. (Theorem 2.5)

The operation of function composition is associative. (Theorem 2.6)

Therefore, the set AF is a group.

Theorem 2.4

$E_{1,0}$ is the identity element of AF under composition.

Proof:

Let $E_{a,b} \in AF$. Then, according to Theorem 2.1,

$$E_{1,0} \circ E_{a,b} = E_{1*a,1*b+0} = E_{a,b}$$

and

$$E_{a,b} \circ E_{1,0} = E_{a*1,a*0+b} = E_{a,b}$$

Therefore, $E_{1,0}$ is the identity of AF under composition.

Theorem 2.5

For any element $E_{a,b} \in AF$, $E_{a,b}^{-1} = E_{(a^{-1}) \% 26, (-a^{-1}b) \% 26}$

Proof:

See [Garret, p.14] for the proof. The multiplicative inverse for $a \in A$ can be found using Table 10.

| | | | | | | | | | | | | |
|----------|---|---|----|----|---|----|----|----|----|----|----|----|
| a | 1 | 3 | 5 | 7 | 9 | 11 | 15 | 17 | 19 | 21 | 23 | 25 |
| a^{-1} | 1 | 9 | 21 | 15 | 3 | 19 | 7 | 23 | 11 | 5 | 17 | 25 |

Table 10

Also $-b = 26 - b$ is the additive inverse of b modulo 26 for $b \in B$.

Theorem 2.6

The operation of function composition for AF is associative.

Proof:

Let $E_{a,b}, E_{c,d}, E_{e,f} \in AF$.

According to Theorem 2.1,

$$E_{a,b} \circ (E_{c,d} \circ E_{e,f}) = E_{a,b} \circ E_{ce,cf+d}$$

$$E_{a,b} \circ (E_{c,d} \circ E_{e,f}) = E_{ace,a(cf+d)+b}$$

$$E_{a,b} \circ (E_{c,d} \circ E_{e,f}) = E_{ace,acf+ad+b}$$

Similarly,

$$(E_{a,b} \circ E_{c,d}) \circ E_{e,f} = E_{ac,ad+b} \circ E_{e,f}$$

$$(E_{a,b} \circ E_{c,d}) \circ E_{e,f} = E_{ace,acf+ad+b}$$

$$\text{So, } E_{a,b} \circ (E_{c,d} \circ E_{e,f}) = (E_{a,b} \circ E_{c,d}) \circ E_{e,f}.$$

Therefore, the function composition of AF is associative.

Now that it has been shown that AF is a group under the function composition, other interesting facts can be discovered. For instance, the order of the Affine function, $E_{25,25}$, in AF can be determined by using

Theorem 2.1 as follows:

$$E_{25,25}^2 = E_{25,25}(E_{25,25}(x)) = E_{(25*25)\%26, (25*25+25)\%26} = E_{625\%26, 650\%26} = E_{1,0}$$

The order of an element refers to the smallest positive integer power that the element must be raised to in order to obtain the identity element, which in this case is $E_{1,0}$. Thus, the order of $E_{25,25}$ is 2,

$$\text{since } E_{25,25}^2 = E_{1,0}. \text{ Symbolically we write, } \text{ord}(E_{25,25}) = 2.$$

The following theorem enables us to find the order of each Affine function in AF more quickly.

Theorem 2.7

$$E_{a,b}^n = E_{a^n, b \left(\frac{a^n - 1}{a - 1} \right)} \text{ when } a > 1 \text{ and } E_{a,b}^n = E_{a^n, nb} \text{ if } a=1$$

Proof:

$$E_{a,b} \circ E_{a,b} = E_{a^2, ab+b}$$

$$E_{a,b} \circ E_{a^2, ab+b} = E_{a^3, a^2b+ab+b}$$

$$E_{a,b} \circ E_{a^3, a^2b+ab+b} = E_{a^4, a^3b+a^2b+ab+b}$$

The pattern that is emerging is $E_{a,b}^n = E_{a^n,b(a^{n-1}+a^{n-2}+\dots+a^1+a^0)}$, which, using the formula for the sum of a geometric series, can be simplified to $E_{a,b}^n = E_{a^n,b\left(\frac{a^n-1}{a-1}\right)}$ when $a > 1$ and $E_{a,b}^n = E_{a^n,nb}$ if $a=1$.

The above can be formalized by using Mathematical Induction.

Using Theorem 2.7 we see that if $a > 1$, then $E_{a,b}^n$ will equal $E_{1,0}$ the first time when $a^n = 1$. That is, $ord(E_{a,b})$ in AF under composition is equal to $ord(a)$ in the group of invertible elements (units) mod 26 under multiplication modulo 26. The latter group is often denoted by $\langle U_{26}, \bullet_{26} \rangle$. It is easy to write a Python Program to compute $ord(a)$ for each $a \in U_{26}$. This in turn gives the $ord(E_{a,b})$ for each $E_{a,b} \in \langle AF, \circ \rangle$ with $a > 1$.

The following table gives $ord(a)$ for each $a \in U_{26}$.

| | | | | | | | | | | | |
|----------------|---|---|----|---|----|----|----|----|----|----|----|
| $a \in U_{26}$ | 3 | 5 | 7 | 9 | 11 | 15 | 17 | 19 | 21 | 23 | 25 |
| $ord(a)$ | 3 | 4 | 12 | 3 | 12 | 12 | 6 | 12 | 4 | 6 | 2 |

Table 11

The table above can be used as follows:

$$ord(15) = 12 \text{ in } \langle U_{26}, \bullet_{26} \rangle, \text{ therefore, } ord(E_{15,b}) = 12 \text{ in } \langle AF, \circ \rangle.$$

Here is the result performing a 12-round Affine cipher of $E_{15,7}$ to the plaintext, merryxmas.

```
>>> affine_e()
Enter the plaintext to be encrypted with the affine cipher:
merryxmas

Enter the first part of the key, a, where a is relatively prime to 26: 15
Enter the second part of the key, b, where 0<=b<=25 : 7
Repetitions? :12
```

```
-----
merryxmas
fpcdfhr
```

```
-----
fpcdfhr
eyllajeic
```

```
-----
eyllajeic
pdqhqmpxl
```

```
-----
pdqhqmpxl
```

```

yannifyoq
-----
yannifyoq
dhuuxedjn
-----
dhuuxedjn
aivvopamu
-----
aivvopamu
hxxkjyhfj
-----
hxxkjyhfj
iobbmdiek
-----
iobbmdiek
xjwwfaxpb
-----
xjwwfaxpb
omzzehoyw
-----
omzzehoyw
jfsspijdz
-----
jfsspijdz
merryxmas
>>>

```

Now consider the case where $a = 1$. According to Theorem 2.7, we see that if $a = 1$, then $E_{a,b}^n$ will equal $E_{1,0}$ the first time when $nb = 0$. That is, $ord(E_{a,b}^n)$ in $\langle AF, \circ \rangle$ is equal to $ord(b)$ in $\langle Z_{26}, +_{26} \rangle$. A Python program to calculate $ord(b)$ for each $b \in \langle Z_{26}, +_{26} \rangle$ produces the following table:

| b | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|----------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $ord(b)$ | 1 | 26 | 13 | 26 | 13 | 26 | 13 | 26 | 13 | 26 | 13 | 26 | 13 | 2 | 13 | 26 | 13 | 26 | 13 | 26 | 13 | 26 | 13 | 26 | 13 | 26 |

Table 12

From Table 11 and 12 it can be summarized that the order of an element in AF with $a > 1$ will be one of the numbers 2,3,4,6,12 and the order of an element in AF with $a=1$ will be one of the numbers 1,2,13,26.

Table 12 can be used as follows:

Since $ord(3) = 26$ in $\langle Z_{26}, +_{26} \rangle$, it follows that $ord(E_{1,3}) = 26$ in $\langle AF, \circ \rangle$.

Applying a 26-round Affine cipher of $E_{1,3}$ to the plaintext, helloworld, produces:

```

>>> affine_e()
Enter the plaintext to be encrypted with the affine cipher:
helloworld

```

```

Enter the first part of the key, a, where a is relatively prime to 26: 1
Enter the second part of the key, b, where 0<=b<=25 : 3

```

Repetitions? :26

helloworld
khoodruog

khoodruog
nkrrucuxrj

nkrrucuxrj
qnuuxfxaum

qnuuxfxaum
tqxxaiadxp

tqxxaiadxp
wtaadldgas

wtaadldgas
zwdgdgjdvd

zwdgdgjdvd
czggjrmgy

czggjrmgy
fcjmmumpjb

fcjmmumpjb
ifmmpxpsme

ifmmpxpsme
lippsasvph

lippsasvph
olssvdvysk

olssvdvysk
rovvygybvn

rovvygybvn
uryyjbeyq

uryyjbeyq
xubbemehbt

xubbemehbt
axeephkew

axeephkew
dahhksknhz

dahhksknhz
gdcknvnqkc

gdcknvnqkc
jgmnqyqtnf

jgnnqyqtnf
mjqqbtwqi

mjqqtbtwqi
pmttwewztl

pmttwewztl
spwwzhzcwo

spwwzhzcwo
vszzckcfzr

vszzckcfzr
yvccfnficu

yvccfnficu
byffiqilfx

byffiqilfx
ebiiltloia

ebiiltloia
helloworld
>>>

Theorem 2.8

$\langle AF, \circ \rangle$ is not an Abelian group, in other words, Affine ciphers are not commutative.

Proof:

Let $E_{a,b} = E_{5,8}$ and $E_{c,d} = E_{19,4}$ where $E_{a,b}, E_{c,d} \in AF$

$$E_{a,b} \circ E_{c,d} = E_{ac,ad+b} = E_{17,2}$$

$$E_{c,d} \circ E_{a,b} = E_{ca,cb+d} = E_{17,0}$$

$$E_{17,2} \neq E_{17,0}$$

Therefore, $E_{a,b} \circ E_{c,d} \neq E_{c,d} \circ E_{a,b}$, so Affine ciphers are not commutative, and $\langle AF, \circ \rangle$ is not an Abelian group.

Theorem 2.9

$\langle AF, \circ \rangle$ is not a cyclic group.

Proof:

According to Charles Pinter, every cyclic group is abelian [Pinter, p. 116]. From Theorem 2.8, it is evident that $\langle AF, \circ \rangle$ is not abelian, and therefore, is not a cyclic group.

Finding Inverses

Many ciphers require the use of inverses in the decryption formulas, such as the Hill cipher and the Affine cipher. The decryption equation for the Affine cipher is $P = (a^{-1} * C - a^{-1} * b) \bmod 26$, where

$C = (a * P + b) \bmod 26$ and $a, a^{-1} \in A, b \in B$. It is simple to see how this decryption equation is derived; the encryption equation is used to solve for P .

$$C = a * P + b$$

$$C - b = a * P + b - b$$

$$a^{-1} * (C - b) = a^{-1} * (a * P)$$

$$a^{-1} * (C - b) = (a^{-1} * a) * P$$

$$a^{-1} * C - a^{-1} * b = P$$

Because the numbers must be integers in the range of a chosen alphabet, instead of using a fraction when dividing a , a^{-1} must be used. Modular arithmetic also plays a large part in these equations. All integers, except zero, have multiplicative inverses that can be written as $\frac{1}{n}$. Since the use of fractions is not an option, a^{-1} must be a number that satisfies $aa^{-1} \bmod 26 = a^{-1}a \bmod 26 = 1 \bmod 26$. This way a can be divided from both sides of the equation without corrupting the requirements of the numbers.

The Hill cipher also operates under the same principle of the Affine cipher. A key matrix must be chosen so that an inverse for it can be found since not all matrices are invertible. The inverse of a matrix must be a matrix that when multiplied with the original matrix yields the identity matrix, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

There are several different techniques for finding the inverse of an invertible matrix. For a 2x2 invertible matrix this formula can be used: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$. For a 3x3 invertible matrix there is a complicated equation that is much more involved.

For an $n \times n$ invertible matrix, Gaussian Elimination will always work. This method also employs the identity matrix, which for any sized matrix contains all 0's except for the diagonal, which contains 1's. The

first step is to place the invertible matrix next to the identity matrix, like this example: $\begin{bmatrix} 2 & 5 & 3 \\ 6 & 8 & 1 \\ 4 & 7 & 9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The goal is to modify the first matrix into an identity matrix by using simple row operations. Row operations can be multiplying a row by a constant, multiplying a row by a constant and adding it to another row, or swapping rows. Whenever a row operation is made, it is not only made to the first matrix, but to the adjacent identity matrix. When the first matrix is modified into an identity matrix, the former identity matrix can now be used as the inverse matrix.

This process can become long and tedious, especially with large matrices. A relatively simple way to calculate the inverse of a matrix is to first create a matrix whose inverse is easy to find. This can be done using triangular matrices, which are matrices that contain only zeroes above or below the diagonal. For

example, this matrix $\begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{bmatrix}$ is an upper triangular matrix, where x can be any number. A lower

triangular matrix looks like this: $\begin{bmatrix} x & 0 & 0 \\ x & x & 0 \\ x & x & x \end{bmatrix}$. To efficiently implement the use of triangular matrices in the

Hill cipher it is best to create a lower triangular matrix that contains only 1's on the diagonal, 0's above the diagonal, and randomly generated numbers below the diagonal. Second, find the transpose of the lower triangular matrix, which will be an upper triangular matrix, and then multiply the lower triangular matrix by its transpose, LL^t . The resulting matrix is a valid key, $K=LL^t$.

Then, to calculate the inverse, use the equation $K^{-1} = (LL^t)^{-1} = (L^t)^{-1} L^{-1} = (L^{-1})^t L^{-1}$. So, you must find the inverse of the lower triangular matrix and multiply the transpose of the inverse matrix by the inverse matrix. There are other ways to simplify the inverse calculation process, but I actually use this method in one of my programs that will be discussed later.

Inverses of Triangular Matrices

By using the Gauss-Jordan method, it is quite easy to find the inverse of a triangular matrix. For example,

let's consider the upper triangular matrix, $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix}$. To find the inverse using the Gauss-Jordan method,

would require three row operations.

$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 4 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad -4r_3 + r_2 \rightarrow r_2$$

$$\begin{bmatrix} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & -4 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad -3r_3 + r_1 \rightarrow r_1$$

$$\begin{bmatrix} 1 & 2 & 0 & 1 & 0 & -3 \\ 0 & 1 & 0 & 0 & 1 & -4 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad -2r_2 + r_1 \rightarrow r_1$$

$$\begin{bmatrix} 1 & 0 & 0 & 1 & -2 & 5 \\ 0 & 1 & 0 & 0 & 1 & -4 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, the inverse of $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix}$ is $\begin{bmatrix} 1 & -2 & 5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$. To ensure that this is indeed the inverse, you can

multiply the two matrices together to yield the identity matrix. This method would be fairly simple to create a program for, but there is another interesting way to find the inverse of these special matrices.

From one matrix multiplication, it is possible to find the inverse of a 3x3 triangular matrix with diagonals composed of only 1's. For example:

$$\begin{bmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

As you can see, the second column of the first matrix is the same as the second column of the original matrix, except that the number above the diagonal has been made negative. The same is true for the third column of the second matrix. When the two matrices are multiplied together, they yield the inverse matrix.

To determine which method is more efficient, it is imperative to calculate the total number of multiplications and additions required. For the Gauss-Jordan method example above, there were eighteen multiplications and eighteen additions required. For the above matrix multiplication, there were twenty-seven multiplications and eighteen additions are required. So, in this instance the matrix multiplication method is less efficient than the Gauss-Jordan method. If one uses the fact that the product of two upper triangular matrices each with 1's on the diagonal is an upper triangular matrix with 1's on the diagonal, the number of calculations can be somewhat reduced. For the 3x3 matrices, there are only nine multiplications and six additions required to obtain the inverse. However, the main attraction of the matrix multiplication over the Gauss-Jordan method is that it is easier to program, especially if you already have an algorithm to multiply matrices together. The only tricky part is multiplying matrices in the correct order. For example, if we had attempted to multiply the matrices above in the reverse order we would have obtained,

$$\begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & -2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -2 & -3 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$$

which is not the correct inverse matrix.

So, it is very important to multiply the matrices in the correct order. One way to verbally explain how to create the correct order is to think of the order needed to correctly employ the Gauss-Jordan method. The first calculations involved changing the 4 and 3 and then changing the 2. Therefore, we can assume that the rightmost matrix in the product should be the one containing the numbers -4 and -3 and the leftmost matrix in the product should be the one containing the number -2.

Now let's consider how this would work for a lower triangular matrix.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 \\ 4 & 5 & 6 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & -3 & 1 & 0 \\ -5 & 13 & -6 & 1 \end{bmatrix}$$

To calculate this inverse using the column-multiplication method, would require two matrix multiplications as illustrated below.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -6 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & -5 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -2 & 0 & 1 & 0 \\ -4 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & -3 & 1 & 0 \\ -5 & 13 & -6 & 1 \end{bmatrix}$$

If the Gauss-Jordan method was used to find the inverse of the matrix, the first changes would involve changing the 1, 2, and 4 below the 1 on the main diagonal in the first column. Therefore, the rightmost matrix in the matrix product should be the matrix with -1, -2, and -4 in the first column. The second changes when using the Gauss-Jordan method would involve changing the 3 and 5 below the 1 on the main diagonal in the second column. Therefore the second matrix from right in matrix product should be the matrix with -3 and -5 in the second column. Finally, the last changes when using the Gauss-Jordan method would involve changing the 6 in the third column. Therefore, the third matrix in the matrix product from the right should be the matrix with a -6 in the third column. The latter matrix is really the first matrix on the left in the matrix product.

Hill Cipher Key Space

Finding the size of the key space for the Hill cipher in the way it is encoded in my program depends on the size of the matrix. Since the key matrix is determined from a lower triangular matrix and its transpose, we can consider only the lower triangular matrix to establish the key space.

$$\begin{bmatrix} 1 & 0 \\ x & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ x & 1 & 0 \\ x & x & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ x & 1 & 0 & 0 \\ x & x & 1 & 0 \\ x & x & x & 1 \end{bmatrix}$$

All of these lower triangular matrices are examples of what could be used in my program. The x 's represent any non-zero integer within the chosen alphabet Z_{26} . Therefore, the 2x2 matrix has a key space size of 26 since there are 26 possible numbers that can be used. For the 3x3 matrix, there are $26^3 = 17,576$ possible key matrices, and for the 4x4 matrix, there are $26^6 = 308,915,776$ possible key matrices. The formula to obtain the number of key matrices for an $n \times n$ matrix is $26^{\frac{n(n-1)}{2}}$.

This formula only works for the type of matrices that are used within my program; there are many more invertible matrices that my program will not create. There is a formula to find the total number of $n \times n$ invertible matrices whose elements are from Z_p , where p is a prime number. If the size of the alphabet, p , is prime and $g(p, n)$ is the number of invertible matrices of size $n \times n$, then

$g(p, n) = (p^n - 1)(p^n - p)(p^n - p^2) \dots (p^n - p^{n-1})$ [Bauer, 81]. Since the alphabet that we are using is of size 26, which is not a prime number, let us consider this for an alphabet of size 29 on a 2x2 matrix. According to the above formula, the number of invertible 2x2 matrices is

$g(29, 2) = (29^2 - 1)(29^2 - 29) = 682,080$. Now, if we use my formula from my program, the number of invertible matrices would be $29^{\frac{2(2-1)}{2}} = 29$.

| Nxn | Total Number of Invertible Matrices | Total Number of Program-Generated Invertible Matrices |
|-----|---|---|
| 3x3 | $g(29,3) = (29^3 - 1)(29^3 - 29)(29^3 - 29^2) = 1.4 * 10^{13}$ | $29^{\frac{3(3-1)}{2}} = 24389$ |
| 4x4 | $g(29,4) = (29^4 - 1)(29^4 - 29)(29^4 - 29^2)(29^4 - 29^3) = 2.41 * 10^{23}$ | $29^{\frac{4(4-1)}{2}} = 5.95 * 10^8$ |
| 5x5 | $g(29,5) = (29^5 - 1)(29^5 - 29)(29^5 - 29^2)(29^5 - 29^3)(29^5 - 29^4) = 3.50 * 10^{36}$ | $29^{\frac{5(5-1)}{2}} = 4.21 * 10^{14}$ |

Table 13

The program using triangular matrices does not create as many invertible matrices as are available because it only generates one lower triangular matrix, L , with ones on the diagonal to compute the key matrix, $K=LL'$.

A way to generate more invertible matrices would be to randomly generate two different triangular matrices, a lower triangular matrix, L , and an upper triangle matrix, U , both with ones on the diagonal, and multiply them together for the key matrix. This method would square the number of matrices created in the

program but would create more work for the program to do in calculating inverses. Instead of calculating one inverse, an inverse matrix would have to be calculated for each triangular matrix to obtain the key inverse matrix of $K^{-1} = (LU)^{-1} = U^{-1}L^{-1}$.

Composition Program

As I mentioned earlier, the program I wrote in the Python language contains the Shift, Affine, Vigenere, Hill, and RSA cipher. It combines the five in any order or frequency desired. I propose that using such a system to encode messages could be very secure. To demonstrate, I will give an example encryption from my program and show how an unauthorized decoder would attack it.

```
>>> comp_e()
Enter the plaintext to be compositely encrypted:
Thanksgiving is my favorite holiday
Please enter the encryption methods you would like to use, in the order you want, separated by commas.
```

- 1 - Affine
- 2 - Hill
- 3 - RSA
- 4 - Shift
- 5 - Vigenere

```
Example: 1, 3, 5, 2
1,5,4
Affine Routine
Choose a number from the set {1,3,5,7,9,11,15,17,19,21,23,25} :
5
Choose a number from 0 to 25 :
9
Vigenere Routine
Enter the keyword :
mcmurry
Shift Routine
Enter the key, k, where 0 < k <= 25:
14
AIJEMAZXAXESCHRPIRPGCXQDAGRJYZZ
>>>
```

This particular encryption first Affine-encrypted the plaintext, “Thanksgiving is my favorite holiday”, with the key (5,9). Then the resulting message was then Vigenere-encrypted with the key, McMurry; following this, it was Shift-encrypted with the key 14. The resulting ciphertext appears as “AIJEMAZXAXESCHRPIRPGCXQDAGRJYZZ”.

Now, the difficult task of breaking the code can begin. First, we will determine the key space for this particular program. We will assume that the unauthorized decoder will know that this message was compositionally encrypted with my program. Although this may not be the case, it is always best to assume that your enemies are aware of all your cryptology techniques. Another assumption that would be convenient to make is to assume that the adversary knows that three ciphers are being used only once. Therefore, the number of possibilities of the order in which we could have used them is equal to $3! = 6$. It is known that the size of the key space for the Shift cipher is 25 and the size of the key space for the Affine cipher is 311. Since McMurry is a seven-letter word, the size of the key space for the Vigenere cipher is $26^7 = 8,031,810,176$. Therefore, the total size of the key space is $6 * 25 * 311 * 8,031,810,176 = 374,683,944,700,000$.

However, there are other methods that can be used to attack ciphers. To determine whether or not the ciphertext has been encrypted with a mono-alphabetic or a poly-alphabetic cipher, the Friedman test should be used. The Friedman test uses the Index of Coincidence, which is the probability of two randomly

selected letters being identical and is equal to $\sum_{i=1}^{26} \frac{n_i(n_i - 1)}{n(n - 1)}$ where n is equal to the number of letters in the

text and n_i is equal to the number of specific letters in the text. For instance, n_0 is the number of a's in the text. The IC for the English language based on the frequencies from Table 2 is .065 and the IC for a 26 letter alphabet in which every letter has the same frequency is .038. Therefore, a mono-alphabetic cipher, the Index of Coincidence will be close to .065, and for a poly-alphabetic cipher, the Index of Coincidence will be close to .038 [Lewand, p.87].

In this instance, there are 31 letters in our ciphertext, and we can run a number frequency program I wrote to determine the frequency of each letter in the text.

```
>>> count()
Enter text to checked for frequency distribution:
aijemazxaxeschrpirpgcxqdagrjyzz
a 4
b 0
c 2
d 1
e 2
f 0
g 2
h 1
i 2
j 2
k 0
l 0
m 1
n 0
o 0
p 2
q 1
r 3
s 1
t 0
u 0
v 0
w 0
x 3
y 1
z 3
0
>>>
```

Therefore, $IC = \frac{4 * 3}{31 * 30} + 3 \left(\frac{3 * 2}{31 * 30} \right) + 6 \left(\frac{2 * 1}{31 * 30} \right) + 6 \left(\frac{1 * 0}{31 * 30} \right) + 10 \left(\frac{0 * -1}{31 * 30} \right) = \frac{42}{930} = .04516$. This number

is much closer to .038 than to .065, so the unauthorized decoder can assume that we have used a poly-alphabetic cipher such as Vigenere, Hill, or RSA cipher somewhere in our composition.

Another use of the Kasiski test is to determine the length of the keyword, if we assume that our adversary thinks we are using the Vigenere cipher. If a message of length n and the Index of Coincidence, IC , is

encoded using a Vigenere cipher, then r , the length of the keyword is $r \approx \frac{.027n}{(n-1)IC-.038n+.065}$ [Lewand,

p.92]. Using our information, $r \approx \frac{.027 * 31}{(30).04516-.038 * 31+.065} = 3.46$. Since we already know that our

keyword has seven letters, the Kasiski test has failed to approximate closely to the actual length. This occurs because our message has a very short length. If the message were longer, say, 300 letters or so, this test would work fairly accurately. Therein lies the importance of the Unicity Distance concept and keeping the message as short as possible.

If our opponent were able to determine the length of the keyword, he would separate the ciphertext into groups of seven and arrange them in table-like form:

| | | | | | | |
|---|---|---|---|---|---|---|
| A | I | J | E | M | A | Z |
| X | A | X | E | S | C | H |
| R | P | I | R | P | G | C |
| X | Q | D | A | G | R | J |
| Y | Z | Z | | | | |

If the message were long enough, he would then determine the letter frequencies of each ‘column’ since each column is Shift encrypted with its corresponding keyword letter. However, because our message is so short, the letter frequencies will not reveal the correct message. Even if the message was long and letter frequencies could be useful, since the Vigenere-encrypted message has been sandwiched between a Shift and Affine-encryption, there would be no way for the opponent to recognize familiar words, which is the usual method that cryptanalysts use in breaking the Vigenere Cipher.

Therefore, this encryption proves to be very secure because even the complicated methods that can be used to aid in decryption cannot be used to fully decrypt the message. The security in using the composition program is in using several different cipher systems in a random order. Many methods that are used to break ciphers, such as the Friedman test, would not work effectively because ideally, the ciphers would be combined in a way to create Shannon’s ‘confusion and diffusion’.

However, efficiency is always crucial to using ciphers, and sharing the key privately with an authorized user ensures the security of the cipher. Most likely, the key and order in which the ciphers are composed would be encrypted by another secure cipher like DES or AES. Another issue to be faced can occur in the authorized decryption. If one mistake is made in the beginning of the decryption, perhaps a key is read incorrectly, then an avalanche effect would take place and the plaintext would never be able to appear. While this makes it even more secure against unauthorized cryptanalysts, it can also make it inconvenient for intended recipients.

Conclusion

The majority of important information is now encoded with ciphers such as RSA, DES, AES, etc. These ciphers are fairly complex, but secure and efficient. Classical ciphers are not used except in elementary cases when secrecy is not vital. However, classical ciphers may still have some place in today’s secrecy society through composition. The program I have written can be used to formulate some very secure ciphers that may only be deciphered through a brute force attack, because the message must be decrypted in a specific order and with specific keys for any plaintext to become recognizable. Although the program would probably never be used for a large-scale operation because of possible security issues, the “pastry dough mixing” effect can create a very secure cipher.

Appendix

Theorem A.1

$$(x + y) \% m = (x \% m + y \% m) \% m$$

Proof:

According to the division algorithm [Pinter, p. 213] there exists unique integers q_1, r_1, q_2, r_2, q , and r such that:

$$x = q_1 m + r_1 \text{ where } 0 \leq r_1 < m$$

$$y = q_2 m + r_2 \text{ where } 0 \leq r_2 < m$$

$$x + y = qm + r \text{ where } 0 \leq r < m$$

The above equations imply that:

$$r_1 - x = -q_1 m$$

$$r_2 - y = -q_2 m$$

$$r - (x + y) = -qm$$

Therefore,

$$r_1 \equiv x \% m$$

$$r_2 \equiv y \% m$$

$$r \equiv (x + y) \% m$$

From the first set of equations, it follows that

$$qm + r = (q_1 m + r_1) + (q_2 m + r_2)$$

$$r_1 + r_2 = qm - q_1 m - q_2 m + r$$

$$r_1 + r_2 = m(q - q_1 - q_2) + r$$

$$r - (r_1 + r_2) = -m(q - q_1 - q_2)$$

$$r \equiv (r_1 + r_2) \% m$$

$$(x + y) \% m \equiv (x \% m + y \% m) \% m$$

Theorem A.2

$$((x \% m) - (y \% m)) \% m = (x - y) \% m$$

According to the division algorithm there exists unique integers q_1, r_1, q_2, r_2, q , and r such that:

$$x = q_1 m + r_1 \text{ where } 0 \leq r_1 < m$$

$$y = q_2 m + r_2 \text{ where } 0 \leq r_2 < m$$

$$x - y = qm + r \text{ where } 0 \leq r < m$$

The above equations imply that:

$$r_1 - x = -q_1 m$$

$$r_2 - y = -q_2 m$$

$$r - (x - y) = -qm$$

Therefore,

$$r_1 \equiv x \% m$$

$$r_2 \equiv y \% m$$

$$r \equiv (x - y) \% m$$

From the first set of equations, it follows that

$$\begin{aligned}
qm + r &= (q_1m + r_1) - (q_2m + r_2) \\
qm + r &= q_1m + r_1 - q_2m - r_2 \\
r - (r_1 - r_2) &= q_1m - q_2m - qm \\
r - (r_1 - r_2) &= (q_1 - q_2 - q)m \\
r &\equiv (r_1 - r_2) \% m \\
(x - y) \% m &\equiv (x \% m - y \% m) \% m
\end{aligned}$$

Theorem A.3

$$((x \% m)(y \% m)) \% m = (xy) \% m$$

According to the division algorithm there exists unique integers q_1, r_1, q_2, r_2, q , and r such that:

$$x = q_1m + r_1 \text{ where } 0 \leq r_1 < m$$

$$y = q_2m + r_2 \text{ where } 0 \leq r_2 < m$$

$$xy = qm + r \text{ where } 0 \leq r < m$$

The above equations imply that:

$$r_1 - x = -q_1m$$

$$r_2 - y = -q_2m$$

$$r - xy = -qm$$

Therefore,

$$r_1 \equiv x \% m$$

$$r_2 \equiv y \% m$$

$$r \equiv (xy) \% m$$

From the first set of equations, it follows that:

$$qm + r = (q_1m + r_1)(q_2m + r_2)$$

$$qm + r = q_1q_2m^2 + q_1mr_2 + q_2mr_1 + r_1r_2$$

$$r_1r_2 = qm - q_1q_2m^2 - q_1mr_2 - q_2mr_1 + r$$

$$r_1r_2 = m(q - q_1q_2m - q_1r_2 - q_2r_1) + r$$

$$r - r_1r_2 = -m(q - q_1q_2m - q_1r_2 - q_2r_1)$$

$$r \equiv (r_1r_2) \% m$$

$$(xy) \% m \equiv ((x \% m)(y \% m)) \% m$$

Composition Program (not available online)

References

- Barr, Thomas H. Invitation to Cryptology. Prentice Hall: New Jersey, 2002.
- Bauer, Friedrich L. Decrypted Secrets. Springer: New York, 2002.
- Caldwell, Chris. “How Many Primes are There?” <http://primes.utm.edu/howmany.shtml#hist>
- Dupuis, Clement. “A Short History of Crypto” Apr 1999, http://www.jproc.ca/crypto/crypto_hist.html
- Garrett, Paul. Making, Breaking Codes. Prentice Hall: New Jersey, 2001.
- Lewand, Robert Edward. Cryptological Mathematics. The Mathematical Association of America:
Washington, D.C., 2000.
- Pinter, Charles C. A Book of Abstract Algebra: Second Edition. McGraw-Hill: New York, 1990.
- Stinson, Douglas. Cryptography: Theory and Practice. CRC Press, 1995.
- Weisstein, Eric W. “RSA-200 Factored”. May 2005, <http://mathworld.wolfram.com/news/2005-05-10/rsa-200/>