The Hill Cipher
A Linear Algebra Perspective

Contents

1 Introduction to Classical Cryptography

Cryptography is defined to be the process of creating ciphers such that when applied to a message it hides the meaning of the message. Cryptanalysis is the process of breaking the cipher and discovering the meaning of the message. Finally, Cryptology is the study of both Cryptography and Cryptanalysis. The terms Cryptography and Cryptology have become synonymous over the years.

Classical cryptography refers to encryption methods that are no longer in use today. Modern cryptography refers to encryption methods that are currently in use. The division between classical and modern is the invention of the computer. The classical methods were strong enough to withstand attacks by humans but are easily broken with high-speed computers, hence they are not in use today. There are many classical methods and there is a rich history of cryptography and cryptanalysis that we do not have the time to explore here. For a good, and mostly non-mathematical, introduction to cryptography please take some time to read *The Code Book* by Simon Singh [5] or Fred Wrixon's book *Codes Ciphers & Other Cryptic & Clandestine Communication* [9]. A more encyclopedic treatment of classical cryptography can be found in David Kahn's book, *The Codebreakers* [2]. For something more mathematical, but very readable, please see the first couple chapters of *An Introduction to Cryptography with Coding Theory* by Wade Trappe and Lawrence Washington [7].

The Hill Cipher was developed by Lester Hill in 1929, setting it firmly in the classical age of cryptography. Lester Hill was a professor at Hunter College in New York City and first published this method in the American Mathematical Monthly with his article *Cryptography in an Algebraic Alphabet* [1]. Although it seems that this method was not used much in practice, it marked the transition cryptography made from a mainly linguistic practice to a mathematical discipline. Prior to World War II most cryptographic and cryptanalysis methods centered around replacing characters in a message with different characters (using one or more alphabets) and mixing up or rearranging the message. Hence the code breakers were primarily people who were highly trained in linguistics, could speak several languages, and were good puzzle solvers. With the invention of the Enigma machine, used by the German's in World War II, cryptanalysis of these ciphertexts required advanced mathematics and an enormous amount of computation, far beyond that of a single person or group of people.

1.1 Alice, Bob & Eve

We will discuss some of the basic concepts of cryptography and define some terms we will need. All classical cryptography techniques were what are called symmetric-key techniques. That is, the sender and receiver of the message had an encryption and decryption key that they shared and no one else knew what that key was. The security of the message depended on two things, keeping the key a secret between the sender and receiver and constructing

both an encryption method and a key for that method that would be difficult for someone to break. In fact, what was the most important aspect of this was the key. This is known as Kerckhoff's principle, which essentially says that one must always assume that the enemy knows the method of encryption and that the strength of the encryption is the strength of the key.

Three people you need to know in the world of cryptography are Alice, Bob and Eve. Alice is the sender of the message, Bob is the receiver of the message and Eve is the eavesdropper.

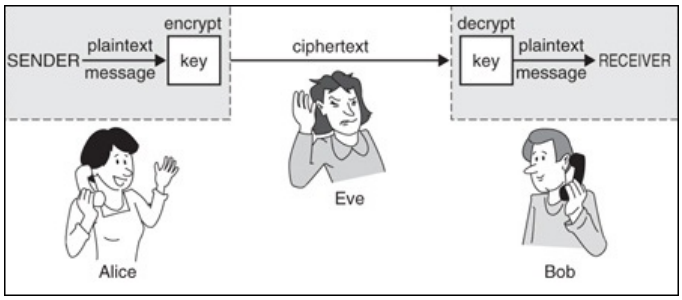


Figure 1: Symmetric-Key Cryptography

In symmetric-key cryptography, the traditional method and the only one used before 1977, Alice and Bob would share an encryption and decryption key that only the two of them knew. When Alice wanted to send a message (plaintext) to Bob she would use the key to encrypt the message into ciphertext, she would then send the ciphertext to Bob where he would use the decryption key to decrypt the message back into plaintext and read what Alice had to say. In all transmissions we assume that a third person, Eve, could and does intercept the message. Now Eve does not have the key, she only has the ciphertext. It is her job to break the code either by finding the key and decrypting the message or by finding the meaning of the message without finding the entire key.

1.2 Types of Attacks

Eve wishes to attack the system. She has several goals in mind but her primary one is to find the encryption and decryption key for the message and then decrypt the message so that she knows the meaning of the message. There are four standard attacks on a cryptographic method, They are,

Ciphertext Only: Eve has only a copy of the ciphertext.

Known Plaintext: Eve has a copy of the ciphertext and the corresponding plaintext from the message or another message she suspects was encrypted with the same key. Obviously, she would not have the plaintext of the entire message or there would be no reason to decrypt the message. For example, suppose that Alice always started her letters to Bob with "Dear Bob," then Eve knows what the first several letters of message and has the corresponding ciphertext. This is called a *crib*, a portion of the plaintext message. On the surface it seems that knowing only seven letters would be of little use to Eve, but in many cases this would be enough information to construct the entire key.

Even with very complicated ciphers, like the German Enigma, this amount of information was useful. In one particular case, during World War II, shortly after 6 AM every morning the Germans sent an encrypted radio transmission for the day's weather. So it was certain that the word WETTER, the German word for weather, would be in that transmission. Also, with the consistency of military-type transmissions it was not too difficult to figure out where that word would be in each transmission.

Chosen Plaintext: Eve temporally gains access to the encryption machine, she carefully chooses some plaintext messages, sends them through the machine to obtain the corresponding ciphertexts. Now she can do a known plaintext attack. If she chooses her plaintext messages well she will have an easier time in finding the key.

Chosen Ciphertext: Eve temporally gains access to the decryption machine, carefully chooses some ciphertexts, sends them through the machine to obtain the corresponding plaintexts. Now she can do a known plaintext attack. Again, if she chooses her ciphertext messages well she will have an easier time in finding the key.

Clearly, the ciphertext only attack is the most difficult since it relies on the least amount of information. Different cryptographic methods have their own particular strengths and weaknesses. So for some methods a chosen plaintext attack will work better and for others a chosen ciphertext attack is preferred. For the Hill Cipher we will be doing known plaintext attacks on the system to find the key.

1.3 Block Ciphers

In a simple substitution cipher, where each letter of the plaintext is replaced with some other letter, changing one letter in the plaintext changes only one letter in the ciphertext. This is a substantial weakness that usually makes finding the key to the cipher fairly easy. One way around this weakness is to encrypt several characters at once. For example, we could take the plaintext message and break it into blocks of 5 characters and then encrypt each of the blocks. To do this we would need to devise a method that encrypts and decrypts 5 characters at one time and not just one character at a time.

A block cipher is simply any cipher that encrypts and decrypts blocks of characters instead of just a single character at a time. The Hill cipher is such a cipher. In fact, as we will see we could devise a Hill cipher to encrypt as many characters as we want at one time.

2 A Quick Reminder on Modular Arithmetic

As we will see shortly, the Hill Cipher is a linear algebra technique but it relies on modular arithmetic. So we will give a quick reminder on modular calculations. Throughout the discussion we will let n be the modulus, so n will be an integer and $n \geq 2$.

2.1 Definition

Definition 1: *Let m and k be two integers, then we say that $m \equiv k \pmod{n}$, if $n|(m-k)$, that is, n divides the quantity $m-k$ evenly.*

Example 1: A few examples of $m \equiv k \pmod{n}$,

1. $12 \equiv 2 \pmod{5}$ since $\frac{12-2}{5} = 2$.
2. $156 \equiv 3 \pmod{17}$ since $\frac{156-3}{17} = 9$.
3. $-24 \equiv 10 \pmod{17}$ since $\frac{-24-10}{17} = -2$.

The next definition is really more of a convention than it is a definition.

Definition 2: *Let k be an integer, when we write $k \pmod{n}$, we mean the number m that is in the range $0 \leq m < n$ such that $m \equiv k \pmod{n}$.*

Example 2: A few examples of $k \pmod{n}$,

1. $12 \pmod{5} \equiv 2$
2. $156 \pmod{17} \equiv 3$
3. $-24 \pmod{17} \equiv 10$

2.2 Arithmetic

Addition, subtraction, and multiplication modulo a number is fairly straight forward. Simply do the addition, subtraction, or multiplication as usual then take the result mod the number.

Example 3: A few examples of modular arithmetic,

1. $34 + 12 \pmod{5} \equiv 46 \pmod{5} \equiv 1$
2. $15 - 23 \pmod{17} \equiv -8 \pmod{17} \equiv 9$
3. $24 \cdot 13 \pmod{17} \equiv 312 \pmod{17} \equiv 6$

When it comes to division we need to be a bit more careful. We also need to consider what division is. When we write the fraction $\frac{2}{3}$ we really mean 2 times the "inverse" of 3, that is, $2 \cdot \frac{1}{3}$. And what is the "inverse" of 3? Well when we are doing arithmetic using real numbers we represent the "inverse" of 3 as $\frac{1}{3}$ so that, with our rules of algebra, when we multiply 3 by its inverse we get 1, that is, $3 \cdot \frac{1}{3} = 1$. Why do we want the result to be 1? Answer, since 1 is the multiplicative identity in the real number system, that is $1 \cdot a = a$ for all real numbers a. The general concept of an inverse, specifically a multiplicative inverse, is the same. To invert a number a in a particular number system we want another b in that same system such that when we multiply a and b together we get the multiplicative identity in that system.

Definition 3: *Let k be an integer, then $k^{-1} \pmod{n}$ is the number m that is in the range $0 \leq m < n$ such that $m \cdot k \pmod{n} \equiv 1$, if m exists. If no such number m exists then we say that k does not have an inverse modulo n or that k is not invertible mod n.*

Example 4: A few examples of $k^{-1} \pmod{n}$,

1. $2^{-1} \pmod{5} \equiv 3$ since $2 \cdot 3 \pmod{5} \equiv 6 \pmod{5} \equiv 1$
2. $12^{-1} \pmod{17} \equiv 10$ since $12 \cdot 10 \pmod{17} \equiv 120 \pmod{17} \equiv 1$
3. $11^{-1} \pmod{26} \equiv 19$ since $11 \cdot 19 \pmod{26} \equiv 209 \pmod{26} \equiv 1$
4. $4^{-1} \pmod{26}$ does not exist since there is no number b between 0 and 25 with $4 \cdot b \pmod{26} \equiv 1$

One obvious question at this point would be, when does a number have an inverse and when does it not have an inverse mod n? For the real number system this is an easy question since we know that 0 is the only number you cannot invert, but the question is a little more difficult when working modulo n. The next theorem will give us the answer, we will not prove this result, you can find a proof in any discrete mathematics text of elementary number theory text.

Theorem 1: *Let k be an integer, then $k^{-1} \pmod{n}$ exists if and only if the greatest common divisor of k and n is 1, that is, $\gcd(k, n) = 1$. Another way to say this is that k and n are relatively prime.*

From our last example,

Example 5:

1. $2^{-1} \pmod{5}$ exists since $\gcd(2, 5) = 1$
2. $12^{-1} \pmod{17}$ exists since $\gcd(12, 17) = 1$
3. $11^{-1} \pmod{26}$ exists since $\gcd(11, 26) = 1$

4. $4^{-1} \pmod{26}$ does not exist since $\gcd(4, 26) = 2 \neq 1$

Now that we know what is necessary and sufficient for a number to have an inverse modulo n the next question is if an inverse exists, how do you find it? There is an algorithm for doing this, it is the Extended Euclidean Algorithm. The process is not difficult but it is a bit tedious and does not add anything to our discussion. One can find detailed discussions of the algorithm and why it works in any elementary number theory textbook. In the section on technology we will go over the commands to find modular inverses using a number of different computer algebra systems.

2.3 Matrices

Our next goal is to generalize the above modular calculations to matrices. Taking a matrix that has only integer entries mod n is simple, we just take each entry mod n. For example,

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \pmod 5 \equiv \begin{bmatrix} 1 & 4 & 2 \\ 2 & 0 & 3 \\ 3 & 1 & 4 \end{bmatrix}$$

Addition, subtraction, and multiplication mod n is just as easy. First do the addition, subtraction, or multiplication as you would normally and then take the result mod n, that is, take each entry mod n. For example,

Example 6:

1.
$$\begin{bmatrix} 2 & 1 & -1 \\ 3 & 5 & 2 \end{bmatrix} + \begin{bmatrix} 2 & -1 & 5 \\ -4 & -7 & 9 \end{bmatrix} \pmod 5 \equiv \begin{bmatrix} 4 & 0 & 4 \\ -1 & -2 & 11 \end{bmatrix} \pmod 5$$
$$\equiv \begin{bmatrix} 4 & 0 & 4 \\ 4 & 3 & 1 \end{bmatrix} \pmod 5$$

2.
$$\begin{bmatrix} 2 & 1 & -1 \\ 3 & 5 & 2 \end{bmatrix} - \begin{bmatrix} 2 & -1 & 5 \\ -4 & -7 & 9 \end{bmatrix} \pmod 5 \equiv \begin{bmatrix} 0 & 2 & -6 \\ 7 & 12 & -7 \end{bmatrix} \pmod 5$$
$$\equiv \begin{bmatrix} 0 & 2 & 4 \\ 2 & 2 & 3 \end{bmatrix} \pmod 5$$

3.
$$\begin{bmatrix} 2 & 1 & -1 \\ 3 & 5 & 2 \end{bmatrix} \begin{bmatrix} 2 & -4 \\ -1 & -7 \\ 5 & 9 \end{bmatrix} \pmod 5 \equiv \begin{bmatrix} -2 & -24 \\ 11 & -29 \end{bmatrix} \pmod 5$$

$$\equiv \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix} \pmod 5$$

As you probably expected, matrix inverses are a little more difficult. When we compute a matrix inverse by hand we usually use the reduction algorithm where we place the matrix beside the identity, reduce, and then extract what the identity had become. For example, say we wanted to find the inverse of

$$\begin{bmatrix} 2 & 1 & -1 \\ 3 & 5 & 2 \\ -4 & -7 & 9 \end{bmatrix}$$

First we set up the matrix,

$$\begin{bmatrix} 2 & 1 & -1 & 1 & 0 & 0 \\ 3 & 5 & 2 & 0 & 1 & 0 \\ -4 & -7 & 9 & 0 & 0 & 1 \end{bmatrix}$$

Reduce this matrix,

$$\begin{bmatrix} 1 & 0 & 0 & \frac{59}{84} & -\frac{1}{42} & \frac{1}{12} \\ 0 & 1 & 0 & -\frac{5}{12} & \frac{1}{6} & -\frac{1}{12} \\ 0 & 0 & 1 & -\frac{1}{84} & \frac{5}{42} & \frac{1}{12} \end{bmatrix}$$

Since the left thee columns are the 3×3 identity we know that the original matrix is invertible and its inverse are the last three columns. We then extract the last three columns to obtain the inverse,

$$\begin{bmatrix} \frac{59}{84} & -\frac{1}{42} & \frac{1}{12} \\ -\frac{5}{12} & \frac{1}{6} & -\frac{1}{12} \\ -\frac{1}{84} & \frac{5}{42} & \frac{1}{12} \end{bmatrix}$$

Although we did not show all of the steps in the reduction, it is clear by the result that at some stage we needed to divide rows by a number. When we are dealing with modular arithmetic this division must be thought of, and executed as, the multiplication of inverses mod n. Say we wanted to take the inverse of this matrix mod 5. First we reduce the matrix mod 5 to obtain,

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

Append the identity,

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

Now we will reduce the matrix to reduced echelon form. During this derivation all arithmetic will be done mod 5, we will drop the $\pmod 5$ notation here.

$$\begin{bmatrix} 2 & 1 & 4 & 1 & 0 & 0 \\ 3 & 0 & 2 & 0 & 1 & 0 \\ 1 & 3 & 4 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{3R_1} \begin{bmatrix} 1 & 3 & 2 & 3 & 0 & 0 \\ 3 & 0 & 2 & 0 & 1 & 0 \\ 1 & 3 & 4 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{array}{rl}
 & \left[\begin{array}{cccccc} 1 & 3 & 2 & 3 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 3 & 4 & 0 & 0 & 1 \end{array}\right] \\
\xrightarrow{-3R_1+R_2} & \left[\begin{array}{cccccc} 1 & 3 & 2 & 3 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 2 & 2 & 0 & 1 \end{array}\right] \\
\xrightarrow{-R_1+R_3} & \left[\begin{array}{cccccc} 1 & 3 & 2 & 3 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 3 \end{array}\right] \\
\xrightarrow{3R_3} & \left[\begin{array}{cccccc} 1 & 3 & 2 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 0 & 3 \end{array}\right] \\
\xrightarrow{-R_3+R_2} & \left[\begin{array}{cccccc} 1 & 3 & 0 & 1 & 0 & 4 \\ 0 & 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 0 & 3 \end{array}\right] \\
\xrightarrow{-2R_3+R_1} & \left[\begin{array}{cccccc} 1 & 0 & 0 & 1 & 2 & 3 \\ 0 & 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 0 & 3 \end{array}\right] \\
\xrightarrow{-3R_2+R_1} &
\end{array}$$

So the inverse of this matrix mod 5 is,

$$\left[\begin{array}{ccc} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 1 & 0 & 3 \end{array}\right]$$

Notice in the reduction above there were two steps where we needed to divide by 2, the first row operation and the fourth row operation. In both cases we multiplied by 3, which is the inverse of 2 modulo 5. We can check our answer by multiplying it by the original and reducing the result mod 5.

$$\left[\begin{array}{ccc} 2 & 1 & 4 \\ 3 & 0 & 2 \\ 1 & 3 & 4 \end{array}\right]\left[\begin{array}{ccc} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 1 & 0 & 3 \end{array}\right] = \left[\begin{array}{ccc} 6 & 5 & 20 \\ 5 & 6 & 15 \\ 5 & 5 & 21 \end{array}\right] \equiv \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \pmod 5$$

Since we got the identity matrix, we know that our process worked.

Another way to calculate an inverse to a matrix is by using the adjugate (or classical adjoint) of the matrix and dividing it by the determinant. Recall that

$$A^{-1} = \frac{1}{\det(A)}\text{adj}(A)$$

Where the adjugate of A is the transpose of the cofactor matrix. This can be done modulo n as well. Cofactors are just determinants and determinants are just multiplications and

additions, hence we simply do all of our operations modulo n. Let's do our above example using the adjugate method. We will also use the notation A_{ij} to denote the i, j minor of the matrix A and C_{ij} as the i, j cofactor of A.

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

$$C_{11} = \begin{vmatrix} 0 & 2 \\ 3 & 4 \end{vmatrix} = 4 \qquad C_{21} = -\begin{vmatrix} 1 & 4 \\ 3 & 4 \end{vmatrix} = 3 \qquad C_{31} = \begin{vmatrix} 1 & 4 \\ 0 & 2 \end{vmatrix} = 2$$

$$C_{12} = -\begin{vmatrix} 3 & 2 \\ 1 & 4 \end{vmatrix} = 0 \qquad C_{22} = \begin{vmatrix} 2 & 4 \\ 1 & 4 \end{vmatrix} = 4 \qquad C_{32} = -\begin{vmatrix} 2 & 4 \\ 3 & 2 \end{vmatrix} = 3$$

$$C_{13} = \begin{vmatrix} 3 & 0 \\ 1 & 3 \end{vmatrix} = 4 \qquad C_{23} = -\begin{vmatrix} 2 & 1 \\ 1 & 3 \end{vmatrix} = 0 \qquad C_{33} = \begin{vmatrix} 2 & 1 \\ 3 & 0 \end{vmatrix} = 2$$

So the cofactor matrix is

$$\begin{bmatrix} 4 & 0 & 4 \\ 3 & 4 & 0 \\ 2 & 3 & 2 \end{bmatrix}$$

and the adjugate matrix is,

$$\begin{bmatrix} 4 & 3 & 2 \\ 0 & 4 & 3 \\ 4 & 0 & 2 \end{bmatrix}$$

Since $\det(A) \pmod 5 \equiv 4$ and $4^{-1} \pmod 5 \equiv 4$ we have

$$A^{-1} = 4 \cdot \begin{bmatrix} 4 & 3 & 2 \\ 0 & 4 & 3 \\ 4 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 1 & 0 & 3 \end{bmatrix}$$

The $A^{-1} = \frac{1}{\det(A)}\text{adj}(A)$ formula for the inverse is what led us to the theorem that a square matrix has an inverse if and only if the determinant of the matrix was not 0. Since 0 was the only number that could not be inverted in the real number system. What does this mean when doing modular arithmetic? As we saw in the beginning of this section, numbers other than 0 may not be able to be inverted modulo n. One example we have seen was $4^{-1} \pmod{26}$ does not exist since $\gcd(4, 26) = 2 \neq 1$, so 4 is not invertible modulo 26. So if the determinant of A modulo 26 turns out to be 4 then we will not be able to invert the matrix. On the other hand, if the determinant of A modulo 26 turns out be 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, or 25, then the matrix will be invertible and its inverse can be computed by the adjucate formula. This gives us a slightly modified version of our determinant criteria for invertibility.

Theorem 2: *Let A be an $n \times n$ integer matrix, then $A^{-1} \pmod n$ exists if and only if $\det(A)$ is invertible modulo n. That is, if and only if $(\det(A))^{-1} \pmod n$ exists.*

3 The Hill Cipher

As we pointed out above the Hill Cipher is a block cipher. Here is how it works in general. After we discuss the general process we will look at an example.

3.1 Encryption with the Hill Cipher

The Hill Cipher Encryption Algorithm

1. Find an $n \times n$ matrix E that is invertible modulo 26. This is actually the encryption key.

2. Take the message that is to be sent (the plaintext), remove all of the spaces and punctuation symbols, and convert the letters into all uppercase.

3. Convert each character to a number between 0 and 25. The usual way to do this is $A = 0$, $B = 1$, $C = 2$, $\ldots$, $Z = 25$.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

   As a historical note, Lester Hill did not use this coding of letters to numbers, he simply mixed up the order. Mixing up the order does not make the method more secure, it simply combines the Hill cipher with a simple substitution cipher, which are easy to break.

4. Divide this string of numbers up into blocks of size n. Note that if E is an $n \times n$ matrix then the block size is n. Another note, if the message does not break evenly into blocks of size n we pad the ending of the message with characters, this can be done at random.

5. Write each block as a column vector of size n. At this point the message is a sequence of n-dimensional vectors, $v_1, v_2, \ldots, v_t$.

6. Take each of the vectors and multiply them by the encryption matrix E, so

$$\begin{aligned} Ev_1 &= w_1 \\ Ev_2 &= w_2 \\ Ev_3 &= w_3 \\ &\vdots \\ Ev_t &= w_t \end{aligned}$$

7. Take the vectors $w_1, w_2, \ldots, w_t$, write the entries of the vectors in order, convert the numbers back to characters and you have your ciphertext.

One note about this algorithm is that we can do step 6 with a single matrix multiplication. If we let the message matrix M be the matrix produced by having the vectors $v_1, v_2, \ldots, v_t$ as columns, that is, $M = [v_1 \ v_2 \ \ldots \ v_t]$ then $EM = [w_1 \ w_2 \ \ldots \ w_t] = C$ would be our ciphertext matrix.

Example 7: Say Alice wants to send Bob the message "Cryptography is cool!"

1. Alice chooses the block size $n = 3$ and chooses the encryption matrix E to be,

$$E = \begin{bmatrix} 2 & 3 & 15 \\ 5 & 8 & 12 \\ 1 & 13 & 4 \end{bmatrix}$$

   Since $\det(E) \pmod{26} = 11$, and 11 is invertible modulo 26, the matrix E is also invertible modulo 26.

2. The message that is to be sent is "Cryptography is cool!", removing the spaces and punctuation symbols, and convert the letters into all uppercase gives

CRYPTOGRAPHYISCOOL

3. Conversion to numbers using $A = 0$, $B = 1$, $C = 2$, $\ldots$, $Z = 25$, gives

2 17 24 15 19 14 6 17 0 15 7 24 8 18 2 14 14 11

4. Dividing this string of numbers up into blocks of size 3.

2 17 24 15 19 14 6 17 0 15 7 24 8 18 2 14 14 11

   so no padding is needed here.

5. Converting these blocks into a message matrix M gives,

$$M = \begin{bmatrix} 2 & 15 & 6 & 15 & 8 & 14 \\ 17 & 19 & 17 & 7 & 18 & 14 \\ 24 & 14 & 0 & 24 & 2 & 11 \end{bmatrix}$$

6. Multiply by the encryption matrix E,

$$EM = \begin{bmatrix} 2 & 3 & 15 \\ 5 & 8 & 12 \\ 1 & 13 & 4 \end{bmatrix} \begin{bmatrix} 2 & 15 & 6 & 15 & 8 & 14 \\ 17 & 19 & 17 & 7 & 18 & 14 \\ 24 & 14 & 0 & 24 & 2 & 11 \end{bmatrix} = \begin{bmatrix} 25 & 11 & 11 & 21 & 22 & 1 \\ 18 & 5 & 10 & 3 & 0 & 2 \\ 7 & 6 & 19 & 20 & 16 & 6 \end{bmatrix} = C$$

7. Convert C into the ciphertext.

$$25\ 18\ 7\ 11\ 5\ 6\ 11\ 10\ 19\ 21\ 3\ 20\ 22\ 0\ 16\ 1\ 2\ 6$$
$$\text{ZSHLFGLKTVDUWAQBCG}$$

So Alice will send "ZSHLFGLKTVDUWAQBCG" to Bob.

Since this is a symmetric cipher, Alice and Bob would have to share this key with each other. They obviously could not simply call or text each other with this information since Eve could easily intercept that call or text and would know the key. So either Alice and Bob would have to meet in person, in a secure location, and exchange the key or they would need some other trusted person to deliver the key from Alice to Bob. This difficulty in exchanging the key securely gave rise to the creation of public-key systems which are commonly used today, for more information on public-key systems please see the references [5] and [7].

3.2 Decryption with the Hill Cipher

Now that Bob has the encrypted message and the encryption key he can decrypt the message that Alice had sent to him. The decryption algorithm is essentially the same as the encryption algorithm, except that we use E^{-1} in place of E. Since $EM = C$, and E is invertible we can calculate $M = E^{-1}C$. We will call $D = E^{-1}$ the decryption matrix, so $DC = M$. Remember that this inverse is the inverse modulo 26.

The Hill Cipher Decryption Algorithm

1. Find $D = E^{-1} \pmod{26}$. This is the decryption key.
2. Take the ciphertext and convert it to the matrix C.
3. Calculate $DC = M$.
4. Convert the matrix M to the plaintext message. You may need to insert the appropriate spaces and punctuation symbols since these were removed.

Example 8: Bob has the encrypted message ZSHLFGLKTVDUWAQBCG.

1. He calculates
$$\begin{bmatrix} 2 & 3 & 15 \\ 5 & 8 & 12 \\ 1 & 13 & 4 \end{bmatrix}^{-1} \pmod{26} = \begin{bmatrix} 10 & 19 & 16 \\ 4 & 23 & 7 \\ 17 & 5 & 19 \end{bmatrix}$$
2. He also converts the ciphertext to the matrix C.

ZSHLFGLKTVDUWAQBCG

25 18 7 11 5 6 11 10 19 21 3 20 22 0 16 1 2 6

and since he knows that the block size is 3 he constructs C as

$$C = \begin{bmatrix} 25 & 11 & 11 & 21 & 22 & 1 \\ 18 & 5 & 10 & 3 & 0 & 2 \\ 7 & 6 & 19 & 20 & 16 & 6 \end{bmatrix}$$

3. Calculate $DC = M$.

$$DC = \begin{bmatrix} 10 & 19 & 16 \\ 4 & 23 & 7 \\ 17 & 5 & 19 \end{bmatrix} \begin{bmatrix} 25 & 11 & 11 & 21 & 22 & 1 \\ 18 & 5 & 10 & 3 & 0 & 2 \\ 7 & 6 & 19 & 20 & 16 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 15 & 6 & 15 & 8 & 14 \\ 17 & 19 & 17 & 7 & 18 & 14 \\ 24 & 14 & 0 & 24 & 2 & 11 \end{bmatrix} = M$$

4. Convert the matrix M to the plaintext message.

2 17 24 15 19 14 6 17 0 15 7 24 8 18 2 14 14 11

CRYPTOGRAPHYISCOOL

So Bob adds in a couple spaces to get CRYPTOGRAPHY IS COOL!

4 Breaking the Hill Cipher

Now it is Eve's turn, how can she find the key to a Hill cipher? Looking at the encryption algorithm we know that $EM = C$. If have a portion of the plaintext and its corresponding ciphertext (a Known Plaintext attack) then we have a little of M and C. If we are lucky, the portion of M that we have might form an invertible $n \times n$ matrix (modulo 26). Then $EM = C$ could be rewritten as $E = CM^{-1}$, giving her the encryption matrix. From there, she simply inverts E modulo 26 to get D and then she can decrypt the entire message.

There is really one more piece of information that Eve needs, if she just has plaintext and ciphertext characters she does not know the block size n and hence she does not know the size of E nor the size of the matrices M and C she needs to use to find E. This is clearly a problem. But there is a way to guess the possible block sizes, if the message is not too long. Since Eve can get the entire ciphertext, she knows the number of letters in the message (possibly padded) and that this number of characters must be a multiple of the block size. So if she has intercepted ZSHLFGLKTVDUWAQBCG she knows that the message has 18 characters in it, so the block sizes could possibly be, 2, 3, 6, 9 or 18. If the block size was 6, 9, or 18 she would not have enough characters to create M and C so it would not be possible for her to find E and hence D. So the only possibilities she would have that would allow her to find the key would be block sizes of 2 or 3. If these both fail to produce a key then she knows that she will not be able to break the code and not know the original message. As an example we will assume that Eve knows that CRYPTOGRAPHYISCOOL encrypts as ZSHLFGLKTVDUWAQBCG and we will follow the process outlined above to find that the block size is 3 and the matrix E.

Example 9: Eve has intercepted the encrypted message ZSHLFGLKTVDUWAQBCG and from other espionage knows that this message was CRYPTOGRAPHYISCOOL. She also knows that other messages sent that day between Alice and Bob are using the same key and she wishes to decrypt them as well, but she has no other information about these other messages.

Since the message size 18 she knows that that block size must be 2, 3, 6, 9 or 18, and since she has only 18 characters to work with her only hope is that the block size is either 2 or 3. If both of these fail it is back to other espionage.

Let's start with a block size of 2. Since,

CRYPTOGRAPHYISCOOL — 2 17 24 15 19 14 6 17 0 15 7 24 8 18 2 14 14 11

encrypts as

ZSHLFGLKTVDUWAQBCG — 25 18 7 11 5 6 11 10 19 21 3 20 22 0 16 1 2 6

we know that

$$\begin{bmatrix} 2 \\ 17 \end{bmatrix} \longrightarrow \begin{bmatrix} 25 \\ 18 \end{bmatrix} \qquad \begin{bmatrix} 24 \\ 15 \end{bmatrix} \longrightarrow \begin{bmatrix} 7 \\ 11 \end{bmatrix} \qquad \begin{bmatrix} 19 \\ 14 \end{bmatrix} \longrightarrow \begin{bmatrix} 5 \\ 6 \end{bmatrix} \cdots$$

So we want to construct a 2×2 matrix out of the plaintext blocks that is invertible modulo 26. If we take the first two blocks 2 17 and 24 15 and build a matrix from them

$$M = \begin{bmatrix} 2 & 24 \\ 17 & 15 \end{bmatrix}$$

then $\det(M) = -378$ which is not relatively prime to 26, so M is not invertible. Actually, we should have seen this coming with what we know about determinants, notice that the first row has all even numbers in it and 2 is a factor of 26.

If we move on to the next block, 19 14 and keep the first block our M matrix becomes,

$$M = \begin{bmatrix} 2 & 19 \\ 17 & 14 \end{bmatrix}$$

then $\det(M) = -295$ which is 17 when we take it modulo 26. Since 17 has an inverse modulo 26 we are in business. Our equation becomes,

$$E \begin{bmatrix} 2 & 19 \\ 17 & 14 \end{bmatrix} = \begin{bmatrix} 25 & 5 \\ 18 & 6 \end{bmatrix}$$

So

$$E = \begin{bmatrix} 25 & 5 \\ 18 & 6 \end{bmatrix} \begin{bmatrix} 2 & 19 \\ 17 & 14 \end{bmatrix}^{-1} = \begin{bmatrix} 25 & 5 \\ 18 & 6 \end{bmatrix} \begin{bmatrix} 10 & 5 \\ 25 & 20 \end{bmatrix} = \begin{bmatrix} 11 & 17 \\ 18 & 2 \end{bmatrix}$$

To see if this works we test it on our plaintext message CRYPTOGRAPHYISCOOL.

$$\begin{aligned} C = EM &= \begin{bmatrix} 11 & 17 \\ 18 & 2 \end{bmatrix} \begin{bmatrix} 2 & 24 & 19 & 6 & 0 & 7 & 8 & 2 & 14 \\ 17 & 15 & 14 & 17 & 15 & 24 & 18 & 14 & 11 \end{bmatrix} \\ &= \begin{bmatrix} 25 & 25 & 5 & 17 & 21 & 17 & 4 & 0 & 3 \\ 18 & 20 & 6 & 12 & 4 & 18 & 24 & 12 & 14 \end{bmatrix} \end{aligned}$$

As we can see, columns 1 and 3 encrypt correctly, they should since those were the ones we used, but the rest of the encryption is incorrect. Conclusion, the block size is not 2. So we move on to block size 3. If it is then there is a 3×3 matrix E with

$$E \begin{bmatrix} 2 & 15 & 6 & 15 & 8 & 14 \\ 17 & 19 & 17 & 7 & 18 & 14 \\ 24 & 14 & 0 & 24 & 2 & 11 \end{bmatrix} = \begin{bmatrix} 25 & 11 & 11 & 21 & 22 & 1 \\ 18 & 5 & 10 & 3 & 0 & 2 \\ 7 & 6 & 19 & 20 & 16 & 6 \end{bmatrix}$$

As before, we want to select three columns from our message matrix that will produce an invertible 3×3 matrix. Looking at the last row, all entries except for 11 are even, so the last

column must be used. Since the last column first row is even we must have at least one odd number in the first row of the columns we use. So we could not select the remaining two columns from columns 1, 3, and 5. Furthermore, column 5 is all even so selecting it would be pointless. So in our selection we must have column 6 and at least one of columns 2 and 4. We will try columns 1, 2, and 6. This gives,

$$M = \begin{bmatrix} 2 & 15 & 14 \\ 17 & 19 & 14 \\ 24 & 14 & 11 \end{bmatrix}$$

Since $\det(M) = -791 \equiv 15 \pmod{26}$, we know that M is invertible. So

$$EM = E \begin{bmatrix} 2 & 15 & 14 \\ 17 & 19 & 14 \\ 24 & 14 & 11 \end{bmatrix} = \begin{bmatrix} 25 & 11 & 1 \\ 18 & 5 & 2 \\ 7 & 6 & 6 \end{bmatrix}$$

and then,

$$E = \begin{bmatrix} 25 & 11 & 1 \\ 18 & 5 & 2 \\ 7 & 6 & 6 \end{bmatrix} \begin{bmatrix} 2 & 15 & 14 \\ 17 & 19 & 14 \\ 24 & 14 & 11 \end{bmatrix}^{-1} = \begin{bmatrix} 25 & 11 & 1 \\ 18 & 5 & 2 \\ 7 & 6 & 6 \end{bmatrix} \begin{bmatrix} 13 & 9 & 24 \\ 3 & 12 & 14 \\ 8 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 15 \\ 5 & 8 & 12 \\ 1 & 13 & 4 \end{bmatrix}$$

Now, we know this is correct but Eve would still need to check her possible solution, doing so she would get,

$$\begin{bmatrix} 2 & 3 & 15 \\ 5 & 8 & 12 \\ 1 & 13 & 4 \end{bmatrix} \begin{bmatrix} 2 & 15 & 6 & 15 & 8 & 14 \\ 17 & 19 & 17 & 7 & 18 & 14 \\ 24 & 14 & 0 & 24 & 2 & 11 \end{bmatrix} = \begin{bmatrix} 25 & 11 & 11 & 21 & 22 & 1 \\ 18 & 5 & 10 & 3 & 0 & 2 \\ 7 & 6 & 19 & 20 & 16 & 6 \end{bmatrix}$$

which checks with the ciphertext and she has found the encryption matrix, E.

5 The Hill Cipher from a Linear Algebra Point of View

We are going to abstract the Hill cipher a little bit and relate it to the material we have been studying for the past several weeks. Most of this will be done in the exercises but we will give a quick introduction here along with some notation.

From the description of the algorithm in Section 3, it is clear that the Hill cipher can be viewed as matrix (and hence linear) transformation. In fact, it is an invertible matrix transformation. Let's put this observation into a more mathematical context. First of all, we are no longer working over the real numbers. We are working with the integers 0–25 with addition and multiplication being done modulo 26.

Notation: Let $\mathbb{A}$ denote the set $\{0, 1, 2, \ldots, 25\}$ where addition and multiplication are done modulo 26.

We use $\mathbb{A}$ as our notation here to designate our "alphabet". If you go further in mathematics and take a course in abstract algebra you will call this structure a ring and denote it as $\mathbb{Z}_{26}$. We will not need this notation nor a digression into ring theory.

As we saw in Section 3, the key to a Hill cipher is an $n \times n$ matrix which has an inverse (modulo 26). So it sends n-dimensional vectors with entries from $\mathbb{A}$ to n-dimensional vectors with entries from $\mathbb{A}$.

Notation: Let $\mathbb{A}^n$ as the set of n-dimensional vectors with entries from $\mathbb{A}$

With this notation, the Hill cipher can be viewed simply as an invertible linear transformation $E : \mathbb{A}^n \to \mathbb{A}^n$. Its decryption is also an invertible linear transformation $D : \mathbb{A}^n \to \mathbb{A}^n$. As we have seen from the Invertible Matrix Theorem, there are many properties of an invertible matrix and the transformation it induces. Some of these properties are crucial when it cones to cryptography. For example, say that E was not invertible but we used it as an encryption matrix for a Hill cipher. One immediate problem comes to mind in that if E is not invertible, finding the decryption matrix D is not possible, hence Bob has a bit of a problem when he receives the ciphertext from Alice. We also know that when we consider E as a linear transformation, this transformation is not one-to-one nor is it onto.

If the transformation is not one-to-one then it is many-to-one which means that there are different vectors in the domain that are mapped to the same vector in the range. Since the domain is associated with the plaintext and the range with the ciphertext this translates to having the two different plaintext words (or segments of words) encrypt to the same ciphertext block. So if we could devise a way to decrypt the message this block of ciphertext would decrypt as two or more possible plaintext blocks. In this situation, it may be possible to figure out which one of the possible decryptions is the correct one but there may be cases where this is not possible either.

Example 10: Say that Alice chooses the following matrix for her Hill cipher,

$$E = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}$$

As we can see, the first row is a multiple of 2 and hence this matrix will not be invertible modulo 26. If we take the determinant modulo 26 we get,

$$\begin{vmatrix} 2 & 4 \\ 3 & 5 \end{vmatrix} \pmod{26} = 24$$

and since the GCD of 24 and 26 is not 1 we have verified that E does not have an inverse. Now say that Alice wants to send the message HELP to Bob. So Alice encodes the message as usual, HELP becomes 7 4 11 15, which produces the message matrix

$$M = \begin{bmatrix} 7 & 11 \\ 4 & 15 \end{bmatrix}$$

Then, modulo 26,

$$EM = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} 7 & 11 \\ 4 & 15 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 15 & 4 \end{bmatrix}$$

Which produces the ciphertext EPEE, which Alice sends to Bob.

Now Bob receives the ciphertext EPEE from Alice and when he goes to decrypt the message he finds out that the encryption matrix E is not invertible modulo 26. So what does he do? Since Bob is not the type to give up, he starts to reason this through. He knows that the first two letters of the message must encrypt to EP and he knows the encryption matrix E, this gives him the equation,

$$\begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 15 \end{bmatrix}$$

So he can find all of the letters xy that encrypt to EP by solving this system. He knows that E is not invertible, hence the encoding transformation is many-to-one and therefore he will get several possible solutions. Unlike when working over the real or complex numbers he is working modulo 26 and so there will not be an infinite number of solutions, so he will have only a finite number of possibilities to worry about. He then solves the system, remember he is working modulo 26.

$$\begin{bmatrix} 2 & 4 & 4 \\ 3 & 5 & 15 \end{bmatrix} \xrightarrow{R_1 \leftrightarrow R_2} \begin{bmatrix} 3 & 5 & 15 \\ 2 & 4 & 4 \end{bmatrix}$$

$$\xrightarrow{9R_1} \begin{bmatrix} 1 & 19 & 5 \\ 2 & 4 & 4 \end{bmatrix}$$

$$\begin{array}{ll} & \begin{bmatrix} 1 & 19 & 5 \\ 0 & 18 & 20 \end{bmatrix} \\ \xrightarrow{25R_2+R_1} & \begin{bmatrix} 1 & 1 & 11 \\ 0 & 18 & 20 \end{bmatrix} \end{array}$$

From this he knows that $18y = 20$ and that $x + y = 11$, that is $x = 11 - y$. If we evaluate $18y$ for $0 \leq y \leq 25$ we see that the only two values of y that give us $18y = 20$ are $y = 4$ and $y = 17$. When $y = 4$ we have $x = 11 - 4 = 7$ and when $y = 17$ we have $x = 11 - 17 = 20$. So the first two letters are either 7 4 or 20 17, that is, either HE or UT. The number of English words that begin with UT is fairly small so Bob probably figures that the letters are HE but we will keep all of our options open at this point, who knows, Alice may be telling Bob she plans to take a trip to UTAH. Now he attacks the last two letters in the same way.

$$\begin{array}{ll} \begin{bmatrix} 2 & 4 & 4 \\ 3 & 5 & 4 \end{bmatrix} & \xrightarrow{R_1 \leftrightarrow R_2} \begin{bmatrix} 3 & 5 & 4 \\ 2 & 4 & 4 \end{bmatrix} \\ & \xrightarrow{9R_1} \begin{bmatrix} 1 & 19 & 10 \\ 2 & 4 & 4 \end{bmatrix} \\ & \xrightarrow{24R_1+R_2} \begin{bmatrix} 1 & 19 & 10 \\ 0 & 18 & 10 \end{bmatrix} \\ & \xrightarrow{25R_2+R_1} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 18 & 10 \end{bmatrix} \end{array}$$

From this he knows that $18y = 10$ and that $x + y = 0$, that is $x = -y$. If we evaluate $18y$ for $0 \leq y \leq 25$ we see that the only two values of y that give us $18y = 10$ are $y = 2$ and $y = 15$. When $y = 2$ we have $x = -2 = 24$ and when $y = 15$ we have $x = -15 = 11$. So the first two letters are either 24 2 or 11 15, that is, either YC or LP. Again Bob probably figures that the letters are LP since the number of English words ending in YC is surprisingly small but again we will consider all possibilities. So there are four words that will encrypt to EPEE under this transformation, they are, HEYC, HELP, UTYC, and UTLP. Under the assumption that Alice sent an English word we would guess that the plaintext for the message was HELP.

A few things to note here,

1. If this message had been longer the HEYC option could have been valid if the original message was "Hey c…".

2. In the reductions we did above we had to stop where we did since 18 does not have an inverse modulo 26.

3. If we take a closer look at our two-letter decryptions, HE and UT, that is, 7 4 and 20 17. Note that,

$$\begin{bmatrix} 20 \\ 17 \end{bmatrix} - \begin{bmatrix} 7 \\ 4 \end{bmatrix} = \begin{bmatrix} 13 \\ 13 \end{bmatrix}$$

and

$$\begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} 13 \\ 13 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We could have also come to this conclusion by solving the homogenous system,

$$\begin{bmatrix} 2 & 4 & 0 \\ 3 & 5 & 0 \end{bmatrix} \xrightarrow{R_1 \leftrightarrow R_2} \begin{bmatrix} 3 & 5 & 0 \\ 2 & 4 & 0 \end{bmatrix}$$
$$\xrightarrow{9R_1} \begin{bmatrix} 1 & 19 & 0 \\ 2 & 4 & 0 \end{bmatrix}$$
$$\xrightarrow{24R_1+R_2} \begin{bmatrix} 1 & 19 & 0 \\ 0 & 18 & 0 \end{bmatrix}$$
$$\xrightarrow{25R_2+R_1} \begin{bmatrix} 1 & 1 & 0 \\ 0 & 18 & 0 \end{bmatrix}$$

From this we know that $18y = 0$ and that $x + y = 0$, that is $x = -y$. If we evaluate $18y$ for $0 \leq y \leq 25$ we see that the only two values of y that give us $18y = 0$ are $y = 0$ and $y = 13$. When $y = 0$ we have $x = 0$ and when $y = 13$ we have $x = -13 = 13$. So the vectors that get sent to the zero vector are

$$\begin{bmatrix} 13 \\ 13 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In other words, thinking of E as the transformation,

$$\ker(E) = \left\{ \begin{bmatrix} 13 \\ 13 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$$

As we know from the course material, the entire set of solutions to a consistent non-homogenous system of equations can be written as a single solution to the non-homogenous system plus any element of the kernel (or null space) of the coefficient matrix. So here that translates to,

$$\begin{bmatrix} 7 \\ 4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 7 \\ 4 \end{bmatrix} + \begin{bmatrix} 13 \\ 13 \end{bmatrix} = \begin{bmatrix} 20 \\ 17 \end{bmatrix}$$

and

$$\begin{bmatrix} 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 24 \\ 2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 24 \\ 2 \end{bmatrix} + \begin{bmatrix} 13 \\ 13 \end{bmatrix} = \begin{bmatrix} 11 \\ 15 \end{bmatrix}$$

Although, this observation may not have saved us too many calculations here, but in some cases, especially with larger block sizes, this method could save a lot of work.

The above discussion and example shows us why we should use an invertible matrix as our encryption matrix for the Hill cipher algorithm. Furthermore, it puts both the theory and calculations we have been doing into the language of linear algebra.

Let's take a quick look at Eve's job, the cryptanalysis process. Recall that we did a known plaintext attack on the system to find the encryption matrix, and hence the decryption matrix. In this discussion, we will be referring to the example from Section 4, we reproduce portions of it in the example below.

Example 11: Recall that Eve knows the plaintext and the corresponding ciphertext of the message,

CRYPTOGRAPHYISCOOL — 2 17 24 15 19 14 6 17 0 15 7 24 8 18 2 14 14 11

encrypts as

ZSHLFGLKTVDUWAQBCG — 25 18 7 11 5 6 11 10 19 21 3 20 22 0 16 1 2 6

When she started with block size of 2, she came up with the vector correspondence,

$$\begin{bmatrix} 2 \\ 17 \end{bmatrix} \longrightarrow \begin{bmatrix} 25 \\ 18 \end{bmatrix} \qquad \begin{bmatrix} 24 \\ 15 \end{bmatrix} \longrightarrow \begin{bmatrix} 7 \\ 11 \end{bmatrix} \qquad \begin{bmatrix} 19 \\ 14 \end{bmatrix} \longrightarrow \begin{bmatrix} 5 \\ 6 \end{bmatrix} \cdots$$

Then proceeded to build a 2×2 matrix out of the plaintext blocks (vectors) that was invertible modulo 26. Obtaining, after some work,

$$M = \begin{bmatrix} 2 & 19 \\ 17 & 14 \end{bmatrix}$$

which was invertible. Then knowing that,

$$E \begin{bmatrix} 2 & 19 \\ 17 & 14 \end{bmatrix} = \begin{bmatrix} 25 & 5 \\ 18 & 6 \end{bmatrix}$$

She calculated,

$$E = \begin{bmatrix} 25 & 5 \\ 18 & 6 \end{bmatrix} \begin{bmatrix} 2 & 19 \\ 17 & 14 \end{bmatrix}^{-1} = \begin{bmatrix} 25 & 5 \\ 18 & 6 \end{bmatrix} \begin{bmatrix} 10 & 5 \\ 25 & 20 \end{bmatrix} = \begin{bmatrix} 11 & 17 \\ 18 & 2 \end{bmatrix}$$

Which turned out not to work for the message, telling her that the block size of 2 was incorrect. When she moved on to block size 3, she constructed,

$$M = \begin{bmatrix} 2 & 15 & 14 \\ 17 & 19 & 14 \\ 24 & 14 & 11 \end{bmatrix}$$

which was invertible. So

$$EM = E \begin{bmatrix} 2 & 15 & 14 \\ 17 & 19 & 14 \\ 24 & 14 & 11 \end{bmatrix} = \begin{bmatrix} 25 & 11 & 1 \\ 18 & 5 & 2 \\ 7 & 6 & 6 \end{bmatrix}$$

and then,

$$E = \begin{bmatrix} 25 & 11 & 1 \\ 18 & 5 & 2 \\ 7 & 6 & 6 \end{bmatrix} \begin{bmatrix} 2 & 15 & 14 \\ 17 & 19 & 14 \\ 24 & 14 & 11 \end{bmatrix}^{-1} = \begin{bmatrix} 25 & 11 & 1 \\ 18 & 5 & 2 \\ 7 & 6 & 6 \end{bmatrix} \begin{bmatrix} 13 & 9 & 24 \\ 3 & 12 & 14 \\ 8 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 15 \\ 5 & 8 & 12 \\ 1 & 13 & 4 \end{bmatrix}$$

And this matrix worked for the entire message and hence we found the key, the encryption matrix, E.

Constructing an invertible matrix M from the message to calculate the encryption matrix makes perfect sense arithmetically. We know that

$$EM = C$$

so if M is invertible, we have,

$$E = E(MM^{-1}) = (EM)M^{-1} = CM^{-1}$$

and we are done. Let's also think about this in terms of vectors and transformations. We are trying to determine the transformation E by what the transformation does. That is, we know some vectors in the domain (the plaintext) and what those vectors get sent to in the range (the ciphertext). The columns of M are our domain vectors and the columns of C are where those vectors are mapped. If M is invertible, the Invertible Matrix Theorem tells us that the columns of M are linearly independent. Since there are n column vectors in M and the dimension of our domain space is n we also know that the columns of M form a basis to $\mathbb{A}^n$. We also know that any linear transformation is completely determined by what it does to a basis for the space. Hence we know that if we can find a set of linearly independent vectors from the plaintext message, and hence construct M, we will have enough information to completely determine E.

This also tells us that if we cannot find a set of linearly independent vectors from the plaintext then we do not have a basis for $\mathbb{A}^n$ and subsequently we will not be able to determine E. We may, in some cases, be able to determine a set of possibilities for E, as we did for the message HELP in the previous example. If we were to try to determine E from a non-basis set of vectors we would need to set up a system of equations and solve the system. Since the set is not a basis we would be guaranteed to have at least one free variable in the solution and hence a set of possible solutions for E, each of which would need to be tested on the plaintext to see which one, or ones, produced the corresponding ciphertext. One positive note here is that since we are working modulo 26 we will have only a finite number of possible matrices E to test, just like we had only four possibilities for the decryption of the message in the previous example.

6 Technology

Now that we know how the Hill cipher works and how to break it, we will look at some software packages that will ease the calculations for us. We will discuss the processes to find the modulus of a matrix as well as modular inverses of both numbers and matrices for four software packages, Linear ME (pre and post version 2.2.1), Mathematica and Maxima. Please refer to the section devoted to the software package you are using.

Linear ME: Linear ME (Maxima Edition) is a freeware program that was developed for the teaching and exploration of concepts in Linear Algebra. It was created and is maintained by faculty and students at Salisbury University. The program links to the Maxima computer algebra system for many of the calculations. You can download Linear ME from [6],

http://facultyfp.salisbury.edu/despickler/personal/LinearME.asp

Maxima: Maxima is an open-source general purpose computer algebra system. It can be downloaded from Sourceforge at [4],

http://maxima.sourceforge.net/

Mathematica: Mathematica is also a general purpose computer algebra system. It is not freeware but can be purchased on the Wolfram site at [8],

http://www.wolfram.com/

6.1 Linear ME (Version 2.2.1 or Later)

In version 2.2.1, we added modular arithmetic functions for integers and matrices so that all of the calculations done in the discussion above can be done without using the Maxima command line feature. We assume that the reader has some familiarity with the use of Linear ME, specifically we will assume that the user knows how to enter matrices and knows the naming convention for matrices and expressions in the workspace. Please consult the help system of Linear ME if you are new to the program.

6.1.1 Matrix Arithmetic

Addition: After the desired matrices are loaded into the workspace, select any matrix in the workspace and then

1. Select Operations > Arithmetic > Add from the main menu.

2. At this point a dialog box will appear with drop-down menus for selecting the two matrices you wish to add. Select the matrices and press the OK button.

If the matrices are the same size the program will find their sum and load it as a new matrix into the workspace. If the matrices are not the same size the program will display an error.

You can also add matrices from the Maxima command line at the bottom of the screen. First load the desired matrices into the workspace and then on the command line type in

```
M1 + M2
```

and hit the evaluate key, where M1 and M2 are the names of the desired matrices. If the matrices are the same size the program will find their sum and load it as a new matrix into the workspace. If the matrices are not the same size the program will display an error.

Subtraction: After the desired matrices are loaded into the workspace, select any matrix in the workspace and then

1. Select Operations > Arithmetic > Subtract from the main menu.
2. At this point a dialog box will appear with drop-down menus for selecting the two matrices you wish to subtract. Select the matrices and press the OK button.

If the matrices are the same size the program will find their difference and load it as a new matrix into the workspace. If the matrices are not the same size the program will display an error.

You can also subtract matrices from the Maxima command line at the bottom of the screen. First load the desired matrices into the workspace and then on the command line type in

```
M1 - M2
```

and hit the evaluate key, where M1 and M2 are the names of the desired matrices. If the matrices are the same size the program will find their difference and load it as a new matrix into the workspace. If the matrices are not the same size the program will display an error.

Multiplication: After the desired matrices are loaded into the workspace, select any matrix in the workspace and then

1. Select Operations > Arithmetic > Multiply from the main menu.

2. At this point a dialog box will appear with drop-down menus for selecting the two matrices you wish to multiply. Select the matrices and press the OK button.

If the matrices are of compatible sizes the program will find their product and load it as a new matrix into the workspace. If the matrices are not of compatible sizes the program will display an error.

You can also multiply matrices from the Maxima command line at the bottom of the screen. First load the desired matrices into the workspace and then on the command line type in

```
M1 . M2
```

and hit the evaluate key, where M1 and M2 are the names of the desired matrices. If the matrices are of compatible sizes the program will find their product and load it as a new matrix into the workspace. If the matrices are not of compatible sizes the program will display an error.

Note: You use `.` and not `*` when doing matrix multiplication.

Scalar Multiplication: Select the matrix you wish to multiply and then

1. Select Operations > Arithmetic > Scalar Multiply from the main menu.
2. At this point a dialog box will appear with an input box for you to input the scalar to multiply by.

The syntax of this scalar must be in Maxima syntax, a description of some of the basic Maxima syntax can be found in the help system in the Linear ME program.

You can also do scalar multiplication from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
c*M1
```

and hit the evaluate key, where M1 is the name of the desired matrix and c is the expression of the scalar, in Maxima syntax.

Note: You use `*` and not `.` when doing scalar multiplication.

Negation: Select the matrix you wish to negate and then select Operations > Arithmetic > Negate from the main menu. At this point the negative of the selected matrix will be loaded into the workspace.

You can also do negation from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in `-M1` and hit the evaluate key, where M1 is the name of the desired matrix.

Powers: Select the matrix you wish to take the power of and then select

1. Operations > Arithmetic > Power from the main menu.
2. At this point a dialog box will appear with an integer input box for you to input the power.

When you select OK the power of the matrix will be loaded as a new matrix in the workspace. Note that this option will only be enabled if the matrix is square.

You can also do a matrix power from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
M1^^#
```

and hit the evaluate key, where M1 is the name of the desired matrix and # is the power.

Notes:

1. Use two caret symbols for a matrix power (^^), using only one will simply take the power of each entry.
2. The matrix must be square for this operation. Non-square matrices will result in an error.
3. The power must be an integer. A non-integer power will result in an error.

6.1.2 Modular Arithmetic

These options will only be enabled if the currently selected matrix or expression has only integer entries.

Number Modulo n: If the expression is an integer you can take its modulus by selecting

1. Operations > Mod n... from the menu.
2. At this point a dialog box will appear allowing you to input the modulus.

Once you select OK a new expression with the result will be loaded into the workspace.

You can also do a modulus from the Maxima command line at the bottom of the screen. First load the desired expression into the workspace and then on the command line type in

```
mod(R1, n)
```

and hit the evaluate key, where R1 is the name of the desired expression and n is the modulus.

Matrix Modulo n: If the matrix has only integer entries you can take its modulus by selecting

1. Operations > Modular Operations > Mod n... from the menu.
2. At this point a dialog box will appear allowing you to input the modulus.

Once you select OK a new matrix with the result will be loaded into the workspace.

You can also do a matrix modulus from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
mod(M1, n)
```

and hit the evaluate key, where M1 is the name of the desired matrix and n is the modulus.

Inverse of a Number Modulo n: If the expression is an integer you can take its inverse modulo a number, if it exists, by selecting

1. Operations > Inverse Mod n... from the menu.
2. At this point a dialog box will appear allowing you to input the modulus.

Once you select OK a new expression with the result will be loaded into the workspace, if the inverse exists. If the inverse does not exist the program will return an error.

You can also do a modular inverse from the Maxima command line at the bottom of the screen. First load the desired expression into the workspace and then on the command line type in

```
inv_mod(R1, n)
```

and hit the evaluate key, where R1 is the name of the desired expression and n is the modulus.

Inverse of a Matrix Modulo n: If the matrix has only integer entries you can take its inverse modulo a number by selecting

1. Operations > Modular Operations > Inverse Mod n... from the menu.
2. At this point a dialog box will appear allowing you to input the modulus.

Once you select OK a new matrix with the result will be loaded into the workspace, if the inverse exists. If the inverse does not exist then the program will return an error.

You can also do a matrix modulus from the Maxima command line at the bottom of the screen. There is not a single command to do this but a short sequence of commands will produce the desired result.

1. First load the desired matrix into the workspace, Say it is called M1.
2. Find the determinant of the matrix, say it is called R1.
3. Make sure that the determinant and the modulus are relatively prime. If this is too hard to see, you can find the GCD of the two numbers from the command line using the command `gcd(R1, n)` where n is the modulus. If the GCD if the two numbers is one you may proceed.
4. Take the modular inverse of the determinant by the command, `inv_mod(R1, n)`, say that this is stored in R2.
5. Find the classical adjoint of the matrix with the command, `adjoint(M1)`, say this is stored in M2. The result of this operation will not be modulo n.
6. Take M2 modulo n using the command, `mod(M2, n)`, say that this is stored in M3.
7. Multiply the inverse of the determinant by the adjoint with, `R2*M3`, say this is stored in M4.
8. Take M4 modulo n, `mod(M4, n)`, say that this is stored in M5. This is the inverse of M1 modulo n.

Note that numeric inverses and the adjoint calculations can also be done through the menu system. Operations > Inverse Mod n... will allow you to find the numeric inverse modulo n of a number and Operations > Arithmetic > Adjoint will find the adjoint of the selected matrix. The reduction modulo n of both numbers and matrices can be done through the menu system as well, as described above.

6.1.3 Some Matrix Operations

Below we give a short description of some of the other matrix operations that can be done in the Linear ME software package. We have only listed those operations that may be of use for the manipulations needed when doing work with Hill ciphers. For a complete list of operations and descriptions of them please see the help system in the Linear ME package.

Triangularize: Returns the upper triangular form of the matrix M, as produced by Gaussian elimination. The return value is the same as Echelon Form, except that the leading nonzero coefficient in each row is not normalized to 1.

To triangularize a matrix in Linear ME, select the desired matrix and then,

Operations > Triangularize

from the menu. At this point the new matrix will be loaded into the workspace.

You can also triangularize a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
triangularize(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Echelon Form: Returns the echelon form of the matrix M, as produced by Gaussian elimination. The echelon form is computed from M by elementary row operations such that the first non-zero element in each row in the resulting matrix is one and the column elements under the first one in each row are all zero.

To find the above echelon form of a matrix in Linear ME, select the desired matrix and then,

Operations > Echelon Form

from the menu. At this point the new matrix will be loaded into the workspace.

You can also produce the echelon form a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
echelon(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Reduced Echelon Form: Returns the echelon form of the matrix M, as produced by Gaussian elimination. The echelon form is computed from M by elementary row operations such that the first non-zero element in each row in the resulting matrix is one and the column elements both above and below the first one in each row are all zero.

To find the reduced echelon form of a matrix in Linear ME, select the desired matrix and then,

Operations > Reduced Echelon Form

from the menu. At this point the new matrix will be loaded into the workspace.

Determinant: Returns the determinant of a square matrix.

To find the determinant of a matrix in Linear ME, select the desired matrix and then,

Operations > Determinant

from the menu. At this point the determinant of the matrix will be loaded into the workspace.

You can also find the determinant of a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
determinant(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Adjoint: Returns the classical adjoint of a square matrix.

To find the classical adjoint of a square matrix in Linear ME, select the desired matrix and then,

Operations > Arithmetic > Adjoint

from the menu. At this point the adjoint of the matrix will be loaded into the workspace.

You can also find the adjoint of a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
adjoint(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Transpose: Returns the transpose of a matrix.

To find the transpose of a matrix in Linear ME, select the desired matrix and then,

Operations > Transpose

from the menu. At this point the transpose of the matrix will be loaded into the workspace.

You can also find the transpose of a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
transpose(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Invert: Returns the inverse of a matrix if the determinant is not 0. Note that this is an inverse over the real or complex number system and not a modular inverse. For modular inverses see the section on Modular Arithmetic.

To find the inverse of a matrix in Linear ME, select the desired matrix and then,

Operations > Invert

from the menu. At this point the inverse of the matrix will be loaded into the workspace, if it exists.

You can also find the inverse of a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
invert(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

6.1.4 Lists

Computer algebra systems (CAS) like Mathematica, Maxima, and Maple are built to work on lists. In fact, the internal structure of most computer algebra systems rely heavily on the storage and manipulation of lists. They can also be very convenient for the end user in that they allow the same operations to be done on many inputs at once without the need to redo each of the operations.

To create a list in Linear ME you use the same syntax as you would using Maxima, begin the list with a `[` end it with a `]` and separate the entries by commas. A list can be input either from the command line or using the new expression dialog box which you can get by selecting Edit > New Expression... from the main menu. In Maxima, and hence Linear ME, many of the functions and arithmetic will work as well on lists as they do on non-list expressions.

Example 12: To create the list 0, 1, 2, 3, 4, 5 simply type `[0, 1, 2, 3, 4, 5]` into the Maxima command line at the bottom of the screen and click the Evaluate button, or select Edit > New Expression... from the main menu and type it into the Expression box of the dialog that appears and click the OK button. At this point you should see

$$[0, 1, 2, 3, 4, 5]$$

in the workspace. Let's suppose that this list was stored as R1 in the workspace.

1. If we were to enter `2*R1` into the Maxima command line and click Evaluate we would have,

$$[0, 2, 4, 6, 8, 10]$$

2. If we were to enter `3*R1+4` into the Maxima command line and click Evaluate we would have,
$$[4, 7, 10, 13, 16, 19]$$

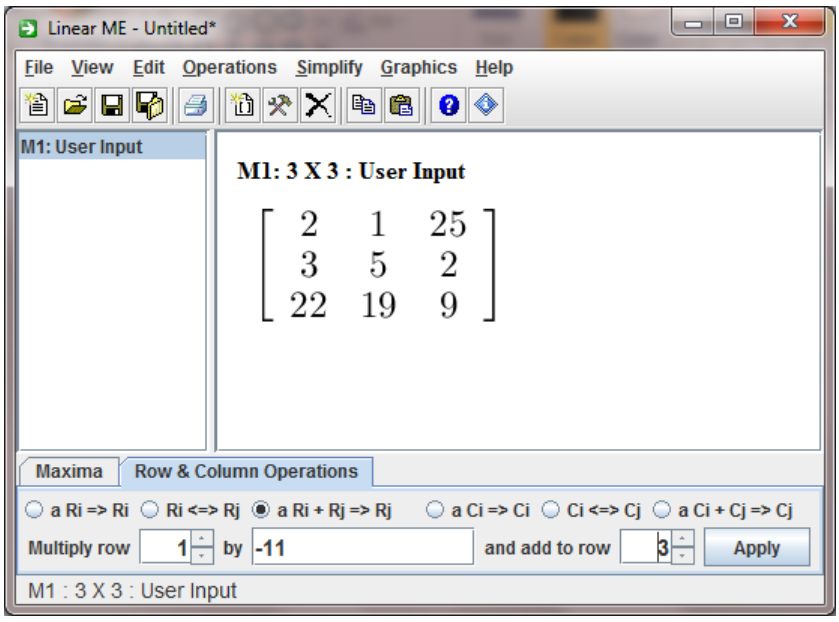
3. If we were to enter `R1^2` into the Maxima command line and click Evaluate we would have,
$$[0, 1, 4, 9, 16, 25]$$

4. If we were to enter `cos(R1)` into the Maxima command line and click Evaluate we would have,
$$[1, \cos 1, \cos 2, \cos 3, \cos 4, \cos 5]$$

5. If we were to enter `mod(R1, 3)` into the Maxima command line and click Evaluate we would have,
$$[0, 1, 2, 0, 1, 2]$$
Note that Linear ME does not recognize a list as either a matrix or simple integer expression so the Mod n... option in the menu will be disabled. Hence to take a modulus of a list you will need to use the command line.

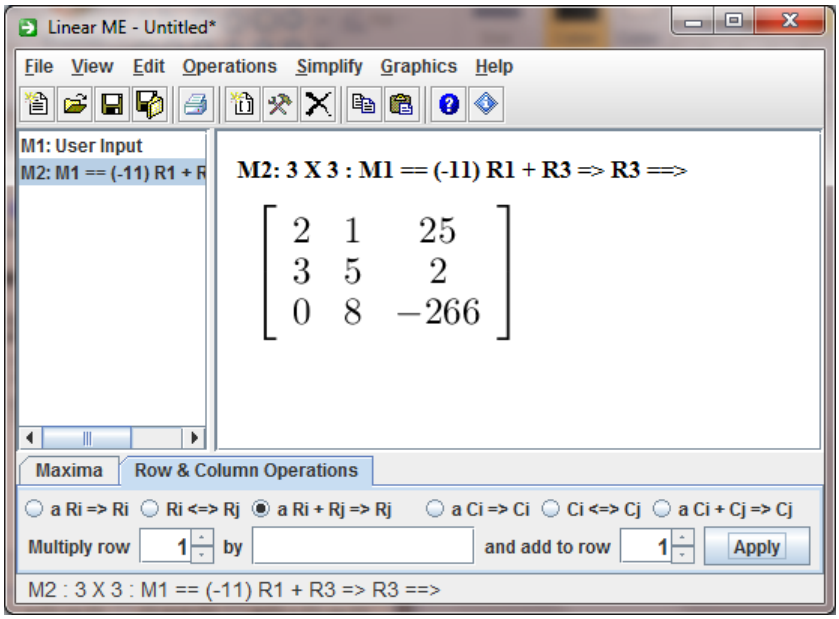
6.1.5 Row & Column Operations Interface

At the bottom of the Linear ME window you will see a tab titled Row & Column Operations. If you click on that tab you will bring up the Row & Column Operation interface, as shown below.



The primary purpose of this interface is to help teach students Gauss-Jordan elimination. It is also a good tool for experimenting with the effects of row and column operations on the determinant. For the working modulo n, as with the Hill cipher, this interface will allow the user to do row operations one at a time and then after each step, use the modular operations to mod the matrix by n. Since the built-in options for solving systems are programed to work over the real and complex number systems, if the user needs to solve a system in a modular setting they will need to do the reduction process by hand, this interface will make that process easier.

Along the top of the tab the user selects the type of operation they want to do. When the selection is made the data boxes below the selection change according to the selection. The user then fills in the information on the operation and clicks Apply to do the operation. When this is done a new matrix, resulting from the operation, will be added to the workspace.



The row or columns you are working with are controlled by spinners that do not allow you to go outside the matrix. The multiplier can be any valid Maxima expression, but in the case of reducing a matrix modulo n the expressions will simply be integers.

6.2 Linear ME (Version 2.1.2 or Earlier)

Prior to version 2.2.1, the user must use the Maxima command line feature for some of the calculations done above. We assume that the reader has some familiarity with the use of Linear ME, specifically we will assume that the user knows how to enter matrices and knows the naming convention for matrices and expressions in the workspace. Please consult the help system of Linear ME if you are new to the program.

6.2.1 Matrix Arithmetic

Addition: First load the desired matrices into the workspace and then on the command line type in

```
M1 + M2
```

and hit the evaluate key, where M1 and M2 are the names of the desired matrices. If the matrices are the same size the program will find their sum and load it as a new matrix into the workspace. If the matrices are not the same size the program will display an error.

Subtraction: First load the desired matrices into the workspace and then on the command line type in

```
M1 - M2
```

and hit the evaluate key, where M1 and M2 are the names of the desired matrices. If the matrices are the same size the program will find their difference and load it as a new matrix into the workspace. If the matrices are not the same size the program will display an error.

Multiplication: First load the desired matrices into the workspace and then on the command line type in

```
M1 . M2
```

and hit the evaluate key, where M1 and M2 are the names of the desired matrices. If the matrices are of compatible sizes the program will find their product and load it as a new matrix into the workspace. If the matrices are not of compatible sizes the program will display an error.

Note: You use `.` and not `*` when doing matrix multiplication.

Scalar Multiplication: First load the desired matrix into the workspace and then on the command line type in

```
c*M1
```

and hit the evaluate key, where M1 is the name of the desired matrix and c is the expression of the scalar, in Maxima syntax.

Note: You use `*` and not `.` when doing scalar multiplication.

Negation: First load the desired matrix into the workspace and then on the command line type in `-M1` and hit the evaluate key, where M1 is the name of the desired matrix.

Powers: First load the desired matrix into the workspace and then on the command line type in

```
M1^^#
```

and hit the evaluate key, where M1 is the name of the desired matrix and # is the power.

Notes:

1. Use two caret symbols for a matrix power (^^), using only one will simply take the power of each entry.
2. The matrix must be square for this operation. Non-square matrices will result in an error.
3. The power must be an integer. A non-integer power will result in an error.

6.2.2 Modular Arithmetic

Number Modulo n: First load the desired expression into the workspace and then on the command line type in

```
mod(R1, n)
```

and hit the evaluate key, where R1 is the name of the desired expression and n is the modulus.

Matrix Modulo n: First load the desired matrix into the workspace and then on the command line type in

```
mod(M1, n)
```

and hit the evaluate key, where M1 is the name of the desired matrix and n is the modulus.

Inverse of a Number Modulo n: First load the desired expression into the workspace and then on the command line type in

```
inv_mod(R1, n)
```

and hit the evaluate key, where R1 is the name of the desired expression and n is the modulus.

Inverse of a Matrix Modulo n: First load the desired matrix into the workspace and then do the following sequence of commands,

1. First load the desired matrix into the workspace, Say it is called M1.
2. Find the determinant of the matrix, say it is called R1.
3. Make sure that the determinant and the modulus are relatively prime. If this is too hard to see, you can find the GCD of the two numbers from the command line using the command `gcd(R1, n)` where n is the modulus. If the GCD if the two numbers is one you may proceed.
4. Take the modular inverse of the determinant by the command, `inv_mod(R1, n)`, say that this is stored in R2.
5. Find the classical adjoint of the matrix with the command, `adjoint(M1)`, say this is stored in M2. The result of this operation will not be modulo n.
6. Take M2 modulo n using the command, `mod(M2, n)`, say that this is stored in M3.
7. Multiply the inverse of the determinant by the adjoint with, `R2*M3`, say this is stored in M4.
8. Take M4 modulo n, `mod(M4, n)`, say that this is stored in M5. This is the inverse of M1 modulo n.

6.2.3 Some Matrix Operations

Below we give a short description of some of the other matrix operations that can be done in the Linear ME software package. We have only listed those operations that may be of use for the manipulations needed when doing work with Hill ciphers. For a complete list of operations and descriptions of them please see the help system in the Linear ME package.

Triangularize: Returns the upper triangular form of the matrix M, as produced by Gaussian elimination. The return value is the same as Echelon Form, except that the leading nonzero coefficient in each row is not normalized to 1.

To triangularize a matrix in Linear ME, select the desired matrix and then,

Operations > Triangularize

from the menu. At this point the new matrix will be loaded into the workspace.

You can also triangularize a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
triangularize(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Echelon Form: Returns the echelon form of the matrix M, as produced by Gaussian elimination. The echelon form is computed from M by elementary row operations such that the first non-zero element in each row in the resulting matrix is one and the column elements under the first one in each row are all zero.

To find the above echelon form of a matrix in Linear ME, select the desired matrix and then,

Operations > Echelon Form

from the menu. At this point the new matrix will be loaded into the workspace.

You can also produce the echelon form a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
echelon(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Reduced Echelon Form: Returns the echelon form of the matrix M, as produced by Gaussian elimination. The echelon form is computed from M by elementary row operations such that the first non-zero element in each row in the resulting matrix is one and the column elements both above and below the first one in each row are all zero.

To find the reduced echelon form of a matrix in Linear ME, select the desired matrix and then,

Operations > Reduced Echelon Form

from the menu. At this point the new matrix will be loaded into the workspace.

Determinant: Returns the determinant of a square matrix.

To find the determinant of a matrix in Linear ME, select the desired matrix and then,

Operations > Determinant

from the menu. At this point the determinant of the matrix will be loaded into the workspace.

You can also find the determinant of a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
determinant(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Adjoint: Returns the classical adjoint of a square matrix.

To find the classical adjoint of a square matrix in Linear ME, first load the desired matrix into the workspace and then on the Maxima command line type in

```
adjoint(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Transpose: Returns the transpose of a matrix.

To find the transpose of a matrix in Linear ME, select the desired matrix and then,

Operations > Transpose

from the menu. At this point the transpose of the matrix will be loaded into the workspace.

You can also find the transpose of a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
transpose(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

Invert: Returns the inverse of a matrix if the determinant is not 0. Note that this is an inverse over the real or complex number system and not a modular inverse. For modular inverses see the section on Modular Arithmetic.

To find the inverse of a matrix in Linear ME, select the desired matrix and then,

Operations > Invert

from the menu. At this point the inverse of the matrix will be loaded into the workspace, if it exists.

You can also find the inverse of a matrix from the Maxima command line at the bottom of the screen. First load the desired matrix into the workspace and then on the command line type in

```
invert(M1)
```

and hit the evaluate key, where M1 is the name of the desired matrix.

6.2.4 Lists

Computer algebra systems (CAS) like Mathematica, Maxima, and Maple are built to work on lists. In fact, the internal structure of most computer algebra systems rely heavily on the storage and manipulation of lists. They can also be very convenient for the end user in that they allow the same operations to be done on many inputs at once without the need to redo each of the operations.

To create a list in Linear ME you use the same syntax as you would using Maxima, begin the list with a `[` end it with a `]` and separate the entries by commas. A list can be input either from the command line or using the new expression dialog box which you can get by selecting Edit > New Expression... from the main menu. In Maxima, and hence Linear ME, many of the functions and arithmetic will work as well on lists as they do on non-list expressions.

Example 13: To create the list 0, 1, 2, 3, 4, 5 simply type `[0, 1, 2, 3, 4, 5]` into the Maxima command line at the bottom of the screen and click the Evaluate button, or select Edit > New Expression... from the main menu and type it into the Expression box of the dialog that appears and click the OK button. At this point you should see

$$[0, 1, 2, 3, 4, 5]$$

in the workspace. Let's suppose that this list was stored as R1 in the workspace.

1. If we were to enter `2*R1` into the Maxima command line and click Evaluate we would have,
$$[0, 2, 4, 6, 8, 10]$$

2. If we were to enter `3*R1+4` into the Maxima command line and click Evaluate we would have,
$$[4, 7, 10, 13, 16, 19]$$

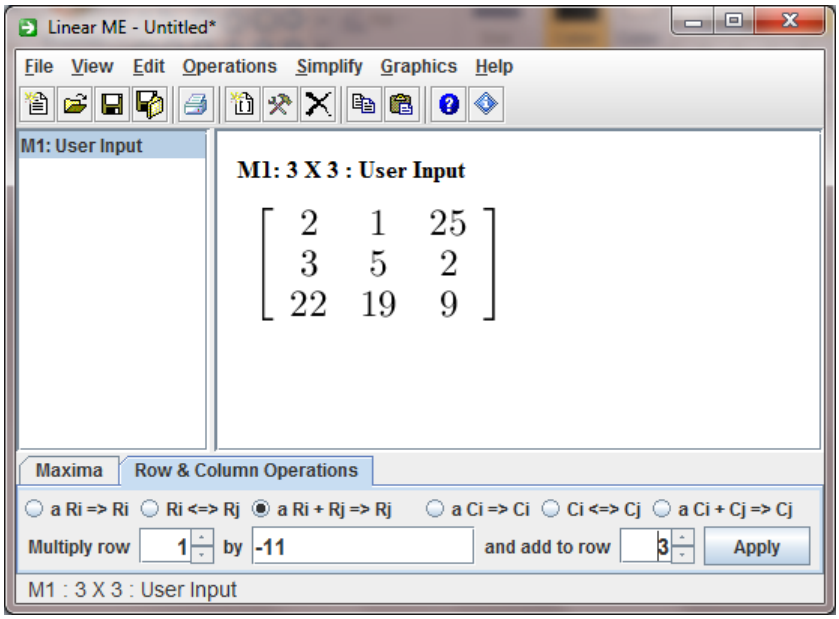
3. If we were to enter `R1^2` into the Maxima command line and click Evaluate we would have,
$$[0, 1, 4, 9, 16, 25]$$

4. If we were to enter `cos(R1)` into the Maxima command line and click Evaluate we would have,
$$[1, \cos 1, \cos 2, \cos 3, \cos 4, \cos 5]$$

5. If we were to enter `mod(R1, 3)` into the Maxima command line and click Evaluate we would have,
$$[0, 1, 2, 0, 1, 2]$$

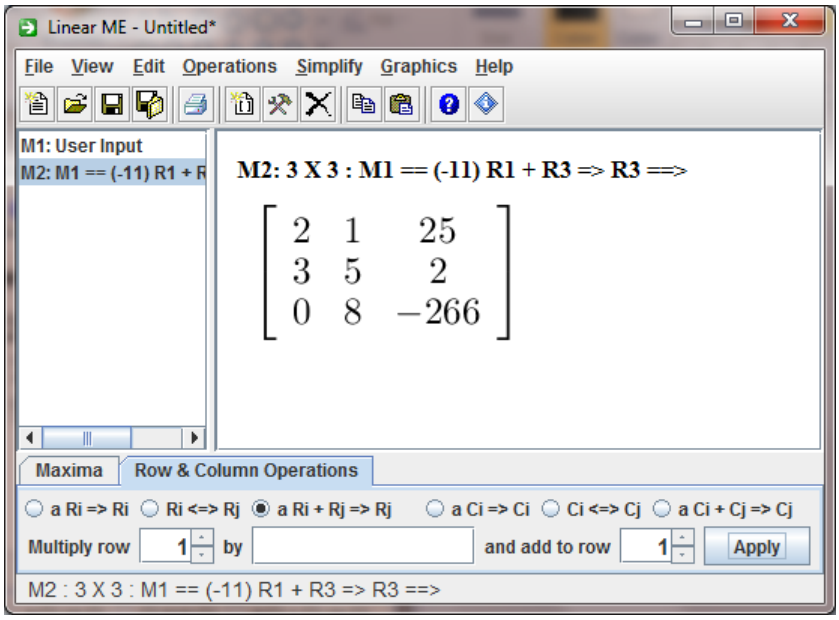
6.2.5 Row & Column Operations Interface

At the bottom of the Linear ME window you will see a tab titled Row & Column Operations. If you click on that tab you will bring up the Row & Column Operation interface, as shown below.



The primary purpose of this interface is to help teach students Gauss-Jordan elimination. It is also a good tool for experimenting with the effects of row and column operations on the determinant. For the working modulo n, as with the Hill cipher, this interface will allow the user to do row operations one at a time and then after each step, use the modular operations to mod the matrix by n. Since the built-in options for solving systems are programed to work over the real and complex number systems, if the user needs to solve a system in a modular setting they will need to do the reduction process by hand, this interface will make that process easier.

Along the top of the tab the user selects the type of operation they want to do. When the selection is made the data boxes below the selection change according to the selection. The user then fills in the information on the operation and clicks Apply to do the operation. When this is done a new matrix, resulting from the operation, will be added to the workspace.



The row or columns you are working with are controlled by spinners that do not allow you to go outside the matrix. The multiplier can be any valid Maxima expression, but in the case of reducing a matrix modulo n the expressions will simply be integers.

6.3 Mathematica

6.3.1 Matrix Arithmetic

To create a matrix in Mathematica you use the following syntax,

```
M1 := {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
```

This will produce the matrix,

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

and assign it to the name M1. So, in general, the first set of brackets defines the matrix and the inside sets define each row. That is, the first row is `{1, 2, 3}`. Each row is separated by a comma and each entry in the row is separated by a comma. You can, of course, use the pallets if you would like. Also, the command `MatrixForm[M1]` will print out the matrix in a nice row/column form.

For the syntax descriptions below we will assume that the matrices have already been loaded into a Mathematica worksheet, their names are M1 and M2 and their sizes are compatible with the operations. If they are not then Mathematics will return some type of error message.

Addition: `M1 + M2`

Subtraction: `M1 + M2`

Multiplication: `M1 . M2`

Note: You use `.` and not `*` when doing matrix multiplication.

Scalar Multiplication: `c*M1`

where c is the scalar.

Note: You use `*` and not `.` when doing scalar multiplication.

Negation: `-M1`

Powers: `MatrixPower[M1, n]`

where n is the integer power.

Note: Do not try `^` or `^^` when doing matrix powers in Mathematica. The single caret will raise each entry to the power and the double caret will produce an error.

6.3.2 Modular Arithmetic

Number Modulo n: `Mod[m, n]`

this will find $m \pmod{n}$.

Matrix Modulo n: `Mod[M1, n]`

this will find the matrix M1, modulo n.

Inverse of a Number Modulo n: The syntax `PowerMod[a,b,n]` will find $a^b \pmod{n}$. So the command `PowerMod[a,-1,n]` will find the inverse of a modulo n, if it exists. If it does not exist then Mathematica will return an error.

Inverse of a Matrix Modulo n: `Inverse[M1, Modulus -> n]`

where n is the modulus.

6.3.3 Some Matrix Operations

Below we give a short description of some of the other matrix operations that can be done in Mathematica. We have only listed those operations that may be of use for the manipulations needed when doing work with Hill ciphers. For a complete list of operations and descriptions of them please see Mathematica's Documentation Center.

RowReduce: RowReduce performs a version of Gaussian elimination, adding multiples of rows together so as to produce zero elements when possible. The final matrix is in reduced row echelon form. The syntax,

RowReduce[M1]

will return the reduced row echelon form of the matrix M1.

Determinant: Returns the determinant of a square matrix. The syntax,

Det[M1]

will return the determinant of the matrix M1.

Transpose: Returns the transpose of a matrix. The syntax,

Transpose[M1]

will return the transpose of the matrix M1.

Inverse: Returns the inverse of a matrix if the determinant is not 0. Note that this is an inverse over the real or complex number system and not a modular inverse. For modular inverses see the section on Modular Arithmetic. The syntax,

Inverse[M1]

will return the transpose of the matrix M1.

6.3.4 Lists

Computer algebra systems (CAS) like Mathematica, Maxima, and Maple are built to work on lists. In fact, the internal structure of most computer algebra systems rely heavily on the storage and manipulation of lists. They can also be very convenient for the end user in that they allow the same operations to be done on many inputs at once without the need to redo each of the operations.

To create a list in Mathematica, begin the list with a `{` end it with a `}` and separate the entries by commas. Equivalently, you can use the List command, begin the command with `List[`, end it with a `]` and separate the entries by commas.

Example 14: To create the list 0, 1, 2, 3, 4, 5 and assign it to the name `A` simply type `A = {0, 1, 2, 3, 4, 5}` or `A = List[0, 1, 2, 3, 4, 5]` into Mathematica and hit Shift+Enter to evaluate the expression. At this point you should see

$$\{0, 1, 2, 3, 4, 5\}$$

in the Mathematica workspace.

1. If we were to enter `2*A` into Mathematica we would have,

$$\{0, 2, 4, 6, 8, 10\}$$

2. If we were to enter `3*A+4` into Mathematica we would have,

$$\{4, 7, 10, 13, 16, 19\}$$

3. If we were to enter `A^2` into Mathematica we would have,

$$\{0, 1, 4, 9, 16, 25\}$$

4. If we were to enter `Cos[A]` into Mathematica we would have,

$$\{1, \text{Cos}[1], \text{Cos}[2], \text{Cos}[3], \text{Cos}[4], \text{Cos}[5]\}$$

5. If we were to enter `Mod[A, 3]` into Mathematica we would have,

$$\{0, 1, 2, 0, 1, 2\}$$

6.4 Maxima

6.4.1 Matrix Arithmetic

To create a matrix in Maxima you use the following syntax,

```
M1 : matrix( [1,2,3], [4,5,6], [7,8,9] );
```

This will produce the matrix,

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

and assign it to the name M1. So, in general, you start out with the the matrix command and in parentheses type in each row inside square brackets. The rows are separated by commas and the row entries are separated by commas.

For the syntax descriptions below we will assume that the matrices have already been loaded into a Maxima worksheet, their names are M1 and M2 and their sizes are compatible with the operations. If they are not then Maxima will return some type of error message.

Addition: `M1 + M2`

Subtraction: `M1 + M2`

Multiplication: `M1 . M2`

Note: You use `.` and not `*` when doing matrix multiplication.

Scalar Multiplication: `c*M1`

where c is the scalar.

Note: You use `*` and not `.` when doing scalar multiplication.

Negation: `-M1`

Powers: `M1^^n`

where n is the integer power.

Note: Use `^^` and not `^` when doing matrix powers in Maxima. The single caret will raise each entry to the power.

6.4.2 Modular Arithmetic

Number Modulo n: `mod(m, n)`

where m is the desired number and n is the modulus.

Matrix Modulo n: `mod(M1, n)`

where M1 is the and n is the modulus.

Inverse of a Number Modulo n: `inv_mod(m, n)`

where m is the desired number and n is the modulus.

Inverse of a Matrix Modulo n: First load the desired matrix into the workspace and then do the following sequence of commands,

1. First load the desired matrix into the workspace, Say it is called M1.
2. Find the determinant of the matrix, say it is called R1.
3. Make sure that the determinant and the modulus are relatively prime. If this is too hard to see, you can find the GCD of the two numbers from the command line using the command `gcd(R1, n)` where n is the modulus. If the GCD if the two numbers is one you may proceed.
4. Take the modular inverse of the determinant by the command, `inv_mod(R1, n)`, say that this is stored in R2.
5. Find the classical adjoint of the matrix with the command, `adjoint(M1)`, say this is stored in M2. The result of this operation will not be modulo n.

6. Take M2 modulo n using the command, `mod(M2, n)`, say that this is stored in M3.
7. Multiply the inverse of the determinant by the adjoint with, `R2*M3`, say this is stored in M4.
8. Take M4 modulo n, `mod(M4, n)`, say that this is stored in M5. This is the inverse of M1 modulo n.

6.4.3 Some Matrix Operations

Below we give a short description of some of the other matrix operations that can be done in the Maxima software package. We have only listed those operations that may be of use for the manipulations needed when doing work with Hill ciphers. For a complete list of operations and descriptions of them please see the Maxima help system.

Triangularize: Returns the upper triangular form of the matrix M, as produced by Gaussian elimination. The return value is the same as Echelon Form, except that the leading nonzero coefficient in each row is not normalized to 1. The syntax,

```
triangularize(M1)
```

will return the triangularized form of the matrix M1.

Echelon: Returns the echelon form of the matrix M, as produced by Gaussian elimination. The echelon form is computed from M by elementary row operations such that the first non-zero element in each row in the resulting matrix is one and the column elements under the first one in each row are all zero. The syntax,

```
echelon(M1)
```

will return the above echelon form of the matrix M1.

Determinant: Returns the determinant of a square matrix. The syntax,

```
determinant(M1)
```

will return the determinant of the matrix M1.

Adjoint: Returns the classical adjoint of a square matrix. The syntax,

```
adjoint(M1)
```

will return the classical adjoint of the matrix M1.

Transpose: Returns the transpose of a matrix. The syntax,

```
transpose(M1)
```

will return the transpose of the matrix M1.

Invert: Returns the inverse of a matrix if the determinant is not 0. Note that this is an inverse over the real or complex number system and not a modular inverse. For modular inverses see the section on Modular Arithmetic. The syntax,

```
invert(M1)
```

will return the inverse of the matrix M1, if it exists.

6.4.4 Lists

Computer algebra systems (CAS) like Mathematica, Maxima, and Maple are built to work on lists. In fact, the internal structure of most computer algebra systems rely heavily on the storage and manipulation of lists. They can also be very convenient for the end user in that they allow the same operations to be done on many inputs at once without the need to redo each of the operations.

To create a list in Maxima, begin the list with a `[` end it with a `]` and separate the entries by commas.

Example 15: To create the list 0, 1, 2, 3, 4, 5 and assign it to the name `list1` simply type `list1: [0, 1, 2, 3, 4, 5]` into Maxima and hit Shift+Enter to evaluate the expression. At this point you should see

$$[0, 1, 2, 3, 4, 5]$$

in the Maxima workspace.

1. If we were to enter `2*list1` into Maxima we would have,

$$[0, 2, 4, 6, 8, 10]$$

2. If we were to enter `3*list1+4` into Maxima we would have,

$$[4, 7, 10, 13, 16, 19]$$

3. If we were to enter `list1^2` into Maxima we would have,

$$[0, 1, 4, 9, 16, 25]$$

4. If we were to enter `cos(list1)` into Maxima we would have,

$$[1, \cos 1, \cos 2, \cos 3, \cos 4, \cos 5]$$

5. If we were to enter `mod(list1, 3)` into Maxima we would have,

$$[0, 1, 2, 0, 1, 2]$$

7 Exercises

In some of these exercises you may want, or be required, to do some matrix reduction. Remember that in this context all operations are done modulo 26. So, as we did above, if you divide a row by a number make sure that the number is invertible modulo 26 and multiply the row by its inverse. If you are using Linear ME to help with the calculations you should use the the Row and Column Operations interface at the bottom of the window and mod your results by 26 whenever necessary.

1. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 2 & 7 \\ 13 & 9 \end{bmatrix}$$

   (a) What is the determinant of E?

   (b) Is E invertible modulo 26? Why or why not?

   (c) If E is invertible modulo 26, find its inverse modulo 26.

   (d) Use E to encrypt the message HELP.

   (e) What message or messages encrypt to XXXX?

2. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 2 & 7 \\ 15 & 8 \end{bmatrix}$$

   (a) What is the determinant of E?

   (b) Is E invertible modulo 26? Why or why not?

   (c) If E is invertible modulo 26, find its inverse modulo 26.

   (d) Use E to encrypt the message HELP.

   (e) What message or messages encrypt to XXXX?

3. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 2 & 6 \\ 15 & 8 \end{bmatrix}$$

   (a) What is the determinant of E?

   (b) Is E invertible modulo 26? Why or why not?

   (c) If E is invertible modulo 26, find its inverse modulo 26.

   (d) Use E to encrypt the message HELP.

(e) What message or messages encrypt to XXXX?

(f) Are there any other plaintext messages that encrypt to the same ciphertext as does HELP? If so, find all plaintext messages that encrypt to the same ciphertext as does HELP. If not, explain why.

4. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 8 & 7 & 17 \\ 19 & 18 & 4 \\ 20 & 1 & 4 \end{bmatrix}$$

(a) What is the determinant of E?

(b) Is E invertible modulo 26? Why or why not?

(c) If E is invertible modulo 26, find its inverse modulo 26.

(d) Use E to encrypt the message ATTACK.

(e) What message or messages encrypt to ATTACK?

(f) Are there any other plaintext messages that encrypt to the same ciphertext as does ATTACK? If so, find all plaintext messages that encrypt to the same ciphertext as does ATTACK. If not, explain why.

(g) Are there any plaintext messages that are sent to the zero vector? If so, find all of them and if not explain why?

(h) Compare your results from the last two parts of this exercise, what is the relation between these two answers?

5. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 8 & 7 & 18 \\ 19 & 18 & 4 \\ 20 & 1 & 4 \end{bmatrix}$$

(a) What is the determinant of E?

(b) Is E invertible modulo 26? Why or why not?

(c) If E is invertible modulo 26, find its inverse modulo 26.

(d) Use E to encrypt the message CHARGE.

(e) What message or messages encrypt to CHARGE?

(f) Are there any other plaintext messages that encrypt to the same ciphertext as does CHARGE? If so, find all plaintext messages that encrypt to the same ciphertext as does CHARGE. If not, explain why.

(g) Are there any plaintext messages that are sent to the zero vector? If so, find all of them and if not explain why?

(h) Compare your results from the last two parts of this exercise, what is the relation between these two answers?

6. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 4 & 8 & 5 & 22 \\ 8 & 16 & 21 & 17 \\ 21 & 25 & 5 & 17 \\ 17 & 4 & 9 & 10 \end{bmatrix}$$

(a) What is the determinant of E?

(b) Is E invertible modulo 26? Why or why not?

(c) If E is invertible modulo 26, find its inverse modulo 26.

(d) Use E to encrypt the message RUN AWAY. Note that since RUN AWAY, when the space is removed, has only seven letters we will need to pad the end with a random letter, in many cases an X is used. we will use this convention and encrypt RUNAWAYX.

(e) What message or messages encrypt to RUNAWAYX?

(f) Are there any other plaintext messages that encrypt to the same ciphertext as does RUNAWAYX? If so, find all plaintext messages that encrypt to the same ciphertext as does RUNAWAYX. If not, explain why.

(g) Are there any plaintext messages that are sent to the zero vector? If so, find all of them and if not explain why?

(h) Compare your results from the last two parts of this exercise, what is the relation between these two answers?

7. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 24 & 17 & 20 & 17 \\ 25 & 21 & 7 & 9 \\ 23 & 22 & 2 & 5 \\ 3 & 17 & 18 & 23 \end{bmatrix}$$

(a) What is the determinant of E?

(b) Is E invertible modulo 26? Why or why not?

(c) If E is invertible modulo 26, find its inverse modulo 26.

(d) Use E to encrypt the message RUN AWAY. Note that since RUN AWAY, when the space is removed, has only seven letters we will need to pad the end with a random letter, in many cases an X is used. we will use this convention and encrypt RUNAWAYX.

(e) What message or messages encrypt to RUNAWAYX?

(f) Are there any other plaintext messages that encrypt to the same ciphertext as does RUNAWAYX? If so, find all plaintext messages that encrypt to the same ciphertext as does RUNAWAYX. If not, explain why.

(g) Are there any plaintext messages that are sent to the zero vector? If so, find all of them and if not explain why?

(h) Compare your results from the last two parts of this exercise, what is the relation between these two answers?

8. Consider the following matrix with entries from $\mathbb{A}$,

$$E = \begin{bmatrix} 20 & 15 & 7 & 17 \\ 23 & 5 & 9 & 19 \\ 3 & 5 & 25 & 22 \\ 9 & 25 & 15 & 1 \end{bmatrix}$$

(a) What is the determinant of E?

(b) Is E invertible modulo 26? Why or why not?

(c) If E is invertible modulo 26, find its inverse modulo 26.

(d) Use E to encrypt the message RUN AWAY. Note that since RUN AWAY, when the space is removed, has only seven letters we will need to pad the end with a random letter, in many cases an X is used. we will use this convention and encrypt RUNAWAYX.

(e) What message or messages encrypt to RUNAWAYX?

(f) Are there any other plaintext messages that encrypt to the same ciphertext as does RUNAWAYX? If so, find all plaintext messages that encrypt to the same ciphertext as does RUNAWAYX. If not, explain why.

(g) Are there any plaintext messages that are sent to the zero vector? If so, find all of them and if not explain why?

(h) Compare your results from the last two parts of this exercise, what is the relation between these two answers?

9. Which of the matrices in the above examples could be used for a Hill cipher and which are not suitable? In each case explain why?

10. For each of the matrices in the previous exercises that are invertible, find their inverses using,

(a) The reduction technique.

(b) The adjoint technique.

11. For each of the matrices in the previous exercises that are not invertible, reduce them as far as possible. Do any reduce to the identity matrix?

12. For each of the matrix transformations in the previous exercises, find the kernel of the transformation.

13. For each of the matrices in the previous exercises, find a maximum set of columns that are linearly independent.

14. For each of the matrices in the previous exercises, which ones have columns that form a basis to $\mathbb{A}^n$? Why?

15. In the description of the Hill cipher we require that the encryption matrix E is invertible modulo 26. What properties of an invertible matrix are important in the encryption and decryption process and why?

16. We have intercepted a segment of the plaintext of a message

    SENDMORETROOPS

    and we have the corresponding ciphertext

    SYTKAWQVHOECOB

    We know that the segment was at the beginning of the message and that there was more ciphertext that followed but we do not know the corresponding plaintext. We, of course, wish to decrypt the entire message, so we need the decryption matrix. Find the encryption and decryption matrices for this cipher, if possible. If there is not enough information to do this, state why.

17. We have intercepted a segment of the plaintext of a message

    WEAREINNEEDOFASSISTANCER

    and we have the corresponding ciphertext

    KWULUEKTYVJEZCQAGSPWRWYJ

    We know that the segment was at the beginning of the message and that there was more ciphertext that followed but we do not know the corresponding plaintext. We, of course, wish to decrypt the entire message, so we need the decryption matrix. Find the encryption and decryption matrices for this cipher, if possible. If there is not enough information to do this, state why.

18. We have intercepted the plaintext of a message

    INEEDABIGCUPOFCOFFEE

and we have the corresponding ciphertext

YSPMYLQNPBTDGWLLXILVV

Find the encryption and decryption matrices for this cipher, if possible. If there is not enough information to do this, state why.

We suspect that the same matrix was used to send messages for that entire day. Later we intercepted the following ciphertext, find the message, if possible.

JRLACWNLZQEYBSHOWQSFMXFFIKFAXSCOT

19. Say that Alice uses the following matrix for her encryption matrix,

$$E = \begin{bmatrix} 2 & 7 \\ 13 & 9 \end{bmatrix}$$

and encrypts the message,

SOLONGANDTHANKSFORALLTHEFISH

into

EWQJQPNNJCONSZTTRXZVZCQXOHHL

When she sends the message to Bob there is one error in the transmission and Bob receives the ciphertext,

EWQJQPNNJCONSZTARXZVZCQXOHHL

When Bob decrypts the message how many errors are in his decryption?

20. Say that Alice uses the following matrix for her encryption matrix,

$$E = \begin{bmatrix} 23 & 7 & 22 & 3 \\ 25 & 16 & 3 & 4 \\ 6 & 22 & 6 & 7 \\ 7 & 20 & 8 & 6 \end{bmatrix}$$

and encrypts the message,

SOLONGANDTHANKSFORALLTHEFISH

into

QJIGQFPDSKKPANZXGQPKGSIRQXZR

When she sends the message to Bob there are two errors in the transmission and Bob receives the ciphertext,

QMIGQFPDSKKPYNZXGQPKGSIRQXZR

When Bob decrypts the message how many errors are in his decryption?

21. From the two exercises above, what is the relationship between transmission errors and errors in the decryption of the message? Does it make any difference where the errors are, for example, if the errors are close together or far apart?

    In World War I the German Military used the ADFGX code. In this code, the first step was to rewrite each letter in the message as a pair of letters from the set $\{A, D, F, G, X\}$, for example, a might be coded as DF, b as AA, and so on. From there the encryption method used only the characters ADFGX. The reason for this choice was that in Morse code, the method of transmission, the letters A, D, F, G, and X were easy to distinguish. So if there was a transmission error, usually human error in this case, the receiver could easily tell what the letter should have been. This was one of the first uses of error correcting codes in cryptography. Today, the use of error correction codes is commonplace in cryptography.

22. Say we use the following matrix to encrypt the message,

    ATTACK

$$E = \begin{bmatrix} 8 & 19 & 9 & 22 & 7 & 10 \\ 6 & 18 & 25 & 25 & 10 & 0 \\ 0 & 25 & 24 & 0 & 24 & 0 \\ 0 & 22 & 0 & 6 & 16 & 3 \\ 19 & 6 & 10 & 0 & 0 & 7 \\ 19 & 23 & 10 & 10 & 21 & 23 \end{bmatrix}$$

    (a) What does the message encrypt to?

    (b) Find the decryption matrix.

    (c) We will now put in a single transmission error. Change the third letter of the ciphertext to a C, and decrypt the new (flawed) message.

    (d) How many errors are in the flawed decryption? Is it possible to recover the original message with this decryption?

    (e) If we do not know the number or position of the errors in the transmission of a single word, as is usually the case, what are all of the possible plantext messages with one word that could be received as the ciphertext ABCDEF?

23. In some of the exercises above we examined what happens if there is an error in the transmission of the ciphertext. This exercise looks at what happens if we have an error in the encryption matrix. Say Alice and Bob share the key, the encryption matrix, but Bob's handwriting is not all that great and he cannot read the entry in the first row and third column but he knows that the rest of the encryption matrix is correct. Bob currently has,

$$E = \begin{bmatrix} 20 & 22 & x \\ 1 & 5 & 5 \\ 7 & 4 & 20 \end{bmatrix}$$

    (a) Just knowing this, what are the possibilities for the value of x? How did you come to this conclusion?

    (b) Later that day Bob receives the following ciphertext from Alice

    IXHSZCMQOERVTCZ

    From this, can Bob determine the value of x? If so what is it and what does the message say?

24. In our discussion above we stated that we can think of the encryption matrix, E, as a linear transformation $E : \mathbb{A}^n \to \mathbb{A}^n$. Formally, we define $E(\mathbf{v}) = E\mathbf{v}$ for all $\mathbf{v} \in \mathbb{A}^n$. With this definition, show that E is a linear transformation.

25. In the definition of the Hill cipher, the matrix E must be invertible modulo 26. As we saw, that meant that the determinant of E had to be invertible in $\mathbb{A}$, which is where the entries of E came from. As we noted above as well, this was in line with the Invertible Matrix Theorem, one part of which states that an $n \times n$ matrix with real coefficients is invertible if and only if the determinant of that matrix is not 0. That is, if the determinant is invertible in the real number system. Since the only real number that is not invertible is 0 we need only exclude it and can simplify the statement. In this exercise we are going to examine some of the other parts of the Invertible Matrix Theorem in relation to $\mathbb{A}$ and its implications to the Hill cipher. Also recall that we can think of the encryption matrix, E, as a linear transformation $E : \mathbb{A}^n \to \mathbb{A}^n$. We will use these two points of view interchangeably.

    (a) Say that E is invertible. What does this imply about the properties of one-to-one and onto of the transformation?

    (b) Say that E is not invertible. What does this imply about the properties of one-to-one and onto of the transformation?

    (c) If the transformation was not one-to-one, what would this imply about encryption and decryption with the Hill cipher? Be specific as to the difficulties that would arise.

    (d) If the transformation was not onto, what would this imply about encryption and decryption with the Hill cipher? Be specific as to the difficulties that would arise.

(e) If E is invertible, what is the kernel of E? How does this relate to your answers above?

(f) If E is not invertible, what is the kernel of E? How does this relate to your answers above?

(g) If E is not invertible, is it possible that two English messages could be sent to the same ciphertext? How does this relate to your answers above? Find two examples of non-invertible encryption matrices E and for each, two different English plaintexts that are encrypted to the same ciphertext.

(h) If E is not invertible, and you know one decryption of a given ciphertext as well as the kernel of E, how can you determine, without knowing E, what all of the possible decryptions are? Justify your answer.

(i) If E is invertible, what can be said about the columns of E in relation to the set $\mathbb{A}^n$? For each observation, discuss the relevance of it to the Hill cipher.

26. In the work we did above, we used a Known Plaintext attack on the cipher to find its key, that is, the encryption matrix. One question would be if other types of attacks could gather enough information for us to determine the encryption, and hence decryption matrices?

(a) Chosen Plaintext: Say that Eve temporally gains access to the encryption machine. Think of it as a black box that she can input messages and get the encryption back.

i. Suppose further that she knows the block size. What plaintexts should she choose to find the encryption matrix? How would she use the output of the machine to construct the encryption matrix? If she knows that she has only a few tries before she is detected and locked out of the system, what is the fewest number of plaintexts she needs to determine the encryption matrix?

ii. Suppose she does not know the block size and the machine automatically pads the input with X's as well as breaks the message into blocks. What plaintexts should she choose to find the encryption matrix? How would she use the output of the machine to construct the encryption matrix? If she knows that she has only a few tries before she is detected and locked out of the system, what is the fewest number of plaintexts she needs to determine the encryption matrix?

(b) Chosen Ciphertext: Say that Eve temporally gains access to the decryption machine. Think of it also as a black box that she can input ciphertext and get the plaintext back.

i. Suppose further that she knows the block size. What ciphertexts should she choose to find the encryption matrix? How would she use the output of the machine to construct the encryption matrix? If she knows that she has only

a few tries before she is detected and locked out of the system, what is the fewest number of ciphertexts she needs to determine the encryption matrix?

ii. Suppose she does not know the block size and the machine automatically pads the input with X's as well as breaks the message into blocks. What ciphertexts should she choose to find the encryption matrix? How would she use the output of the machine to construct the encryption matrix? If she knows that she has only a few tries before she is detected and locked out of the system, what is the fewest number of ciphertexts she needs to determine the encryption matrix?

(c) Ciphertext Only: Here Eve has only the ciphertext of several messages.

i. How could she determine the block size? Or at least a small set of possible block sizes.

ii. Once she has determined the set of possible block sizes, how would she proceed to find the encryption matrix?

iii. In many cases a simple substitution cipher can be broken using only a single ciphertext, if it is sufficiently long. Could Eve break the Hill cipher with only a single, fairly long, ciphertext? If so, what restrictions would need to be assumed?

27. Look back at the definition of a vector space. If we let our set of scalars be $\mathbb{A}$ instead of $\mathbb{R}$ and if we let $V = \mathbb{A}^n$ instead of $\mathbb{R}^n$, which of the properties of a vector space still hold for V and which do not? For each, if the property holds prove it and if the property does not hold then find a counter example to show that the property fails.

28. The Hill cipher is defined to work with square encryption matrices E. Since the decryption matrix $D = E^{-1}$, it is clear why this restriction is imposed. One question would be if we could relax this condition and let E be an $n \times m$ matrix where $n \neq m$? Our immediate answer to this would be no since if E was not square, it would not have an inverse D for decryption.

    On the other hand, if we think about E and D we see that there is a little overkill on this restriction. Specifically, when Alice encrypts her message M, she divides the message into blocks of n letters that is then converted into a column of n numbers. Each block, or column, can, and is, looked at as a vector $\mathbf{v} \in \mathbb{A}^n$. Then when she encrypts her message, she applies E to each of the vectors, getting an encrypted vector $\mathbf{w} = E\mathbf{v}$. Since D is the inverse of E we know that $D\mathbf{w} = \mathbf{v}$ and hence we can decrypt the message. So what we require is that $DE\mathbf{v} = D(E\mathbf{v}) = D\mathbf{w} = \mathbf{v}$ for all vectors $\mathbf{v} \in \mathbb{A}^n$. But if D is the inverse of E we also have that $ED\mathbf{v} = \mathbf{v}$ for all vectors $\mathbf{v} \in \mathbb{A}^n$, which is not required by the Hill cipher algorithm. That is, we only need to get back to $\mathbf{v}$ with the product DE and not ED.

    Before we get to far into this lets step back and ask the same question but with matrices over the real number system. That is, can we produce two non-square matrices A and

B such that $AB = I$ but $BA \neq I$? From the products it is clear that if A is $n \times m$ then B must be $m \times n$ and the products AB will be $n \times n$ and BA will be $m \times m$.

(a) Consider the matrix

$$E = \begin{bmatrix} -6 & -4 & 7 \\ 1 & 5 & 5 \end{bmatrix}$$

over the real numbers, does there exist a matrix D with $DE = I$? To prove or disprove this construct a generic 3×2 matrix D,

$$D = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

and compute

$$DE = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix} \begin{bmatrix} -6 & -4 & 7 \\ 1 & 5 & 5 \end{bmatrix}$$

From this, can we solve for each of the components of D? If so, then we have $DE = I$ as desired. Also, if it is possible one would ask if D is unique?

i. Do the above product and solve the resulting system, over the real numbers, does a solution exist?

ii. If a solution exists write the matrix D.

iii. Is D unique?

iv. If not, write a general expression for all matrices D with $DE = I$.

(b) Consider the matrix

$$E = \begin{bmatrix} -6 & 1 \\ 1 & 5 \\ 7 & 4 \end{bmatrix}$$

over the real numbers, does there exist a matrix D with $DE = I$? To prove or disprove this construct a generic 2×3 matrix D,

$$D = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

and compute

$$DE = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} -6 & 1 \\ 1 & 5 \\ 7 & 4 \end{bmatrix}$$

From this, can we solve for each of the components of D? If so, then we have $DE = I$ as desired. Also, if it is possible one would ask if D is unique?

i. Do the above product and solve the resulting system, over the real numbers, does a solution exist?

ii. If a solution exists write the matrix D.

iii. Is D unique?

iv. If not, write a general expression for all matrices D with $DE = I$.

(c) From the above calculations what would you conjecture as the answer to the question,

> Given an $n \times m$ matrix A does there exist an $m \times n$ matrix B with $AB = I$?

What restrictions must be placed on n and m? With these restrictions, is it always possible? Why or why not?

(d) Think of the matrices A and B as transformations between $\mathbb{R}^n$ and $\mathbb{R}^m$. Revise your statement about the restrictions on the sizes of n and m to the language of transformations. That is, discuss the properties of one-to-one and onto of the transformation induced by A and B and the necessity of these restrictions needed for AB to be both one-to-one and onto, as is the identity matrix.

(e) If we can find two non-square matrices A and B such that the matrix AB produces a one-to-one and onto transformation can we find another matrix C with $AC = I$ or can we find another matrix D with $DB = I$? If so, how would we create either C or D? If not find two matrices A and B with the matrix AB producing a one-to-one and onto transformation but there exists no matrices C and D with either $AC = I$ or $DB = I$.

(f) Discuss this situation with matrices over $\mathbb{A}$ in place of matrices over $\mathbb{R}$. Specifically,

i. Is it possible to relax the Hill requirement of using a square matrix?

ii. If so, what restrictions must be placed on n and m?

iii. If so, with these restrictions, can this always be done? Why or why not?

iv. If so, give examples of non-square matrices E and D over $\mathbb{A}$ with $ED = I$ modulo 26.

29. In the Hill cipher we do all of calculations modulo 26 since, of course, there are 26 letters in the English alphabet. What if we had a different alphabet? Say we had an alphabet with 27 letters, or 29 letters. What if our language symbolism was based on syllables instead of letters, then we would probably have around 80 to 100 characters? What if our language were based on pictures, then we could have thousands of characters. Updating the Hill cipher would be easy in this case, all we do is change the modulus.

(a) Say our alphabet had only 5 letters. What would the determinant of a Hill cipher encryption matrix need to be?

(b) Say our alphabet had only 12 letters. What would the determinant of a Hill cipher encryption matrix need to be?

(c) Say our alphabet had 29 letters. What would the determinant of a Hill cipher encryption matrix need to be?

(d) What would the advantage be to the Hill cipher if we had a prime number of letters in our alphabet?

(e) Say we had an alphabet with 5 letters in it, $\{A, B, C, D, E\}$. So in this case $\mathbb{A} = \{0, 1, 2, 3, 4\}$, with addition and multiplication done modulo 5. Consider the following matrix with entries in $\mathbb{A}$.

$$E = \begin{bmatrix} 4 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 4 & 4 \end{bmatrix}$$

i. What is the determinant of E?

ii. Could E be used as an encryption matrix for a Hill cipher?

iii. If so, find the decryption matrix D using both the reduction technique and the adjoint technique.

iv. If so, encrypt the message ABBEDA.

v. If so, decrypt the message AAABBB.

(f) Say we had an alphabet with 5 letters in it, $\{A, B, C, D, E\}$. So in this case $\mathbb{A} = \{0, 1, 2, 3, 4\}$, with addition and multiplication done modulo 5. Consider the following matrix with entries in $\mathbb{A}$.

$$E = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 4 & 2 \\ 1 & 1 & 3 \end{bmatrix}$$

i. What is the determinant of E?

ii. Could E be used as an encryption matrix for a Hill cipher?

iii. If so, find the decryption matrix D using both the reduction technique and the adjoint technique.

iv. If so, encrypt the message ABBEDA.

v. If so, decrypt the message CCDDEE.

30. Pair up with another student in your class. Both of you create your own plaintext message (in English) and an invertible matrix modulo 26. Make the message fairly long, at least 80 alphabetic characters. Keep the matrix you use for the cipher fairly small, say 3×3 to 5×5. Encrypt your message using your matrix. Give the other person the ciphertext of the message and a portion of the plaintext. The portion of plaintext you give must be at least 40 consecutive characters. The plaintext can start anywhere in the message but you need to tell the person where the plaintext starts. Break the code your partner gives you and decrypt their message.

References

[1] Lester S. Hill. Cryptography in an Algebraic Alphabet. *Amer. Math. Monthly*, 36(6):306–312, 1929.

[2] David Kahn. *The Codebreakers: The Story of Secret Writing*. MacMillan Company, New York, 1967.

[3] David Lay. *Linear Algebra and its Applications*. Pearson, Boston, 3^{rd} edition, 2006.

[4] Maxima. Maxima, a Computer Algebra System. `http://maxima.sourceforge.net/`, 2013. [Online; accessed January 2, 2014].

[5] Simon Singh. *The Code Book*. Anchor Books, New York, 1^{st} edition, 1999.

[6] Don Spickler. Linear ME. `http://facultyfp.salisbury.edu/despickler/personal/LinearME.asp`. [Online; accessed January 2, 2014].

[7] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory*. Prentice Hall, Upper Saddle River, NJ 07458, 2^{nd} edition, 2006.

[8] Wolfram. Mathematica. `http://www.wolfram.com/`, 2013. [Online; accessed January 2, 2014].

[9] Fred B. Wrixon. *Codes Ciphers & Other Cryptic & Clandestine Communication*. Tess Press, New York, 1998.