Spring 2016

Cryptanalysis of the Hill Cipher

Jessica Lehr
Elizabethtown College, lehrj@etown.edu

Cryptanalysis of the Hill Cipher

By: Jessica Lehr

This thesis is submitted in partial fulfillment of the requirements for Honors in the Discipline in Mathematics and the Elizabethtown College Honors Program.

May 4, 2016

Thesis Director: Dr. Tim McDevitt _____

Abstract

The Hill cipher is a classical block cipher based upon matrix multiplication. In 2007, Bauer and Millward completed a ciphertext-only attack in which they recovered the individual rows of the encrypting matrix to reduce the work previously necessary to recover the entire matrix at one time. In 2015, Leap et al. improved Bauer and Millward's attack by changing the scoring statistic to the Index of Coincidence, making it possible to score all members of entire classes of rows by testing a single member of each class and decreasing the necessary work by a factor of $\phi(L)$, where ϕ is the Euler totient function and L is the length of the alphabet. This paper presents further improvements by focusing attention on subsequences of the putative plaintext instead of the rows of the matrix, thereby making the search more efficient and more amenable to implementation on multiple computer processors.

Contents

1 Literature Review

The Hill Cipher, created by Lester Hill in 1929, is a cipher that has not been successfully decrypted in a ciphertext only attack [3]. Numerous attempts have been made over the years by various cryptanalysts, but to date no complete successes have been achieved for all block sizes of the key matrix when analyzing ciphertext only. The Hill cipher is entirely vulnerable if the plaintext is known. First efforts towards cryptanalysis were made by Jack Levine [8, 9, 10], a mathematics professor at North Carolina State University and cryptanalyst for the government during World War II. Later, Craig Bauer and Katherine Millward, a professor of mathematics at York College and his student, made further advancements in analyzing the Hill Cipher via examining its key row by row [1]. Two years following Bauer and Millward's advancements, Dae Hyun Yum and Pil Joong Lee, professors at Pohang University of Science and Technology furthered prior research through enhancing scoring analysis of potential solutions to the Hill Cipher [12] . Most recently, Tom Leap and Tim McDevitt, professors at Elizabethtown College along with two students, Kayla Novak and Nicolette Siermine, offered further improvements on the Bauer-Millward attack to allow analysis of larger matrices and the use of more efficient scoring [7].

1.1 Levine

Jack Levine was one of the first individuals to delve into cryptanalysis of the Hill Cipher. In his 1961 paper, *Elementary Cryptanalysis of Algebraic Cryptography* [9], he recognized two different potential situations posed by the Hill cipher. He started with having knowledge that the alphabet length is 26 and knowledge that the numbers corresponding to those given letters are as such: A=1, B=2...Y=25, Z=0. He also assumed knowledge of a portion of plaintext. Here, he does not know what portion of ciphertext with which the known plaintext corresponds or the key matrix which he calls matrix A. In order to solve the first problem of trying to determine the location of the known plaintext, Levine considered the system of congruences where

$$C_{i\beta} \equiv \alpha_{\beta_1} P_{i1} + ... + \alpha_{\beta_n} P_{in} \mod 2$$

all modulus two such that the plain and ciphertext sequences become binary sequences. This is enciphering the plain block $P_{i1}...P_{in}$ into cipher block $C_{i1}...C_{in}$ where n is the block size, P_{β_i} are the plaintext characters in block i and C_β are the ciphertext characters. Levine's example worked as such where he assumed an alphabet 'a' to 'z' corresponding to integers 1 through 26:

Plaintext: CRY PTO GRA PHY ...

Numerical Plaintext: 3 18 25 16 20 15 ...

Plaintext Mod 2: 1 0 1 0 0 1 ...

Ciphertext: SUI RIM AYG DIK ...

Numerical Ciphertext: 19 21 9 18 9 13 ...

Ciphertext Mod 2: 1 1 1 0 1 1

Using the single digit scheme such that

000 = 0, 001 = 1, 010 = 2, 011 = 3, 100 = 4, 101 = 5, 110 = 6, 111 = 7

Making:

Plaintext: 5 1 ...

Ciphertext: 7 3 ...

Modulo two, each block of n characters is converted to an integer in $\{0, 1, \ldots, 2^n - 1\}$ which is the binary value of the block. So in the above example we are working with blocks of 3, making the values of the block range from 0 to 7. This then creates a pattern of binary value pairings where each plaintext binary value will map to the same ciphertext binary value and vice versa for the duration of the text, i.e. for the above example, 5 will always map to 7 and 1 will always map to 3 and vice versa. This causes the same pattern to be present in both the plain and ciphertext such that if the plaintext follows the pattern 'abbd' the the ciphertext will follow the same pattern, but for the plaintext 'a' may correspond to 5, 'b' to 7 and 'd' to 3, but in the ciphertext this would translate to 'a' being 7, 'b' as 5 and 'd' as 1. This allowed Levine to obtain preliminary locations for the plaintext by comparing the patterns within the plain and ciphertext. The problem with this approach arose when there were longer messages, shorter probable texts or larger key matrices making it more challenging to pair the plain and cipher text to only one possible solution.

In response to the second situation that Levine posed, determining the key matrix A, he assumes that the location of the given plaintext has already been determined. He also uses knowledge that the determinant must be relatively prime to 26, the alphabet length, because otherwise the solution will not be unique and an inverse would not be possible to assist with decryption. Additionally, he assumes that $A^2 = I$, the identity matrix. He then set up the equation

$$P = A^{-1}C$$

where P is the plaintext matrix, A^{-1} is the inverse of the key matrix, which can be inverted to retrieve the original

key, and C is the ciphertext matrix. The congruences resulting from this relationship allows him to solve for the values of the key matrix [9].

In Levine's next paper, *Some Applications of High-Speed Computers to the Case $n = 2$ of Algebraic Cryptography* [8], he proposed the idea that high speed computers are capable of deciphering the Hill cipher exhaustively in the simple case of a 2×2 key matrix. For this situation, a known-plaintext attack, the ciphertext is known as is the plaintext and the size of the key matrix leaving only the elements of the decryption key matrix as unknown. Here, as the size of the key matrix increases, so does the complexity of cryptanalysis. His first attempt at cryptanalysis considered all constructions of the inverse key matrix of two forms, both of which satisfy the key matrix constraints:

$$\text{Type 1} = \begin{pmatrix} a & b \\ c & -a \end{pmatrix}$$

$$\text{Type 2} = \begin{pmatrix} a & b \\ c & a \end{pmatrix}$$

These two forms present 740 possible mod 26 matrices to be considered. A 1960's machine was able to run through all of these in approximately 30 minutes to identify which was the correct for decipherment.

The second method to determine A is dependent upon trigraphs present in the given plaintext. Levine assumes that a type one key matrix has been used and that the plaintext contains three consecutive letters with another character before or behind them to create a string of characters of length four. Based upon the assumption that it is a type one matrix, along with the formation of the string of four characters, multiple congruences can be formed between the plaintext and ciphertext characters using the key, or encryption marix. To solve these congruences, one can look at the congruences using the same key matrix elements (i.e. 'a' and 'b') and form a matrix using these congruences whose determinant is zero mod 26. This relationship is able to be constructed whether the string of plaintext characters is padded with an unknown character before or after the known triple of plaintext. A list of possible combinations for the elements of the key matrix are generated from matrices formed from the congruences and are dependent on whether the equations are solved mod 13 or mod 2. Levine also suggests compiling a list of the highest frequency English trigraphs to be tested in every ciphertext position for the consecutive characters where the correct solution will be easily visible from assumptions made prior to testing the trigraph. In general, Levine found that the longer the ciphertext, the fewer the number of trigraphs it was necessary to test.[8]

1.2 Bauer-Millward

After briefly reviewing the basics of matrix encryption necessary to understand the Hill cipher, Bauer and Millward review previous attacks on the cipher made by Jack Levine. Specifically, they reference Levine's use of involutory matrices for the key matrix so that the key matrix and its inverse are the same. In order to reinforce the need for

their research, they present the statistics for number of invertible matrices mod 26, the length of their alphabet, which increase exponentially as the size of their key matrix increases. See Table 1.

Of Levine's prior research, Bauer and Millward emphasized the case for which a 2×2 involutory matrix is used for encryption where there are only 740 matrices to check. They also analyzed his probable word attack in which the location of a probable word within the ciphertext is known and the researchers are attempting to recover the key matrix. Levine showed in his work that the location does not actually need to be known, but can be easily recovered using knowledge that the product of two integers is odd if and only if both integers are odd and the sum of two integers is odd if only one of the integers is odd. Levine's presentation of use of a specific key matrix of the form

$$\begin{pmatrix} \text{even} & \text{odd} \\ \text{odd} & \text{even} \end{pmatrix}$$

made it easier to analyze four different enciphering possibilities which were dependent upon the numerical components of the letters being encoded. For example, the enciphering matrix would send text of form

$$\begin{pmatrix} \text{even} \\ \text{even} \end{pmatrix}$$

to another pair of letters having the same form where as a set of text of form

$$\begin{pmatrix} \text{even} \\ \text{odd} \end{pmatrix}$$

would be sent to a set of text in the form

$$\begin{pmatrix} \text{odd} \\ \text{even} \end{pmatrix}$$

and vice versa for the opposite situation. For example, enciphering 'MA' corresponding to numerical equivalents (12 0) is considered the even-even case and would therefore result in an even-even form for ciphertext. On the other hand, 'AT' corresponds to (0 19) which is an even-odd situation and will therefore result in ciphertext of the form odd-even. This creates an easy to detect pattern if we name the even-even form 0, even-odd form 1, odd-even form 2 and odd-odd form 3. Then, form 0 always goes to 0, form 3 always goes to form 3 and forms 1 and 2 go to one another. The encipherment constraints and this labeling make it easier to examine the ciphertext to see where the same pattern is visible as that in plaintext. This helps to determine the corresponding set of ciphertext for the plaintext allowing a quicker recovery of the encryption matrix whose correctness can be verified by application to the ciphertext.

Bauer and Millward expand upon the research done by Levine by proposing a new attack on the Hill Cipher in which the inverse of the key matrix is recovered one row at a time. Their method includes making guesses for the first row of the key matrix, which in the 2×2 case results in only the odd or only the even plaintext

elements. While this does not tell them outright whether their guess is correct, they can statistically analyze their results to see whether their guess is reasonable. This is done by comparing frequencies of the recovered putative plaintext letters with the expected frequencies of the letters in the English language. The row that results in the frequencies most similar to that of the English language generally provides the correct values for the first row. The same procedure is completed for the second row to obtain those matrix values as well. Upon obtaining the two combinations for the key matrix, it is necessary to make sure they are not multiples of one another, otherwise the matrix determinant will not be relatively prime to the modulus of 26. Occasionally, they found that the top combination for either trial was not the correct one and more than just the two top rows had to be considered to obtain the correct matrix. The other problem that Bauer and Millward considered was how to score the putative plaintext resulting from these rows to more efficiently obtain the true key matrix values. Their first attempt was to give points equal to the sum of the frequencies of the putative plaintext characters as they corresponded to English characters such that more frequently occurring letters in English that showed up in the putative plaintext would receive more points. For example, a row that contained 'x' and 'v' would receive significantly fewer points than one containing 'e' and 't'. This was not as successful as desired so they came up with an alternative scoring method where points were awarded, or subtracted, depending on the letters present after the application of the inverse key matrix to the ciphertext.

The first example considered was the 2×2 case in which all of the highest scoring rows were often deemed impossible because they were not relatively prime to the modulus of 26. They found that for 64% of the trials, the two highest scoring rows constructed the correct key matrix. In order to obtain the correct matrix 100% of the time, it was necessary to consider the top seven rows in which there were 210 possible ordered pairings of the seven rows. This is a significant reduction of necessary work by humans or computers in comparison to checking all 157,248 invertible 2×2 matrices.

When considering the 3×3 case for the key matrix, it was found that the correct three rows of the key matrix were obtained over 50% of the time if the top 17 were considered. To have all three rows 100% of the time, it was necessary to consider the top 394 potential rows which results in 60,698,064 possible key matrices. This seemed large to Bauer and Millward, but they point out that this is actually only .0037% of the total invertible 3×3 matrices. The same analysis was performed on the 4×4 key matrix and again yielded similar results of a large reduction in necessary work to obtain the correct key matrix in comparison to the brute force method. Bauer and Millward concluded that their new attack was successful, but still not optimal, believing that a stronger scoring mechanism could yield better results. More reliable results could also be obtained, they thought, by using larger samples of ciphertext. Finally, they believed that if the numeric equivalents of the letters of the text were not

known, the attack would fail. [1]

1.3 Yum-Lee

Yum and Lee's research [12] builds upon the ciphertext-only attack produced by Bauer and Millward. They aim to present a more reliable scoring system for cryptanalysis of the cipher based upon goodness-of-fit statistics while also showing how to apply attacks to the Hill cipher without knowledge of the numeric equivalents of the pre-declared alphabet, a concern of Bauer and Millward.

The two researchers start out their paper by presenting background information similar to Bauer and Millward's; first on the basics of the Hill Cipher in addition to touching upon research done by both Levine and Bauer and Millward. For their use, Yum and Lee fix the encoding scheme with A=0 through Z=25 and the length of the ciphertext to be 100. From there, they work with the goodness-of-fit statistic to see how well the model has been described by each potential key matrix using the observed and expected values. This statistic is commonly used in hypothesis testing where the null hypothesis generally assumes the two, observed and expected values, are the same. Yum and Lee chose to use the χ^2 statistic to analyze their null hypothesis that the character frequencies of the putative plaintext are the same as the frequencies of letters in the English language. They then ranked potential inverse key matrix rows based upon the resulting χ^2 score for each row where the better rows had lower χ^2 scores implying smaller differences between observed and expected values.

The results of Yum and Lee's research were not as ideal as they had hoped due to the new scoring failing to significantly excel beyond the Bauer-Millward scoring system. This was attributed to the χ^2 distribution approximation failing to hold if the expected frequencies are too low which is the case with various less common English letters such as 'x' and 'q'. To attempt improvement, Yum and Lee found the exact probability of the recovered plaintext equivalents. Using this, they constructed a new score by dividing the product of the expected probabilities of the letters by the product of the observed probabilities of the letters. Larger values of this score denoted better potential rows for the decryption matrix. Yum and Lee discovered this method, which they called the simplified multinomial statistic method, to be much more effective than both the Bauer Millward and χ^2 methods.

Yum and Lee's other research examined the possibility of an unknown encoding scheme for assigning numbers to the letters for the predetermined alphabet. For an alphabet of length 26, there are 26! possible encoding schemes where with an unknown encoding scheme, it is not possible to count the observed frequencies of a specific character. A solution would be to observe the shape of the distribution of tallied plaintext numerical equivalents. Despite not knowing the numeric counterparts of the alphabet, the characters can still be sorted in descending order by frequency and a new score can be computed. The new score is the product of the probabilities of the

i-th most frequent English letters raised to the numerical equivalent of the i-th most frequent character in the recovered plaintext and then divided by the product of the factorial of the numerical equivalent of the i-th most frequent character in the recovered plaintext as seen below.

$$h(S;q,m) = \frac{\prod_{i=0}^{25} q_i^{S_i}}{\prod_{i=0}^{25} S_i!}$$

Here S_i is the numerical equivalent of the i-th most frequently occurring character in the recovered plaintext, while q_i is the probability of the i-th most frequent English letter. A higher score denoted better rows for the decryption matrix. Yum and Lee found this method to be very powerful for large messages.[12]

1.4 Leap-McDevitt-Novak-Siermine

The Elizabethtown College team begins their paper [7] in a similar fashion to other researchers with a brief introduction and small recap on the research of prior cryptanalysts. This team modified the Bauer-Millward attack to reduce computational complexity such that the scoring method they use decreases the complexity by approximately $\phi(L)$ where ϕ is the Euler totient function and L is the length of their alphabet. Further scoring is done using goodness-of-fit statistics to find the best rows for the inverse of the key and to discover the unique inverse of the key matrix using potential plaintext digraphs. The team also worked to ultimately produce the entire decryption matrix instead of simply the rows. In previous papers, it is not discussed how to construct the inverse of the key matrix from the recovered rows, nor are matrices larger than 4×4 examined, most likely due to computational restrictions. The Elizabethtown team presents a method to build the key's inverse requiring b^2 instead of $b!$ ways to arrange the rows. This is accomplished through scoring English digraphs produced when selected rows are paired.

The team presents the example of a 3×3 key matrix inverse with only the first row of the matrix guessed and then applied to the ciphertext. They then compute the index of coincidence of the resulting text, the likelihood of picking two of the same characters in a set of text. This index of coincidence, they prove, is the index of coincidence for 18 different vectors. This is because each vector is of the form m times the elements of a "base" vector modulo the length of the alphabet where m is a number relatively prime to the length of the alphabet. Here, it is 18 different vectors because the alphabet length has been set to 27 and there are 18 numbers relatively prime to 27 mod 27. Elizabethtown also points out that despite having 27^3 potential vectors of length three, they can omit 9^3 of these vectors as there are nine different rows of which all are multiples of three and can each be arranged three ways and therefore fail to yield invertible keys. This omission leaves $27^3 - 9^3$ potential rows which can then be divided by 18. In turn, this leaves only 1053 rows to consider compared to the original 19,683 rows. Each row is given a χ^2 score based upon its resulting text where low scores denote closer to English text. Finally, to recover

the actual matrix, the team plugs in combinations of two of the best rows and recovers putative plaintext that is missing every third letter. This putative plaintext is scored with a χ^2 goodness-of-fit test to determine which pairings are the best. When the best pair is recovered, it is easy to figure out the final row of the inverse key matrix based upon the guessed plaintext and given ciphertext.

In the general case, Elizabethtown presents an analysis of various scoring methods. They find that the power, the likelihood of rejecting a false null hypothesis, of the Bauer-Millward scoring method is practically the same as the power of the multinomial method utilized by Yum and Lee. They determined that the index of coincidence is most powerful for text greater than 50 characters in length and the χ^2 statistic is most powerful for text over 65 characters in length. Many of the potential problems result from shorter texts due to lack of information or larger block sizes for the key matrix because they are not easily computed with computers.

The Elizabethtown team also discusses scoring in a more general nature for when the length of the alphabet is different that the length they set at the beginning of their paper. They consider the situation where the length is simply prime, is a power of a prime or is a product of distinct primes. Regardless of length of the alphabet, the Elizabethtown team found that they can reduce the work done by Bauer and Millward by a factor of ϕ.

1.5 Literature Review Thoughts

After examining prior cryptanalysis research for the Hill Cipher, there is still area for improvement as declared by each researcher at the conclusion of their papers. Most need for improvement lies in developing ways to further reduce complexity of the Bauer-Millward scoring method that Yum and Lee and the Elizabethtown team both tackled, or in developing a new scoring method in order to recover the inverse of the key matrix to be used for decryption. New methods to analyze the Hill Matrix aside from the row by row method proposed by Bauer and Millward should also be investigated. Additionally, a method for large block sizes is necessary as they are typically not conducive to most methods that have been suggested so far.

2 Cryptography

2.1 History

Cryptography is an ancient art dating back to before 1900 B.C. The word is derived from the Greek words "kryptos" and "graphein" literally translating to hidden writing. In ancient times, the goal of cryptography was to convert a message into unreadable text or figures in order to protect the contents as it was carried from one city to another between two higher officials by a messenger [4].

The concept of cryptography is said to have originated with Egyptian scribes who used hieroglyphs to hide the meanings of their messages. Not long after, the Greeks, wrapped their secret messages on tape that was wound

around a stick with which the correct circumference stick was then necessary for decryption. This is often referred to as the Spartan Scytale technique. Elsewhere in Europe, the Romans used the monoalphabetic Caesar Cipher shift in which each letter was shifted the same number of characters within the cipher. For a three letter shift Caesar cipher, an 'a' would become 'd', 'b' would become 'e' and so on [4]. The size of the shift is referred to as the *key* or the information necessary to solve the cipher.

The Middle Ages presented the next set of advancements in evolving the monoalphabetic cipher into a polyalphabetic cipher. Polyalphabetic ciphers use multiple characters to generate one ciphertext character. The most common 21st century polyalphabetic substitution cipher is the Vigenere cipher which utilizes the Vigenere Tableau to complete character substitution [4]. The Vigenere tableau is unique because, depending on how the tableau is set up, each row or column corresponds to a Caesar Cipher with shifts from zero to one less than length of the alphabet. Using the tableau, the ciphertext character is given by the intersection of the row given by the keyword letter and the column given by the plain text character [11].

2.2 Breaking Ancient Ciphers

Throughout history, ciphers have become increasingly more challenging to crack. The Ancient Greek Spartan Scytale technique simply required obtaining a stick with the correct circumference. To crack this, one could try varying sized sticks until they found a stick that produced intelligible text. The Caesar Cipher is similarly uncomplicated to crack using either the brute force or the frequency analysis method. The brute force method attempts every possible shift within the alphabet length. If the alphabet length is 26, like the English alphabet, then there are 25 different shifts that can be attempted since the shift of zero or 26, which is zero modulo 26, should not be attempted because that will result in the cipher text. For frequency analysis, it is necessary to know what language the plaintext will be in. From there, the cryptanalyst can analyze the frequency of each character in the cipher and attempt to match the more frequently occurring characters with the high frequency characters in the language of the text. For instance, 'e' is the most frequent letter in English and therefore could be substituted for the highest frequency cipher text character [4].

Cracking the Vigenere cipher is more challenging than cracking prior ciphers due to each plaintext character being encrypted as multiple different ciphertext characters. One of the major benefits of a polyalphabetic cipher is the plaintext to ciphertext relation depends upon which character from the keyword the plaintext was paired with. This thwarts the use of frequency analysis to crack the cipher and therefore increases the difficulty of obtaining the original message.

All of the ciphers discussed so far have been stream ciphers where plaintext is encrypted and ciphertext is decrypted one character at a time. These forms of ciphers are much less complicated to cryptanalyze than their

counterparts, block ciphers. Block ciphers work on encryption and decryption over multiple characters at a time. This added complexity of characters being dependent upon more than one other character increases the security of the cipher and also increases the difficulty of cryptanalysis.

3 The Hill Cipher

3.1 Introduction

The Hill Cipher, a modular arithmetic, matrix based block cipher, is a final classical cipher and the main focus of this paper. This cipher "[made] polygraphic cryptography practical for the first time" according to the author of *The Codebreakers*, David Kahn. Lester Hill, creator of the Hill cipher, was a late 19th - early 20th century mathematician. Hill spent the majority of his working life as a college professor at four different universities during which time he also earned his PhD in mathematics, only taking a short break to fight in World War I. Throughout his scholastic career, Hill authored numerous papers and directed abundant research in areas of error detection and cryptography [3]

Hill's first paper on the Hill Cipher appeared in the June-July 1929 issue of *The American Mathematical Monthly* under the title of "Cryptography in an Algebraic Alphabet". Here, the Hill cipher was described as matrix encryption in which a series of plaintext n characters long was converted into n characters of ciphertext. To change the plaintext to ciphertext, the plaintext must be converted into its numerical equivalent. Prior to converting to numerical equivalents, it is necessary to 'clean' the text such that it only consists of characters within the given, predetermined alphabet. If the alphabet consists only of lowercase letters, than all letters within the text are converted to lowercase and all other characters are omitted. In the most common situation, the letter to number substitutions are such that 'a' is zero, 'b' is one, through 'z' is 25. Hill's original encoding scheme for the Hill cipher is not the most common situation. Here, his letter to number conversion was not linear meaning that 'b' was not necessarily mapped to the number that immediately follows the number to which 'a' was mapped. Numerous cryptanalysts praised this encryption method for the added security that resulted from the nonlinear substitution [6]. Occasionally, more characters such as space, punctuation or special characters from other languages are appended to the end of the alphabet causing the length of the "alphabet" to be longer than the traditional 26. For the remainder of this paper, we will assume an alphabet of length 29 consisting of all lowercase characters 'a' through 'z', space, period and comma. The length of the alphabet is a crucial piece of information to anyone trying to encrypt or decrypt the series of text. The modular arithmetic of the Hill matrix is dependent upon the length as it is the modulus used to reduce the numbers obtained from matrix multiplication so that text can be retrieved from their numerical counterparts. Example 3.1 shows the basic process for generating the

numerical representation of a set of plaintext.

Example 3.1.

Alphabet = abcdefghijklmnopqrstuvwxyz .,

Plaintext: OnCe UPON a time, in a far off land, THERE liveD a mathematician!

Cleaned Text: once upon a time, in a far off land, there lived a mathematician

Numerical Representation: 14 13 2 4 26 20 15 14 13 26 0 26 19 8 12 4 28 26 8 13 26 0 26 5 0 17 ...

3.2 How it Works

Upon obtaining the numerical representation of cleaned text, the next step is to arrange it properly for encoding. To determine the proper array for the characters, it is necessary to decide upon a 'key size'. The key, the matrix used for encryption via matrix multiplication, is an $n \times n$ matrix. The numerical plaintext then needs to be arranged into an $n \times m$ matrix since the key premultiplies the plaintext matrix. The plaintext characters are arranged in their matrix in a vertical manner where the first n characters are arranged in the first column and then the text is continued at the top of the second column [12]. In the situation that there is not enough plaintext to fill the entire $n \times m$ matrix such that the last column is only partially filled, one may pad the text with artificial letters, commonly 'x', 'q' or 'z' as they are easily distinguished as padding upon decoding due to being less frequent occurring letters in English.

The key itself has many special rules. Most importantly, the key is required to be invertible otherwise decryption is not possible. To be invertible, the determinant of the matrix must be nonzero and relatively prime to the length of the alphabet or modulus. If the alphabet is 26 characters long, including only lowercase letters 'a' through 'z', then the determinant of the key matrix can not be divisible by 2 or 13 as they are both multiples of 26 and therefore are not relatively prime to the modulus. We chose an alphabet of length 29 since it is prime and therefore we do not encounter this issue. The determinant of the key matrix must follow these rules otherwise the key used for decryption would be unobtainable as it is the inverse of the original key matrix [12].

For actual encryption to occur, the $n \times n$ key matrix pre-multiplies the $n \times m$ matrix of plaintext. This matrix multiplication returns a $n \times m$ matrix of ciphertext. The ciphertext matrix must then be reduced modulo the length of the alphabet to easily convert the numerical representation of the letters back into their alphabetic counterpart. If n is two, the key is constructed as an invertible 2×2 matrix. Multiplication of the first column (two characters) of plaintext by the key returns the first two characters of ciphertext. In generating these ciphertext characters, they are both dependent on both of the first two plaintext characters. To obtain the first ciphertext character, the first row of the key is multiplied by the first column of plaintext, or the first two characters of

plaintext. To obtain the second ciphertext character, the second row of the key is multiplied by the first column of plaintext. The following gives the steps for a full 2×2 key matrix encryption example from start to finish. The plaintext I LOVE MATH. is converted into the numerical sequence (8 26 11 14 21 4 26 12 0 19 7 27)

The key used for this 2×2 matrix encryption is given by

$$\text{Key} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \end{pmatrix}.$$

To complete encryption, the numerical plaintext is arranged into a 6×2 matrix in order to be compatible with the 2×2 key and then matrix multiplication occurs to result in the ciphertext. The resulting numerical ciphertext is then reduced modulo 29, the length of the alphabet.

$$\begin{aligned} \text{Ciphertext} &= \begin{pmatrix} 4 & 3 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} 8 & 11 & 21 & 26 & 0 & 7 \\ 26 & 14 & 4 & 12 & 19 & 27 \end{pmatrix} \\ &= \begin{pmatrix} 110 & 86 & 96 & 140 & 57 & 109 \\ 76 & 61 & 71 & 102 & 38 & 75 \end{pmatrix} \\ &\equiv \begin{pmatrix} 23 & 28 & 9 & 24 & 28 & 22 \\ 18 & 3 & 13 & 15 & 9 & 17 \end{pmatrix} \mod 29, \end{aligned}$$

resulting in the numerical ciphertext Sequence: (23 18 28 3 9 13 24 15 28 9 18 24), which corresponds to XS.DJNYPXJSY as the resulting ciphertext.

Note, each character of ciphertext is dependent on numerous factors including multiple key and plaintext characters. This causes the difficulty of cryptanalysis of the Hill cipher. The Hill Cipher enables varying encryption of the same plaintext character due to different pairings of plaintext characters and key matrix characters. The original plaintext has two spaces, denoted as 26 in the numerical plaintext, but when examining the resulting ciphertext, you can see that the first space was encoded as an 'S' while the second space was encoded as a 'Y'. This displays how the Hill Cipher masks letter frequencies which is information commonly used for decryption.

3.3 Basic Idea of Decryption

Decrypting the Hill matrix is not complex if one has access to the original key matrix and the set of ciphertext. First, the decrypter must compute the inverse of the key matrix. Using the previous example, the inverse of the key matrix is

$$\text{Inverse Key} = \begin{pmatrix} 4 & 3 \\ 3 & 2 \end{pmatrix}^{-1} = \begin{pmatrix} -2 & 3 \\ 3 & -4 \end{pmatrix}$$

Upon obtaining the inverse of the key matrix, the decrypter pre-multiplies the $n \times m$ matrix of ciphertext by the inverted key matrix. The resulting $n \times m$ matrix of plaintext must then be reduced modulo the length of

Cipher	Keyspace
Caesar	25
2×2 Hill	$682,080 \approx 2^{19.4}$
3×3 Hill	$13,989,670,880,640 \approx 2^{43.7}$
4×4 Hill	$241,319,751,100,575,994,828,800 \approx 2^{77.7}$
5×5 Hill	$3,500,860,685,395,554,849,102,157,075,077,734,400 \approx 2^{121.4}$
6×6 Hill	$42,712,284,908,737,393,674,385,678,513,029,803,119,097,861,332,992,000 \approx 2^{174.8}$

Table 1: Table indicating the size of the keyspace for different ciphers.

the alphabet to obtain the characters' numerical representation which can easily be converted back into their alphabetic counterparts.

$$\begin{aligned}\text{Plaintext} &= \begin{pmatrix}-2 & 3\\ 3 & -4\end{pmatrix}\begin{pmatrix}23 & 28 & 9 & 24 & 28 & 18\\ 18 & 3 & 13 & 15 & 9 & 24\end{pmatrix}\\ &= \begin{pmatrix}8 & -47 & 21 & -3 & -29 & 7\\ -3 & 72 & -25 & 12 & 48 & -2\end{pmatrix}\\ &\equiv \begin{pmatrix}8 & 11 & 21 & 26 & 0 & 7\\ 26 & 14 & 4 & 12 & 19 & 27\end{pmatrix} \mod 29,\end{aligned}$$

which results in our original plaintext from before, "I LOVE MATH.".

The difficulty in Hill Cipher decryption arises when knowledge of the original key matrix is unavailable. Then, decryption is dependent only upon knowledge of the ciphertext and, typically, the language of the original plaintext. Much of the complexity of cracking the Hill Cipher lies in the number of possible keys compared to previous ciphers. In the Caesar cipher there were only 25 different keys to try before the brute force method was exhausted. In the situation of the Hill cipher, the simplest enciphering occurs with a 2×2 key matrix. For this 2×2 matrix, there are 682,080 possible invertible matrices meaning that to exhaust the brute force method meaning $682,080 \approx 2^{19.4}$ possible matrices would have to be tried compared to the 25 previously. These numbers grow exponentially as the key size increases. For example, a 3×3 matrix encryption keyspace is $13,989,670,880,640 \approx 2^{43.7}$ and a 4×4 keyspace is $241,319,751,100,575,994,828,800 \approx 2^{77.7}$ [1]. Table 1 shows a comparison of various ciphers and their corresponding keyspaces. It is important to note that Table 1 simply displays the number of *invertible* matrices, not the total number of $n \times n$ matrices. The process of recovering only the invertible matrices is yet another challenge to which there also is not a simple and efficient method. To solve a general case of the Hill cipher with key size $n \times n$, a method more efficient than brute force is necessary. The remainder of this paper analyzes possible methods of cracking the Hill Cipher having strictly knowledge of the ciphertext.

4 Hill Cipher Decryption

4.1 Bauer-Millward

Progress towards complete cryptanalysis of the Hill Cipher in the twenty-first century has stemmed from the 2007 efforts made by Craig Bauer and Katherine Millward of York College, who recovered individual rows of the encryption matrix, in contrast to previous attempts to recover the entire encryption matrix at one time. Bauer and Millward worked on a ciphertext-only attack in which the plaintext is unknown, but they assumed they had knowledge of the size of the key matrix. In the 2×2 case, having knowledge of one row of the decryption matrix supplies them with the plaintext equivalents for all of the odd or even positioned characters. While these characters can not be visually examined to see if they are correct, they can be statistically compared to the typical occurrence of characters in English via frequency comparison. The same procedure is done for the second row, but the highest scoring row will appear as the top choice for the second row as well. Bauer and Millward consider the top choices for the rows such that the two rows are distinct in order for the key matrix to be invertible. Various combinations of the top choices are tried in different orders until when the key matrix is applied to the ciphertext, readable plaintext is obtained. The main issue that Bauer and Millward examined was how to score and rank these rows such that every possible row did not have to be considered.[1]

Bauer and Millward originally considered the 2×2 case in which there are various rows that are impossible to be used within the key matrix due to rendering it not invertible. These rows for an alphabet of length 26 include (0,13), (13,0), (13,13) and anything of the form (even,even) because then the determinant of the matrix would not be relatively prime to the modulus of the alphabet length. Each row was given a score based upon how the resulting frequencies of its putative plaintext compared to English character frequencies and then the rows were ranked based upon this score. Commonly, only a small percentage of rows, relatively speaking to the entire key space, had to be considered to obtain the correct rows for decryption. While Bauer and Millward were successful with their approach, they recognized that their scoring method was most likely not the optimal method.

4.1.1 2×2 Bauer Millward Case using Gettysburg Address

Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ .,

Ciphertext: “EOPHUAUSHGK DD... A,WFKNMR.”

Numerical Ciphertext: 4 14 15 7... 12 17

Matrix form:

$$\begin{pmatrix} 4 & 15 & ... & 12 \\ 14 & 7 & ... & 17 \end{pmatrix}$$

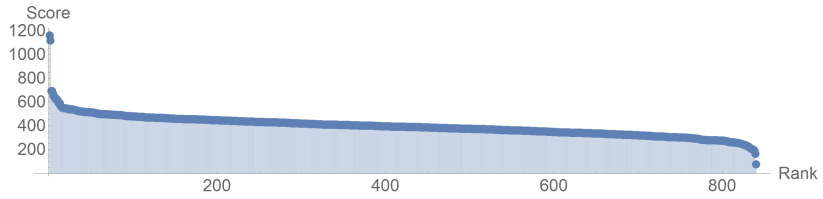


Figure 1: Bauer-Millward scores for the $29^2 = 841$ possible encryption matrix rows for the ciphertext "EOPHUAUSHGK DD... A,WFKNMR." The two highest scores are 1159.0 and 1118.5, which correspond to $(27, 3)$ and $(3, 25)$ respectively.

Potential Rows:

$$\begin{pmatrix}0 & 0\end{pmatrix}, \begin{pmatrix}0 & 1\end{pmatrix}, ..., \begin{pmatrix}28 & 28\end{pmatrix}$$

Putative Plaintext:

$$\begin{pmatrix}28 & 28\end{pmatrix}\begin{pmatrix}4 & 15 & ... & 12\\ 14 & 7 & ... & 17\end{pmatrix} \equiv \begin{pmatrix}504 & 616 & ... & 812\end{pmatrix} \equiv \begin{pmatrix}11 & 7 & ... & 0\end{pmatrix} \mod 29$$

Scored Using Point system:

(Space): 3 Points

AETNO: 2 Points

HILRS.: 1 Point

FGP,: 1/2 Point

MBCDUVWY: 0 Points

QJKXZ: -1 Points

The highest scoring rows are $\begin{pmatrix}27 & 3\end{pmatrix}$ with a score of 1159.0, $\begin{pmatrix}3 & 25\end{pmatrix}$ with a score of 1118.5, and $\begin{pmatrix}15 & 9\end{pmatrix}$ with a score of 693.5. Note the significant drop off in point values for pairs $\begin{pmatrix}27 & 3\end{pmatrix}$ and $\begin{pmatrix}3 & 25\end{pmatrix}$ versus $\begin{pmatrix}15 & 9\end{pmatrix}$, which is more clearly evident in Figure 1. Note in Figure 1 that there are two high scoring pairs on the vertical axis by 1200 that distinguish themselves from the rest corresponding to rows $\begin{pmatrix}27 & 3\end{pmatrix}$ and $\begin{pmatrix}3 & 25\end{pmatrix}$.

Now that we are confident that the rows of the decryption matrix are $\begin{pmatrix}27 & 3\end{pmatrix}$ and $\begin{pmatrix}3 & 25\end{pmatrix}$, there are two ways to order them:

$$\begin{pmatrix}27 & 3\\ 3 & 25\end{pmatrix} \quad \text{or} \quad \begin{pmatrix}3 & 25\\ 27 & 3\end{pmatrix}.$$

The first results in the putative plaintext

$$\begin{pmatrix}27 & 3\\ 3 & 25\end{pmatrix}\begin{pmatrix}4 & 15 & ... & 12\\ 14 & 7 & ... & 17\end{pmatrix} \equiv \begin{pmatrix}5 & 20 & ... & 27\\ 14 & 17 & ... & 26\end{pmatrix} \mod 29$$

that corresponds to FOURSCORE AND SEVEN YEARS AGO... The second gives

$$\begin{pmatrix} 3 & 25 \\ 27 & 3 \end{pmatrix} \begin{pmatrix} 4 & 15 & ... & 12 \\ 14 & 7 & ... & 17 \end{pmatrix} \equiv \begin{pmatrix} 14 & 17 & ... & 26 \\ 5 & 20 & ... & 27 \end{pmatrix} \mod 29,$$

which corresponds to OFRUCSRO ENA DESEV.... The first choice for the decryption key matrix of

$$\begin{pmatrix} 27 & 3 \\ 3 & 25 \end{pmatrix}$$

is clearly the correct choice.

It was easy to determine the correct ordering of the rows in this example because there are only two of them. However, in the general case for an $n \times n$ matrix, there are $n!$ ways to arrange the rows making it much more complex to correctly order the potential rows of the decryption matrix.

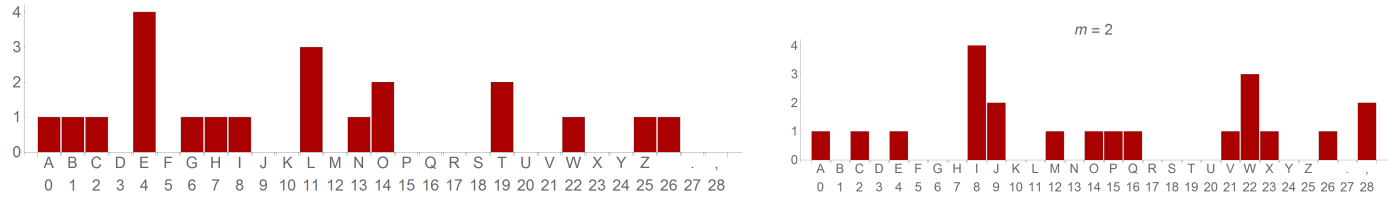
4.2 Elizabethtown Team

The Elizabethtown College team consisting of professors Dr. Tom Leap and Dr. Tim McDevitt along with students Kayla Novak and Nicolette Siermine modified the attack made by Bauer and Millward to reduce computational complexity and also to produce the entire decryption matrix instead of just the rows. The alternative scoring method of use of the Index of Coincidence proposed by the Elizabethtown team reduced overall complexity by a factor of $\phi(L)$ where L is the length of the alphabet and ϕ is the Euler totient function. This reduction occurs because when testing a row of (1 2 3) for example, one is actually checking all possibilities of the form m(1 2 3) modulo L where m is relatively prime to L. In the case of the alphabet of length 27 considered by the Elizabethtown team, computational complexity is reduced by a factor of $\phi(27) = 18$. This is because there are 18 rows with the same index of coincidence making it possible to test only one of those, the base row, since we only need to check each index of coincidence (IoC) once. The index of coincidence measures the likelihood of two randomly selected characters in a set of text being the same and is given by

$$\text{IoC} = \frac{1}{N(N-1))} \sum_{i=1}^{n} (F_i(F_i - 1))$$

where N is the length of the text, n is the length of the alphabet, and F_i is the number of times the i^{th} character appears in the text. The IoC of English text is typically around 0.072 for our 29-character alphabet, whereas the IoC of Hill ciphertext is typically close to 0.035, which is close to completely flat text where the IoC is $1/29 \approx 0.0344$.

The IoC is invariant with respect to multiples of the same text. For example, the character frequencies in



(a) (b)

Figure 2: Character frequencies in the text (a) "ELIZABETHTOWN COLLEGE" and in (b) two times that text.

"ELIZABETHTOWN COLLEGE" are shown in Figure 2(a) and

$$\begin{aligned}\text{IoC} &= P(\text{Two randomly selected characters are identical}) \\ &= P(\text{E and E}) + P(\text{L and L}) + P(\text{O and O}) + P(\text{T and T}) \\ &= \frac{4(3) + 3(2) + 2(1) + 2(1)}{21(20)} \\ &\approx 0.0524.\end{aligned}$$

Multiplying the text by any integer relatively prime to 29 simply permutes the frequencies while leaving the IoC unchanged. For example, Figure 2(b) shows "ELIZABETHTOWN COLLEGE" multiplied by 2 and

$$\begin{aligned}\text{IoC} &= P(\text{I and I}) + P(\text{J and J}) + P(\text{W and W}) + P(\text{, and ,}) \\ &= \frac{4(3) + 2(1) + 3(2) + 2(1)}{21(20)} \\ &\approx 0.0524.\end{aligned}$$

This invariance allows us to use the IoC to examine a partial string of putative plaintext resulting from a base class where the characters are not necessarily the ones present in the plaintext, but rather a multiple of them, in order to see if it is English. This differs from using something such as the χ^2 score to analyze the text because in that situation, each character is compared to itself such that the frequency of 'A' in the putative plaintext is compared to the frequency of 'A' in English text.

Once the highest IoC scoring base rows are obtained, multiples of the form $m \times \textit{baserow}$ are checked to obtain the best multiples of each base row. The top IoC row multiples are ranked using goodness-of-fit scoring, typically χ^2. The χ^2 score compares observed data, character frequencies from the putative plaintext, with expected frequencies of characters taken from the Brown Corpus and is calculated for an alphabet of length 29 using

$$\chi^2 = \sum_{i=0}^{29} \frac{(\text{observed} - \text{expected})^2}{\text{expected}}.$$

Once the best n rows are identified, to correctly order the prospective rows, digraph combinations are considered. A digraph is a pair of sequential characters in a body of text. The putative plaintext digraphs are assembled

by trying pairs of two sequential decryption matrix rows at one time resulting in text that has pairs of putative plaintext characters surrounded by ciphertext. χ^2 scores are then computed for these digraph combinations by comparing to English digraph frequencies taken from the Brown Corpus. The lowest χ^2 scoring pairs denote most likely English combinations and tell us how to order the rows of the decryption matrix. It is also possible to deduce a missing row from the decryption matrix in the situation that all but one are recovered from the index of coincidence and goodness-of-fit scorings. The putative plaintext is assembled using the decryption matrix formed from the ordering found in the digraph putative plaintext scoring. In the correct ordering, it will be visible where missing characters are located and will tell us where the other row should be placed. Then the putative plaintext will allow the cryptanalyst to discern the necessary letters to complete the text which will in turn allow them to figure out the solution of the missing row using the characters and the ciphertext.[7]

4.2.1 2×2 Case using Gettysburg Address

Let's examine an example.

Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ .,

Ciphertext: EOPHUAUSHGK DD ... AA ,WFKNMR

Numerical Ciphertext: 4 14 15 7 ... 10 13 12 17

Applying each base row of the form $\begin{pmatrix}0 & 26\end{pmatrix}$ and $\begin{pmatrix}26 & p\end{pmatrix}$, $p = 0, 1, \ldots, 28$, to the ciphertext and scoring the resulting putative plaintext yields the scores displayed in Figure 3. Recall that we don't need to calculate the IoC of every possible row because, as stated above, each base row has the same IoC as $\phi(L) = 28$ other rows.

The two best IoCs in this example correspond to $\begin{pmatrix}26 & 19\end{pmatrix}$ and $\begin{pmatrix}26 & 4\end{pmatrix}$, so we next calculate putative plaintext subsequences using $m\begin{pmatrix}26 & 19\end{pmatrix}$ and $m\begin{pmatrix}26 & 4\end{pmatrix}$, $m = 1, 2, \ldots, 28$, and score them using the χ^2 statistic. See Figure 4. The multiple $20\begin{pmatrix}26 & 19\end{pmatrix} = \begin{pmatrix}27 & 3\end{pmatrix}$ clearly has a significantly lower $\ln \chi^2$ score than the others making it a likely candidate row of the decryption matrix. The natural log was taken of each score in order to make the graph of all scores easier to examine. Similarly, the multiple $28\begin{pmatrix}26 & 4\end{pmatrix} = \begin{pmatrix}3 & 25\end{pmatrix}$ clearly has a significantly lower $\ln \chi^2$ score than its competitors, making it a the probable other row of the decryption matrix. This suggests two likely decryption matrices, $\begin{pmatrix}27 & 3\\ 3 & 25\end{pmatrix}$ or $\begin{pmatrix}3 & 25\\ 27 & 3\end{pmatrix}$. The first matrix gives the putative plaintext "OFRUCSRO ENA..." and the second gives "FOURSCORE AND SEVEN...", so $\begin{pmatrix}3 & 25\\ 27 & 3\end{pmatrix}$ is the correct decryption matrix.

For the $n \times n$ case, we would have n candidate rows to order to construct the decryption matrix, where each candidate row is the highest scoring χ^2 multiple of the n highest IoC base rows. Normally, there would be $n!$ ways to arrange these n possible rows unless a more insightful method is utilized, which is what the Elizabethtown team did. They plug two potential decryption rows into the key inverse sequentially at a time, and use this to develop

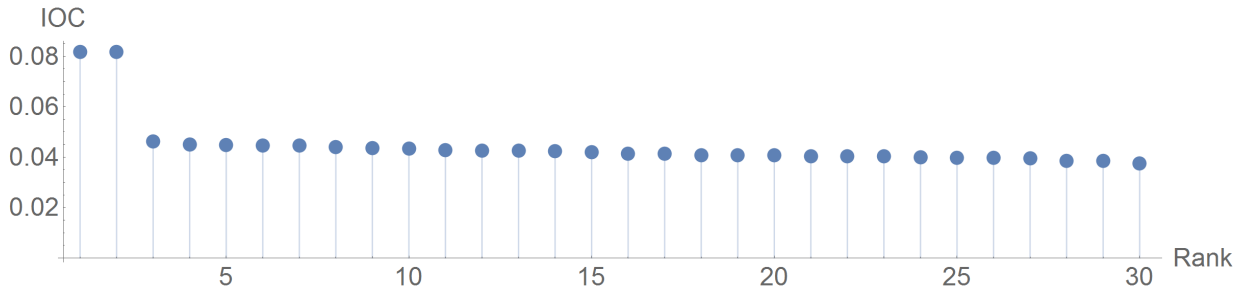


Figure 3: Index of Coincidence scores for the putative plaintext subsequences generated by the base rows for the ciphertext "EOPHUAUSHGK DD ... AA ,WFKNMR." The best scores correspond to $(26, 19)$ and $(26, 4)$, respectively.

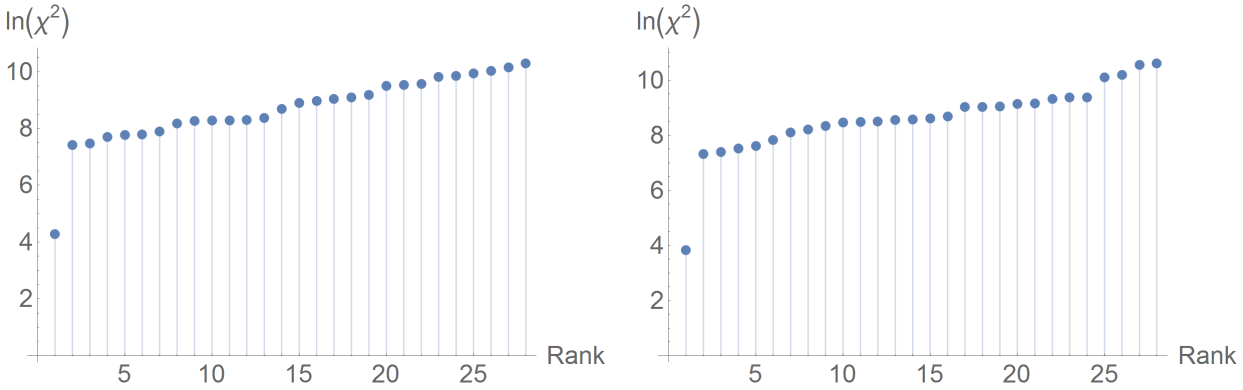


Figure 4: Goodness of fit scores for the putative plaintext subsequences corresponding to (a) $m(26, 19)$ and (b) $m(26, 4)$, $m = 1, 2, \ldots, 28$. The best (lowest) scores correspond to $20(26, 19) \equiv (27, 3)$ and $28(26, 4) \equiv (3, 25)$ mod 29, respectively.

putative plaintext in digraph segments. The digraph distribution of the putative plaintext is compared to that of English using a χ^2 score. The row pairs forming digraphs that resulted in the lowest χ^2 scores would display the appropriate ordering for the decryption matrix. For example, if the lowest scoring digraph combinations for a 4×4 key matrix were $\begin{pmatrix} B \\ A \end{pmatrix} \begin{pmatrix} A \\ C \end{pmatrix} \begin{pmatrix} D \\ B \end{pmatrix}$, then the correct ordering of the decryption rows is $\begin{pmatrix} D \\ B \\ A \\ C \end{pmatrix}$.

4.3 Present Progress

This thesis presents further improvements upon the work done by the Elizabethtown team by dramatically decreasing the amount of time to complete a ciphertext-only attack. Our first improvement involves a change to working with plaintext instead of the entries in the decryption matrix. Working directly with the plaintext allows us to do random searches that are guided by expected character frequencies in a given language. For instance, in English, it makes much more sense to choose space as a character in a sequence than an 'X' because space is the most frequently occurring character in English whereas 'X' occurs only rarely. The second significant outcome of this thesis is that the resulting attack can easily be parallelized to run on multiple processors.

Consider the ciphertext from the last example shown in (1). Before cryptanalysis, the h_{ij} and p_i are all unknown and the goal is to recover them. The attacks explained above strive to determine the h_{ij} first, but now we strive to find the p_i first.

$$\begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} 4 & 15 & ... & 12 \\ 14 & 7 & ... & 17 \end{pmatrix} \equiv \begin{pmatrix} p_1 & p_3 & \dots & p_{1433} \\ p_2 & p_4 & \dots & p_{1434} \end{pmatrix} \mod 29 \tag{1}$$

Consistency conditions for (1) lead to a plaintext sequence of the form

$$\{p_1, p_2, p_3, p_4, 26p_1 + 6p_3, 26p_2 + 6p_4, 9p_1 + 26p_3, 9p_2 + 26p_4, \dots\}.$$

In general, the sequence looks like

$$\{p_1, p_2, p_3, p_4, Ap_1 + Bp_3, Ap_2 + Bp_4, Cp_1 + Dp_3, Cp_2 + Dp_4, \dots\}$$

where A, B, C, D, ... are integers from 0 to 28. Note that odd-even pairs have the same coefficients. This similarity in coefficients allows us to break this larger sequence into smaller subsequences or either the odd p_i or the even pi. For example, the odd subsequence has the form

$$\{p_1, p_3, Ap_1 + Bp_3, Cp_1 + Dp_3, Ep_1 + Fp_3, Gp_1 + Hp_3, \dots\}.$$

Once we have this smaller subsequence we can generalize the possibilities for p_1 and p_3 via formation of base rows. The idea of forming base rows results from the $\phi(L)$ reduction that the Elizabethtown team discovered. Our base

rows take the form of $\begin{pmatrix}26 & c_1 & c_2 & \dots\end{pmatrix}$, $\begin{pmatrix}0 & 26 & c_1 & \dots\end{pmatrix}$, $\begin{pmatrix}0 & 0 & 26 & \dots\end{pmatrix}$ where for an $n \times n$ matrix, the base rows would have n elements. These base rows start with 26 because that corresponds to the space character in our alphabet and space is the most frequently occurring character in the English language. The c_i may be any integer from 0 to 28, but are strategically chosen in an order based upon letter frequency in English given as such where higher frequency characters are chosen first:

26	4	19	0	14	8	13	18	17	7	11	3	2	20	12	5	15	6	22	24	1	28	27	21	10	23	9	16	25
space	E	T	A	O	I	N	S	R	H	L	D	C	U	M	F	P	G	W	Y	B	,	.	V	K	X	J	Q	Z

Each of these base rows are then substituted for the original odd (or even) plaintext characters and expanded to complete the odd (or even) subsequence of the larger text. For example, substituting 0 for p_1 and 26 for p_3 gives

$$\{p_1, p_3, Ap_1 + Bp_3, Cp_1 + Dp_3, Ep_1 + Fp_3, Gp_1 + Hp_3, \dots\} = \{0, 26, 26B, 26D, 26F, 26H, \dots\}.$$

The even/odd partition is only relevant for the 2×2 case. For the 3×3 case, the sequences would be in steps of three starting at p_1, p_2 and p_3 where the first would be in terms of p_1, p_4 and p_7, the second in terms of p_2, p_5, and p_8 and the third in terms of p_3, p_6 and p_9. Each putative plaintext subsequence formed from a base row is then analyzed using the IoC score and the IoC scores are ordered from highest to lowest. The n highest IoC scores should be significantly higher than the rest, but it is acceptable to use only the top $(n-1)$ IoC scores as explained in [7]. The top n IoC scores represent the base element combinations that will build the actual decryption elements.

Once these base elements are obtained, all 28 multiples of each base element are calculated and scored with the χ^2 score. With our alphabet length of 29, all numbers less than 29 with the exception of zero are relatively prime to 29, so there are 28 multiples to check for each base element combination. The smallest χ^2 scores most correspond to elements that can be used to recover the entire plaintext. Once all of the proper subsequences have been obtained, it is necessary to arrange them to form the entire plaintext sequence. In a simple 2×2 case that means simply deciding which multiple should be assigned to the even subsequence and which should be assigned to the odd subsequence, and a visual examination of the putative plaintext will display the correct selection.

In a larger $n \times n$ case, there are $n!$ ways to select the elements for the base p's which leads us back to Elizabethtown's method of scoring digraphs. In a 4×4 case, we would try different combinations of the suggested decryption elements for p_1 and it's group and p_2 and it's group until we have found the putative plaintext digraphs for each set of combinations. The χ^2 score of these digraphs compared to English digraphs resulting from the Brown Corpus in which a lower score corresponds to English will ultimately show which order to place the base

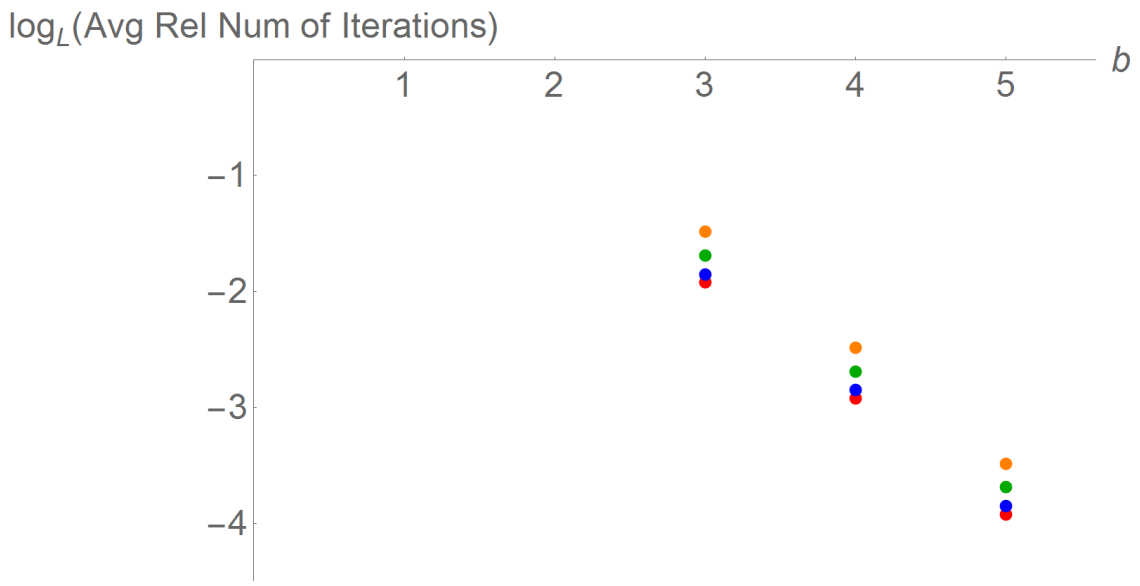


Figure 5: As b, the block size, increases, the average relative number of iterations attempted decreases linearly independent of text length shown by the varying colors of points

p_i in. This decreases the amount of work necessary from $n!$ attempted orderings to $\binom{n}{2} = \frac{n!}{2(n-2)!}$ possibilities.

The Elizabethtown method is difficult to parallelize, but our approach can easily be modified to run efficiently on multiple processors. For example, consider the $n = 2$ case in which each term in the odd subsequence is written as a linear combination of p_1 and p_3. The choices of p_1 and p_3 are largely arbitrary, so we could express the terms in the subsequence as a linear combination of p_3 and p_5, or p_5 and p_7, and so on.The maximum number of different subsequences possible is

$$\left\lfloor \frac{\text{Text Length}}{n} \right\rfloor - n + 1$$

where $\lfloor x \rfloor$ is the greatest integer less than or equal to x.

If we rename the base elements in each subsequence p_1 and p_3 then we can test a great number of subsequences simulataneously, and if each subsequence is tested on a separate processor, then the speedup can be dramatic, especially if processing is stopped once n quality base elements are discovered.

We tested this approach for $n = 3$, 4, and 5 and, on average, less than 0.5% of all base classes needed to be considered to recover the plaintext. For a text of length 500, it was found that for the 3×3 case only 0.23% of all possible base classes had to be considered, for the 4×4 case only 0.28% were considered and for the 5×5 case only 0.017% of all possible base classes were considered as shown in Figure 5 .

Furthermore, we found that for a text length of 1000, in the 3×3 case, 4×4 case, and 5×5 case, we were able to recover all of the correct scoring p combinations 99.7%, 98.9% and 96.0% of the time, respectively. In all cases, when we didn't recover all of the combinations, we did recover all but one. This shows that as long as the text

is sufficiently long, the method works efficiently. Not recovering all n subsequences is acceptable because having $(n-1)$ out of n subsequences still usually produces intelligible plaintext as outlined in [7].

Ultimately, by trying plaintext subsequences based on character frequencies and the use of parallelization across all possible plaintext subsequences, we are able to cut the necessary work by a factor of approximately 400-500, obtained by considering the fact that we only need to consider approximately 0.002 to 0.0025 of all possible base class multiples. We implemented our method in *Mathematica*, so further advancements can be achieved by re-programming our method in C or Java and by using multiple processors as outlined above, but that work is left for future research.

4.3.1 2×2 Example using Gettysburg Address

For completeness, let's resume the example begun in (1). If we regard the p_i as known (which they're not) and the h_{ij} as unknown, then (1) gives us 1434 congruences for 4 unknowns, h_{11}, h_{12}, h_{21}, and h_{22}. Solving for those unknowns generates consistency constraints on the p_i and tell us that the plaintext has the following form

$$\begin{aligned}\{p_1, p_2, p_3, p_4,\ & 26p_1 + 6p_3, 26p_2 + 6p_4, 9p_1 + 26p_3, 9p_2 + 26p_4,\\ & 16p_1 + 2p_3, 16p_2 + 2p_4, 11p_1 + 19p_3, 11p_2 + 19p_4, \ldots,\\ & 12p_1 + 11p_3, 12p_2 + 11p_4, 25p_1 + 27p_3, 25p_2 + 27p_4\}\end{aligned}$$

for some p_1, p_2, p_3, and p_4. Highlighting the even and odd terms,

$$\begin{aligned}\{p_1, p_2, p_3, p_4,\ & 26p_1 + 6p_3, 26p_2 + 6p_4, 9p_1 + 26p_3, 9p_2 + 26p_4,\\ & 16p_1 + 2p_3, 16p_2 + 2p_4, 11p_1 + 19p_3, 11p_2 + 19p_4, \ldots,\\ & 12p_1 + 11p_3, 12p_2 + 11p_4, 25p_1 + 27p_3, 25p_2 + 27p_4\}\end{aligned}$$

allows us to extract a representative subsequence

$$\{p_1, p_3, 26p_1 + 6p_3, 9p_1 + 26p_3, 16p_1 + 2p_3, 11p_1 + 19p_3, \ldots, 12p_1 + 11p_3, 25p_1 + 27p_3\}$$

that contains only two unknowns.

Suppose that we intend to implement this method on a machine with four CPUs. Then we can re-write the representative subsequence relative to 3 other base pairs as follows.

$$\{p_1, p_3, 26p_1 + 6p_3, 9p_1 + 26p_3, 16p_1 + 2p_3, 11p_1 + 19p_3, \ldots, 12p_1 + 11p_3, 25p_1 + 27p_3\}$$

$$\{2p_3 + 19p_5, p_3, p_5, 15p_3 + 26p_5, 5p_3 + 14p_5, 12p_3 + 6p_5, \ldots, 6p_3 + 25p_5, 19p_3 + 11p_5\}$$

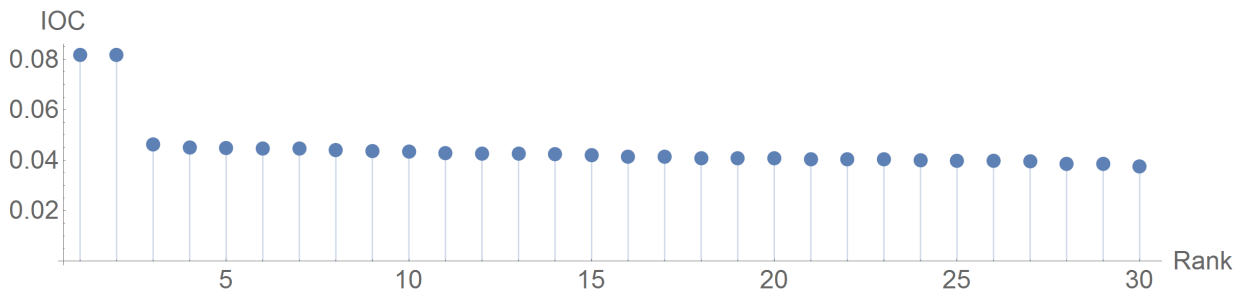


Figure 6: Index of coincidence scores for the illustration of our cryptanalysis method. The two base elements with the highest IoCs are $(26, 17)$ and $(26, 15)$.

$$\{2p_5 + 4p_7, 6p_5 + 2p_7, p_5, p_7, 15p_5 + 10p_7, 20p_5 + 24p_7, \ldots, 3p_5 + 12p_7, 9p_5 + 9p_7\}$$

$$\{22p_7 + 4p_9, 27p_7 + 12p_9, 9p_7 + 2p_9, p_7, p_9, p_7 + 11p_9, \ldots, 10p_7 + 6p_9, 3p_7 + 18p_9\}$$

Then we rename all of these base pair entries p_1 and p_3 as follows.

$$\{p_1, p_3, 26p_1 + 6p_3, 9p_1 + 26p_3, 16p_1 + 2p_3, 11p_1 + 19p_3, \ldots, 12p_1 + 11p_3, 25p_1 + 27p_3\}$$

$$\{2p_1 + 19p_3, p_1, p_3, 15p_1 + 26p_3, 5p_1 + 14p_3, 12p_1 + 6p_3, \ldots, 6p_1 + 25p_3, 19p_1 + 11p_3\}$$

$$\{2p_1 + 4p_3, 6p_1 + 2p_3, p_1, p_3, 15p_1 + 10p_3, 20p_1 + 24p_3, \ldots, 3p_1 + 12p_3, 9p_1 + 9p_3\}$$

$$\{22p_1 + 4p_3, 27p_1 + 12p_3, 9p_1 + 2p_3, p_1, p_3, p_1 + 11p_3, \ldots, 10p_1 + 6p_3, 3p_1 + 18p_3\}$$

These final four subsequences are then assigned to different processors to speed up the run-time by a factor of close to 4.

We loop over all base class elements of the form $(p_1, p_3) \equiv (0, 26)$ or $(26, p)$, $p = 0, 1, \ldots, 28$, stopping when we have found two high-scoring elements. We also order the choices of p to correspond to the likelihoods of their appearing in English plaintext. Figure 6 shows the indices of coincidence for all of the base pairs, but in practice computation stops once the two scores near 0.08 are discovered.

Once the promising base elements are discovered, we test the multiples $m(26, 17)$ and $m(26, 15)$, $m = 1, 2, \ldots, 28$ using the χ^2 score. The multiples with the best fit are $(5, 20)$ and $(14, 17)$, respectively. This gives two possibilities for (p_1, p_2, p_3, p_4), namely $(5, 14, 20, 17)$ and $(14, 5, 17, 20)$. The first gives the plaintext sequence

$$(5, 14, 20, 17, 28, 2, 14, 17, \ldots)$$

that corresponds to "FOURSCORE AND SEVEN...," and the second gives

$$(14, 5, 17, 20, 2, 18, 17, 14, \ldots),$$

which corresponds to "OFRUCSRO ENA..." Clearly, the first choice is correct and we have recovered the original plaintext.

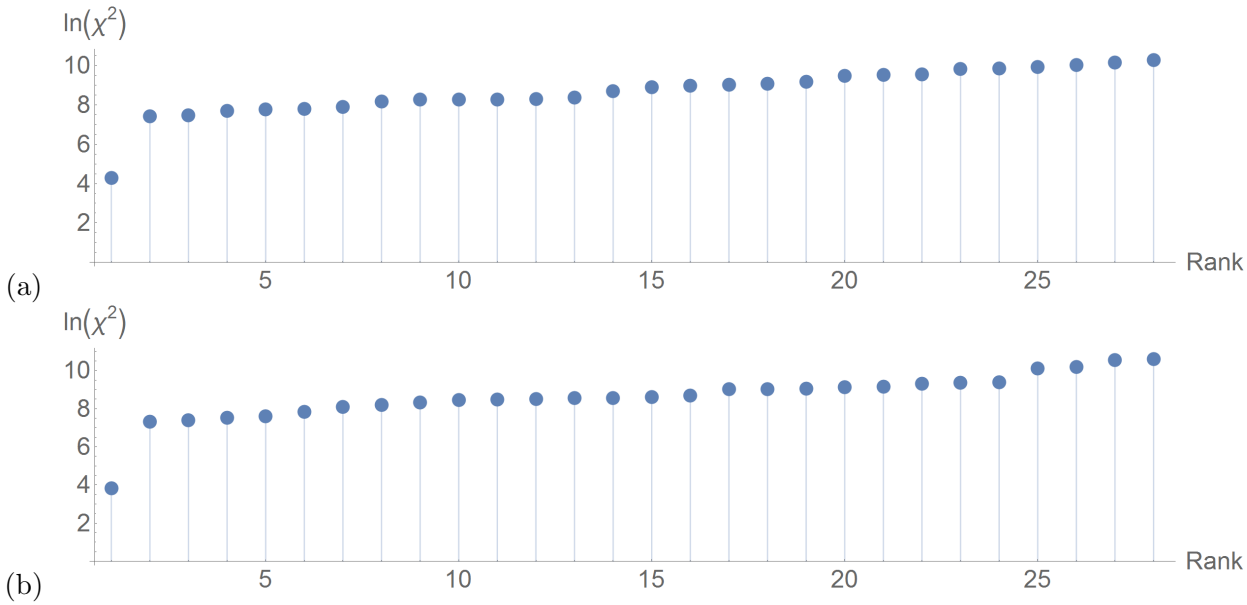


Figure 7: $\ln(\chi^2)$ scores for all multiples of (a) $(26, 17)$ and (b) $(26, 15)$ illustrating our cryptanalysis method. The lowest scores correspond to $(5, 20)$ and $(14, 17)$ respectively.

5 Conclusion

5.1 Other Research Efforts

Although we have made some significant advancements, there are many more areas to be pursued in future research of cryptanalysis of the Hill Cipher. As all previous cryptanalysts have stated, there is still a need for a better scoring algorithm if a row-by-row decryption of the inverse key matrix is utilized. Progress has been made in this area by moving from the Bauer-Millward ad hoc approach to the Elizabethtown team's Index of Coincidence method.

We have tried a variety of other approaches that are not described in this thesis because they were ultimately unsuccessful. Some attempted methods included using different goodness-of-fit tests including the Kolmogorov-Smirnov test and making various changes to the χ^2 test including working with an ordered χ^2 test. This version compared the frequencies of the characters in ranked order, highest to lowest between the plaintext and English, with no regard to which character they were representing. Alterations were also made to the χ^2 formula in which the denominator was transformed into it's inverse, but no significant improvements resulted from any of these methods. Other attempted scoring methods included the creation of a dictionary score where the plaintext was given a score based upon how closely it matched an English word from the *Mathematica* dictionary and we also tried to work with our putative plaintext subsequence to use this dictionary score to more accurately choose characters so that they would be creating words. Two important aspects that we considered with the dictionary

score was the frequency of spaces and average word length. Space is the most frequently occurring character in the English language, so an abnormally low number should be a warning. Average word length was considered because if we were to receive a string of putative plaintext that consisted of multiple words over 20 characters long, it was obvious that it was not the correct sequence and it could be discarded early on in the testing sequence.

One technique that frequently reappeared in what we were trying was the Markov matrix. Due to the nature of the English language, there are various relationships between characters and distance between characters. If they are sequential characters, there are very strong relationships between the characters and which follows which. For example, it is highly likely that 'q' would be followed by 'u', but highly unlikely that it would be followed by 'x'. There are also relationships between characters that have one or more characters between them. These relationships are easily modeled using a Markov matrix which can be assembled using a large set of text like the Brown Corpus. The benefit of using a Markov Matrix is that it only needs to be constructed once for each distance between characters, whether they be next to one another, one character apart, etc and then it may be continuously referenced. The difficulty with the Markov matrix arose when we were considering large key sizes because of the very poor relationships between characters that are significantly far apart, such as ten characters apart, which led to us abandoning this technique for large cases.

At various points in our research, we also considered working with digraphs or triples of letters. In English, as stated above, there are clear relationships between characters and which are more likely to be followed by others. We tried to use this technique when working with our putative plaintext string to intelligently choose characters using something similar to our dictionary scoring mechanism due to these relationships. This method collapsed when we considered the magnitude of how many possible triples, quadruples, and so forth exist. The chance of starting a triple or quadruple and having the first character wrong from the start, but proceeding with testing that first character through every possible second and third character before proceeding to a different first character of the sequence also generated concern. This would ultimately be very time consuming and therefore would not be very efficient.

5.2 Closing Thoughts

Two major advancements were made through our efforts in writing this thesis. Looping over putative plaintext instead of random matrix enties enables us to use a much more efficient method to choose characters. Through the use of putative plaintext, we were able to use the common knowledge of English character frequencies and the relationships between characters. When we are working with plaintext, we have a reason to start a series of text with a space or an 'e' than with a 'q' or an 'x'. This is an improvement over working with matrix entries for the decryption key because there all of the entries were randomly selected.

The second advancement came in the form of easier parallelization of the search process. In our putative plaintext case, it was simple to parallelize by considering every possible subsequence in the text length and trying base classes that were strategically constructed based upon a frequency sorted alphabet. If a base class attained a high IoC score, then all of the multiples of the base class were calculated and then applied to the plaintext subsequence to calculate regular χ^2 scores. If the χ^2 score was sufficiently low, then the multiple was set aside as a likely candidate for one of the subsequences of the plaintext sequence. When n candidate χ^2 scoring base class multiples were obtained for an $n \times n$ block size, the search was halted and the cipher was deemed solved as these n base class multiples assembled the plaintext. Ultimately, less than 0.5% of all possible base class multiples had to be attempted, resulting in a reduction of necessary work by a factor of 400 to 500. For a text length of 1000, all of the correct 'p' combinations were obtained 99.7%, 98.9% and 96.0% of the time for the 3×3, 4×4 and 5×5 block size cases respectively. Additionally, it is not necessary for every row to be obtained, as it is possible to reconstruct the plaintext with knowledge of all but one row. This reduction in work is a major improvement over the $\phi(L)$ reduction from the Elizabethtown team and stands to be further reduced if the program is written to run in a computer programming software that can run it over multiple computer cores.

The combination of use of plaintext, a sorted alphabet based upon commonly known frequencies and parallelization over multiple subsequences led to a significant reduction in work necessary to perform cryptanalysis on the Hill Cipher.

References

[1] Bauer, Craig, and Katherine Millward. "Cracking Matrix Encryption Row by Row." *Cryptologia* 31 (2007). Print.

[2] The Brown Corpus of Standard American English, available for download at www.cs.toronto.edu/gpenn/scs401/a1res.html.

[3] Christensen, Chris. "Lester Hill Revisited." *Cryptologia* 38 (2014). Print.

[4] Damico, Tony M. "A Brief History of Cryptography." *Student Pulse* 1.11 (2009). Student Pulse. Web. 17 Dec. 2015. http://www.studentpulse.com/articles/41/a-brief-history-of-cryptography.

[5] Hill, Lester. "Concerning certain linear transformation apparatus of cryptography." *American Mathematical Monthly* 38 (1931): 135-54. Print.

[6] Hill, Lester. "Cryptography in Algebraic Alphabet." *American Mathematical Monthly* 36.1929: 306-12. Print.

[7] Leap, Tom, et al. "Further Improvements to the Bauer-Millward Attack on the Hill Cipher." (2015). Print.

[8] Levine, Jack. "Some Applications of High-Speed Computers to the Case $n = 2$ of Algebraic Cryptography." *Mathematics of Computation* 15.75 (1961). Print.

[9] Levine, Jack. "Some Elementary Cryptanalysis of Algebraic Cryptography." *American Mathematical Monthly* 68.5 (1961): 411-18. Print.

[10] Levine, Jack, and Richard Chandler. "The Hill Cryptographic System with Unknown Cipher Alphabet but Known Plaintext." *Cryptologia* 13.1 (1989): 1-28. Print.

[11] Morelli, R. "The Vigenere Cipher." *Cryptography.* Trinity College, 10 June 2014. Web. 17 Dec. 2015. http://www.cs.trincoll.edu/ crypto/historical/vigenere.html.

[12] Yum, Dae Hyun, and Pil Joong Lee. "Cracking Hill Ciphers with Goodness-of-Fit Statistics." *Cryptologia* 33 (2009). Print.