APPLIED CRYPTOGRAPHY LAB 1

CHUKWU IKECHUKWU JOHN
20231006041

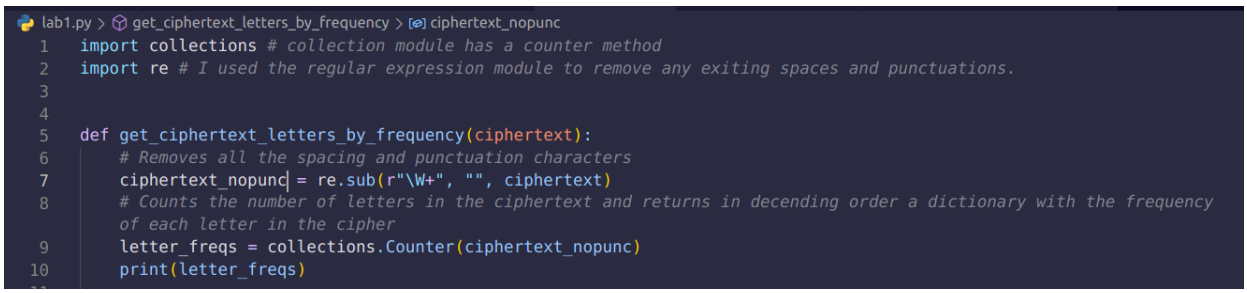
CIPHER TEXT RECEIVED:

qgshzorydzhblgahtizhjrovwrhyacbjrhtjztgchrtvdrzahjrpvizhlqgshxd
zovwigdvbszhcadorozjsvwdvzibguxdvzbrghgujvzorhtsrcizdpsvnzc
hbqacbwdznhbgcbgdrvcsvwdvzibguyarjwrhturdvnzjjcxdzubrhtrifvh
vbdzyjvcsrvjwcztzrhcbbsvwrtrbzjbsdvzbcfjztarhtbsvngdjwvovhrhtc
ugahwsrisahxsvwgovdzngdhbzyjvuavjvwylxsvzfbvzzhwu…

ANALYSIS:

STEP 1

I started by writing a Python code that read the cipher and extracted the frequency of each letter in the cipher:



Output:

Counter(
{'v': 328, 'b': 229, 'r': 214, 'h': 194, 'z': 189, 'g': 178, 'c': 176, 'd': 165, 's': 150, 'j': 122, 'w': 117, 'x': 91, 'a': 80, 't': 71, 'u': 65, 'i': 57, 'n': 50, 'f': 46, 'o': 41, 'y': 40, 'l': 40, 'p': 25, 'q': 16, 'm': 7, 'e': 2}
)

With this output, I found the letters that occurred more frequently in their order.

STEP 2

I wrote a function with an ordered list of the most commonly used letters in the English alphabet to replace the most frequent letters on my cipher with these letters.

```
14
15  def reconstruct_key(ciphertext_letters):
16      letters_ordered_by_freq = ["E", "T", "A", "O", "I", "N", "S", "R", "H", "D", "L",
17                                 "U", "C", "M", "F", "Y", "W", "G", "P", "B", "V", "K", "X", "Q", "J", "Z"]
18      return zip(ciphertext_letters, letters_ordered_by_freq)
19
20  def decipher(ciphertext, key):
21      for x, y in key:
22          ciphertext = ciphertext.replace(x, y)
23      return ciphertext
24
```

Output:

XNHOIPABRIOTVNCOMYIODAPELAOBCSTDAOMDIMNSOAMERAICODAKEYIOVXNHOURIPELYNRETHIOSCRPAPIDHELREIYTNFUREITANONFDEIPAOMHASYIRKHEWISOTXCSTLRIWOTNSTNRAESHELREIYTNFBCADLAOMFAREWIDDSURIFTAOMAYGEOETRIBDESHAEDLSIMIAOSTTHELAMATIDTHREITSGDIMCAOMTHEWNRDLEPEOAOMSFNCOLHAYHCOUHELNPERIWNROTIBDEFCEDELBVUHEIGTEIIOLFDAUKERAOMDAMHTDAOESNFU…

STEP3

This result was not helpful, so I conducted a **statistical analysis**. However, statistical analysis produced a plaintext that was still garbled, but certain words stood out, and a key pair that I used.

Stage 1

```
16
17  def reconstruct_key(ciphertext_letters):
18      letters_ordered_by_freq = ["E", "T", "A", "O", "I", "N", "S", "R", "H", "D", "L",
19                                 "U", "C", "M", "F", "Y", "W", "G", "P", "B", "V", "K", "X", "Q", "J", "Z"]
20      print(ciphertext_letters, letters_ordered_by_freq)
21      print(sorted(zip(ciphertext_letters, letters_ordered_by_freq), key=lambda x: x[1]))
22      return sorted(zip(ciphertext_letters, letters_ordered_by_freq), key=lambda x: x[1])
23
```

```
56  def print_deciphered_plaintext(type, key, plaintext):
57      header = "{} Statistical Key Reconstruction:".format(type)
58      print("=" * len(header), "\n", header, "\n", "=" * len(header))
59      print("\n", "Key:", "\n")
60      print(key, "\n")
61      i = 0
62      for x, y in key:
63          print("{} = {},".format(y, x)),
64          i += 1
65          if not i % 5:
66              print
67      print("\n", plaintext)
68
```

Generated Key Pair based on my statistical analysis:

[('r', 'A'), ('y', 'B'), ('a', 'C'), ('j', 'D'), ('v', 'E'), ('u', 'F'), ('f', 'G'), ('s', 'H'), ('z', 'I'), ('e', 'J'), ('p', 'K'), ('w', 'L'), ('t', 'M'), ('g', 'N'), ('h', 'O'), ('o', 'P'), ('m', 'Q'), ('d', 'R'), ('c', 'S'), ('b', 'T'), ('x', 'U'), ('l', 'V'), ('n', 'W'), ('q', 'X'), ('i', 'Y')]

STEP 4

Finally, I wrote a function that returned this key. I manually kept on tweaking the keys until I found the perfect pairs.

Version 1.0

```
def get_manually_tweaked_key():
    return [
        ('r', 'V'), ('b', 'T'), ('s', 'H'),
        ('z', 'A'), ('q', 'J'), ('i', 'Y'),
        ('g', 'O'), ('v', 'E'), ('h', 'N'),
        ('r', 'A'), ('y', 'B'), ('a', 'C'),
        ('j', 'D'), ('v', 'E'), ('u', 'F'),
        ('f', 'G'), ('s', 'H'), ('z', 'I'),
        ('e', 'J'), ('p', 'K'), ('w', 'L'),
        ('t', 'M'), ('g', 'N'), ('h', 'O'),
        ('o', 'P'), ('m', 'Q'), ('d', 'R'),
        ('c', 'S'), ('b', 'T'), ('x', 'U'),
        ('l', 'V'), ('n', 'W'), ('q', 'X'),
    ]
```

JOHNA**PVBRANT**VOCNMYANDVPELVNBCSTDVNMDAMOSNV
MERVACNDVKEYANV**JOHN**URAPELYORETHANSCRPVPADH
ELREAYTOFUREATVONOFDEAPVNMHVSYARK**HEWAS**NTJC
STLRAWNTOSTORVESHELREAYT**OF**BCVDLVNMFVREWADD
SURAFTVNMVYGENETRABDESHVEDLSAMAVNSTTHELVMVT
ADTHREATSGDAMCVNMTHEWORDLEPENVNMSFOCNLH…

Version 2.0

```
def get_manually_tweaked_key():
    return [
        ('r', 'I'), ('b', 'T'), ('s', 'H'),
        ('z', 'A'), ('q', 'J'), ('i', 'Y'),
        ('g', 'O'), ('v', 'E'), ('h', 'N'),
        ('r', 'A'), ('y', 'B'), ('a', 'C'),
        ('j', 'D'), ('v', 'E'), ('u', 'F'),
        ('f', 'G'), ('s', 'H'), ('z', 'I'),
        ('e', 'J'), ('p', 'K'), ('w', 'L'),
        ('t', 'M'), ('g', 'N'), ('h', 'O'),
        ('o', 'V'), ('m', 'Q'), ('d', 'R'),
        ('c', 'S'), ('b', 'T'), ('x', 'U'),
        ('l', 'V'), ('n', 'W'), ('q', 'X'),
    ]
```

JOHNAVIBRANTVOCNMYANDIVELINBCSTDINMDAMOSNIMERIACNDIKEYANVJOHNURAVELYORETHANSCRVIVADHELREAYTOFUREATIONOFDEAVINMHISYARKHEWASNTJCSTLRAWNTOSTORIESHELREAYTOFBCIDLINMFIREWADDSURAFTINMIYGENETRABDESHIEDLSAMAINSTTHELIMITAD**THREATS**GDAMCINMTHEWORDLEVENINMSFOCNLHIYHCNUHELOVERAWORNTABDEFCEDELBVUHEAGTEAANLFDIUKERINMDIMHTDINESOFUOLEFDOWELFROYHISFINMERTIGSEAUHONEABRIUKBCIDLINMHISLIM

Version 3.0

```python
def get_manually_tweaked_key():
    return [
        ('r', 'I'), ('b', 'T'), ('s', 'H'),
        ('z', 'A'), ('q', 'J'), ('i', 'M'),
        ('g', 'O'), ('v', 'E'), ('h', 'N'),
        ('y', 'B'), ('a', 'U'),
        ('j', 'L'), ('v', 'E'), ('u', 'A'),
        ('f', 'U'), ('s', 'H'), ('z', 'I'),
        ('e', 'B'), ('p', 'K'), ('w', 'D'),
        ('t', 'G'), ('g', 'N'), ('h', 'O'),
        ('o', 'V'), ('m', 'Q'), ('d', 'R'),
        ('c', 'S'), ('b', 'T'), ('x', 'C'),
        ('l', 'Y'), ('n', 'W'), ('q', 'X'),
    ]
```

JOHNAVIBRANTYOUNGMANLIVEDINBUSTLINGLAGOSNIGERIAUNLIKEMANYJOHNCRAVEDMORETHANSURVIVALHEDREAMTOACREATIONOALEAVINGHISMARKHEWASNTJUSTDRAWNTOSTORIESHEDREAMTOABUILDINGAIREWALLSCRAATINGIMUENETRABLESHIELDSAGAINSTTHEDIGITALTHREATSULAGUINGTHEWORLDEVENINGSA

…

RECOVERED PLAINTEXT

JOHNAVIBRANTYOUNGMANLIVEDINBUSTLINGLAGOSNIGERIAUNLIKEMANYJOHNCRAVEDMORETHANSURVIVALHEDREAMTOFCREATIONOFLEAVINGHISMARKHEWASNTJUSTDRAWNTOSTORIESHEDREAMTOFBUILDINGFIREWALLS…