

Assignment-1

Keshav Gambhir (2019249)
Tanmay Rajore (2019118)

Polyalphabetic Substitution Cipher

- Monoalphabetic substitution with increased number of substitutions

$$p = p_0 p_1 p_2 \dots p_n$$

$$k = k_0 k_1 k_2 \dots k_m$$

Making the length of k equal to length of p

$$c = (p_0 + k_0)(p_1 + k_1) \dots (p_m + k_m)(p_{m+1} + k_0) \dots (p_{2m} + k_m) \dots$$

$c \rightarrow$ cipher text

$p_i \rightarrow$ alphabet in plain text


$k_i \rightarrow$ alphabet in key

$+$ \rightarrow modulo 26 addition

Vigenere Cipher


	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

Construction of Vigenere Cipher Table




```
1  def constructSquare():
2      global forwardMapping,reverseMapping
3      table = []
4      for i in range(0,26):
5          cntRow = []
6          for j in range (0,26):
7              cipherNumber = (j + i)%26
8              cntRow.append(reverseMapping[cipherNumber])
9          table.append(cntRow)
10     return table
```

Encryption Algorithm




```
1 def constructSameLengthKey(plainText,p_key):
2     q = int(len(plainText)/len(p_key))
3     r = int(len(plainText)%len(p_key))
4     key = p_key*q
5     key += p_key[:r]
6     return key
7
8 def encrypt(plainText,p_key):
9     encryptionKey = constructSameLengthKey(plainText,p_key)
10    table = constructSquare()
11    cipherText = ""
12    for char,keyChar in zip(plainText,encryptionKey):
13        cipherText += table[forwardMapping[keyChar]][forwardMapping[char]]
14    return cipherText
15
```

Decryption Algorithm




```
1 def constructSameLengthKey(cipherText, p_key):
2     q = int(len(cipherText)/len(p_key))
3     r = int(len(cipherText) % len(p_key))
4     key = p_key*q
5     key += p_key[:r]
6     return key
7
8
9 def decrypt(cipherText, p_key):
10    key = constructSameLengthKey(cipherText, p_key)
11    table = constructSquare()
12    plainText = ""
13    for keyChar, char in zip(key, cipherText):
14        plainText += reverseMapping[table[forwardMapping[keyChar]].index(char)]
15    return plainText
```

Brute Force Algorithm



```
1 def bruteForceKeyLength1(cipherTextList):
2     for i in range(0, 26):
3         generatedKey = reverseMapping[i]
4         if (testBruteForcedKey(cipherTextList, generatedKey)):
5             return True, generatedKey
6     print("Key Length not equal to 1")
7     return False, None
```



```
1 def bruteForceKeyLength2(cipherTextList):
2     for i in range(0, 26):
3         for j in range(0, 26):
4             generatedKey = reverseMapping[i] + reverseMapping[j]
5             if (testBruteForcedKey(cipherTextList, generatedKey)):
6                 return True, generatedKey
7     print("Key Length not equal to 2")
8     return False, None
9
```



```
1 def bruteForceKeyLength3(cipherTextList):
2     for i in range(0, 26):
3         for j in range(0, 26):
4             for k in range(0, 26):
5                 generatedKey = reverseMapping[i] + \
6                     reverseMapping[j] + reverseMapping[k]
7                 if (testBruteForcedKey(cipherTextList, generatedKey)):
8                     return True, generatedKey
9     print("Key Length not equal to 3")
10    return False, None
11
```




```
1 def bruteForceKeyLength4(cipherTextList):
2     for i in range(0, 26):
3         for j in range(0, 26):
4             for k in range(0, 26):
5                 for l in range(0, 26):
6                     generatedKey = reverseMapping[i] + reverseMapping[j] + reverseMapping[k] + reverseMapping[l]
7                     if (testBruteForcedKey(cipherTextList, generatedKey)):
8                         return True, generatedKey
9     print("Key Length not equal to 4")
10    return False, None
```


Key Testing



```
1 def testBruteForcedKey(cipherTextList, generatedKey):
2     for cipherTextWithHash in cipherTextList:
3         cipherText, appendedHash = seperateHashAndCipherText(cipherTextWithHash)
4         decryptedPlainText = decrypt(cipherText, generatedKey)
5         if appendedHash != getHashedString(decryptedPlainText):
6             return False
7     return True
```

Chosen Test Cases and key



```
1 def main():
2     plainTextList = ["keshav", "tanmayrajore", "helloworldthisiskeshav",
3                     "hellothisiskeshavandwearetestingbruteforcingakey", "thisisthefinalteststringwhichisgreaterthanthepreviousone"]
4     key = "lopr"
```

Result

```
C:\Users\kesha\Desktop\NSC Assignment-1>python main.py
keshav:    vshyljac877a7dcb8adb06cf81d1edca866e6ea5116766770b5f5fe06b5371c03ad043
tanmayrajore:    eocdlmgrucgv967f318cb0f01cd33419ce9cae1db35c5511a299bd762d5b560afc4a9659904c
helloworldthisiskeshav:    ssaczkdriyrtgxjvshyljb98c321e774c746baf3e9440adc9a40bc6ed3c31aaa179c4f05b0d6a053d4d83
hellothisiskeshavandwearetestingbruteforcingakey:    ssaczhwzdwhbpgwrgocuhspihjtjwecxmfmjkptdinwxclytp6a3a6250a850045c3ae8eb26a481ca68d7f7878d9aa2b020e6a7498
02204d4f0
thisisthefinalteststringwhichisgreaterthanthepreviousone:    evxjtgipytxelzivdhhkcwcxhvxtswhxcspkpfyilbiypdgvgwdldccvf041bced48ca40d65bf625da3214689332c1ff1
9e142f8059574a604c2f80abf

keshav : keshav : True
tanmayrajore : tanmayrajore : True
helloworldthisiskeshav : helloworldthisiskeshav : True
hellothisiskeshavandwearetestingbruteforcingakey : hellothisiskeshavandwearetestingbruteforcingakey : True
thisisthefinalteststringwhichisgreaterthanthepreviousone : thisisthefinalteststringwhichisgreaterthanthepreviousone : True

Key Length not equal to 1
Key Length not equal to 2
Key Length not equal to 3
(True, 'lopr')
```