Assignment-1

Keshav Gambhir (2019249)
Tanmay Rajore (2019118)

Polyalphabetic Substitution Cipher

- Monoalphabetic substitution with increased number of substitutions

p = p0p1p2…….pn

k = k0k1k2……..km

Making the length of k equal to length of p

c = (p0+k0)(p1+k1)......(pm+km)(pm+1+k0).......(p2m + km)…..

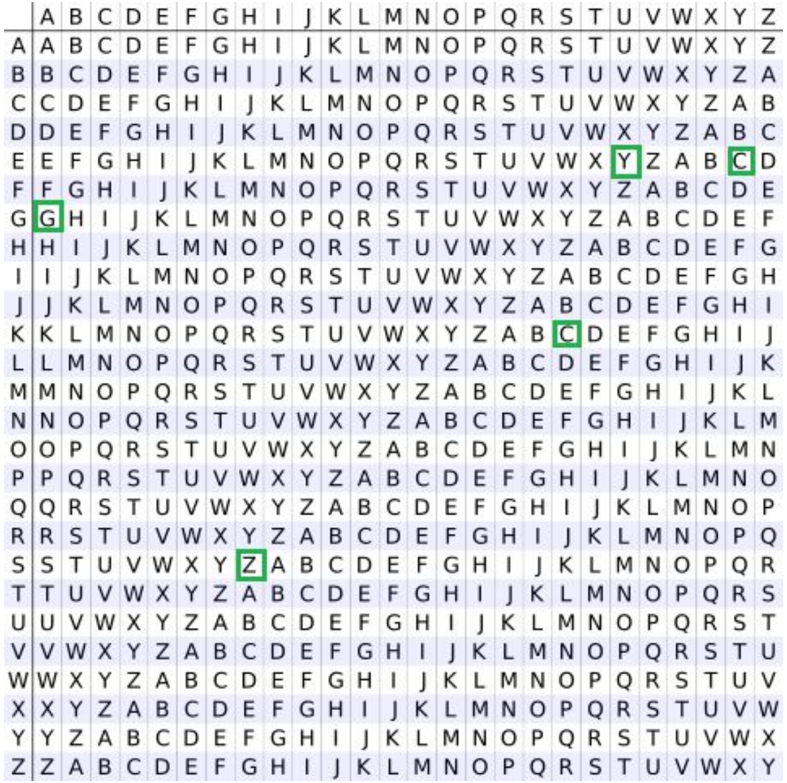
c → cipher text

pi → alphabet in plain text

ki → alphabet in key

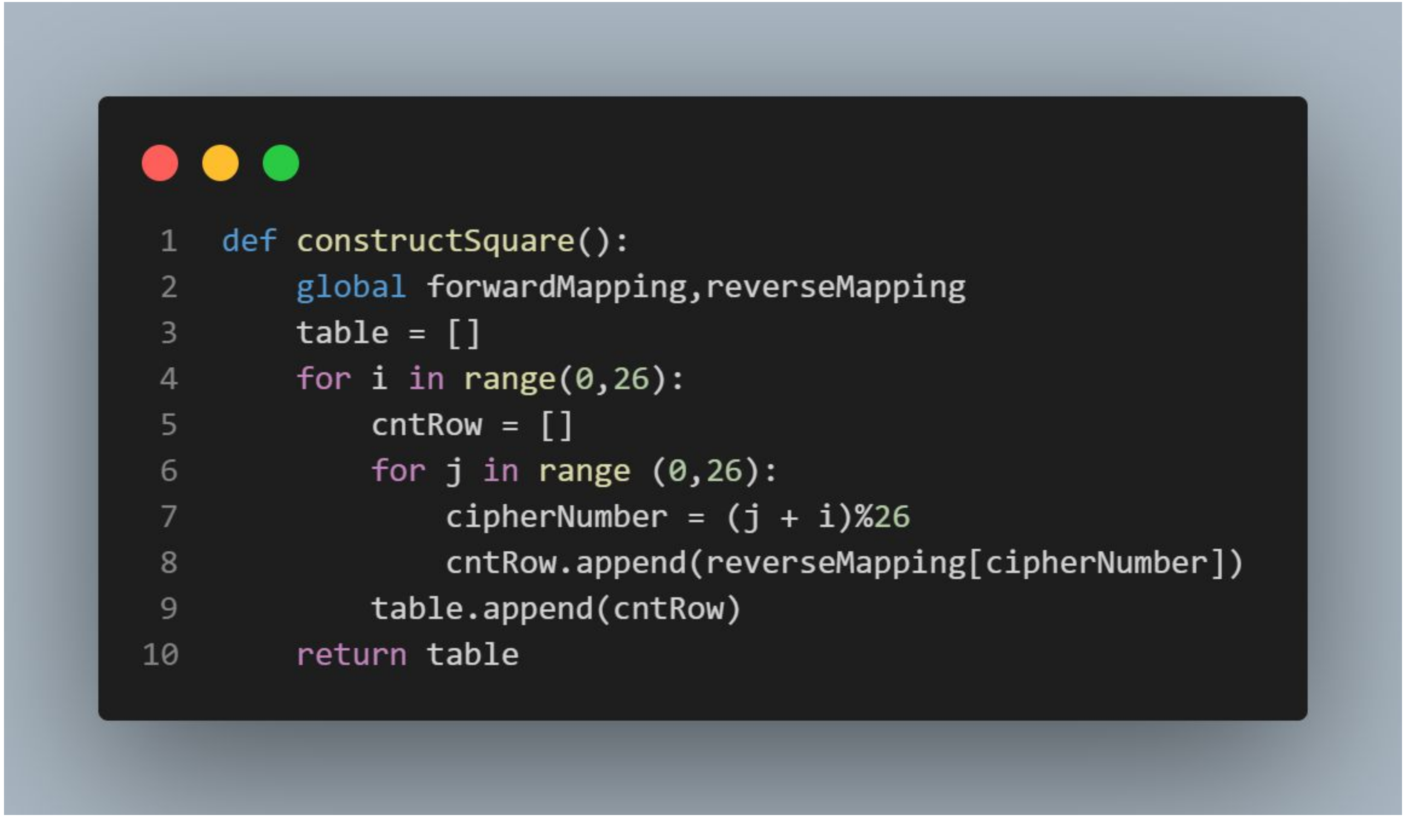
+ → modulo 26 addition

Vigenere Cipher



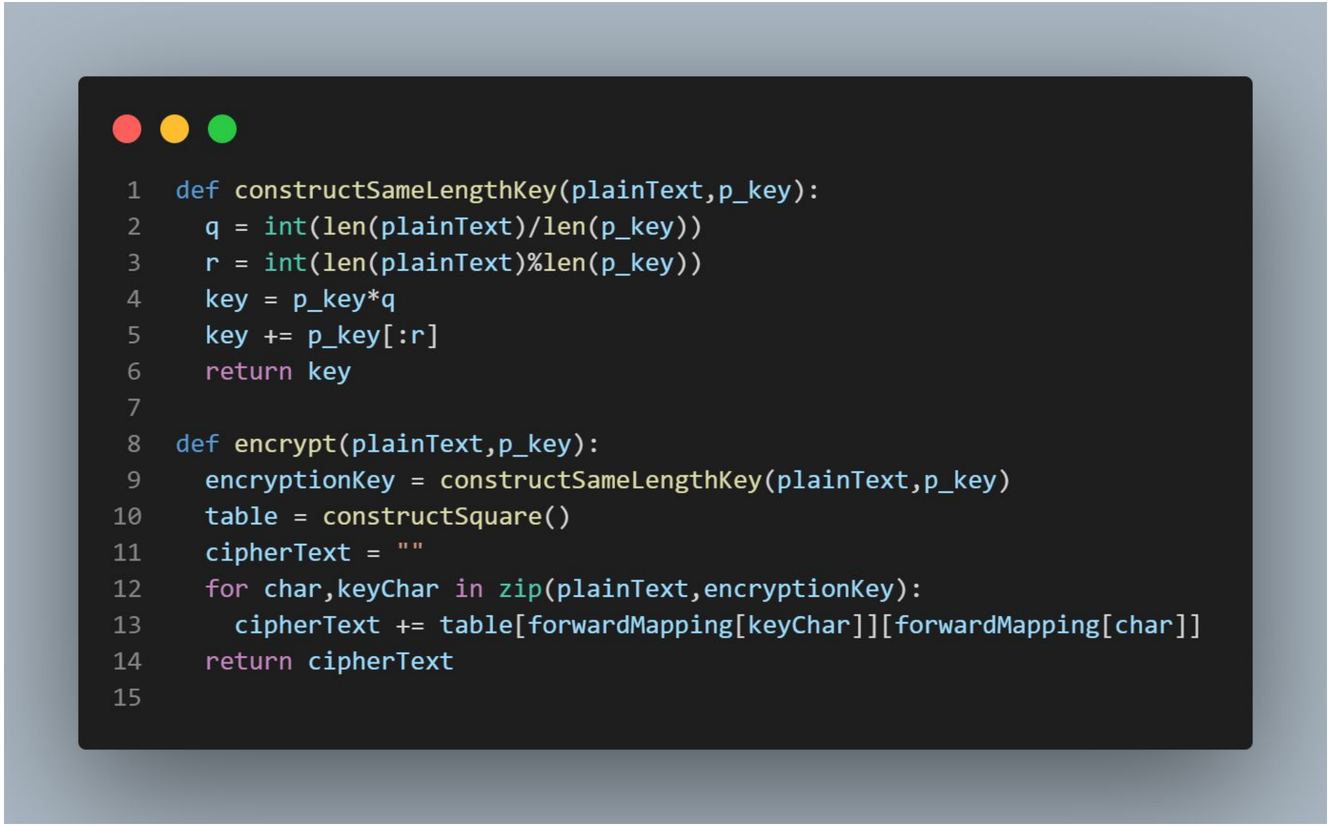
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T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



```python
def constructSquare():
    global forwardMapping,reverseMapping
    table = []
    for i in range(0,26):
        cntRow = []
        for j in range (0,26):
            cipherNumber = (j + i)%26
            cntRow.append(reverseMapping[cipherNumber])
        table.append(cntRow)
    return table
```

Encryption Algorithm

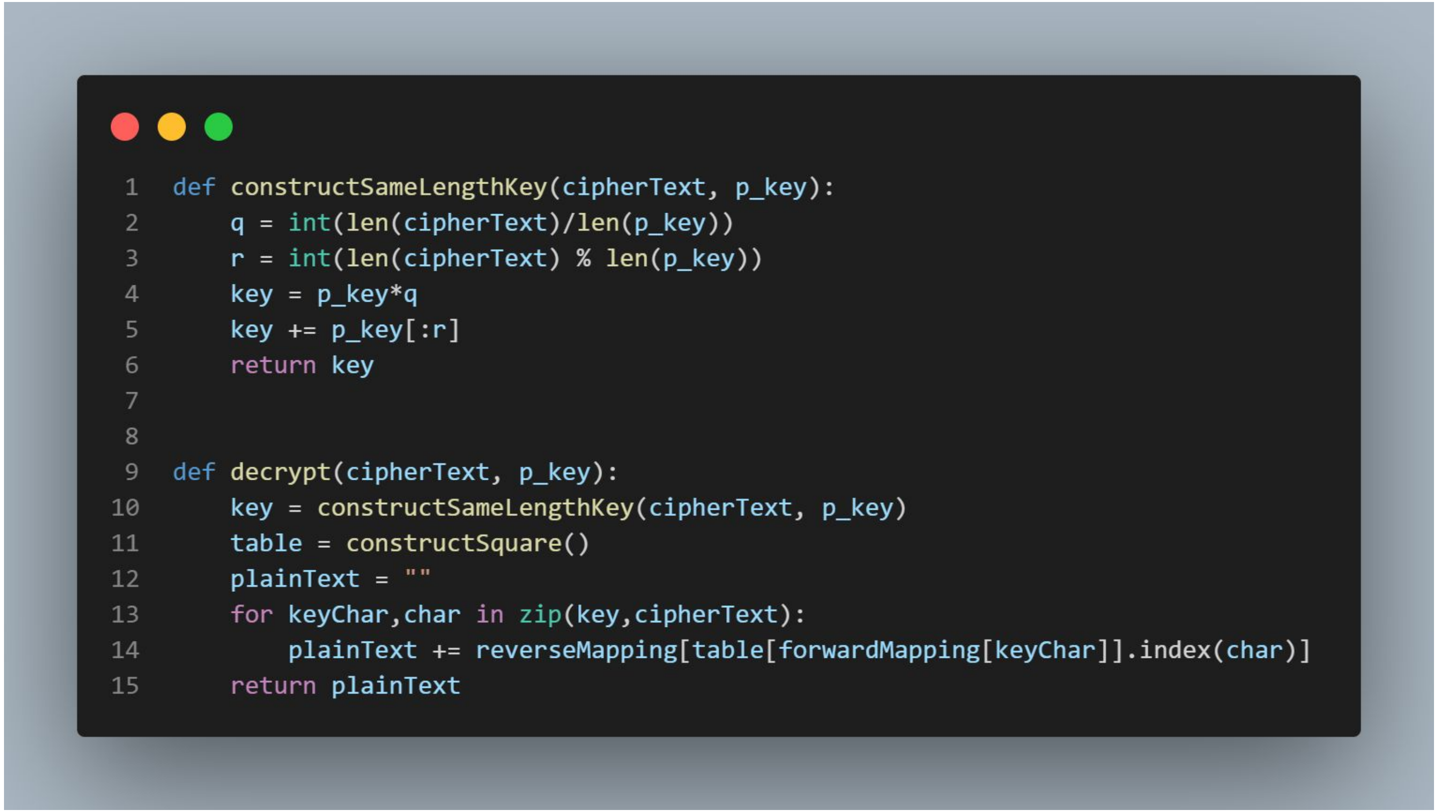


```python
def constructSameLengthKey(plainText,p_key):
  q = int(len(plainText)/len(p_key))
  r = int(len(plainText)%len(p_key))
  key = p_key*q
  key += p_key[:r]
  return key

def encrypt(plainText,p_key):
  encryptionKey = constructSameLengthKey(plainText,p_key)
  table = constructSquare()
  cipherText = ""
  for char,keyChar in zip(plainText,encryptionKey):
    cipherText += table[forwardMapping[keyChar]][forwardMapping[char]]
  return cipherText

```

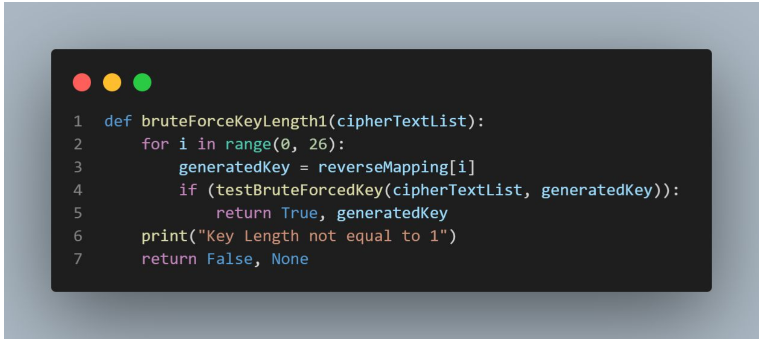
Decryption Algorithm



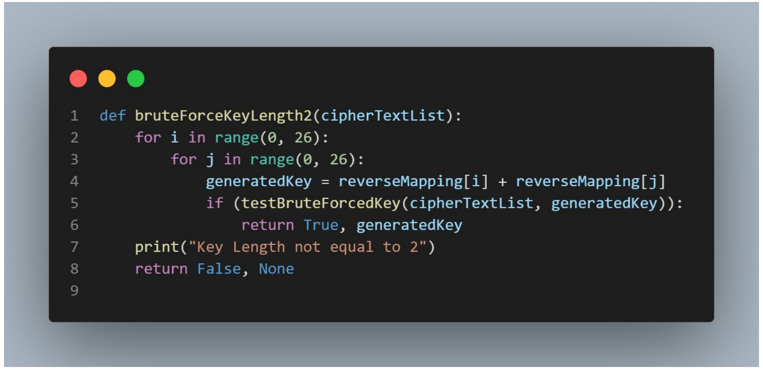
```python
def constructSameLengthKey(cipherText, p_key):
    q = int(len(cipherText)/len(p_key))
    r = int(len(cipherText) % len(p_key))
    key = p_key*q
    key += p_key[:r]
    return key


def decrypt(cipherText, p_key):
    key = constructSameLengthKey(cipherText, p_key)
    table = constructSquare()
    plainText = ""
    for keyChar,char in zip(key,cipherText):
        plainText += reverseMapping[table[forwardMapping[keyChar]].index(char)]
    return plainText
```

Brute Force Algorithm

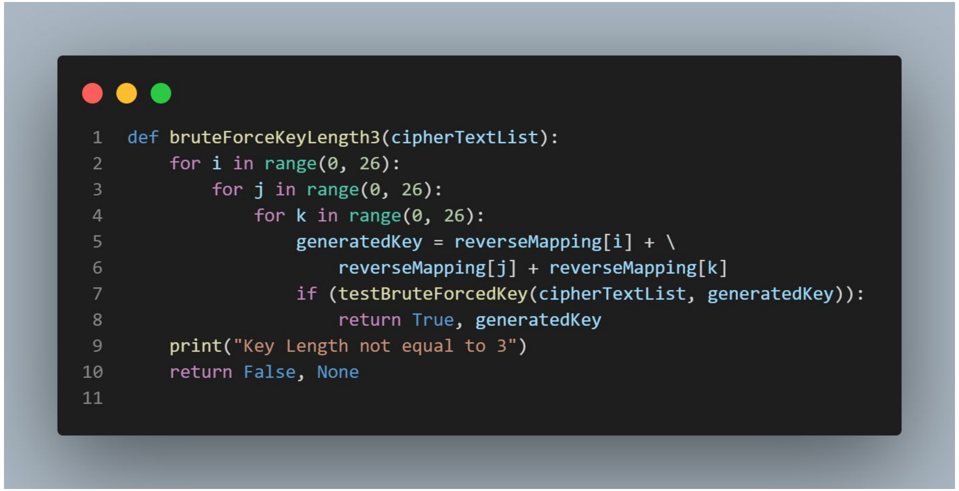


```python
def bruteForceKeyLength1(cipherTextList):
    for i in range(0, 26):
        generatedKey = reverseMapping[i]
        if (testBruteForcedKey(cipherTextList, generatedKey)):
            return True, generatedKey
    print("Key Length not equal to 1")
    return False, None
```



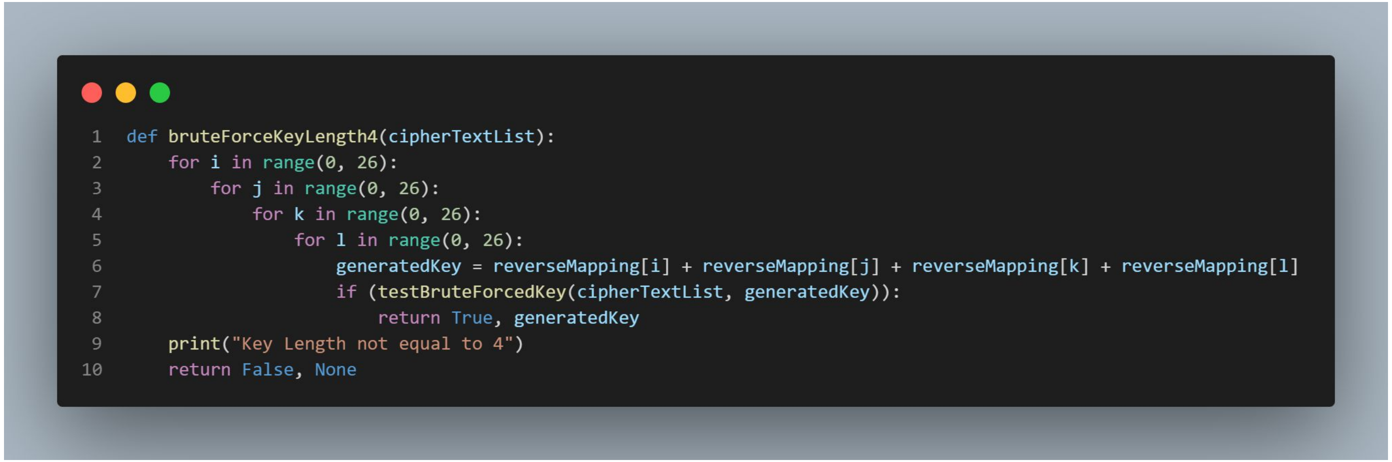
```python
def bruteForceKeyLength2(cipherTextList):
    for i in range(0, 26):
        for j in range(0, 26):
            generatedKey = reverseMapping[i] + reverseMapping[j]
            if (testBruteForcedKey(cipherTextList, generatedKey)):
                return True, generatedKey
    print("Key Length not equal to 2")
    return False, None

```



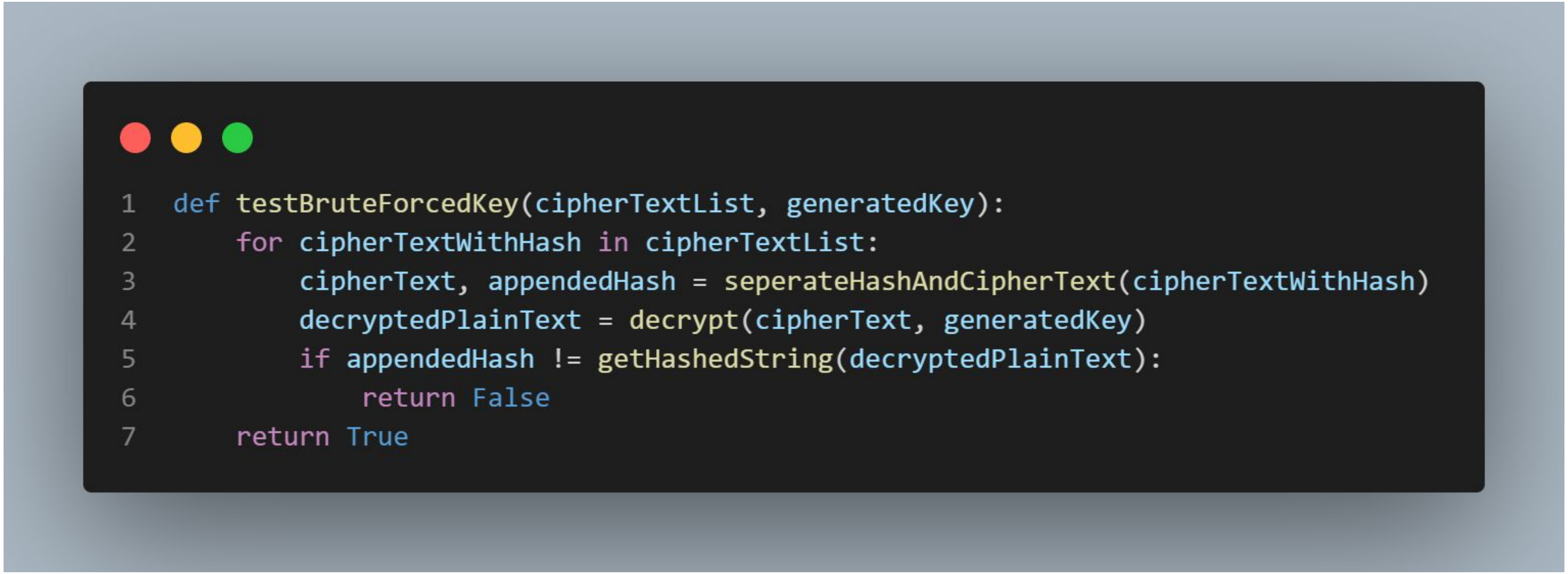
```python
def bruteForceKeyLength3(cipherTextList):
    for i in range(0, 26):
        for j in range(0, 26):
            for k in range(0, 26):
                generatedKey = reverseMapping[i] + \
                    reverseMapping[j] + reverseMapping[k]
                if (testBruteForcedKey(cipherTextList, generatedKey)):
                    return True, generatedKey
    print("Key Length not equal to 3")
    return False, None

```



```python
def bruteForceKeyLength4(cipherTextList):
    for i in range(0, 26):
        for j in range(0, 26):
            for k in range(0, 26):
                for l in range(0, 26):
                    generatedKey = reverseMapping[i] + reverseMapping[j] + reverseMapping[k] + reverseMapping[l]
                    if (testBruteForcedKey(cipherTextList, generatedKey)):
                        return True, generatedKey
    print("Key Length not equal to 4")
    return False, None
```

Key Testing



```python
def testBruteForcedKey(cipherTextList, generatedKey):
    for cipherTextWithHash in cipherTextList:
        cipherText, appendedHash = seperateHashAndCipherText(cipherTextWithHash)
        decryptedPlainText = decrypt(cipherText, generatedKey)
        if appendedHash != getHashedString(decryptedPlainText):
            return False
    return True
```