African Journal of Advanced Pure and Applied Sciences (AJAPAS)
Online ISSN: 2957-644X
Volume 3, Issue 1, January-March 2024, Page No: 58-71
Website: *https://aaasjournals.com/index.php/ajapas/index*
معامل التأثير العربي 2023: (1.55) SJIFactor 2023: 5.689 ISI 2022-2023: 0.557

The Hill 2-Cipher with Python

Noura A. A. [1*], Raiga A. A [2], Fouad S. A. [3]
[1,2,3] Department of Mathematics, Faculty of Science, Derna University, Al-Qubbah branch, Libya

[*]Corresponding author: nouraambark1988@gmail.com

Received: November 30, 2023	Accepted: January 21, 2024	Published: January 26, 2024

Abstract:
This research paper focuses on the utilization of the Hill 29 cipher to encrypt short sentences using symbols and spaces between words. The Hill 29 cipher relies on linear algebra and number theory concepts. The paper deals with matrix multiplication and inverse matrix, as well as working with the modulo 29 system. The mathematical formulas of the Hill 29 cipher are presented, along with the selection of a suitable key that meets certain conditions to ensure the effectiveness of the encryption process. The cipher is applied to a sentence consisting of 13 characters and three symbols, including spaces between words, and the encryption is successfully achieved. To verify the effectiveness of the encryption, the deciphering of the obtained crossed symbols was performed. The results confirmed that the key is necessary to reconstruct the original formula with its symbols and spaces. The key is considered as a password and is represented by a 2x2 matrix. The original formula with its symbols and spaces was successfully restored, thereby proving the effectiveness of the Hill 29 cipher. It can be relied upon, provided that the appropriate key selected conditions are met. At the end of this research paper, we used a high-level, general-purpose, and very popular programming language which is called Python (latest Python 3) to ensure and get fast results without a classical manual mathematical calculation to encrypt, decrypt sentences by a special code written in Python.

Keywords: Plaintext, cipher text, modular29, inverse, matrices, encryption, decipheration, Python, SymPy.

Cite this article as: Noura A. A., Raiga A. A, Fouad S. A, “The Hill 2-Cipher with Python,” *African Journal of Advanced Pure and Applied Sciences (AJAPAS)*, vol. 3, no. 1, pp. 58-00, January-March 2024.

هيل 2- سايفر مع بايثون

نورا امبارك عبد الرازق[1*]، رايقه أيوب أحمد[2]، فؤاد صبحي علي [3]
[3,2,1] قسم الرياضيات، كلية العلوم، جامعة درنة، فرع القبة، ليبيا

الملخــص
تركز هذه الورقة البحثية على استخدام تشفير هيل 29 لتشفير الجمل القصيرة باستخدام الرموز والمسافات بين الكلمات. يعتمد تشفير هيل 29 على مفاهيم الجبر الخطي ونظرية الأعداد. الورقة التي تتناول ضرب المصفوفة والمصفوفة العكسية، وكذلك العمل مع نظام موديول 29. يتم تقديم الصيغ الرياضية لتشفير هيل29 ، إلى جانب اختيار مفتاح مناسب يلبي شروطا معينة لضمان فعالية عملية التشفير. يتم تطبيق التشفير على جملة تتكون من 13 حرفا وثلاثة رموز، بما في ذلك المسافات بين الكلمات، ويتم تحقيق التشفير بنجاح. للتحقق من فعالية التشفير، تم إجراء فك الرموز المتقاطعة التي تم الحصول عليها. أكدت النتائج أن المفتاح ضروري لإعادة بناء الصيغة الأصلية برموزها ومسافاتها. يعتبر المفتاح كلمة مرور ويتم تمثيله بمصفوفة2 x 2. تمت استعادة الصيغة الأصلية برموزها ومساحاتها بنجاح، مما يثبت فعالية تشفير هيل 29 . يمكن الاعتماد عليها، بشرط استيفاء الشروط الرئيسية المحددة المناسبة. في نهاية هذه الورقة البحثية، استخدمنا لغة برمجة عالية المستوى وذات أغراض عامة وشائعة جدا تسمى بايثون (أحدث اصدار بايثون 3) لضمان الحصول على نتائج سريعة دون حساب رياضي يدوي كلاسيكي لتشفير وفك تشفير الجمل بواسطة رمز خاص مكتوب بلغة بايثون.

الكلمات المفتاحية: نص عادي، نص تشفير، معياري 29، معكوس، مصفوفات، تشفير، فك تشفير، بايثون، سمباي.

Introduction:

The Hill cipher, developed in 1929 by Lester S. Hill, is a cryptographic technique based on matrices and linear algebra. It operates on blocks of plaintext letters, replacing them with cipher text letters using matrix calculations. In this study, we specifically focus on the Hill 2-cipher, utilizing a set of 29 letters commonly used in written messages. Our research draws from the works of Murray Eisenberg (November 1999) on the Hill cipher and modular linear algebra, as well as Jonaki B Ghosh (November 2015) on the Hill cipher. Through this discussion, we aim to present a concise and simplified overview of the Hill cipher and its application.

Definition1: A given n*n invertible matrix, whose entries are non-negative integers from (0, 1 ,..., m-1). Where m is an even number of characters required for encoding, serves as the key for an arbitrary hill n-cipher. Let's imagine we want to employ all 26 alphabets, ranging from A to Z, along with an additional three characters, like (“.”,”?” and -). This indicates that we will have 29 characters available to us for writing plaintext. The 29 characters have been numbers ranging from 0 to 28 in the provided substitution table1.

Table 1. The substitution table for the Hill Cipher

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

Definition 2. Given an integer m>1, Called the modulus. Then we say that two integers a and b are congruent to one another modulo m (congruent “mod m” for short) and we write $a \equiv b(mod\ m)$.
This means that the difference a-b is an integral multiple of m .in other words a $\equiv b(mod\ m)$ when $a = b + km$ for some integer k (positive, negative or zero).
For our hill 2-cipher, we have

$$36 \equiv 7(mod29) \quad \text{And} \quad 25 \equiv 25(mod29)$$

Because

$$36 = 7 + 1 \times 29 \quad \text{And} \quad 25 = 25 + 0 \times 29.$$

Definition 3: the invers of matrix in Z_{29}

There are some methods to calculate the invers of matrixes. we shall now study the easier method of finding the invers of a 2×2 matrix inZ_{29}.here goes:

We suppose we have the key matrix A; we compute the invers key matrix by

$$A^{-1} = \frac{1}{Det(A)} Adj(A)(mod29)$$

$$A^{-1} = [Det(A)]^{-1} Adj(A)(mod29)$$

Not: there is condition for getting on invers key matrix to success the cipher operation, is called

$$\gcd\big(mod29, Detmod(A)\big) = 1$$

If this condition not founded, currently will fail the cipher operation and you have to choose another key matrix

Definition 3. Now we want to calculate $[Det(A)]^{-1}$ in above definition, we will call it the Multiplicative inverses modulo29 and will is x^{-1} and to suggest the x^{-1} we apply the following condition

$$ax^{-1} = 1(\ mod29)$$
$$ax^{-1} = 1(mod29)$$
$$1 \times 1 = 1(mod29)\ so\ x^{-1}\ for\ 1\ is\ 1$$
$$2 \times 15 = 1(mod29\ ) \quad so\ x^{-1}\ for2\ is\ 15$$
$$7 \times 25 = 1(\ mod29) \quad so\ x^{-1}\ for\ 7\ is\ 25$$

Similarly, with $\{N = 1,2,3 \ldots\ldots\ldots..29\}$. We list in Table the reciprocals (multiplicative inverses) of the nonzero elements of Z_{29}

Table2: Multiplicative inverses modulo in Z29

Numbers	**Multiplicative inverses modulo29**	**Numbers**	**Multiplicative inverses modulo29**
1	**1**	**15**	**2**
2	**15**	**16**	**20**

3	**10**	**17**	**12**
4	**22**	**18**	**21**
5	**6**	**19**	**26**
6	**5**	**20**	**16**
7	**25**	**21**	**18**
8	**11**	**22**	**4**
9	**13**	**23**	**24**
10	**3**	**24**	**23**
11	**8**	**25**	**7**
12	**17**	**26**	**19**
13	**9**	**27**	**14**
14	**27**	**28**	**28**

Hill 2-Cipher Encryption

When encrypting a sentence using the Hill $2 \times n$ cipher, it is necessary to have a key that serves as a password. Both the sender and the recipient must be aware of this key, as without it, they will not be able to encrypt or decrypt the sentence. Follow the steps below:

1. Convert the plaintext that we want to cipher it to the corresponding values from the hill cipher table1.
2. Put each two numbers as pairs in columns of a $2 \times n$ matrix, we get a matrix P (is called the plaintext matrix)
3. Compute the cipher operation $C = K * P \ (mod29)$
4. Corresponding the numbers that we have it in step3 with hill cipher table1 and will get a cipher text for plaintext

Example1: - cipher the text "**YOU ARE WELCOME"** with $K = \begin{bmatrix} 5 & 11 \\ 3 & 7 \end{bmatrix}$

Solve: - separate the plaint text to pairs and we do not forget the spaces as

YO U AR E WE LC OM E.

Put each pair in column in p matrix as

$$p = \begin{bmatrix} Y & U & A & E & W & L & O & E \\ O & - & R & - & E & C & M & . \end{bmatrix}$$

Every letter of p matrix corresponding with hill cipher table1

$$P = \begin{bmatrix} 24 & 20 & 0 & 4 & 22 & 11 & 14 & 4 \\ 14 & 28 & 17 & 28 & 4 & 2 & 12 & 26 \end{bmatrix}$$

We calculate the hill cipher operation

$$C = K * P \ (mod29)$$

$$C = \begin{bmatrix} 5 & 11 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} 24 & 20 & 0 & 4 & 22 & 11 & 14 & 4 \\ 14 & 28 & 17 & 28 & 4 & 2 & 12 & 26 \end{bmatrix} (mod29)$$

$$C = \begin{bmatrix} 274 & 408 & 187 & 328 & 154 & 77 & 202 & 306 \\ 170 & 256 & 119 & 208 & 94 & 47 & 126 & 194 \end{bmatrix} (mod29)$$

$$C = \begin{bmatrix} 13 & 2 & 13 & 9 & 9 & 19 & 28 & 16 \\ 25 & 24 & 3 & 5 & 7 & 18 & 10 & 20 \end{bmatrix}$$

Now we corresponding the C matrix with hill cipher table1, we get

$$C^{*} = \begin{bmatrix} N & C & N & J & J & T & - & Q \\ Z & Y & D & F & H & S & K & U \end{bmatrix}$$

The Cipher text will be: "**NZCYNDJFJHTS-KQU"**

Hill Cipher description

As previously mentioned, both the sender and the recipient must be aware of the cipher key, which is represented by a $2 \times n$ matrix. Follow the steps below:

1- Convert the cipher text that we want to break it to the corresponding values from the hill cipher table1.
2- Put each two numbers as pairs in columns of a $2 \times n$ matrix, we get a matrix C (is called the cipher text matrix)
3- Compute the operation $P = K^{-1} * C \ mod 29$
4- Corresponding the numbers that we have it in step3 with hill cipher table1 and will get a plaintext for cipher text.

Example2: - Broke the cipher text **"NZCYNDJFJHTS-KQU"** with $k = \begin{bmatrix} 5 & 11 \\ 3 & 7 \end{bmatrix}$

Solution: - calculate the decipheration product

$$P = K^{-1} * C \ mod 29$$

Compute K^{-1} by one of methods of linear algebra as:

$$K^{-1} = \frac{1}{DetK} \ Adj \ K \ mod 29$$

$$K^{-1} = \frac{1}{2} \begin{bmatrix} 7 & -11 \\ -3 & 5 \end{bmatrix} mod 29$$

$$K^{-1} = 2^{-1} \begin{bmatrix} 7 & -11 \\ -3 & 5 \end{bmatrix} mod 29$$

The Multiplicative inverse modulo29 of 2 in table2 is 15

$$K^{-1} = 15 \begin{bmatrix} 7 & -11 \\ -3 & 5 \end{bmatrix} Mod 29$$

$$K^{-1} = \begin{bmatrix} 105 & -165 \\ -45 & 75 \end{bmatrix} Mod 29$$

$$K^{-1} = \begin{bmatrix} 18 & 9 \\ 13 & 17 \end{bmatrix}$$

There are two Conditions for the effectiveness of the key matrix must be satisfied: -

1- $K * K^{-1} = I \ Mod 29$

Let check:

$$\begin{bmatrix} 5 & 11 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} 18 & 9 \\ 13 & 17 \end{bmatrix} = \begin{bmatrix} 233 & 232 \\ 145 & 146 \end{bmatrix} Mod 29$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

2- $gcd \ (det(K), 29) = 1$

Let check:

$$gcd \ (2,29) = 1$$

Because $29 \div 2 = 14 \quad 1$

$2 \div 1 = 2 \quad 0$

When the remainder is zero, we take the preceding value, which is 1.
So, the conditions is satisfied and cipher operation will success.

Write the cipher text as pairs in columns in C matrix as

$$C^{*} = \begin{bmatrix} N & C & N & J & J & T & - & Q \\ Z & Y & D & F & H & S & K & U \end{bmatrix}$$

And corresponding with hill cipher table1, we have

$$C = \begin{bmatrix} 13 & 2 & 13 & 9 & 9 & 19 & 28 & 16 \\ 25 & 24 & 3 & 5 & 7 & 18 & 10 & 20 \end{bmatrix}$$

Now, apply the decipheration operation

$$P = K^{-1} * C \ mod 29$$

$$P = \begin{bmatrix} 18 & 9 \\ 13 & 17 \end{bmatrix} \begin{bmatrix} 13 & 2 & 13 & 9 & 9 & 19 & 28 & 16 \\ 25 & 24 & 3 & 5 & 7 & 18 & 10 & 20 \end{bmatrix} mod29$$

$$P = \begin{bmatrix} 459 & 252 & 261 & 207 & 225 & 504 & 594 & 468 \\ 594 & 434 & 220 & 202 & 236 & 553 & 534 & 548 \end{bmatrix} mod29$$

$$P = \begin{bmatrix} 24 & 20 & 0 & 4 & 22 & 11 & 14 & 4 \\ 14 & 28 & 17 & 28 & 4 & 2 & 12 & 26 \end{bmatrix}$$

$$p = \begin{bmatrix} Y & U & A & E & W & L & O & E \\ O & - & R & - & E & C & M & . \end{bmatrix}$$

The plaintext will be: YOU ARE WELCOME. the remainder is YOU ARE WELCOME.

Hill Cipher Encryption and Decryption using Python

What is Python?

Python is a high-level, general-purpose, and very popular programming language. It was created by Guido van Rossum, and released in 1991. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry. It is used for: web development (server-side), software development, mathematics, system scripting. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. Official website is (https://www.python.org/).

The biggest strength of Python is huge collection of standard library modules which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Scientific computing

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Interacting with Python

The following features make for easy code development and debugging in python:

- Python code is interpreted: There is no need to compile the code. Your code is read by a python interpreter and made into executable instructions for your computer in real time.
- Python is dynamically typed: There is no need to declare the type of a variable or the type of an input to a function.
- Python has automatic garbage collection or memory management: There is no need to explicitly allocate memory for variables before you use them or deallocate them after use.

There are at least four ways to interact with your Python 3 installation.

1. Use a python shell.
2. Use an iPython shell.
3. Put code in a python file ending in .py.
4. Write code+text in Jupyter notebook.

Python shell

Type the python command you use in your system (python or python3) to get this shell. We will use python3 since that is what my system requires, but please do make sure to replace it by python if that's what is needed on your system.

iPython shell

A more powerful shell interactive environment is provided by the iPython shell (type in iPython or ipython3 into your command prompt, or launch it from Anaconda navigator).The iPython shell has features like auto-completion, coloring, history of commands, automatic help by tacking on?, ability to interact with your operating system’s commands,etc.

Python file

Open your favorite text editor, type in some python code, and then save the file as my firstpy.py.

Jupyter Notebook

The Jupyter notebook is a web-browser based graphical environment consisting of cells, which can consist of code, or text. The text cells should contain text in markdown syntax, which allows you to type not just words in bold and italic, but also tables, mathematical formula using latex, etc. The code cells of Jupyter can contain code in various languages, but here we will exclusively focus on code cells with Python 3. This is which we used in our programming Python cod in this paper.

Main standard libraries used in coding Hill Cipher algorithm to Python

- SymPy (computer algebra system)
- Pandas (data analyze)
- NumPy (multi-dimensional arrays and matrices)

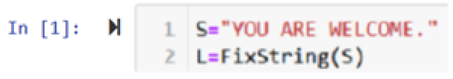
How to Install Python, SymPy, Pandas, NumPy and Jupyter Notebook on Windows:

1. Download and install the latest version of Python from the official website (https://www.python.org/downloads/). Make sure to check the option “Add Python to PATH”.
2. Go to Start and search for Python.
3. You can see Python 3.7 (64-bit) and IDLE. Let’s open IDLE, which is the short form for Integrated Development Environment, and run a simple print statement.
4. In the command prompt install jupyter using command: pip install Jupyter.
5. In command prompt using cd command. and open it using command: *jupyter-notebook.*
6. To install the NumPy, Sympy and Pandas module, run the following commands in a Jupyter Notebook cell: *!pip install numpy, sympy, pandas*
7. Copy the python code from appendix and past it in Notebook cell and run it before.

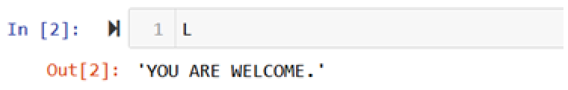
How to use Python for Hill Cipher Encryption?

From the previous example1:

- Put plaintext "YOU ARE WELCOME." in a string variable S and apply on its python function FixString(S) to add a space at the end of the string if the number of letters is odd, nothing if else form in Jupyter Notebook (see appendix)

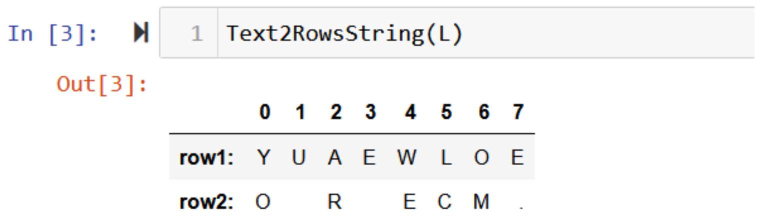


```
In [1]:  S="YOU ARE WELCOME."
         L=FixString(S)
```



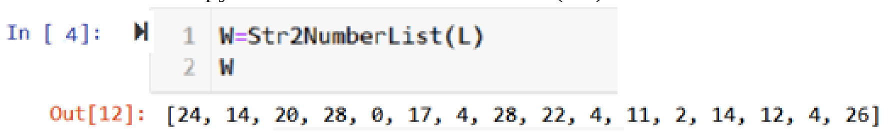
```
In [2]:  L

Out[2]: 'YOU ARE WELCOME.'
```

- Convert the plaintext in list L to output pairs in column as in p matrix as before with python function Text2RowsString(L) (see appendix)



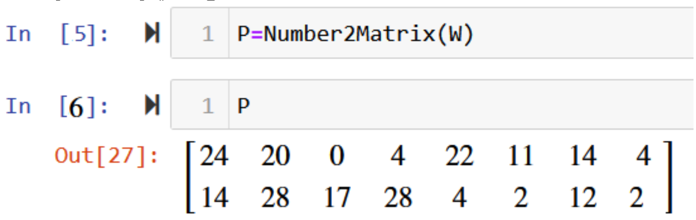
```
In [3]:  Text2RowsString(L)

Out[3]:
        0  1  2  3  4  5  6  7
row1:   Y  U  A  E  W  L  O  E
row2:   O     R     E  C  M  .
```

- Convert the plaintext stored in list L that we want to cipher it to the corresponding values from the hill cipher table1 which programed in python function chr_to_int(char) (see appendix) and get the new coded list W with python function Str2NumberList(list):



```
In [ 4]:  W=Str2NumberList(L)
          W

Out[12]: [24, 14, 20, 28, 0, 17, 4, 28, 22, 4, 11, 2, 14, 12, 4, 26]
```

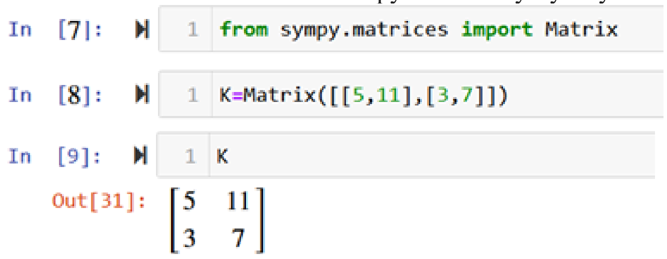
- Convert coded list W with python function Number2Matrix(W) which contains the super python library SumPy() to get coded matrix P :



```
In [5]:   P=Number2Matrix(W)

In [6]:   P
Out[27]:
```

$$\begin{bmatrix} 24 & 20 & 0 & 4 & 22 & 11 & 14 & 4 \\ 14 & 28 & 17 & 28 & 4 & 2 & 12 & 2 \end{bmatrix}$$

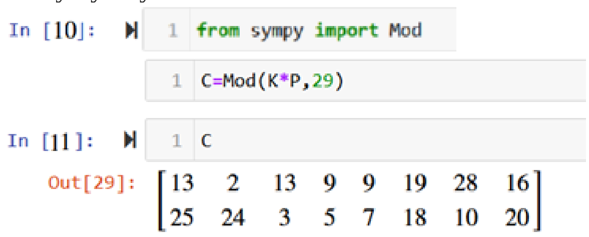
- Let's call the Matrix function from python library SymPy and define the key matrix K :



```
In [7]:   from sympy.matrices import Matrix

In [8]:   K=Matrix([[5,11],[3,7]])

In [9]:   K
Out[31]:
```

$$\begin{bmatrix} 5 & 11 \\ 3 & 7 \end{bmatrix}$$

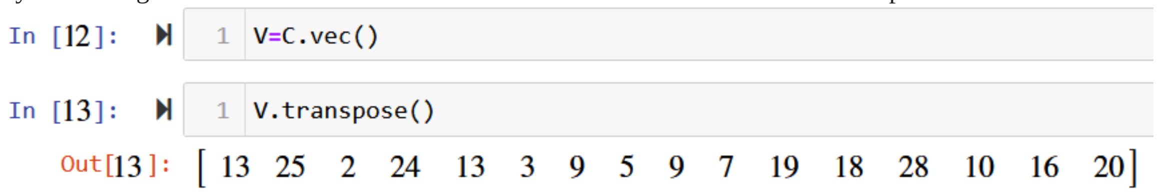
- Calculate the hill cipher operation $C = K * P \ (mod\ 29)$ by calling the Mod function from python library SymPy as follows:



```
In [10]:  from sympy import Mod

          C=Mod(K*P,29)

In [11]:  C
Out[29]:
```

$$\begin{bmatrix} 13 & 2 & 13 & 9 & 9 & 19 & 28 & 16 \\ 25 & 24 & 3 & 5 & 7 & 18 & 10 & 20 \end{bmatrix}$$

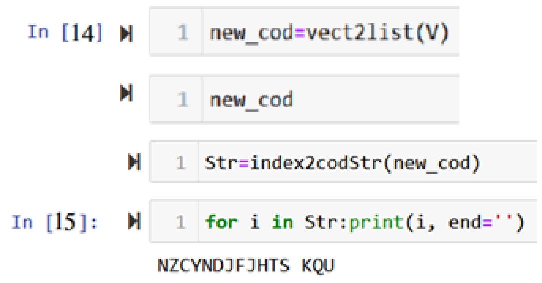
- By converting matrix C to vector V then to show it as a row vector we take its transpose:



```
In [12]:  V=C.vec()

In [13]:  V.transpose()
Out[13]:
```

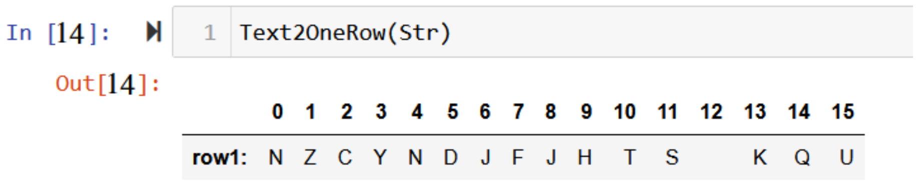
$$\begin{bmatrix} 13 & 25 & 2 & 24 & 13 & 3 & 9 & 5 & 9 & 7 & 19 & 18 & 28 & 10 & 16 & 20 \end{bmatrix}$$

Corresponding to the C matrix with hill cipher table1 and python functions int_to_chr(int), indix2codStr(L) stored in list Str, we get the coded text "**NZCYNDJFJHTS KQU**" as expected before:



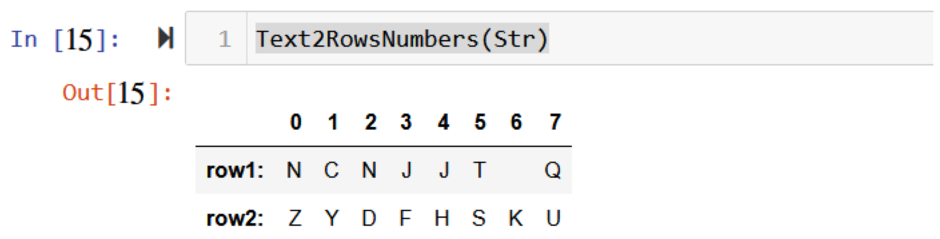
```
In [14]   new_cod=vect2list(V)

          new_cod

          Str=index2codStr(new_cod)

In [15]:  for i in Str:print(i, end='')

NZCYNDJFJHTS KQU
```

- With the help of python library pandas coded in python functions Text2OneRow(Str) and Text2RowsNumbers(Str),we can write this code text in a beautiful one and two views :



```
In [14]:  Text2OneRow(Str)
Out[14]:
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
row1:	N	Z	C	Y	N	D	J	F	J	H	T	S		K	Q	U

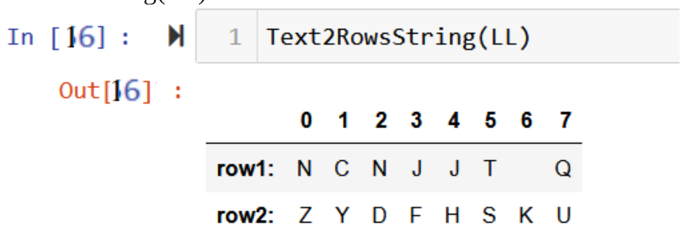


```
In [15]:  Text2RowsNumbers(Str)

Out[15]:
           0  1  2  3  4  5  6  7
   row1:   N  C  N  J  J  T     Q
   row2:   Z  Y  D  F  H  S  K  U
```

How to use Python for Hill Cipher Decryption?

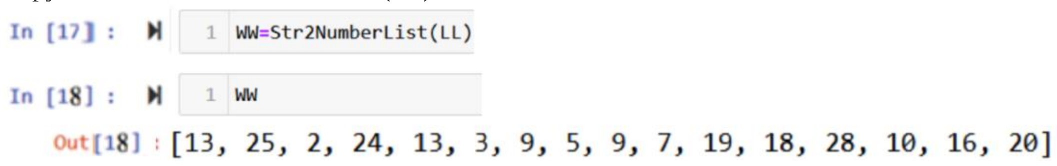
From the previous example2:

- Put plaintext "**NZCYNDJFJHTS KQU**" in a list LL form and use python function Text2RowsString(LL) to view it



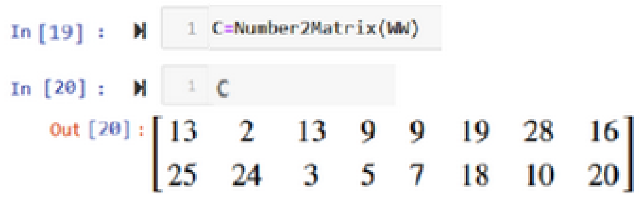
```
In [16] :  Text2RowsString(LL)

Out[16] :
           0  1  2  3  4  5  6  7
   row1:   N  C  N  J  J  T     Q
   row2:   Z  Y  D  F  H  S  K  U
```

- Convert the cipher text stored in list LL that we want to break it to the corresponding values from the hill cipher table1 which programed in python function chr_to_int(char) and get the new coded list WW with python function Str2NumberList(LL):



```
In [17] :  WW=Str2NumberList(LL)

In [18] :  WW

Out[18] : [13, 25, 2, 24, 13, 3, 9, 5, 9, 7, 19, 18, 28, 10, 16, 20]
```

- Convert the list to matrix with the python function Number2Matrix(list)

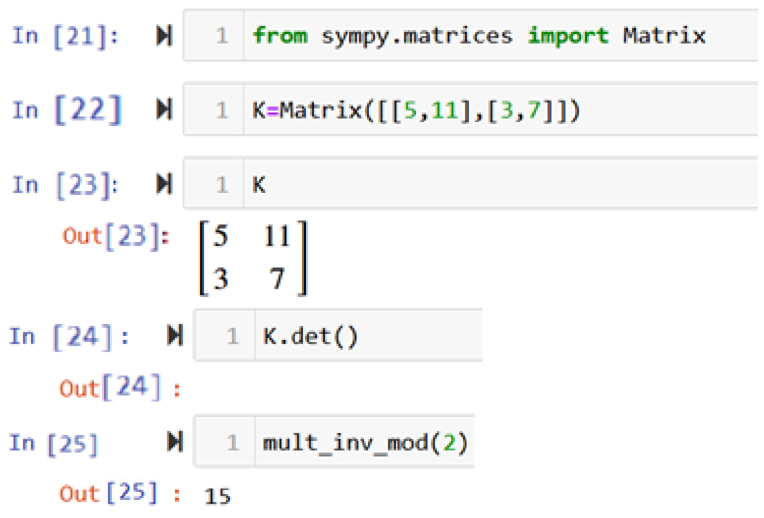


```
In [19] :  C=Number2Matrix(WW)

In [20] :  C
```

Out [20] :

$$\begin{bmatrix} 13 & 2 & 13 & 9 & 9 & 19 & 28 & 16 \\ 25 & 24 & 3 & 5 & 7 & 18 & 10 & 20 \end{bmatrix}$$

- Let's call the Matrix function from python library SymPy and define the key matrix K and find its determinant and find multiplicative inverses modulo 29 in table2 which programmed in python function mult_inv_mod(determinant) or find_multiplicative_inverse(determinant) :



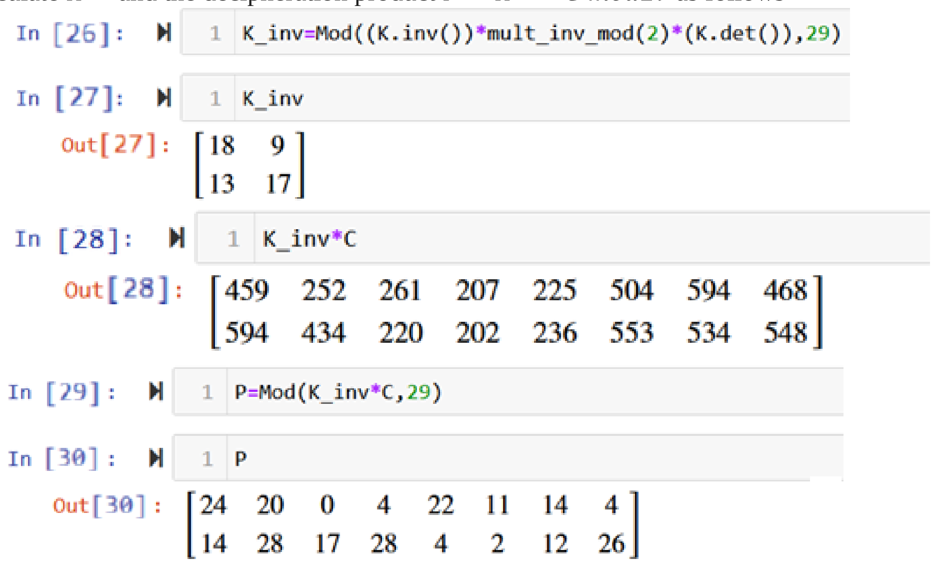
```
In [21]:  from sympy.matrices import Matrix

In [22]   K=Matrix([[5,11],[3,7]])

In [23]:  K
```

Out[23]:

$$\begin{bmatrix} 5 & 11 \\ 3 & 7 \end{bmatrix}$$

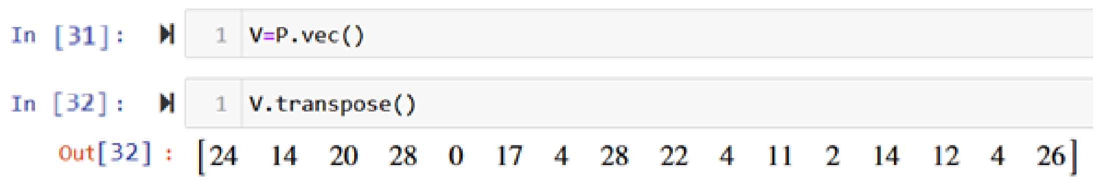
```
In [24]:  K.det()

Out[24] :

In [25]   mult_inv_mod(2)

Out [25] :  15
```

- Calculate K^{-1} and the decipheration product $P = K^{-1} * C \; mod29$ as follows



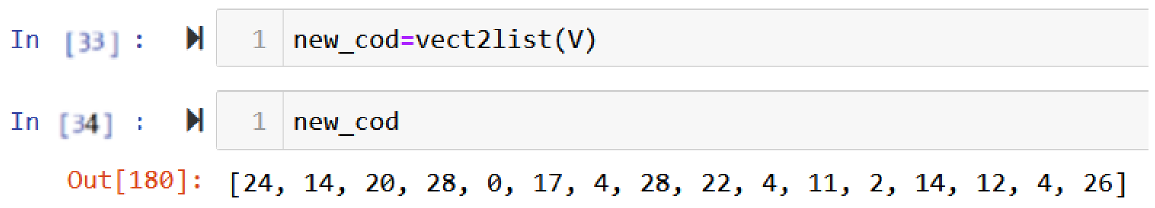
```
In [26]:  K_inv=Mod((K.inv())*mult_inv_mod(2)*(K.det()),29)

In [27]:  K_inv

Out[27]: [18   9]
         [13  17]

In [28]:  K_inv*C

Out[28]: [459  252  261  207  225  504  594  468]
         [594  434  220  202  236  553  534  548]

In [29]:  P=Mod(K_inv*C,29)

In [30]:  P

Out[30]: [24  20   0   4  22  11  14   4]
         [14  28  17  28   4   2  12  26]
```

- By converting matrix P to a V vector then to show it as a row vector we take its transpose:



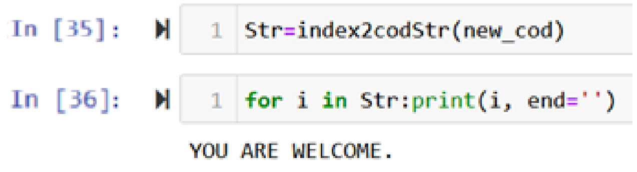
```
In [31]:  V=P.vec()

In [32]:  V.transpose()

Out[32]: [24  14  20  28  0  17  4  28  22  4  11  2  14  12  4  26]
```

- By converting V vector using python function vect2list(vec) to list



```
In [33] :  new_cod=vect2list(V)

In [34] :  new_cod

Out[180]: [24, 14, 20, 28, 0, 17, 4, 28, 22, 4, 11, 2, 14, 12, 4, 26]
```

- Corresponding to the V vector with hill cipher table1 and python functions int_to_chr(int), indix2codStr(L) stored in list Str, we get the coded text "YOU ARE WELCOME." as we expected .



```
In [35]:  Str=index2codStr(new_cod)

In [36]:  for i in Str:print(i, end='')

          YOU ARE WELCOME.
```

Conclusion

The research paper focuses on the process of encrypting a short sentence using the Hill cipher with a key size of 29 and the subsequent decryption process. The cipher relies on matrix operations in linear algebra and important concepts in number theory. The paper highlights the selection of an appropriate cipher key and its significance in ensuring successful encryption with the help of data science modeling by using Python language not just for small text but a huge text. The results demonstrate the effectiveness of the cipher in data protection and successful retrieval of the original text. This research paper contributes to enhancing our understanding of modern ciphers and the associated mathematical and programming techniques.

References

[1] D. Beazley, Python Essential Reference, Addison-Wesley Professional, 717 pages, 2009.
[2] C. Christensen, Finding multiplicative inverses modulo n, Cryptology Notes, Fall 2005, 2005
[3] M. Eisenberg , "Hill cipher and modular linear algebra". November 3, 1999.
[4] J. B. Ghosh, "Hill cipher". November, 2015, 33-43 ,2015
[5] L. S. Hill, "Cryptography in an algebraic alphabet". The American Mathematical Monthly, 36, 306-312, 1929.

[6] L. S. Hill, "Concerning certain linear transformation apparatus of cryptography", The American Mathematical Monthly, 38, 135-154, 1931.
[7] D. Kahn, "The codebreakers: The story of secret writing", 1967.
[8] J. Overbey , W. Traves, J. Wojdylo , "On the key space of the Hill Cipher", Cryptologia, 29(1), 59-72, 2005.
[9] M. Summerfield, Programming in Python 3, Addison-Wesley,2010.

Appendix: Some code fragments

The following python code functions text are used in this paper, copy it in alone cell in Jupyter and run it firstly:

```
def FixString(str):
    if len(str) % 2 == 0:
        S=str
    else:
        S=str+' '
    return S

def Text2RowsString(list):
    import pandas as pd
    Row1[] =
    Row2[] =
    count = 0
    for i in list :
        if count % 2 == 0:
            Row1.append(i)
        if count % 2 == 1:
            Row2.append(i)
        count += 1
    df = pd.DataFrame({'row1:':Row1 ,
                             ' row2:':Row2})
 #    Print the dataframe
    df.transpose    ()
    return df.transpose    ()

def chr_to_int(char):
 #    Uppercase the char to get into range 65-90 in ascii table
    char = char.upper()
 #    Cast chr to int and subtract 65 to get 0-25
    integer = ord(char) - 65
    if char =='.' : integer=26
    elif char =='?':integer=27
    elif char ==' ':integer=28
    return integer

def Str2NumberList(L):
    U[]=
    for items in L:
        U.append(chr_to_int(items))
    return U

def chr_to_int(char):
 #    Uppercase the char to get into range 65-90 in ascii table
    char = char.upper()
 #    Cast chr to int and subtract 65 to get 0-25
    integer = ord(char) - 65
    if char =='.' : integer=26
    elif char =='?':integer=27
    elif char ==' ':integer=28
    return integer

def Str2NumberList(L):
    U[]=
    for items in L:
        U.append(chr_to_int(items))
    return U
def Text2RowsString(list):
    import pandas as pd
    Row1[] =
    Row2[] =
    count = 0
    for i in list :
        if count % 2 == 0:
            Row1.append(i)
        if count % 2 == 1:
            Row2.append(i)
        count += 1
    df = pd.DataFrame({'row1:':Row1 ,
                        ' row2:':Row2})
 #    Print the dataframe
    df.transpose    ()
    return df.transpose    ()

def Text2RowsNumbers(list):
    import pandas as pd
    Row1[] =
    Row2[] =
    count = 0
    for i in list :
        if count % 2 == 0:
            Row1.append(i)
        if count % 2 == 1:
            Row2.append(i)
        count += 1
    df = pd.DataFrame({'row1:':Row1 ,
                        ' row2:':Row2})
 #    Print the dataframe
    df.transpose    ()
    return df.transpose    ()

def chr_to_int(char):
 #    Uppercase the char to get into range 65-90 in ascii table
    char = char.upper()
 #    Cast chr to int and subtract 65 to get 0-25
    integer = ord(char) - 65
    if char =='.' : integer=26
    elif char =='?':integer=27
    elif char ==' ':integer=28
    return integer

def Str2NumberList(L):
    U[]=
    for items in L:
        U.append(chr_to_int(items))
    return U

def Number2Matrix(W):
    from sympy.matrices import Matrix
    Row1[] =
    Row2[] =
    count = 0
    for i in W :
        if count % 2 == 0:
            Row1.append(i)
        if count % 2 == 1:
            Row2.append(i)
        count += 1
    return Matrix([Row1,Row2])
```

```
def vect2list(vec):
    import numpy as np
    c = np.array(vec)
    new_cod=c.ravel().tolist()
    return new_cod
def int_to_chr(int):
    num=int+65
    str=chr(num)
    str=str.upper()
    if int==26:
        str'.'=
    if int==27:
        str'?'=
    if int==28:
        str    ' '=
    return str

def index2codStr(list):
    UU[]=
    for items in list:
        UU.append(int_to_chr(items))
    return UU

def Text2OneRow(list):
    import pandas as pd
    df = pd.DataFrame({'row1:':list})
 #    Print the dataframe
    df.transpose    ()
    return df.transpose    ()

def Text2RowsString(list):
    import pandas as pd
    Row1[] =
    Row2[] =
    count = 0
    for i in list :
        if count % 2 == 0:
            Row1.append(i)
        if count % 2 == 1:
            Row2.append(i)
        count += 1
    df = pd.DataFrame({'row1:':Row1 ,
'                  row2:':Row2 ({
 #    Print the dataframe
    df.transpose    ()
    return df.transpose    ()

def mult_inv_mod(determinant):
    data_source = {1:1,2:15,3:10,4:22,5:6,6:5,7:25,8:11,9:13,10:3,11:8,12:17,
,13:9,14:27,15:2,16:20,17:12,18:21,19:26,20:16,21:18,22:4
{23:24,24:23,25:7,26:19,27:14,28:28
    UU=data_source[determinant]
    return UU

def find_multiplicative_inverse(determinant):
    multiplicative_inverse = -1
    for i in range:(29)
        inverse = determinant * i
        if inverse % 29 == 1:
            multiplicative_inverse = i
            break
    return multiplicative_inverse

def vect2list(vec):
    import numpy as np
    c = np.array(vec)
    new_cod=c.ravel().tolist()
    return new_cod
```

Hill Cipher Encryption python commands:

The following python code are used in this paper as commands to get results where every line must be in alone cell in Jupyter:

```
L="YOU ARE WELCOME.";    L
Text2RowsString(L)
W=Str2NumberList(L);  W
P=Number2Matrix(W);  P
from sympy.matrices import Matrix
K=Matrix([[5,11],[3,7]]);   K
from sympy import Mod
C=Mod(K*P,29);    C
V=C.vec()
V.transpose()
new_cod=vect2list(V);   new_cod
Str=index2codStr(new_cod)
for i in Str:print(i, end='')
Text2OneRow(Str)
```

Hill Cipher Decryption python commands:

```
LL="NZCYNDJFJHTS KQU";     LL
Text2OneRow(list(LL))
Text2RowsString(LL)
WW=Str2NumberList(LL);      WW
C=Number2Matrix(WW);    C
from sympy.matrices import Matrix
K=Matrix([[5,11],[3,7]])     ;K
K.det()
mult_inv_mod(2)
find_multiplicative_inverse(K.det())
K.inv()
K_inv=Mod((K.inv())*mult_inv_mod(2)*(K.det()),29);     K_inv
K_inv*C
P=Mod(K_inv*C,29);    P
V=P.vec();    V.transpose()
new_cod=vect2list(V);   new_cod
Str=index2codStr(new_cod)
for i in Str:print(i, end='')
```

Satisfying the Conditions for the effectiveness of the key matrix $(K * K^{-1})\ Mod29 = I$ by python: -

```
N=find_multiplicative_inverse(K.det())    ;N
KK=K.inv()*(K.det()*N);   KK
KKM=Mod(KK,29);      KKM
K*KKM;    K*KKM
Mod(K*KKM,29)
```