Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
DOI : 10.47974/JDMSC-2688

Pseudo-homophonic Hill enigma ciphers encrypting to trigram ciphertext

Porter E. Coggins
School of Education
Bemidji State University
Bemidji 56601
Minnesota
U.S.A.

Abstract

A novel Hill Cipher – type encryption scheme is presented that has characteristics of a homophonic cipher and in which each plaintext character is mapped to a trigram of ciphertext characters. Both the Homophonic Cipher and Hill Cipher continue to generate research today from the time of the seminal works in the 20th century. The Homophonic Cipher is a polyalphabetic substitution stream cipher that encrypts as close to a uniform frequency distribution of ciphertext characters as possible by allowing some or all plaintext characters to map to a different ciphertext character with each occurrence of the particular plaintext character. The Hill Cipher is a polyalphabetic block cipher that encrypts blocks of plaintext through techniques of linear algebra in the context of matrices and produce ciphertext that take advantage of the process of multiplication between a plaintext vector and a key matrix. There is a gap in the literature explicitly relating the homophonic cipher to the Hill Cipher under certain conditions. Thus, a novel, new encryption scheme is presented in which each plaintext character is mapped to several ciphertext characters at the same time combining a homophonic cipher and a Hill Cipher. Due to the rotating elements of the 2 x 2 matrices and similarity to the German Enigma Encoder, these encryption scheme variations are referred to as *Hill Enigma Ciphers*. Multiple examples are given.

Subject Classification: *94A26, 68P25, 20G45.*

Keywords: *Encryption, Homophonic cipher, Hill cipher, Matrix theory.*

ORCID-ID: **0000-0003-3734-5478**

E-mail: **porter.coggins@bemidjistate.edu**

© Taru Publications

1. Introduction

Both the homophonic cipher and the Hill cipher are still active areas of research today largely involving image security and encryption. The academic literature on the homophonic cipher began to appear in the early 1970s [1]. The academic literature on the Hill cipher first appeared in the late 1920s [2] and early 1930s [3]. The Homophonic Cipher is a polyalphabetic substitution stream cipher used to create as close to a uniform distribution of ciphertext characters as possible by allowing some or all plaintext characters to map to a different ciphertext character with each occurrence of the particular plaintext character. One such example is given below in the next section under Homophonic Cipher Example. The Hill Cipher represented a novel application of linear algebra and matrix theory to an encryption scheme. This is a polyalphabetic block cipher used to map blocks of plaintext through the use of linear algebra in the context of matrices and produce ciphertext that take advantage of the process of multiplication between a plaintext vector and a key matrix producing an avalanche effect up to the dimension of the key matrix used..

In the examples that follow, two versions of the plaintext are used [4]. The *modified original plaintext* paragraph with approximate distribution frequency of the English Language as represented in Table 1 appears in Appendix A. The modified original plaintext has all punctuation removed with these exceptions: all spaces, periods, commas, uppercase letters and numbers remain, and all hyphens " - ", semi-colons " ; ", apostrophes " ' " have been removed. The modified original plaintext is used in complex versions of the Hill Cipher, referred to as *Hill Enigma* due to rotating elements of the 2 x 2 matrices similar to the rotor wheels of a typical German Enigma Encoder [5]. The *reduced plaintext*, the plaintext with all spaces remaining, but punctuation removed and all uppercase letters reduced to lowercase letters appears in Appendix B [4]. The frequency distribution of the plaintext version used, original or reduced, will be stated in the appropriate sections, tables, and figures. Figure 1 gives the letter frequency distribution of the original plaintext of Appendix A. Figure 2 gives the letter frequency distribution of the reduced plaintext of the reduced plaintext of Appendix B.

2.1 *Homophonic Cipher Example*

The Homophonic Cipher [1] is a polyalphabetic stream cipher. A web search of "Homophonic Cipher" in the last 5 years returns over 300 scholarly publications. The goal of a homophonic cipher is to distribute

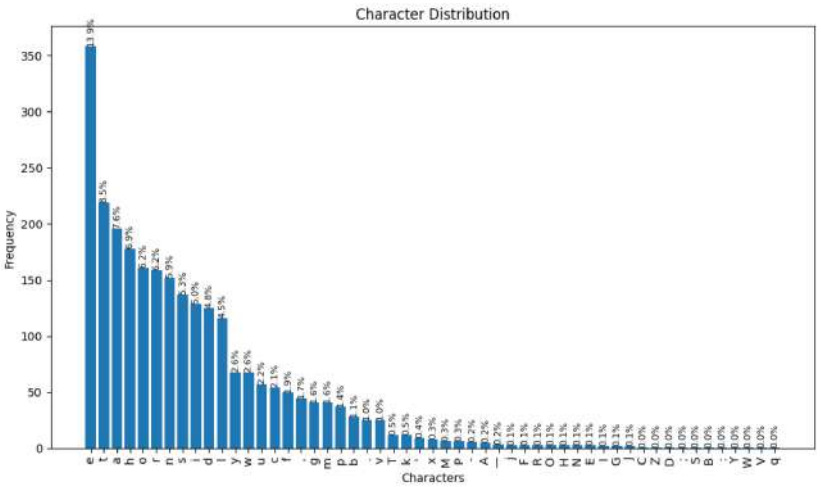

Figure 1

Original Plaintext Letter Frequency Distribution demonstrates the typical English letter frequency distribution within probability of an English text of sufficient length. Note that due to punctuation, spaces, and the use of upper-case letters of the plaintext, the distribution is typical of the English Language letter frequency distribution but varies depending on corpora.

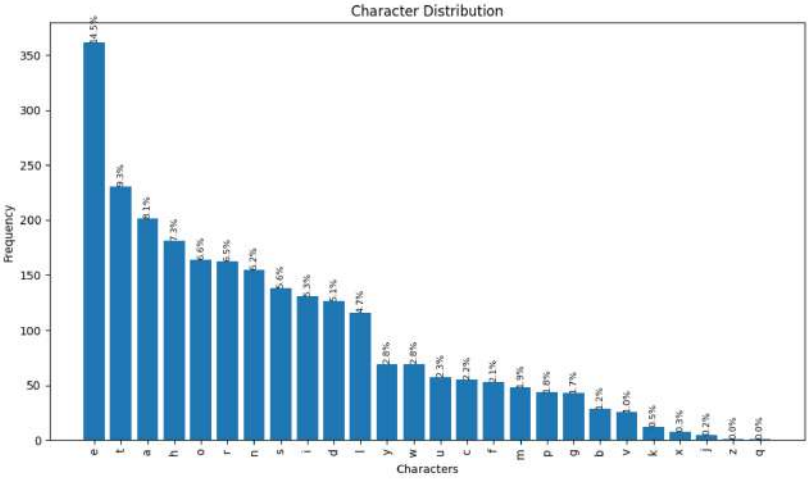

Figure 2A

Reduced plaintext (Appendix B) frequency distribution demonstrates the typical English letter frequency distribution within probability of an English text of sufficient length and varies depending on corpora.

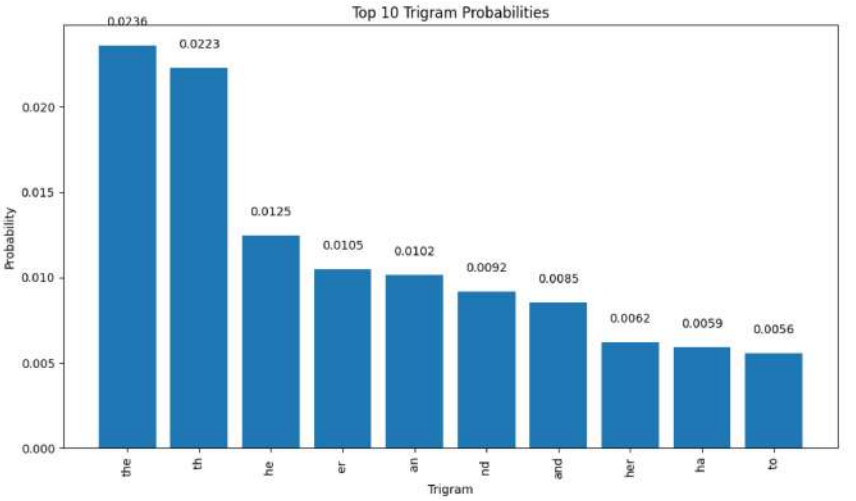

Figure 2B

Reduced plaintext (Appendix B) top ten trigram probability frequency distribution ranging from probability of the most frequently occurring trigram THE with 0.0236 down to the least frequently occurring trigram (in the top ten) with 0.0056 *to where "*" represents a free character, and then there is a gradual decline in probability of the remaining eight of the top ten trigrams between the probability of the most frequently occurring to the probability of the least frequently occurring. This demonstrates the typical English trigram frequency distribution within probability of an English text of sufficient length and varies depending on corpora.

the ciphertext as uniformly as possible. By assigning plaintext characters to the ciphertext characters in a particular way, a near uniform distribution of the ciphertext can be created which decreases the success of ciphertext frequency cryptanalysis. Table 1 presents a three – column example of a homophonic encryption scheme: the left column shows the distribution of letters of the English alphabet in percentage from greatest percentage (most frequently occurring letter) to lowest percentage (least frequently occurring letter) but a typical analysis of the English language [6], the middle column shows the English alphabet in letter – order from the most frequently occurring letter to the least frequently occurring letter, and the right column represents the corresponding ciphertext letter assignments from the plaintext letter above. Note that the distribution of English plaintext letters vary slightly depending on the corpora used.

The basic scheme of a homomorphic cipher is that the number of the ciphertext characters available for encryption from the plaintext characters

is dependent upon the frequency of the plaintext characters. The greater the plaintext character frequency, the greater the number of ciphertext characters available to the plaintext for encryption.

Table 1 gives one possible set (naively constructed) of homophonic substitutions for each plaintext character that produces a much more

Table 1

Example of Homophonic Plaintext to Ciphertext Mapping

Letter Frequency in Percentage	Plaintext Letter	Ciphertext Substitutions
12.02 %	E	A B C D E F G H I J K L
9.10 %	T	M N O P Q R S T U
8.12 %	A	V W X Y Z A B C
7.68 %	O	D E F G H I J K
7.31 %	I	L M N O P Q R
6.95 %	N	S T U V W X X (X repeats twice)
6.28 %	S	Z A B C D E
6.02 %	R	F G H V J K
5.92 %	H	L M N T P Q
4.32 %	D	R S U U (U repeats twice)
3.98 %	L	V W X Y
2.88 %	U	Z A B
2.71 %	C	C D E
2.61 %	M	F G H
2.30 %	F	Y J
2.11 %	Y	K L
2.09 %	W	M N
2.03 %	G	O P
1.82 %	P	Q R
1.49 %	B	S
1.11 %	V	I
0.69 %	K	U
0.17 %	X	H
0.11 %	Q	I
0.10 %	J	Z
0.07 %	Z	Z

uniform distribution with respect to the plaintext. The plaintext to ciphertext mappings were naively constructed to illustrate a more uniform ciphertext distribution compared to the plaintext distribution. More sophisticated methods of constructing homophones for the ciphertext are given in the literature [7], [8], [9].

Section 2.1.A gives the Python pseudo-code for an example of a homophonic cipher. Section 2.1.1 gives the homophonic ciphertext from the reduced plaintext in Appendix B. Section 2.1.B gives the ciphertext frequency distribution of the reduced plaintext. Section 2.1.C gives the ciphertext probability frequency distribution of the top ten trigrams.

A comparison between the reduced plaintext in Figure 2A above and in Figure 3A below highlight the more uniform distribution of the reduced homophonic cipher with the reduced plaintext to ciphertext letter substitutions. The reduced plaintext in Appendix B has been encrypted with the homophonic cipher substitutions (Table 1 above) to the ciphertext below (Section 2.1.1).

2.1.A Python Pseudo-Code for Homophonic Cipher:

```
# Define a mapping of characters to their substitutes
mapping = {
   # ...
}
# Define a function to substitute characters in a plaintext using the
mapping
def homophonic_substitute(text, mapping):
   # Initialize counters for each character in the mapping
   counters = {...}
   # Initialize an empty list to store the result
   result = []
  # Iterate over each character in the text
   for ch in text:
      # Check if the character is in the mapping
      if ch.upper() in mapping:
         # Get the substitutes for the character
         subs = mapping[ch.upper()]
          # Get the next substitute
         rep = subs[counters[ch.upper()] % len(subs)]
         # Preserve the case of the original character
         rep = rep.lower() if ch.islower() else rep
```

```
            # Append the substitute to the result
            result.append(rep)
            # Increment the counter for the character
            counters[ch.upper()] += 1
        else:
            # If the character is not in the mapping, append it as-is
            result.append(ch)
    # Join the result into a string and return it
    return ''.join(result)
# Define a function to print ciphertext wrapped at a certain width
def print_wrapped(text, width=58):
    # Iterate over the plaintext in chunks of the given width
    for i in range(0, len(text), width):
        # Print the current chunk
        print(text[i:i+width])
# Define the plaintext
plaintext = "..."
# Substitute the characters in the plaintext using the mapping
ciphertext = homophonic_substitute(plaintext, mapping)
# Print the ciphertext wrapped at 58 characters per line
print_wrapped(ciphertext, width=58)
```

2.1.1 Reduced Homophonic Ciphertext Paragraph From Appendix B:

Mlakmbfcsdngvlvnwlcmmwrhdtoeseonumopnpsfvqzzqrgafrtgkpxuhiivs
qhapltysibujzqxnalbjhdkqmbccflvwdjyoanorxppkgweubykxcxsuedeuzq
mdrytrfhlfezghviomczwpakrfuabvsgcuvhgszbcwaudtnhlmifjzbeujgrnw
xhkzzsavsyxdtoxsjkxzsumpllypiaunqnqolbrkjapmcfbtrfkqndglu cvdhsm
wvvkjeuskneuyfexrfuio vwwmpstcfxgbggroyxabqxtphjhhsusijvymrviuq
zcdapjnqboncedvwxku wlxyvyszpmjkdgmlqnteratrefwsnbslkaxbctfmds
cyvzygduepmnvugnacwuvohirsewwjjmqoufrxpxmyxugihhlhkuivscdti
juqkvdlzbnlamezcbapcwodebkpndjenfkgqtfhifyrpjfggskhscmvztqlaxrba
uhjeiavbcgjmldjcxifkmwigyxikbfrnhdszoljpnimghdqtjhpysqzuaybieavnr
kosfpoltmcilsabxjfsckuvsdhmcwgeqfngfnngmqethuohsixtjuprvklgnpaw
qbohcm bryxrpldlizxpezqnzaqmkyvlbmnjhfasrghsvhfiwlnjxygkttkjsmu
tducrluub vwpakmnhbovqckpworaxydvv wdnesbxyvwuqcexxuzroplxp
rzwxlyqqffdmzmrnsojgfsayvwauoeashrbk ecxybtilfgcmgnvdwvgjeuxw
mylzmnhhivjzuknkqxalnpotppkcqqyrjltbuskacuqdrlbhvvumcvrsndfter
futerfpggfwgabhwcshdlsiemqmzaljbbkunimgnolcpjhoadykvqmbhnnpx
crtdkmejfwsedpdeyxxukgyrtqhfzvfzsajuifmjbcxuzlkvwnwlagxulaqyb
stkrqaebbgjqhcnglyihomdrvsljpnrykczusjjkkqtefpgfardcdbdmevezzepca
sqgclhnbthwoxofyumixjmupmnkscgvonljajaktqdtruphyppbuixyeqjbzc
kfqldrlhenmfvsnglmhgiojtumwpdikhmtlfzgranpabboqvcdvadukel

nwujexvxnwlcmxrfigkkpnoxoqthlsijskuqflrskcyetpadpfgujpgbeyvhicwlt
bhzdvkmquqlezraiymjlbaurbhk nmfjodecgncxyqvrhkoniktwsxurvuywu
fjfglxzhdkgzlraxsbufpmpgncedvvwluzwhazxsrmqhnyaplbmvicjkimtq
mdlzxbjnasbjcrteknpkmvczadsqffygetuhiigchdezvjikvbwzzauwxuxyny
kbjlzuljekaobfacmvbvcnnywoxxkpogcqdmydsfqfezrllfawnbkemcsvtgh
dhxyiurgjwzgvjhhpsxwukhrvyqxlutnavbzaixujmtceprwrjdsbkltnekfco
mupxlavsyawxnykpqbqrlgljhvsdbfimjcdzwuxtnkxqtohelyykeampvwu
pqsumqcnolalfbompqgnvibscdtiegktwrqlxuuzhrpfknghhszstqilrrsviuu
wjmlknmyrqkmlxynnjrvkoplvmjlnxqtajbuiowxzmaadickbxuresdnp
rpctssqeklvutfyqifwigfluryjhgiuuvlyjmvrklsmmauhbxujawdnywpcezfdi
bnnebzrsfxxkqgohvtaubsojppuclqpcqixrlvggsomqdhsnjkrfkelttzmwaxh
iraybuprvcfjgdqehyfcumtywkkaddbvsgihvpziiujjagkuejnqlfolakrfbbcxp
cndgeyzwxgjembhhsaqrnfgxolshmcdybyvivhjctvjfspwtxuliytqjjulkk
enlfammzdmyggwzanbrnvoobjzxyqchydeppkwjdvehacqvkrnjfgsblcst
hxemqxpirtpjkufrkmlfdlsahqbgmctddghrqkwxnezohiuounevopjwanlq
emncusfgxvvpthjqkibcrxpkfqddehjfrpkgcwaelsqabsoltczbrsduylmevtg
yjnmulxsywpzzffamabchyhnommgxhunnbotinjxpizqcsurostjrdkgksv
euwvwuxkyxyspludzqdavuraeqvbmlfncjdpvtmeufwugkmhfisggujx
mhsqnesaoxipnrspkwzlnbatbhqahcbrrspcqjkvxltuydetomufcvrmqgjhivn
wpbdyrjnkjollepmakcdvosnbdfpcufidgqqszrgjehaheesilzjysfxukxdtcvk
temgihjuikfrswjmpkngwolrlaahxlsbnqmdlsyhucsomdnvxfpnebzouqtar
ppcfkgdmhjikjuskebbelqalcrvcgys

2.1.B Figure 3A of Frequency Distribution Graph of Homophonic Ciphertext

This simple, naïvely constructed plaintext to ciphertext letter substitutions illustrates that the homophonic cipher can more uniformly distribute the ciphertext frequency and increase resistance to a simple cryptanalysis by letter frequency. More specifically, comparing Figure 2 and Figure 3A, the range of the character frequency distribution of Figure 2A is 14.5% with the letter E appearing in 14.5% of the text and both the letters Z and Q appearing 0.0% whereas the range of the ciphertext for Figure 3A is 2.2 with the letter U accounting for the most frequently occurring letter at 4.9% of the ciphertext characters and the letter O accounting for the least frequently letter occurring at 2.7%. All ciphertext characters are represented in the ciphertext with at least a frequency of 2.7% in contrast to Figure 2 where some letters had no representation in the text.

Figure 3B represents the top 10 most frequently occurring trigram probabilities. Of note is that the top 10 occurring trigrams have a probability of occurring at 0.0012. Again, this demonstrates a uniform distribution of the top 10 trigrams. This will serve as a comparison point when variations of the Hill Enigma Cipher encoding to trigrams will be discussed.

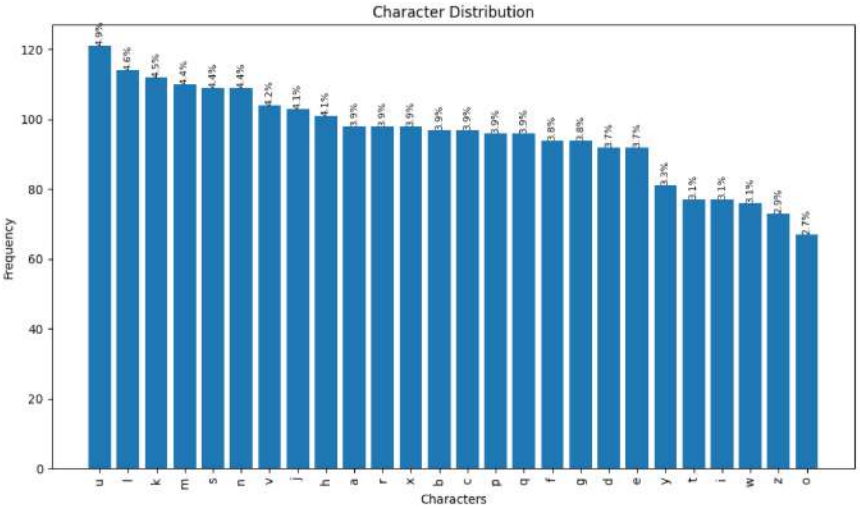

Figure 3A

Reduced Plaintext Homophonic Cipher Ciphertext Frequency Distribution. This frequency distribution shows a range of ciphertext letter frequency percentage between 4.9% to 2.7% which deviates from the typical distribution of English letter frequency percentage noted above in Figure 2.

2.1.C Figure 3B Probability Frequency Distribution Graph of Top 10 Trigrams of Homophonic Ciphertext Trigrams

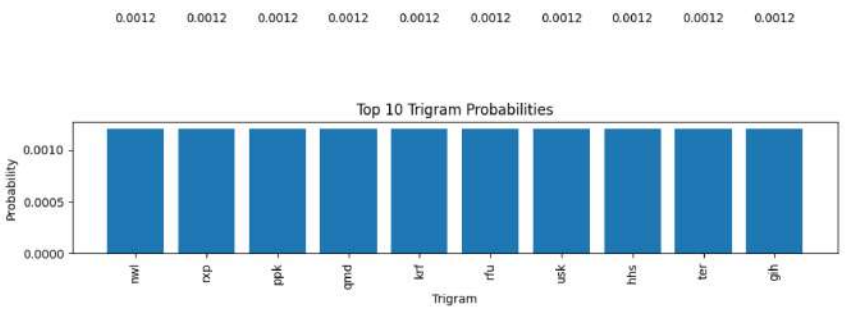

Figure 3B

Reduced Plaintext Homophonic Cipher Trigram Ciphertext Probability Frequency Distribution. This ciphertext character frequency distribution shows a range of the probability of trigram letter frequencies of the top 10 trigrams is 0.0012.

3.1 *Hill Cipher Examples*

The Hill Cipher [2], [3] is a block, polyalphabetic cipher that uses invertible matrices as the keys. The plaintext is taken in blocks represented as vectors corresponding to the size of the invertible matrix. A current web search of "Hill Cipher" produced over 2,000 results in 2025 alone at the time of the first draft of this paper in late 2025. The following examples will step through several versions of the Hill Cipher, simple to complex, finishing with a novel Hill Cipher, referred to as a *Hill Enigma Cipher*, represented in the literature demonstrating a pseudo-homophonic encryption that encrypts each plaintext character with a ciphertext trigram.

As a simple example using Hill's original 26-letter English alphabet as the domain for the plaintext, let the reduced plaintext be as stated in Appendix B.

Let the blocks be [2*x*1] column vectors. Let key matrix 1 be $\begin{bmatrix} 18 & 11 \\ 23 & 5 \end{bmatrix}$. All work is completed in modulo 26. Let the letter to number assignment map a = 0, b = 1, c = 2, … z = 25. Padding will be used to ensure a reduced plaintext character length is even. Section 3.1.A gives the Python pseudo-code for an example of a Hill Cipher. Section 3.1.1 gives the Hill ciphertext from the reduced plaintext in Appendix B for key matrix 1. Section 3.1.B gives the ciphertext frequency distribution of the reduced plaintext for key matrix 1. Section 3.1.C gives the ciphertext probability frequency distribution of the top ten trigrams for key matrix 1. Section Section 3.2 gives the ciphertext for key matrix 2. Section 3.2.B gives the ciphertext frequency distribution of the reduced plaintext for key matrix 2. Section 3.2.C gives the ciphertext probability frequency distribution of the top ten trigrams for key matrix 2.

3.1.A Python Pseudo-Code for a Hill Cipher using a Single 2 x 2 Invertible Key Matrix

```
# Define a function to calculate the modular inverse of a number
def mod_inv(a, m):
    # Iterate over possible values to find the modular inverse
    for x in range(1, m):
        if (a * x) % m == 1:
            return x
    raise ValueError(f"No mod inverse for {a} mod {m}")
# Define a function to encrypt a plaintext using the Hill cipher
def hill_encrypt(plaintext, key_matrix):
```

```
        # Prepare the plaintext by removing non-alphabetic characters and
padding if necessary
        chars = [...]
        non_alpha_positions = [...]
        padded = False
        if len(chars) % 2 != 0:
            chars.append('x')  # Padding for even length
            padded = True
         # Encrypt the plaintext
        cipher_chars = []
        for i in range(0, len(chars), 2):
            pair = [...]
            enc = np.dot(key_matrix, pair) % 26
            cipher_chars += [...]
        # Return the ciphertext and other necessary information
                return ''.join(cipher_chars), non_alpha_positions, padded,
len(chars)
    # Define a function to decrypt a ciphertext using the Hill cipher
    def hill_decrypt(ciphertext, non_alpha_positions, key_matrix,
padded, length):
        # Calculate the inverse of the key matrix
        det = int(round(np.linalg.det(key_matrix))) % 26
        det_inv = mod_inv(det, 26)
        # ...
        inv_matrix = [...]
       # Decrypt the ciphertext
        plain_chars = []
        for i in range(0, length, 2):
            pair = [...]
            dec = np.dot(inv_matrix, pair) % 26
            plain_chars += [...]
        # Remove padding and restore non-alphabetic characters
        if padded:
            plain_chars = plain_chars[:-1]
        decrypted_text = list(plain_chars)
        for pos, char in non_alpha_positions:
            decrypted_text.insert(pos, char)
         # Return the decrypted plaintext
        return ''.join(decrypted_text)
    # Define a function to print a text wrapped at a certain width
```

```
    def print_wrapped(text, width=58):
        # Iterate over the text in chunks of the given width
        for i in range(0, len(text), width):
            # Print the current chunk
            print(text[i:i+width])
    # Define the key matrix
    key_matrix = np.array([...])
    # Define the plaintext
    plaintext = "..."
    # Encrypt the plaintext
    ciphertext, non_alpha_positions, padded, length = hill_
encrypt(plaintext, key_matrix)
    # Print the ciphertext
    print("\nciphertext:")
    print_wrapped(ciphertext)
    # Decrypt the ciphertext
    decrypted_text = hill_decrypt(ciphertext, non_alpha_positions, key_
matrix, padded, length)
    # Print the decrypted plaintext
    print("\ndecrypted (original) plaintext:", decrypted_text)
```

3.1.1 Reduced Hill Ciphertext Paragraph with Single 2 x 2 Invertible Matrix 1

Deyeygmvyfjckoyzeqjdffhyhbcnkrowitptgtajbeyuteiwvfozpphprpzvde
egdrevqpxjubffgmgsnzcolyqmci uoqzvjwcfmbpfzusqmqqujshnnybqcoid
edlnncnseakgvjngokmrdeibqhbmyhciqcrakqrckmwsddeyeygmvsvnpxr
mfanyryuetfgnnjdffhyhbnndiozhjldbdqadedeecxabrozubhcgide
zvbpnnxrmfanyrmvnwepqsrsfxwpzsffqtqadeimolbzbdvlqmwmbeozrs
fxlkqcubojinloqmmywcdebrgmiirdincrwfhwlfbrbmqcqwptvotbnnybjn
kxbrzvlrqyjckmkrhwpxmvwxelbpajwynncrcazzulrdvjfvptdlkoben
nulgjyrgiulhoimxohbgwwxuonpozuxmeuowcbeckeqdezvcuzvvfbmmiv
jdewwstkrvsigqmdemixikacnivegknwwtbozknshgjtkxbxnxbegepwpxdx
jrbdeecwjtboztkhpwfcrmlmhbzptecgdcgxwwfgjkrhbstkrvsnnnzafcidlvf
zvfvaqeoozdidpuoshorbpmichozzwismvsikhhcdeyeegorkambqmjqhjah
ueuxaknnwpzamvakujyghwgidezvmfpetbvvhbcrhwozvsqkecjkeqbplxnv
mdrdgfcnonhwqtmfkannsdptojoronhwqgdeuscukoojorkdbqnvhwhpgw
npbheqkuhwnprtzvlhcuqmrdginprdgmgsdezvzvuoghcnbkeqqadedeqy
gtximvanhcmvhpcndewwnnbalrlyozjhzztboztjuxakinckfxxdkavrkadeeo
lhgkesdiaqepwpnpxrgwpxxrdewwfvffvfozymzvuouhjdqcqthcmvhpde

wwrdegcrnczvwpaahclhuoyzeqoghpgkngjfeduvqykavjizvfyrkdvsoztjc
lvsdueqmyhckdstdezvjpuocngeqmmuqzyujdqmdeqybmeowpkborvjde
hbcuikfmhbkqvsozmvyhhoeogtorvjdebdjngfuxmesgvsozojakfvhbdeyey
gmvwcfmbpcagjvsozumrnnpozesdemvlrsccmjdffhyhbrdgmgswfulgjyr
debpfzozdtmiullomvfmmqoldekacazzwzmvmyolgjujmjknzvymptvouo
atvjcngsnncrgidezvcaiwcurdzqjxzvdeyewfcrikhcnnezzvyrmjknwwkocr
hchpgofvaqqmkuhwbdgzakmyhcfvaqqmdeecrczvoytzdeyerdkqwfcrml
dezvfvaqqmsvnpcbxaqyrirpzvpbiwzlrpzvjfsvnpnncryzeqrfuoghkdube
iakbrinqmojmlevedojakozgmzltbmvhpppuoyzeqlhhpnnnzcohwhbjcuok
qdmxrptnnulhgqtbpbddezvuokqbetbozjdffhyhbkqbpwpmvnpbpvlqghc
yhnaffrxwffmoztflruhbztxqyegxibeozzqgtyclrqbegqahwdebpznwffmoz
oykwxwcnwfgjkrhbgjyrwfteqghckqdeyeygmvetfmozxvvryfktyffvaqwft
erxffrkozpovfilahefbpdezvbdydhwigqmplzvnnjgfxcuptwfhcdeyewfdijnl
duogjyrmfpkzvhbvyrbechcmidiozhyuofqnvjdnnwckuakoaptkadhulrqz
dvfzvwfcrclvsdueqgtnphbdecifxdreotbozdymvesbeueqmcizzlrhbowshg
imvcbxaqyfmwfedegimnvulgjtkxbbdmvcibdvjdesgdezvpokahbuhygmvy
hhwvjjdfhtvxjdemiorbpcuqmyhhwzvvjjdfhjhwfanedvjdesgdezvuezvecu
hjdfhztqmwpmfkbuhyhhwcbxaqyojfxkdmvyjllujdemvbdeqnpozyfqaore
cdezvcncbwpknbpnzcbephbcoxrjqhjahckcazzbpwpbemuqzfvkmvohpkr
hbubdezvpokawmorxrcnkqaktbozqlmwesdemiorbpkadhulrqymhbcnpk
qadelfnnzbcikagmckvjepdrwpshdiwfdecuhbcnxjnnjgvryfafusftwkywh
wcrqghwdehbskmrfhhtsgbkcuuxygmvbpdevfhbulchzvftstqqbedpptxd
xjyfdebpwcgfcuqmibvsnzcolydedlqcleptvymiorbpuohcdexnmiahorkqvi
vfzvdesgdezvckkobaqyepbdplzvptetcnivegknftwakqnchbajyfafyruxyg
gjvszvepjavfozjcoebdgwogeqkrfvkmdufhqqbeoztjoldekakodiwfdeeoyek
aygmvmvnwqymwesozwfcrimxi

3.1.B Figure 4A Frequency Distribution Graph of Hill Ciphertext for 2 x 2 Matrix 1

Note the difference between Figure 3A and Figure 4A. The homophonic cipher yields a much more uniform ciphertext character distribution than the Hill cipher using key matrix 1. Further, comparing Figure 3B with 4B, the ciphertext of the top ten trigrams is completely uniform for the homophonic cipher with respect to the spikes of the first two of the top ten trigrams for the ciphertext using key matrix 1.

Comparing Figures 4A and 5A using different key matrices, the letter E is still the most frequently occurring letter at 6.2% using key matrix 1. Using key matrix 2, the most frequently occurring letter at 6.9% is the letter O. Both of these letters are still half the expected frequency of a typical English text.

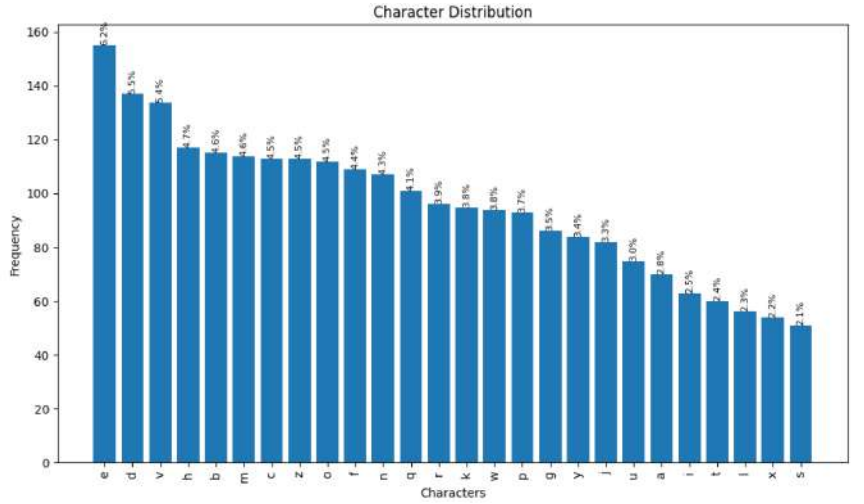

Figure 4A

Reduced Plaintext Hill Cipher Ciphertext Frequency Distribution for Key Matrix 1. This frequency distribution shows a range of letter frequency percentage between 6.2% to 2.1% which somewhat deviates from the typical distribution of English letter frequency percentage noted above in Figure 2A.

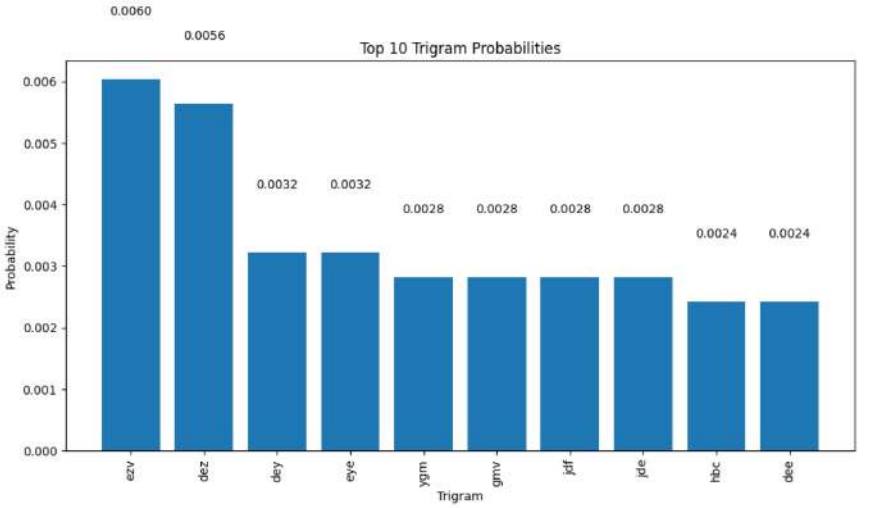

Figure 4B

Reduced Plaintext Hill Cipher Trigram Ciphertext Probability Frequency Distribution Key Matrix 1.

3.1.C Figure 4B of Frequency Probability Distribution Graph of Hill Ciphertext Trigram with Key Matrix 1

For comparison, the key matrix used in the Hill Cipher can have an effect on the ciphertext frequency distribution as illustrated in Figure 5 using the key matrix 2 as $\begin{bmatrix} 9 & 2 \\ 1 & 15 \end{bmatrix}$. The ciphertext and frequence distribution are given below.

3.2 *Reduced Hill Ciphertext Paragraph with Single 2 x 2 Invertible Matrix 2*

Dugayefzppxoqqndwwgdqrjykrrvrjswffghbxdfzmugjggywdtpwzgtahsz
dukcqhptldktxrqrqwsielakrokkyykexxglkobsuvofumkkusrhvvanfhyqw
uduovanrvucmutbsxiaymwjcgdkkrgstgembdmudhqcoolndugayefzrxsr
ejrtnntnugxpzgangdqrjykrannctpehklqxgmduduauhomztpxrtggoduszu
vanejrtnntnfzdsnhmwbawbxbrgqrvlgmduuexngpqxypkkwizmtpbawbv
myqxrfjdtfukkgskodumzqwkwwjdtbdlhrucpmztcyqweghziinanfhsxtv
mzszgnoixoymrjruqpfzrrdzuvdfaganbdeichjlwjglexghovqqzmanjlpdtng
ojlxeuedqkrcqrrkesrtpnjiekekozmqcwwduszcwszwdtcsmglduicztrjjqssk
kdusmbeuorvxjkchbicintphbvvpdlsqdubqdkcnhxbihktefduauvpintplsgt
lhbdtfjxgpghaunrguxglhpdrjkrztrjjqaneltxyyovwdszexogestpncydkevvz
zuvsmpjtpbowqfzekfptgdugakczzuohlkktqehlbyknjmuanxblwfzmurhye
rugoduszrtbqinkjkrbdrutpjqsgaurewwuvizudzpwjfvrvprruvlrtuoanlng
hfjzzprruiyduumcwqqfjzzvhdkudrugtcqsrafwwscrusrkpszutcwkkwjgos
rwjqwsiduszszkexzrvbywwgmduduoibxbefznntgfzgtrvduicanpegnrotp
qlchintpcdnjmudtqcwbihuoabuoduesutysoancognhxbsrejcqqpejduicexq
rwdtpaqszkezdgdyqvltgfzgtduicwjkcbdfeszxbaatgutkendwweqgtyspm
yhjjvfoiuoglhrwdtnvhjqtpcdzrjqrawwgstgvhztduszkbkervwgkkwkxxug
gdkkduoitcesxbddzzgldukrcwcabskriujqtpfzhtxeesbxzzgldуqxsxfvnjie
mgjqtpfjmuexkrdugayefzkobsuveipdjqtpsaidsrtpoadufzgncyiggdqrjykr
wjqwsilhjlpdtnduuvoftpilsmjlfufzbsmcxnduuoeichjvfzgsxnpdrhbbhbsza
qghzikepzglrvsianbdgoduszeigycwwjzcerszdugalhbdcatganzbsztnbbhb
icqqbdtggtiaexogkkscruqxdjmugstgexogkkduaunuszkaqjdugawjiulhbdt
fduszexogkkrxsrnxhooipgahszaxgygfahszyhrxsranbdndwwonkexzvhxr
cgmumzdtkkfjtfptjjfjmutpqwgfinfzgtwzkendwwutgtanelakrukrxokeiux
kejghanjldovluvqxduszkeiuzmintpgdqrjykriuuvxbfzsruvypiytghtnaqru
xlhbstpsvgnzdgpyfoikcbezmtpzcbxowgnpbkcgmruduuvyjlhbstpkakgxg
rvlhpdrjkrpdtnlhjgiytgiudugayefzxpbstporabppjnppexoglhjguxqrhktpn
kwdlplbbnuvduszqxxlrusskkmrszanhwwbcwghlhtgdugalhncsxklkepdt
nrtdcszkrlcefautgsmnctpjykelaudgdankoscmuceghuoenjljwmpwdszlhb
dzrjqrawwbxsrkrduyywbqhesintpbifzoazmykkkyychgnkrswvvgofznxh
ooibslhjjkcueudjlpdlsqdqxfzyyqxgldumgdusznkuokrzdyefzhtruglgdivg

bktdusmzzuvcwkkhtruszglgdivqllhnnjjgldumgduszykszauzdgdivavkk
xbrtddzdhtrunxhooifjwbvhfzzxebrhdufzqxwwsrtpppgmzzauduszrvnxx
bhbuvelnxnhkrakejtqehlbqceichuvxbzmwkxxexymzigtrjkrxrdusznkuow
izzejrviumuintpbvoooadusmzzuvuoenjljwaqkrrvdcgmducpanulyyuoq
wqcglnhqhxbvvnclhducwkrrvktanhwabpptxummteemkrubdiyrudukrw
ovrivqbmgbycwnjyefzuvduwdkrjlpjszmtztuszmydghihktppduuvkofvc
wkkzvjqelakroduovyqtaghlcsmzzuvketgduubsmlbzziuxwwdszdumgd
uszqcqqpeoinhqxmrszghxprvxjkchbmtskiufekrdfpptxtnnjyepdjqsznhfk
wdtpxommqxcqeqwwrjexymraivuszmtpcdxnduuoqqnclhduesgauoyefz
fzdsoioooatplhbduebe

3.2.B Figure 5A Frequency Distribution Graph of Hill Ciphertext for 2 x 2 Matrix 2

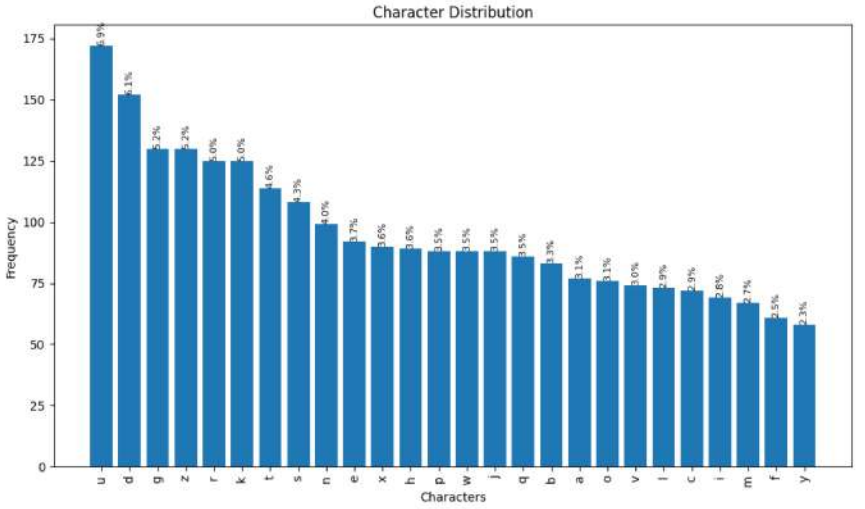

Figure 5A

Reduced Plaintext Hill Cipher Ciphertext Probability Frequency Distribution using a different key matrix 2 from Figure 4A. This frequency distribution shows a range of letter frequency percentage between 6.9% to 2.3% which somewhat deviates from both the typical distribution of English letter frequency percentage noted above in Figure 2 and the frequency distribution generated by the key matrix from Figure 4A.

3.2.C Figure 5B Frequency Distribution Graph of Hill Ciphertext Trigrams for 2 x 2 Matrix 2

Comparing Figure 2A, Figure 4A, and Figure 5A, both key matrices of the Hill Cipher do deviate in their respective ciphertext letter frequency

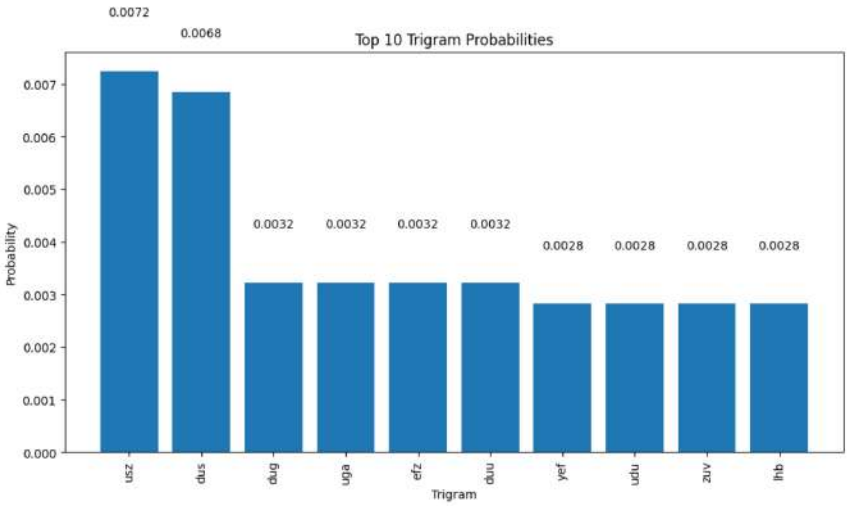

Figure 5B

Reduced Plaintext Hill Cipher Ciphertext Probability Frequency Distribution key matrix 2. Both top ten ciphertext trigrams using key matrix 1 compared to using key matrix 2 each have two trigrams with probabilities above 0.0032 and the other eight trigrams of the 10 most frequently occurring are 0.0032 to 0.0028.

distributions from the general letter frequency distribution of English. Both Figure 4A and Figure 5A indicate a gradual decline in contrast with Figure 2A which has a steep decline in letter frequency. Comparing Figure 2B, Figure 4B, and Figure 5B, the top ten trigrams using key matrix 1 are similar but not exact with respect to the trigram frequency distribution. Figure 2B indicates a somewhat gradual decline in the trigram probability distribution in contrast with the spike in the first two trigram probabilities followed by a more uniform trigram probability frequency distribution.

Trigrams were presented for the ciphertext of Hill's original cipher method for comparison of trigrams of more complicated ciphers in sections 4.1, 5.1, and 6.1.

4. Introducing Pseudo-Homophonic Cipher and Trigram Ciphertext Techniques into the Hill Cipher

In the sections below, as in [10] and [11], the Hill Enigma Ciphers use rotating 2 x 2 matrices (which shift their elements 90 degrees after encrypting each block). This creates a pseudo-homophonic effect: each plaintext block produces different ciphertext for the same plaintext character, with the pattern

repeating only after a number of blocks equal to the least common multiple of the number of matrices and the plaintext length. As the elements of the invertible 2 x 2 matrices rotate, new invertible key matrices are created by the rotation of matrix elements and in conjunction with initializing vectors raised to the ith power mod 29 for each block i the same plaintext characters encrypt to different ciphertext characters. This leads to a pseudo-homophonic cipher effect. Additionally, in this variation of the Hill Cipher scheme, each ciphertext output is encrypted with a trigram per plaintext character. N-gram analysis indicate the probability that a word, token, or character follows a known word, token, or character [13]. In plaintext, n – gram analysis matters to disambiguate. In ciphertext, n – gram analysis can be used as a cryptanalysis tool. The remaining sections, 4.1 – 6.1 each give an example of the Python pseudo-code, the ciphertext, the ciphertext frequency distribution, and the trigram probability frequency distribution for comparison. In section 4.1, four rotating 2 x 2 invertible matrices are used in sequence, along with initialization vectors raised to the ith power mod 29 for each block. This causes the same plaintext to encrypt to different trigrams each time. In section 5.1, four rotating 2 x 2 matrices feed into two 3 x 3 matrices, along with initialization vectors raised to the ith power mod 29 for each block. This causes the same plaintext to encrypt to different trigram ciphertext each time. In section 6.1, four rotating 2 x 2 matrices feed into two 3 x 3 matrices, which then feed into a 6 x 6 matrix, along with initialization vectors raised to the ith power mod 29 for each block. This causes the same plaintext to encrypt to different trigram ciphertext each time.

4.1 *Modified Original Hill Enigma with Four Rotating 2 x 2 Invertible Matrices*

Two papers in 2020 [10] and 2024 [11] described enhanced versions of the Hill Cipher referred to as *Hill Enigma Ciphers*. These improvements made the Hill Cipher stronger by:

1. **Rotating matrix elements** – The 2×2 matrices used for encryption rotated their elements 90 degrees after processing each block of text, similar to how German Enigma Encoding Machines worked.
2. **Expanding character support** – By using modulo 29 arithmetic instead of modulo 26, the plaintext could encrypt spaces, periods, and commas, in addition to English letters and digits, making encrypted messages more difficult to decrypt than using only English letters.
3. **Using multiple matrices** – Four invertible matrices were arranged in sequence, with an initializing vector applied each plaintext character block in both [10] and [11].

4. **Pseudo-homophonic techniques** – By rotating the elements of the 2 x 2 matrices after each block passes through, no character is encrypted with the same ciphertext character up to a point depending on the dimension of the matrices used and length of plaintext.
5. **Increasing Range to include ASCII ISO – 8858 – 1 Characters –** Two papers in 2024 [11] and [12] suggested using ASCII characters for the ciphertext (the encrypted output) to make unauthorized decryption more difficult.
6. **Key difference from standard Hill Cipher:** Instead of using fixed matrices, these matrices "rotate" like the rotors in an Enigma machine, changing how they encrypt as the plaintext message progresses giving the appearance of a pseudo-homophonic effect of using multiple ciphertext characters to encrypt plaintext characters.

In [10], an initializing vector was applied to each block prior to multiplication with the first of four matrices. In total, the four invertible matrices were organized in series with a rotation scheme similar to a German Enigma Machine [5] in which the matrices rotate matrix element values in a prescribed manner. Spaces, period, and comma were allowed as plaintext characters in [10], so modulo 29 was used. The following example will also use spaces, period, and comma for the original plaintext, except the one place in the original plaintext where the semi-colon " ; " was changed to a comma " , ". In this new variation of [10], the novel use of a trigram composed of three ciphertext characters mapping to each plaintext character, the example here of the modified original plaintext is from the first sentence due to the length of the generated ciphertext. Using initializing vectors applied to each block of plaintext and then multiplied four 2 x 2 matrices series in which the elements of each matrix is rotated one position clockwise, each modified original plaintext character is mapped to a trigram ciphertext from the possible range of ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9. The initializing vectors are applied to each matrix output in prior to being applied as input to the matrix (via the operation of vector-matrix multiplication). In order to implement this version of the Hill Cipher [10], all punctuation of the original plaintext was removed other than the space, period, and comma. Using the entire modified original plaintext, the frequency distribution of the most frequently occurring ciphertext characters are given in Figure 6A and a probability frequency distribution of trigrams is given in Figure 6B.

4.1.A Python Pseudo-code for Hill Cipher with Four Rotating 2 x 2 Invertible Matrices

```
# Define constants and functions
domain_chars = [...]
modulus = 29
ascii_printable = (...)
latin1_entities = (...)
cipher_domain = [...]
triplet_count = len(domain_chars)
# Generate random triplets
random.seed(42)
triplets = [...]
triplet_to_num = {...}
num_to_triplet = {...}
symbol_to_num = {...}
num_to_symbol = {...}
# Define functions for matrix operations
def is_invertible(mat, mod):
   # Check if the matrix is invertible
def random_invertible_matrix(mod):
   # Generate a random invertible matrix
def mod_exp_vector(vec, power, mod):
   # Calculate the modular exponentiation of a vector
def add_pairs(pair1, pair2):
   # Add two pairs of numbers modulo the modulus
def matrix_mod_inv(mat, mod):
   # Calculate the modular inverse of a matrix
def rotate_matrix(mat, n):
   # Rotate a matrix 90 degrees clockwise n times
def next_rot_state(state):
   # Advance the rotor state
# Define the encryption function
def encrypt(plaintext):
   # Convert the plaintext to numbers
   nums = [...]
  # Pad the plaintext if necessary
   if len(nums) % 2 != 0:
      nums.append(symbol_to_num[' '])
   # Encrypt the plaintext
```

```
        enc_nums = []
        state = [0, 0, 0, 0]
        for pair_idx in range(0, len(nums), 2):
            pair = np.array(nums[pair_idx:pair_idx+2])
            power_i = pair_idx // 2 + 1
            rotations = state.copy()
            cur_pair = pair.copy()
            for j in range(4):
                mat = rotate_matrix(matrices[j], rotations[j])
                cur_pair = np.dot(mat, cur_pair) % modulus
                vec = mod_exp_vector(vectors[j], power_i, modulus)
                cur_pair = add_pairs(cur_pair, vec)
            enc_nums.extend(cur_pair)
            state = next_rot_state(state)
        # Apply exclusive-or with the ith power of the vector
        xor_vec = np.array([2, 11])
        power_i = 1
        xor_vec_block = np.array([...])
        xor_vec_repeated = np.tile(xor_vec_block, len(enc_nums) // 2)
        if len(enc_nums) % 2 != 0:
            xor_vec_repeated = np.concatenate((xor_vec_repeated, xor_vec_
block[:1]))
        xor_result = (np.array(enc_nums) ^ xor_vec_repeated) % triplet_count
        # Map the output to the triplet character set
        ciphertext = ''.join(num_to_triplet[int(num) % triplet_count] for num
in xor_result)
        return ciphertext
    # Main program
    if __name__ == "__main__":
        # Print allowed plaintext characters
        allowed = "'a'-'z', '0'-'9', space, comma, period"
        print(f"Allowed plaintext: {allowed}")
        print("Ciphertext triplets: random fixed mapping from printable ASCII
+ ISO-8859-1 domain")
        # Get user input
        user_plaintext = input("Enter plaintext to encrypt: ")
        # Check for invalid characters
        invalid_chars = [ch for ch in user_plaintext.lower() if ch not in symbol_
to_num]
        if invalid_chars:
```

```
 print(f"Error: Invalid characters in input: {sorted(set(invalid_chars))}")
else:
   # Encrypt the plaintext
   ciphertext = encrypt(user_plaintext)
# Print the ciphertext
   print("Ciphertext:")
   for i in range(0, len(ciphertext), 58):
      print(ciphertext[i:i+58])
```

4.1.1 Modified Original Hill Enigma Ciphertext Paragraph with Four Rotating 2 x 2 Invertible Matrices

For a simple example in consideration of space regarding the length of this manuscript, a single sentence of the modified original plaintext is given here. Thus, because each plaintext character is mapped to a trigram ciphertext, reproduction of the complete ciphertext of the original modified plaintext is space-prohibitive. The reader can enter the modified original plaintext into the python code to see the complete ciphertext output.

Modified Original Plaintext: *They were not railway children to begin with.*

Ciphertext generated from the above sentence:

Æ`=p,h4¸Ø»$TÆ`=RO¬ß·å2â´úhªR°ì©J±%E½SHõp,hHRóÌ?hÛ!ÚHRó%I¡ R°ìë0p4¸ØJÍÂÛÌ§KW/¼µ@÷`1ß·åRO¬R°ìSHõë0pÛ!ÚL3Uúhªºef%E½Æ`=úhª L3U¯Ç(Ì?hHRókïw%E½R°ì

4.1.B Figure 6A Frequency Distribution Graph of Modified Original Plaintext using Hill Enigma Ciphertext with Four Rotating 2 x 2 Invertible Matrices

Comparing Figures 5A and 6A, there are two distinguishing differences in these two encryption schemes. First, of the ciphertext characters in Figure 6A, all but the first seven have a frequency distribution smaller in percentage than all of the percentages in the frequency distribution of Figure 5A, meaning there is a greater distribution spread. Second, the range of the ciphertext in Figure 6A is much larger due to the use of both uppercase and lowercase letters, digits, and the inclusion of the ASCII ISO-8859-1 character set. This greatly improves diffusion of the plaintext.

4.1.C Figure 6B Probability Frequency Distribution Graph of Hill Enigma Ciphertext Trigrams using Four Rotating 2 x 2 Invertible Matrices

Compared to Figure 4B and 5B with spikes in the first two trigrams of both figures, Figure 6B has a nearly uniform trigram probability distribution which indicates the effect of a static matrix compared to rotating matrices.

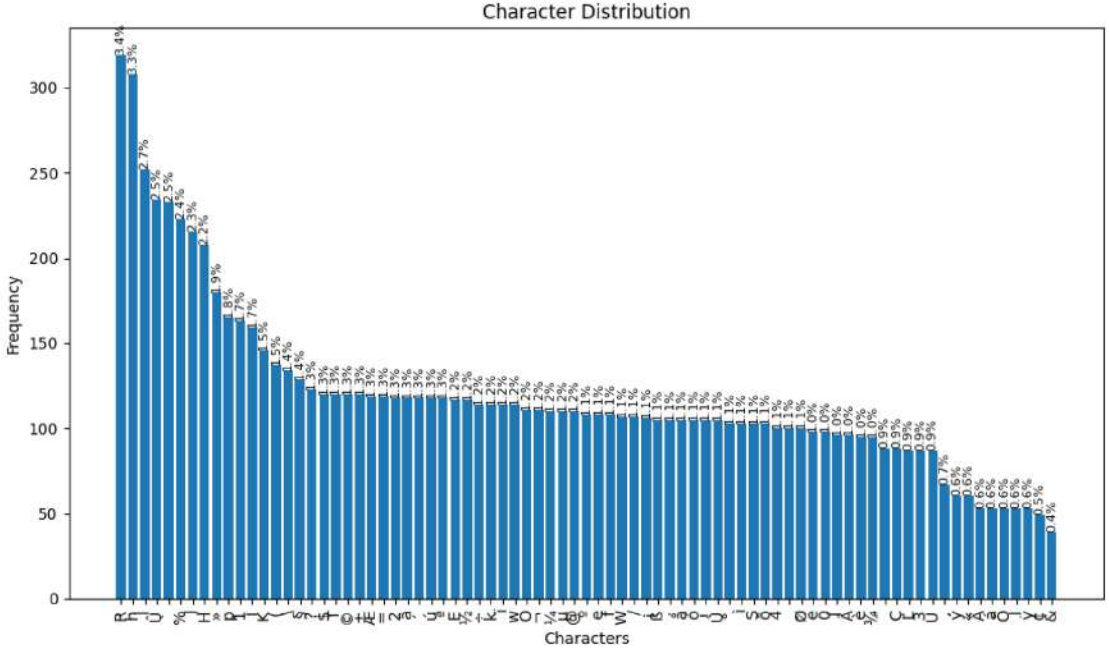

Figure 6A

Modified Original Plaintext Hill Cipher Ciphertext Frequency Distribution using four rotating 2 x 2 invertible matrices in series. This frequency distribution shows a range of letter frequency percentage between 3.4% to 0.4% of the most frequently occurring ciphertext characters from the possible range of ciphertext characters ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9. Contrast this with Figure 1 in which the first 11 characters have a frequency greater the two most frequently occurring ciphertext characters in Figure 6A.

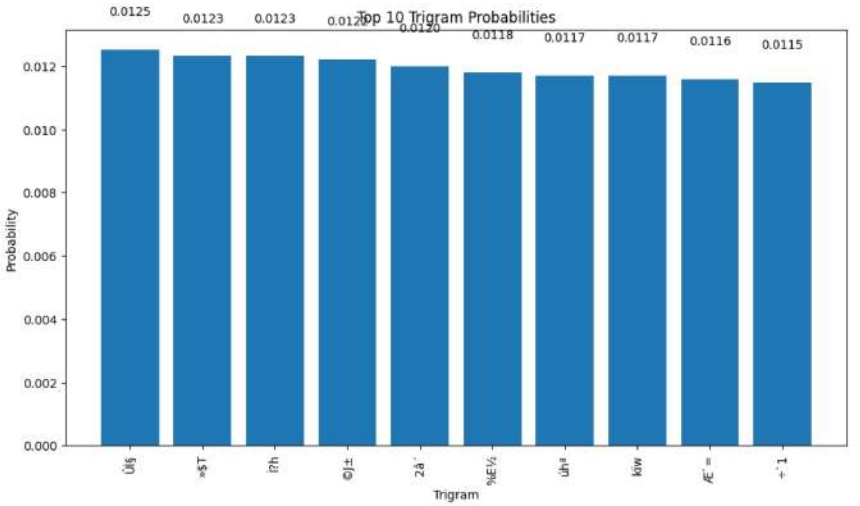

Figure 6B

Modified Original Plaintext Hill Cipher Ciphertext Probability Frequency Distribution of Trigrams using four rotating 2 x 2 invertible matrices in series. This probability frequency distribution of trigrams shows a probability frequency percentage between 0.0125 to 0.0115 of the ten most frequently occurring ciphertext trigrams from the possible range of ciphertext characters ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9.

5.1 *Modified Original Hill using Four Rotating 2 x 2 Invertible Matrices and Two 3x3 Matrices*

1. **Rotating matrix elements** – The 2 x 2 invertible matrices used for encryption rotated their elements 90 degrees after processing each block of text, similar to how German Enigma Encoding Machines worked.
2. **Higher – Dimension Matrices –** Two 3 x 3 invertible matrices accept input from the four rotating 2 x 2 matrices increasing diffusion of the plaintext.
3. **Expanding character support** – By using modulo 29 arithmetic instead of modulo 26, the plaintext could encrypt spaces, periods, and commas, in addition to English letters and digits, making encrypted messages more difficult to decrypt than using only English letters.
4. **Using multiple matrices** – Four invertible matrices were arranged in sequence, with an initializing vector applied each plaintext character block in both [10] and [11].

5. **Pseudo-homophonic techniques** – By rotating the elements of the 2 x 2 matrices after each block passes through, no character is encrypted with the same ciphertext character up to a point depending on the dimension of the matrices used.
6. **Increasing Range to include ASCII ISO – 8858 – 1 Characters –** Two papers in 2024 [11] and [12] suggested using ASCII characters for the ciphertext (the encrypted output) to make unauthorized decryption more difficult.
7. **Key difference from standard Hill Cipher:** Instead of using fixed matrices, these matrices "rotate" like the rotors in an Enigma machine, changing how they encrypt as the plaintext message progresses giving the appearance of a pseudo-homophonic effect of using multiple ciphertext characters to encrypt plaintext characters. In addition, following [11], multi – dimensional matrices are introduced into the encryption scheme.

In this variation of [10] and [11], four invertible 2 x 2 matrices in series with initialization vectors applied to each block prior to input to each matrix are fed into two invertible 3 x 3 matrices. The elements of each 2 x 2 matrix are rotated one position clockwise, each modified original plaintext character is mapped to a trigram ciphertext from the possible range of ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9. In order to implement a multi-dimensional version of rotating matrices of the Hill Cipher in [11], all punctuation of the original plaintext was removed other than the space, period, and comma. The novel use of three ciphertext characters, a trigram, are mapped to each plaintext character. The example ciphertext reproduced here is from the first sentence of the modified original plaintext . The frequence distribution in Figure 7A of the most frequently occurring ciphertext characters is from the fully encrypted modified original plaintext and a probability frequency distribution of trigrams is given in Figure 7B.

5.1.A Python Pseudo-code for Hill Cipher with Four Rotating 2 x 2 Invertible Matrices and Two 3 x 3 Matrices

```
# Define constants and functions
domain_chars = [...]
modulus = 29
ascii_printable = (...)
latin1_entities = (...)
```

```
cipher_domain = [...]
triplet_count = len(domain_chars)
# Generate random triplets
random.seed(42)
triplets = [...]
triplet_to_num = {...}
num_to_triplet = {...}
symbol_to_num = {...}
num_to_symbol = {...}
# Define matrices and vectors
matrices = [...]
vectors = [...]
# Define functions for matrix operations
def mod_exp_vector(vec, power, modulus):
    # Calculate the modular exponentiation of a vector
def add_pairs(pair1, pair2):
    # Add two pairs of numbers modulo the modulus
def matrix_mod_inv(mat, modulus):
    # Calculate the modular inverse of a matrix
def rotate_matrix(mat, n):
    # Rotate a matrix 90 degrees clockwise n times
def next_rot_state(state):
    # Advance the rotor state
def matrix_mod_inv_3x3(mat, modulus):
    # Calculate the modular inverse of a 3x3 matrix
# Define the encryption function
def encrypt(plaintext):
    # Convert the plaintext to numbers
    nums = [...]
    # Pad the plaintext if necessary
    if len(nums) % 2 != 0:
        nums.append(symbol_to_num[' '])
  # Encrypt the plaintext
    enc_nums = []
    state = [0, 0, 0, 0]
    for i in range(0, len(nums), 2):
        pair = np.array(nums[i:i+2])
        power_i = i // 2 + 1
        rotations = state.copy()
        for j in range(4):
```

```
                mat = rotate_matrix(matrices[j], rotations[j])
                pair = np.dot(mat, pair) % modulus
                vec = mod_exp_vector(vectors[j], power_i, modulus)
                pair = add_pairs(pair, vec)
            enc_nums.extend(pair)
            state = next_rot_state(state)
        # Apply 3x3 matrix multiplication
        enc_nums_3x1 = [...]
        A = np.array([...])
        B = np.array([...])
        result_3x1 = [...]
        combined = np.concatenate(result_3x1)
        # Apply XOR operation. The same effect can be accomplished by adding
vectors
        xor_vec = np.array([...])
        power_i = 1
        xor_vec_exp = np.array([...])
        # Map the output to the triplet character set
        ciphertext = ''.join(num_to_triplet[int(num) % triplet_count] for num
in xor_result)
        return ciphertext, original_length
    # Define the decryption function
    def decrypt(ciphertext, original_length):
        # Convert the ciphertext to numbers
        nums = [...]
      # Apply inverse XOR operation
        xor_vec = np.array([...])
        power_i = 1
        xor_vec_exp = np.array([...])
        # Apply inverse 3x3 matrix multiplication
        A = np.array([...])
        B = np.array([...])
        invA = matrix_mod_inv_3x3(A, modulus)
        invB = matrix_mod_inv_3x3(B, modulus)
        nums_2x1 = [...]
        for i in range(0, len(nums), 3):
        # Decrypt the ciphertext
        dec_nums = []
        state = [0, 0, 0, 0]
        pair_vectors = [...]
```

```
    for pair_idx, pair in enumerate(pair_vectors):
        # ...
    # Map the output to the original character set
    plaintext = ''.join(num_to_symbol[int(x)] for x in dec_nums)
    plaintext = plaintext[:original_length]
    return plaintext
# Main program
if __name__ == "__main__":
    # Get user input
    plaintext = input("Enter plaintext: ")
 # Check for invalid characters
    invalid = [...]
    if invalid:
        print("Error: Invalid characters")
        return
   # Encrypt the plaintext
    ciphertext, orig_len = encrypt(plaintext)
   # Print the ciphertext
    print("Ciphertext:")
    for i in range(0, len(ciphertext), 58):
        print(ciphertext[i:i+58])
    # Decrypt the ciphertext
    decrypted = decrypt(ciphertext, orig_len)
  # Print the decrypted plaintext
    print("Decrypted plaintext:")
    print(decrypted)
```

5.1.1 Modified Original Hill Enigma Ciphertext Paragraph with Four Rotating 2 x 2 Invertible Matrices and Two 3 x 3 Matrices

Modified Original Plaintext: *They were not railway children to begin with.*

Ciphertext generated from the above sentence:

K&\SHõß·åúhªHRó2â´R°ìúhªý»«è\¾è\¾%E½÷`1Ì?hºefÛ!Ú%E½L3U2â´ú

hªè\¾è\¾è\¾%E½K&\©J±ºefHRóSHõ%I¡RO¬¼µ@2â´kïwRO¬ÃáÕSHõè\¾¼µ

@Æ`=úhªÛ!ÚL3UÆ`=JÍÂKW/2â´úhªRO¬Æ`=©J±R°ìK&\R°ìÌ?hKW/©J±úhª

KW/¯Ç(ß·åR°ìSHõ»$Tß·åß·åRO¬ë0pÌ?h

5.1.B Figure 6A Frequency Distribution Graph of Hill Enigma Ciphertext Four Rotating 2 x 2 Invertible Matrices and Two 3 x 3 Matrices

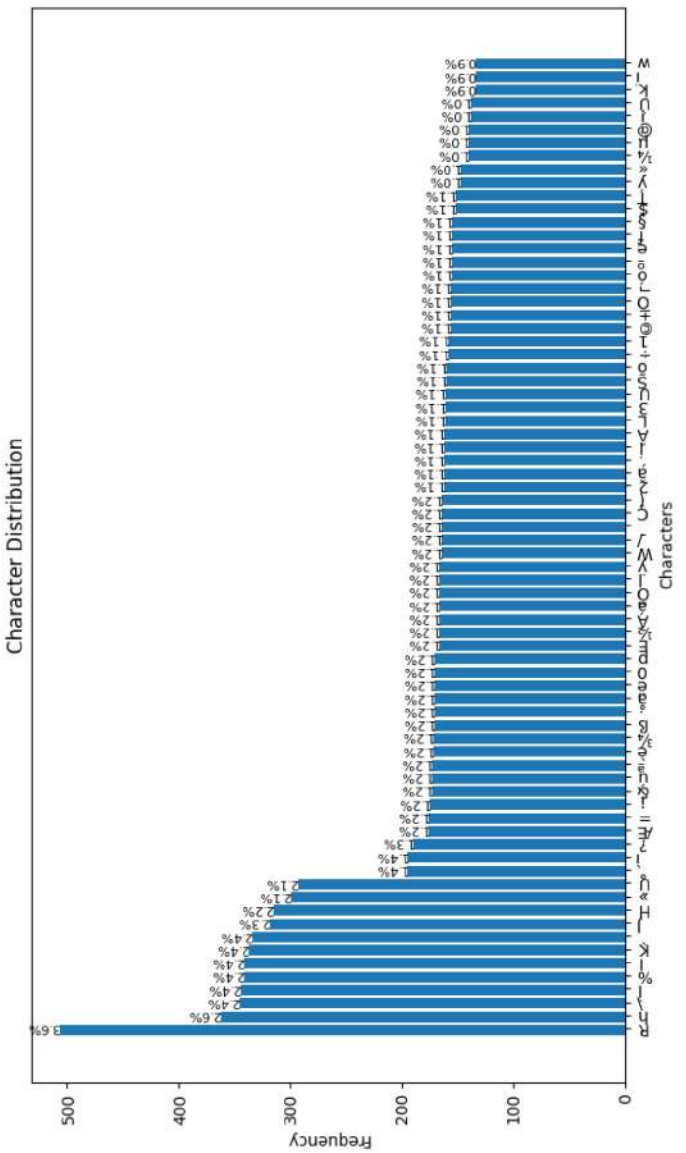

Figure 7A

Modified Original Plaintext Hill Cipher Ciphertext Frequency Distribution using four rotating invertible matrices in series. This frequency distribution shows a range of letter frequency percentage between 3.6% to 0.9% of the most frequently occurring ciphertext characters from the possible range of ciphertext characters ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9.

Comparing Figures 6A and 7A, after the 12th ciphertext character in Figure 7, the remaining most frequently occurring ciphertext characters are nearly uniform in distribution from a frequency of 1.4% to 0.9%. This differs from the distribution of Figure 6 above with the four invertible 2 x 2 matrices in series in which there is a gradual decline in the frequency distribution of the first 16 ciphertext characters until a more uniform frequency distribution of the remaining most frequently occurring characters.

5.1.C Figure 7B Probability Frequency Distribution Graph of Hill Enigma Ciphertext Trigrams using Four Rotating 2 x 2 Invertible Matrices and Two 3 x 3 Matrices

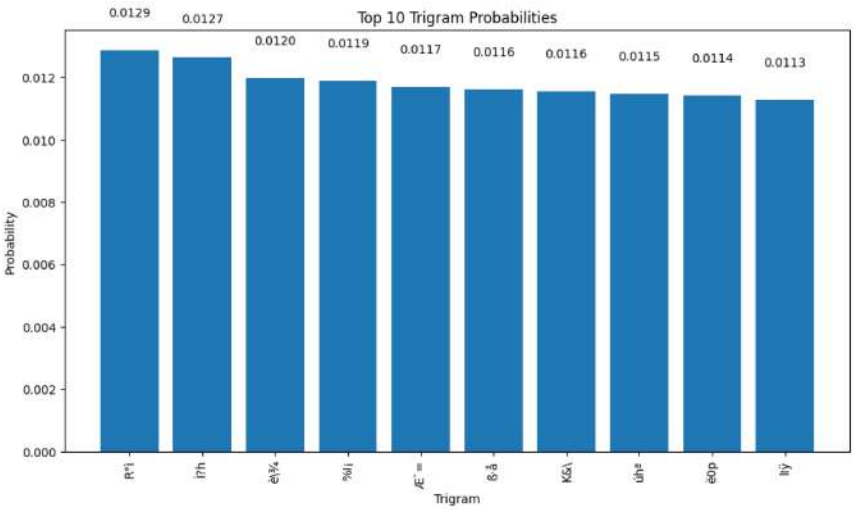

Figure 7B

Modified Original Plaintext Hill Cipher Ciphertext Probability Frequency Distribution of Trigrams using four rotating 2 x 2 invertible matrices in series and two 3 x 3 invertible matrices. This frequency distribution shows a range of letter frequency percentage between 3.4% to 0.4% of the most frequently occurring ciphertext characters from the possible range of ciphertext characters ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9.

Comparing Figure 6B and Figure 7B, in Figure 6B the difference between the probability of the most frequently occurring trigram (0.0125) and the probability of the tenth most frequently occurring trigram (0.0115) is 0.001. In Figure 7B, the difference between the probability of the most frequently occurring trigram (0.0129) and the probability of the tenth most frequently

occurring trigram (0.0113) is 0.0016. There is a small variation between the probability of the trigrams in the two figures.

6.1 *Modified Original Hill Enigma using Four Rotating 2 x 2 Invertible Matrices, Two 3 x 3 Matrices, and One 6 x 6 Matrix*

1. **Rotating matrix elements** – The 2 x 2 invertible matrices used for encryption rotated their elements 90 degrees after processing each block of text, similar to how German Enigma Encoding Machines worked.
2. **Higher – Dimension Matrices –** One 6 x 6 invertible matrix accepts input from two 3 x 3 invertible matrices accept input from the four rotating 2 x 2 matrices increasing diffusion of the plaintext.
3. **Expanding character support** – By using modulo 29 arithmetic instead of modulo 26, the plaintext could encrypt spaces, periods, and commas, in addition to English letters and digits, making encrypted messages more difficult to decrypt than using only English letters.
4. **Using multiple matrices** – Four invertible matrices were arranged in sequence, with an initializing vector applied each plaintext character block in both [10] and [11].
5. **Pseudo-homophonic techniques** – By rotating the elements of the 2 x 2 matrices after each block passes through, no character is encrypted with the same ciphertext character up to a point depending on the dimension of the matrices used.
6. **Increasing Range to include ASCII ISO – 8858 – 1 Characters –** Two papers in 2024 [11] and [12] suggested using ASCII characters for the ciphertext (the encrypted output) to make unauthorized decryption more difficult.
7. **Key difference from standard Hill Cipher:** Instead of using fixed matrices, these matrices "rotate" like the rotors in an Enigma machine, changing how they encrypt as the plaintext message progresses giving the appearance of a pseudo-homophonic effect of using multiple ciphertext characters to encrypt plaintext characters. In addition, following [11], multi – dimensional matrices are introduced into the encryption scheme.

In this variation, in this new variation of [10] and [11] using four invertible 2 x 2 matrices in series which feed into two invertible 3 x 3 matrices which feed into a single 6 x 6 invertible matrix which the elements of each 2 x 2 matrix is

rotated one position clockwise, each modified original plaintext character is mapped to a trigram ciphertext from the possible range of ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9. In order to implement a multi-dimensional version of rotating matrices of the Hill Cipher in [11], all punctuation of the original plaintext was removed other than the space, period, and comma. The novel use of three ciphertext characters mapping to each plaintext character in this version. The example here of the modified original plaintext is from the first sentence for brevity of the length of the ciphertext due to the block size of a 6-dimensional vector. Each ciphertext character of the output vector maps to a trigram. The frequence distribution in Figure 8A of the most frequently occurring ciphertext characters is from the fully encrypted modified original plaintext and a probability frequency distribution of trigrams is given in Figure 8B.

6.1.A Python Pseudo-code for Hill Enigma Cipher with Four Rotating 2 x 2 Invertible Matrices and Two 3x3 Matrices and One 6 x 6 Matrix.

```
# Define constants and functions
domain_chars = [...]
modulus = 29
ascii_printable = (...)
latin1_entities = (...)
cipher_domain = [...]
triplet_count = len(domain_chars)
# Generate random triplets
random.seed(42)
triplets = [...]
triplet_to_num = {...}
num_to_triplet = {...}
symbol_to_num = {...}
num_to_symbol = {...}
# Define matrices and vectors
matrices = [...]
vectors = [...]
# Define functions for matrix operations
def mod_exp_vector(vec, power, modulus):
   # Calculate the modular exponentiation of a vector
def add_pairs(pair1, pair2):
   # Add two pairs of numbers modulo the modulus
def matrix_mod_inv(mat, modulus):
```

```
    # Calculate the modular inverse of a matrix
def rotate_matrix(mat, n):
    # Rotate a matrix 90 degrees clockwise n times
def next_rot_state(state):
    # Advance the rotor state
def matrix_mod_inv_general(mat, modulus):
   # Calculate the modular inverse of a matrix using Gaussian elimination
def matrix_mod_inv_3x3(mat, modulus):
    # Calculate the modular inverse of a 3x3 matrix
# Define the encryption function
def encrypt(plaintext):
    # Convert the plaintext to numbers
    nums = [...]
    original_length = len(nums)
    if len(nums) % 2 != 0:
        nums.append(symbol_to_num[' '])
   # Apply the Hill cipher and other transformations
    enc_nums = [...]
    state = [0, 0, 0, 0]
    for i in range(0, len(nums), 2):
        pair = np.array(nums[i:i + 2])
        power_i = i // 2 + 1
        rotations = state.copy()
        for j in range(4):
            mat = rotate_matrix(matrices[j], rotations[j])
            pair = np.dot(mat, pair) % modulus
            vec = mod_exp_vector(vectors[j], power_i, modulus)
            pair = add_pairs(pair, vec)
        enc_nums.extend(pair)
        state = next_rot_state(state)
    # Apply 3x3 matrix multiplication
    enc_nums_3x1 = [...]
    A = np.array([...])
    B = np.array([...])
    result_3x1 = [...]
    combined = np.concatenate(result_3x1)
    # Apply XOR operation
    xor_vec = np.array([...])
    power_i = 1
    xor_vec_exp = np.array([...])
```

```
        xor_vec_repeated = np.tile(xor_vec_exp, len(combined) // len(xor_
vec_exp))
        if len(combined) % len(xor_vec_exp) != 0:
            xor_vec_repeated = np.concatenate((xor_vec_repeated, xor_vec_
exp[:len(combined) % len(xor_vec_exp)]))
        xor_result = (combined + xor_vec_repeated) % modulus
        # Apply 6x6 matrix multiplication
        matrix_6x6 = np.array([...])
        pad_len = (6 - len(xor_result) % 6) % 6
        xor_result_padded = np.concatenate((xor_result, np.zeros(pad_len,
dtype=int)))
        result_6x1 = []
        for i in range(0, len(xor_result_padded), 6):
            vec = np.dot(matrix_6x6, xor_result_padded[i:i + 6]) % modulus
            result_6x1.extend(vec)
        # Return the ciphertext
        ciphertext = ''.join(num_to_triplet[int(num) % triplet_count] for num
in result_6x1)
        return ciphertext, original_length
    # Define the decryption function
    def decrypt(ciphertext, original_length):
        # Convert the ciphertext to numbers
        nums = [...]
        # Apply inverse 6x6 matrix multiplication
        matrix_6x6 = np.array([...])
        inv_matrix_6x6 = matrix_mod_inv_general(matrix_6x6, modulus)
        nums_6x1 = []
        for i in range(0, len(nums), 6):
            vec = np.dot(inv_matrix_6x6, nums[i:i + 6]) % modulus
            nums_6x1.extend(vec)
        # Apply inverse XOR operation
        xor_vec = np.array([...])
        power_i = 1
        xor_vec_exp = np.array([...])
        xor_vec_repeated = np.tile(xor_vec_exp, len(nums_6x1) // len(xor_
vec_exp))
        if len(nums_6x1) % len(xor_vec_exp) != 0:
            xor_vec_repeated = np.concatenate((xor_vec_repeated, xor_vec_
exp[:len(nums_6x1) % len(xor_vec_exp)]))
        nums_6x1 = (np.array(nums_6x1) - xor_vec_repeated) % modulus
```

```
    # Apply inverse 3x3 matrix multiplication
    A = np.array([...])
    B = np.array([...])
    invA = matrix_mod_inv_3x3(A, modulus)
    invB = matrix_mod_inv_3x3(B, modulus)
    nums_2x1 = []
    for i in range(0, len(nums_6x1), 3):
      vec = nums_6x1[i:i + 3]
      if (i // 3) % 2 == 0:
        orig = np.dot(invA, vec) % modulus
      else:
        orig = np.dot(invB, vec) % modulus
      nums_2x1.extend(orig[:2])
    # Apply inverse Hill cipher and other transformations
    dec_nums = [...]
    state = [0, 0, 0, 0]
    pair_vectors = [...]
    for pair_idx, pair in enumerate(pair_vectors):
      power_i = pair_idx + 1
      rotations = state.copy()
      for j in reversed(range(4)):
        vec = mod_exp_vector(vectors[j], power_i, modulus)
        pair = (pair - vec) % modulus
        mat = rotate_matrix(matrices[j], rotations[j])
        inv_mat = matrix_mod_inv(mat, modulus)
        pair = np.dot(inv_mat, pair) % modulus
      dec_nums.extend(pair)
      state = next_rot_state(state)
    # Return the decrypted plaintext
    plaintext = ''.join(num_to_symbol[int(x)] for x in dec_nums)
    plaintext = plaintext[:original_length]
    return plaintext
# Main program
if __name__ == "__main__":
    # Get user input
    plaintext = input("Enter plaintext: ")
```

6.1.1 Modified Original Hill Enigma Ciphertext Paragraph with Four Rotating 2 x 2 Invertible Matrices and Two 3 x 3 Matrices and One 6 x 6 Invertible Matrix.

Modified Original Plaintext Sentence: *They were not railway children to begin with*.

Ciphertext generated from the above sentence:
RO¬ºefÃáÕúhªKW/K&\©J±R°ìÛÌ§HRó%I¡2â´R°ì%E½lIÿJÍÂL3UÆ`=HRó÷
`1KW/RO¬¼µ@ë0pSHõ»$Tè\¾kïwý»«Û!Ú%I¡©J±kïw¯Ç(HRó%E½ÛÌ§÷`1kï
w%E½SHõÆ`=K&\kïwlIÿlIÿHRóÆ`=Ì?h¼µ@©J±K&\2â´SHõÃáÕúhªè\¾Ì?h
%I¡kïwSHõÛ!ÚºeflIÿý»«ß·å¼µ@¼µ@%E½Û!ÚÆ`=ý»«

6.1.B Figure 8A Frequency Distribution Graph of Hill Enigma Ciphertext Four Rotating 2 x 2 Invertible Matrices and Two 3 x 3 Matrices and One 6 x 6 Invertible Matrix.

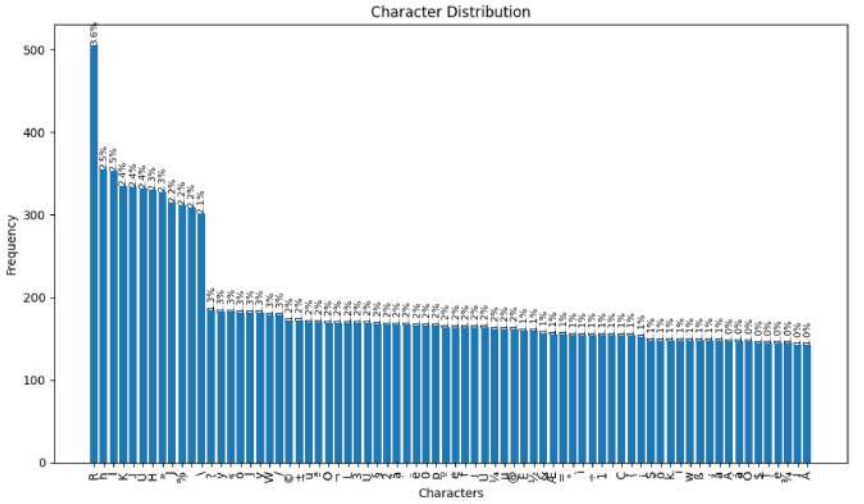

Figure 8A

Modified Original Plaintext Hill Cipher Ciphertext Frequency Distribution using four rotating invertible 2 x 2 matrices in series that feed into two 3 x 3 invertible matrices which feed into one invertible 6 x 6 matrix. This frequency distribution shows a range of letter frequency percentage between 3.6% to 1.0% of the most frequently occurring ciphertext characters from the possible range of ciphertext characters ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9.

Comparing Figure 7A and Figure 8A, Figure 8A has a smaller difference between the frequency distribution after the first 12 ciphertext characters of

0.3 compared to a difference of 1.3 after the first 12 ciphertext characters. The difference between the most frequently occurring ciphertext characters in Figures 6A, 7A, and 8A are 3%, 2.7%, and 2.6% respectively indicating a smaller range as the complexity of the number of dimensions of the matrices used increases.

6.1.C Figure 8B Probability Frequency Distribution Graph of Hill Enigma Ciphertext Four Rotating 2 x 2 Invertible Matrices and Two 3 x 3 Matrices and One 6 x 6 Invertible Matrix.

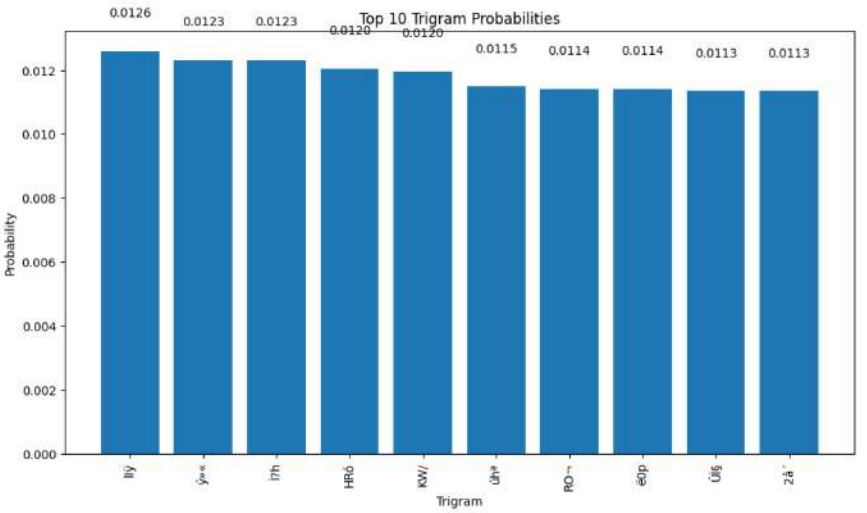

Figure 8B

Modified Original Plaintext Hill Cipher Ciphertext Probability Frequency Distribution of top 10 Trigrams using four rotating 2 x 2 invertible matrices in series with two 3 x 3 matrices and one 6 x 6 matrix. This frequency distribution shows a range of letter frequency percentage between 3.4% to 0.4% of the most frequently occurring ciphertext characters from the possible range of ciphertext characters ASCII and ISO-8859-1 (Latin 1), uppercase and lowercase English letters, and digits 0 – 9.

Comparing Figures 6B, 7B, and 8B, there is a slight difference in the range of the difference in probabilities of the trigram between the trigram with the highest probability and the trigram of the lowest probability among the top 10 trigrams. It is possible in a larger text, the differences might disappear altogether suggesting these variations for encrypting ciphertext are in practice

indistinguishable, but reasonable to consider the higher the dimension of the invertible matrices, the greater the avalanche effect.

Conclusion and Recommendations

Examples of homophonic ciphers and Hill Ciphers were introduced for background information before presenting a connection between the two ciphers. Modifications to previously reported Hill Cipher variations, Hill Enigma Ciphers, were used as a basis to illustrate a new encryption technique that mapped each plaintext character to a trigram ciphertext from a range of ciphertext characters among ASCII and ISO-8859-1 (Latin 1). The novel use of this type of mapping increases confusion. The use of multiple dimensions of the invertible key matrices and rotating the elements of the 2 x 2 invertible key matrices introduces diffusion into the Hill Enigma Cipher encryption schemes. Areas that need further investigation are a) the spike in a ciphertext character in each of Figures 4.1.B, 5.1.B, and 6.1.B., b) the ciphertext character frequency distribution across multiples of trigrams to determine resistance to ciphertext character frequency analysis, c) the extent of the avalanche effect of these variations of the Hill Enigma Ciphers, d) possible error correcting from ciphertext to plaintext due to the number of ciphertext characters each plaintext character encrypts to, e) whether there is any advantage to encrypting each plaintext character into a bigram vs trigram vs 4-gram vs 5-gram, etc., f) whether a generalizable rotation scheme exists for invertible matrices greater than 2 x 2 matrices, and g) the use of modulo 191 to account for the full printable, non-escape character set of ASCII and ISO-8859-1 (Latin 1).

Acknowledgements

The author expresses sincere appreciation for the helpful suggestions and comments from the editor and reviewers that helped increase the clarity of this manuscript.

Declarations

The author has received no grant funding nor has any known conflict of interest. Further, this manuscript is not under consideration for acceptance in any other publication or conference. Finaly, this manuscript has not been publicly presented in any form.

APPENDIES

APPENDIX A: Modified Original Plaintext Passage [4]

They were not railway children to begin with. I dont suppose they had ever thought about railways except as a means of getting to Maskelyne and Cooks, the Pantomime, Zoological Gardens, and Madame Tussauds. They were just ordinary suburban children, and they lived with their Father and Mother in an ordinary red brick fronted villa, with coloured glass in the front door, a tiled passage that was called a hall, a bathroom with hot and cold water, electric bells, French windows, and a good deal of white paint, and every modern convenience, as the house agents say.

There were three of them. Roberta was the eldest. Of course, Mothers never have favourites, but if their Mother HAD had a favourite, it might have been Roberta. Next came Peter, who wished to be an Engineer when he grew up, and the youngest was Phyllis, who meant extremely well.

Mother did not spend all her time in paying dull calls to dull ladies, and sitting dully at home waiting for dull ladies to pay calls to her. She was almost always there, ready to play with the children, and read to them, and help them to do their home lessons. Besides this she used to write stories for them while they were at school, and read them aloud after tea, and she always made up funny pieces of poetry for their birthdays and for other great occasions, such as the christening of the new kittens, or the refurnishing of the dolls house, or the time when they were getting over the mumps.

These three lucky children always had everything they needed pretty clothes, good fires, a lovely nursery with heaps of toys, and a Mother Goose wall paper. They had a kind and merry nursemaid, and a dog who was called James, and who was their very own. They also had a Father who was just perfect, never cross, never unjust, and always ready for a game, at least, if at any time he was NOT ready, he always had an excellent reason for it, and explained the reason to the children so interestingly and funnily that they felt sure he could not help himself.

You will think that they ought to have been very happy. And so they were, but they did not know HOW happy till the pretty life in the Red Villa was over and done with, and they had to live a very different life indeed.

The dreadful change came quite suddenly.

Peter had a birthday, his tenth. Among his other presents was a model engine more perfect than you could ever have dreamed of. The other presents were full of charm, but the Engine was fuller of charm than any of the others were.

Its charm lasted in its full perfection for exactly three days. Then, owing either to Peters inexperience or Phylliss good intentions, which had been rather pressing, or to some other cause, the Engine suddenly went off with a bang. James was so frightened that he went out and did not come back all day. All the Noahs Ark people who were in the tender were broken to bits, but nothing else was hurt except the poor little engine and the feelings of Peter. The others said he cried over it, but of course boys of ten do not cry, however terrible the tragedies may be which darken their lot. He said that his eyes were red because he had a cold.

APPENDIX B: Reduced Plaintext Stripped of Punctuation and Upper Case Letters [4]

they were not railway children to begin with i dont suppose they had ever thought about railways except as a means of getting to maskelyne and cooks the pantomime zoological gardens and madame tussauds they were just ordinary suburban children and they lived with their father and mother in an ordinary redbrickfronted villa with coloured glass in the front door a tiled passage that was called a hall a bathroom with hot and cold water electric bells french windows and a good deal of white paint and every modern convenience as the houseagents say there were three of them roberta was the eldest of course mothers never have favourites but if their mother had had a favourite it might have been roberta next came peter who wished to be an engineer when he grew up and the youngest was phyllis who meant extremely well mother did not spend all her time in paying dull calls to dull ladies and sitting dully at home waiting for dull ladies to pay calls to her she was almost always there ready to play with the children and read to them and help them to do their homelessons besides this she used to write stories for them while they were at school and read them aloud after tea and she always made up funny pieces of poetry for their birthdays and for other great occasions such as the christening of the new kittens or the refurnishing of the dolls house or the time when they were getting over the mumps these three lucky children always had everything they needed pretty clothes good fires a lovely nursery with heaps of toys and a mother goose wallpaper they had a kind and merry nursemaid and a dog who was called james and who was their very own they also had a father who was just perfectnever cross never unjust and always ready for a gameat least if at any time he was not ready he always had an excellent reason for it and explained the reason to the children so interestingly and funnily that they felt sure he couldnt help himself you will think that they ought to have been very happy and so they were but they did not know how

happy till the pretty life in the red villa was over and done with and they had to live a very different life indeed the dreadful change came quite suddenly peter had a birthdayhis tenth among his other presents was a model engine more perfect than you could ever have dreamed of the other presents were full of charm but the engine was fuller of charm than any of the others were its charm lasted in its full perfection for exactly three days then owing either to peters inexperience or phylliss good intentions which had been rather pressing or to some other cause the engine suddenly went off with a bang james was so frightened that he went out and did not come back all day all the noahs ark people who were in the tender were broken to bits but nothing else was hurt except the poor little engine and the feelings of peter the others said he cried over itbut of course boys of ten do not cry however terrible the tragedies may be which darken their lot he said that his eyes were red because he had a cold

References

[1] F. Stahl, "A homophonic cipher for computational cryptography," Report R-599, UILU-ENG 73-2201, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign (1973). [Online]. Available: https://www.ideals.illinois.edu/items/99887

[2] L. S. Hill, "Cryptography in an algebraic alphabet," *The American Mathematical Monthly*, Vol. 36, No. 6, pp. 306-312. (Jun. - Jul., 1929) [Online]. Available: https://ubicua.cua.uam.mx/pluginfile.php/52448/mod_resource/content/3/Hill1.pdf

[3] L. S. Hill, "Concerning certain linear transformation apparatus of cryptography," *The American Mathematical Monthly*, Vol. 38, No. 3., pp. 135-154, (March 1931). [Online]. Available: https://www.jstor.org/stable/2300969?origin=JSTOR-pdf&seq=1

[4] E. Nesbit, The Railway Children, Project Gutenberg, [Online]. Available: https://www.gutenberg.org/ebooks/1874

[5] German Enigma Encoding Machine [Online]. Available: https://www.cia.gov/legacy/museum/artifact/enigma-machine/

[6] English Language Frequency Distribution [Online]. Available: https://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

[7] B. Ryabko and A. Fionov, "Efficient homophonic coding", *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2083-2091, (1999).

[Online]. Available: https://ieeexplore.ieee.org/document/782143 DOI:10.1109/18.782143

[8] V.C. da Rocha Jr., C. Pimentel, and D.P.B.A. Camara, "Redundancy in homophonic coding and a new homophonic coding technique", in *Proceedings IEEE International Symposium on Information Theory*, (June 2006). [Online]. Available: https://www.researchgate.net/publication/258222630_Redundancy_in_homophonic_coding_and_a_new_homophonic_coding_technique DOI:10.1109/ISIT.2006.262006

[9] D.K. Simões, J. Portugheis, and V.C. da Rocha Jr., "Universal homophonic coding scheme using differential encoding and interleaving", *Information Processing Letters,* vol. 113 no. 17, pp. 628-633, (August 2013). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002001901300152X DOI: 10.1016/j.ipl.2013.05.010

[10] P.E. Coggins, III and T. Glatzer, "An algorithm for a matrix-based Enigma encoder from a variation of the Hill Cipher as an application of 2 x 2 matrices", *Primus*, vol. 30, n. 1, pp. 1-18, (2020). DOI: 10.1080/10511970.2018.1493010

[11] P.E. Coggins, "Two novel multidimensional affine variations of the Hill Cipher", *Mathematics and Computer Science*, vol. 9, no. 3, pp. 46-56, (2024). DOI: 10.11648/j.mcs.20240903.11

[12] D. Hamdani and Junaidi, "Modifikasi karaketer kode pada Cipher Hill menggunakan kode ASCII", *Eigen Mathematics Journal*, vol. 03, no. 01, pp. 23-28, (June 2020). DOI: org/10.29303/emj.v3i1.54

[13] D. Jurafsky and J.H. Martin, N-Gram Language Models, Chapter 3, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models Third Edition draft. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ed3book_aug25.pdf

Received November, 2025