NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

LAB 04

LAB 04: Write a program in Java or Python to perform Cryptanalysis or decoding of Vigenere Cipher.

ROLL NO	52
NAME	Sarvesh Patil
CLASS	D15A
SUBJECT	Internet Security Lab
LO MAPPED	LO1: To apply the knowledge of symmetric cryptography to implement classical ciphers

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

AIM:
Write a program in Java or Python to perform Cryptanalysis or decoding of Vigenere cipher

INTRODUCTION:
Introduction
The vigenere cipher is an algorithm that is used to encrypting and decrypting the text. The vigenere cipher is an algorithm of encrypting an alphabetic text that uses a series of interwoven caesar ciphers. It is based on a keyword's letters. It is an example of a polyalphabetic substitution cipher. This algorithm is easy to understand and implement. This algorithm was first described in 1553 by Giovan Battista Bellaso. It uses a Vigenere table or Vigenere square for encryption and decryption of the text. The vigenere table is also called the tabula recta.

Two methods perform the vigenere cipher.

Method 1
When the vigenere table is given, the encryption and decryption are done using the vigenere table (26 * 26 matrix) in this method.

Plaintext

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

Example: The plaintext is "JAVATPOINT", and the key is "BEST".
To generate a new key, the given key is repeated in a circular manner, as long as the length of the plain text does not equal to the new key.

J	A	V	A	T	P	O	I	N	T
B	E	S	T	B	E	S	T	B	E

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

Encryption
The first letter of the plaintext is combined with the first letter of the key. The column of plain text "J" and row of key "B" intersects the alphabet of "K" in the vigenere table, so the first letter of ciphertext is "K".
Similarly, the second letter of the plaintext is combined with the second letter of the key. The column of plain text "A" and row of key "E" intersects the alphabet of "E" in the vigenere table, so the second letter of ciphertext is "E".
This process continues continuously until the plaintext is finished.
Ciphertext = KENTUTGBOX

Decryption
Decryption is done by the row of keys in the vigenere table. First, select the row of the key letter, find the ciphertext letter's position in that row, and then select the column label of the corresponding ciphertext as the plaintext.

K	E	N	T	U	T	G	B	O	X
B	E	S	T	B	E	S	T	B	E

For example, in the row of the key is "B" and the ciphertext is "K" and this ciphertext letter appears in the column "J", that means the first plaintext letter is "J".
Next, in the row of the key is "E" and the ciphertext is "E" and this ciphertext letter appears in the column "A", that means the second plaintext letter is "A".
This process continues continuously until the ciphertext is finished.
Plaintext = JAVATPOINT

Method 2
When the vigenere table is not given, the encryption and decryption are done by Vigenar algebraically formula in this method (convert the letters (A-Z) into the numbers (0-25)).
Formula of encryption is,
$E_i = (P_i + K_i) \bmod 26$
Formula of decryption is,
$D_i = (E_i - K_i) \bmod 26$
If any case (D_i) value becomes negative (-ve), in this case, we will add 26 in the negative value.
Where,
E denotes the encryption.
D denotes the decryption.
P denotes the plaintext.
K denotes the key.
Note: "i" denotes the offset of the ith number of the letters, as shown in the table below.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Example: The plaintext is "JAVATPOINT", and the key is "BEST".

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

Encryption: $E_i = (P_i + K_i) \bmod 26$

Plaintext	J	A	V	A	T	P	O	I	N	T
Plaintext value (P)	09	00	21	00	19	15	14	08	13	19
Key	B	E	S	T	B	E	S	T	B	E
Key value (K)	01	04	18	19	01	04	18	19	01	04
Ciphertext value (E)	10	04	13	19	20	19	06	01	14	23
Ciphertext	K	E	N	T	U	T	G	B	O	X

Decryption: $D_i = (E_i - K_i) \bmod 26$
If any case (Di) value becomes negative (-ve), in this case, we will add 26 in the negative value. Like, the third letter of the ciphertext;
N = 13 and S = 18
$D_i = (E_i - K_i) \bmod 26$
$D_i = (13 - 18) \bmod 26$
$D_i = -5 \bmod 26$
$D_i = (-5 + 26) \bmod 26$
$D_i = 21$

Ciphertext	K	E	N	T	U	T	G	B	O	X
Ciphertext value (E)	10	04	13	19	20	19	06	01	14	23
Key	B	E	S	T	B	E	S	T	B	E
Key value (K)	01	04	18	19	01	04	18	19	01	04
Plaintext value (P)	09	00	21	00	19	15	14	08	13	19
Plaintext	J	A	V	A	T	P	O	I	N	T

The idea behind the Vigenère cipher, like all other polyalphabetic ciphers, is to disguise the plaintext letter frequency to interfere with a straightforward application of frequency analysis. For instance, if P is the most frequent letter in a ciphertext whose plaintext is in English, one might suspect that P corresponds to e since e is the most frequently used letter in English. However, by using the Vigenère cipher, e can be enciphered as different ciphertext letters at different points in the message, which defeats simple frequency analysis.

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

The primary weakness of the Vigenère cipher is the repeating nature of its key. If a cryptanalyst correctly guesses the key's length n, the cipher text can be treated as n interleaved Caesar ciphers, which can easily be broken individually. The key length may be discovered by brute force testing each possible value of n, or Kasiski examination and the Friedman test can help to determine the key length (see below: § Kasiski examination and § Friedman test).

Kasiski examination

Further information: Kasiski examination

In 1863, Friedrich Kasiski was the first to publish a successful general attack on the Vigenère cipher.[17] Earlier attacks relied on knowledge of the plaintext or the use of a recognizable word as a key. Kasiski's method had no such dependencies. Although Kasiski was the first to publish an account of the attack, it is clear that others had been aware of it. In 1854, Charles Babbage was goaded into breaking the Vigenère cipher when John Hall Brock Thwaites submitted a "new" cipher to the Journal of the Society of the Arts.[18][19] When Babbage showed that Thwaites' cipher was essentially just another recreation of the Vigenère cipher, Thwaites presented a challenge to Babbage: given an original text (from Shakespeare's The Tempest : Act 1, Scene 2) and its enciphered version, he was to find the key words that Thwaites had used to encipher the original text. Babbage soon found the key words: "two" and "combined". Babbage then enciphered the same passage from Shakespeare using different key words and challenged Thwaites to find Babbage's key words.[20] Babbage never explained the method that he used. Studies of Babbage's notes reveal that he had used the method later published by Kasiski and suggest that he had been using the method as early as 1846.[21]

The Kasiski examination, also called the Kasiski test, takes advantage of the fact that repeated words are, by chance, sometimes encrypted using the same key letters, leading to repeated groups in the ciphertext. For example, consider the following encryption using the keyword ABCD:

Key: ABCDABCDABCDABCDABCDABCDABCD
Plaintext: cryptoisshortforcryptography
Ciphertext: CSASTPKVSIQUTGQUCSASTPIUAQJB
There is an easily noticed repetition in the ciphertext, and so the Kasiski test will be effective.

The distance between the repetitions of CSASTP is 16. If it is assumed that the repeated segments represent the same plaintext segments, that implies that the key is 16, 8, 4, 2, or 1 characters long. (All factors of the distance are possible key lengths; a key of length one is just a simple Caesar cipher, and its cryptanalysis is much easier.) Since key lengths 2 and 1 are unrealistically short, one needs to try only lengths 16, 8 or 4. Longer messages make the test more accurate because they usually contain more repeated ciphertext segments. The following ciphertext has two segments that are repeated:

Ciphertext: VHVSSPQUCEMRVBVBBBVHVSURQGIBDUGRNICJQUCERVUAXSSR
The distance between the repetitions of VHVS is 18. If it is assumed that the repeated segments represent the same plaintext segments, that implies that the key is 18, 9, 6, 3, 2 or 1 character long. The distance between the repetitions of QUCE is 30 characters. That means that the key length could be 30, 15, 10, 6, 5, 3, 2 or 1 character long. By taking the intersection of those sets, one could safely conclude that the most likely key length is 6 since 3, 2, and 1 are unrealistically short..

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

Frequency analysis

Once the length of the key is known, the ciphertext can be rewritten into that many columns, with each column corresponding to a single letter of the key. Each column consists of plaintext that has been encrypted by a single Caesar cipher. The Caesar key (shift) is just the letter of the Vigenère key that was used for that column. Using methods similar to those used to break the Caesar cipher, the letters in the ciphertext can be discovered.

An improvement to the Kasiski examination, known as Kerckhoffs' method, matches each column's letter frequencies to shifted plaintext frequencies to discover the key letter (Caesar shift) for that column. Once every letter in the key is known, all the cryptanalyst has to do is to decrypt the ciphertext and reveal the plaintext.[24] Kerckhoffs' method is not applicable if the Vigenère table has been scrambled, rather than using normal alphabetic sequences, but Kasiski examination and coincidence tests can still be used to determine key length.

Key elimination

The Vigenère cipher, with normal alphabets, essentially uses modulo arithmetic, which is commutative. Therefore, if the key length is known (or guessed), subtracting the cipher text from itself, offset by the key length, will produce the plain text subtracted from itself, also offset by the key length. If any "probable word" in the plain text is known or can be guessed, its self-subtraction can be recognized, which allows recovery of the key by subtracting the known plaintext from the cipher text. Key elimination is especially useful against short messages. For example using LION as the key below:

Plaintext: thequickbrownfoxjumpsoverthelazydog
Key: LIONLIONLIONLIONLIONLIONLIONLIONLIO
Ciphertext: EPSDFQQXMZCJYNCKUCACDWJRCBVRWINLOWU
Then subtract the ciphertext from itself with a shift of the key length 4 for LION.

Ciphertext (original): EPSDFQQXMZCJYNCKUCACDWJRCBVRWINLOWU
Ciphertext (shifted): FQQXMZCJYNCKUCACDWJRCBVRWINLOWU____
Result (difference): ZZCGTROOOMAZELCIRGRLBVOAGTIGIMTLOWU
Which is nearly equivalent to subtracting the plaintext from itself by the same shift.

Plaintext (original): thequickbrownfoxjumpsoverthelazydog
Plaintext (shifted): uickbrownfoxjumpsoverthelazydog____
Result (difference): zzcgtrooomazelcirgrlbvoagtigimtydog
Which is algebraically represented for as:

In this example, the words brownfox are known.

Plaintext (original): brownfox
Plaintext (shifted): nfox____
Result (difference): omaznfox
This result omaz corresponds with the 9th through 12th letters in the result of the larger examples above. The known section and its location is verified.

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

Subtract brow from that range of the ciphertext.

Ciphertext: EPSDFQQXMZCJYNCKUCACDWJRCBVRWINLOWU
Plaintext: _____brow_____
Key: EPSDFQQXLIONYNCKUCACDWJRCBVRWINLOWU
This produces the final result, the reveal of the key LION.

ALGORITHM:
Step 1 : Read the plaintext from the user.
Step 2 : Read the key from the user.
Step 3 : The key will be repeated as many times as the length of the plaintext.
Step 4 : The plaintext is converted into cipher text using the values of plaintext and key .
Step 5 : The values of plaintext and key are added and taken mod with 26 to get the value of cipher text.
Step 6 : The values of ciphertext are converted into the corresponding alphabets.
Step 7 : Display the obtained Cipher text.
Step 8 : Again decrypt the Cipher text into the corresponding plaintext using the same method instead subtracting the cipher values from key values.
Step 9 : Again display the plaintext which we obtain .

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

PROGRAM:

```
import java.util.*;
public class VigenereCipher
{
static String generateKey(String str, String key)
{
   int x = str.length();

   for (int i = 0; ; i++)
   {
      if (x == i)
         i = 0;
      if (key.length() == str.length())
         break;
      key+=(key.charAt(i));
   }
   return key;
}

static String cipherText(String str, String key)
{
   String cipher_text="";

   for (int i = 0; i < str.length(); i++)
   {
      int x = (str.charAt(i) + key.charAt(i)) %26;

      x += 'A';

      cipher_text+=(char)(x);
   }
   return cipher_text;
}

static String originalText(String cipher_text, String key)
{
   String orig_text="";

   for (int i = 0 ; i < cipher_text.length() &&
                       i < key.length(); i++)
   {

    int x = (cipher_text.charAt(i) -
                key.charAt(i) + 26) %26;

      x += 'A';
```

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

```
        orig_text+=(char)(x);
    }
    return orig_text;
}

static String LowerToUpper(String s)
{
    StringBuffer str =new StringBuffer(s);
    for(int i = 0; i < s.length(); i++)
    {
        if(Character.isLowerCase(s.charAt(i)))
        {
            str.setCharAt(i, Character.toUpperCase(s.charAt(i)));
        }
    }
   s = str.toString();
    return s;
}

public static void main(String[] args)
{
    Scanner myObj = new Scanner(System.in);
    System.out.println("Enter Plaintext:");
    String Str = myObj.nextLine();

    Scanner myObj2 = new Scanner(System.in);
    System.out.println("Enter key:");
    String Keyword = myObj2.nextLine();


     String str = LowerToUpper(Str);
     String keyword = LowerToUpper(Keyword);

    String key = generateKey(str, keyword);
    System.out.println("Generated key : "+key);
    String cipher_text = cipherText(str, key);

    System.out.println("Ciphertext : "
        + cipher_text + "\n");

    System.out.println("Original/Decrypted Text : "
        + originalText(cipher_text, key));
    }
}
```

NAME: Sarvesh Patil
CLASS: D15A
ROLL NO: 52

RESULTS:

TEST CASE 1:

```
❯ cd "/Users/sarveshpatil/Desktop/java/" && javac VigenereCipher.java && java VigenereCipher
Enter Plaintext:
sarveshpatil
Enter key:
d15a
Generated key : D15AD15AD15A
Ciphertext : VKFVHCVPDDWL

Original/Decrypted Text : SARVESHPATIL
```

TEST CASE 2:

```
❯ cd "/Users/sarveshpatil/Desktop/java/" && javac VigenereCipher.java && java VigenereCipher
Enter Plaintext:
vesitchembur
Enter key:
sarvesh
Generated key : SARVESHSARVE
Ciphertext : NEJDXUOWMSPV

Original/Decrypted Text : VESITCHEMBUR
```

CONCLUSION:

We have encrypted the plaintext and converted it into Vigenere Cipher and decode the same to obtain the plaintext back . We encrypted the plaintext using the key which repeats itself according to the length of the paintext.