2. Vigenere暗号解読

2.1. 解読した平文と暗号鍵

平文はintelligentinformationengineeringで,

Intelligent information engineering と読むことができ, インテリジェント情報工学と訳すことができた.

暗号鍵はVIGENERECODEである.

2.2. 解読の手順とソースコード

「最初の8文字は INTELLIG」と与えられていたのでリスト1のAttack関数で平文の最初4文字がi,n,t,eとなる鍵を見つけることにした. 8文字すべてを探すことも考慮したが, for文を8回回すのに時間がかかりすぎるため4文字ずつにしようと考えた.

最初の文字が「i,n,t,e,l,l,I,g」 となる鍵が「VIGENERE」だと分かったので, 次に続く文字が「e,n,t」か「e,n,c,e」なのではないかと推測し, 順々に試した. 「e,n,t」が後に続くと推測したところ複数のリスト1のコードの出力結果のうち, 表１のような結果がでた.

　最後に, tの次に来るのが「i」だと推測し,当てはめたところうまく解読できた.

リスト1に今回使ったコードを示す.使用言語はPythonで, 実行環境はWindows10である.

表１

|  |
| --- |
| Key might be VIGENERECODZ  Plaintext might be intelligentnnformationesgineering...  intelligentnnformationesgineering |

リスト1 Pythonによる暗号解読

|  |  |
| --- | --- |
| 1 | # 復号アルゴリズム |
| 2 | def Dec(cipher, key, table): |
| 3 | plain = "" |
| 4 | text\_l = "abcdefghijklmnopqrstuvwxyz" |
| 5 | text\_u = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" |
| 6 | for i,c in enumerate(cipher): |
| 7 | plain += text\_l[table[text\_u.index(key[i % len(key)])].index(c)] |
| 8 | return plain |
| 9 | def TableGen(): |
| 10 | table = [] |
| 11 | text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" |
| 12 | for i in range(26): |
| 13 | table.append(text) |
| 14 | text = text[1:] + text[0] |
| 15 | return table |
| 16 | def Attack(cipher): |
| 17 | cipher = cipher |
| 18 | alphabet\_upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" |
| 19 | #DVZIYPZKGBWMINUVZEKMQBHRBQTIRVZRI |
| 20 | for letter\_1 in alphabet\_upper: |
| 21 | for letter\_2 in alphabet\_upper: |
| 22 | for letter\_3 in alphabet\_upper: |
| 23 | for letter\_4 in alphabet\_upper: |
| 24 | tmp\_key = letter\_1 + letter\_2 + letter\_3 + letter\_4 |
| 25 | tmp\_plain = Dec(cipher, tmp\_key, TableGen()) |
| 26 | if ("i" in tmp\_plain[0]) and ("n" in tmp\_plain[1]) and ("t" in tmp\_plain[2]) and ("e" in tmp\_plain[3]): |
| 27 | print("Key might be {}".format(tmp\_key)) |
| 28 | print("Plaintext might be {}".format(tmp\_plain[:40] + "...")) |
| 29 | print(tmp\_plain[0:]) |
| 30 | return tmp\_plain |
| 31 | def Attack2(cipher): |
| 32 | cipher = cipher |
| 33 | alphabet\_upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" |
| 34 | #DVZIYPZKGBWMINUVZEKMQBHRBQTIRVZRI |
| 35 | for letter\_1 in alphabet\_upper: |
| 36 | for letter\_2 in alphabet\_upper: |
| 37 | for letter\_3 in alphabet\_upper: |
| 38 | for letter\_4 in alphabet\_upper: |
| 39 | tmp\_key = "VIGENERE"+letter\_1 + letter\_2 + letter\_3 + letter\_4 |
| 40 | tmp\_plain = Dec(cipher, tmp\_key, TableGen()) |
| 41 | #if ("l" in tmp\_plain[4]) and ("l" in tmp\_plain[5]) and ("i" in tmp\_plain[6]) and ("g" in tmp\_plain[7]): |
| 42 | if ("e" in tmp\_plain[8]) and ("n" in tmp\_plain[9]) and ("t" in tmp\_plain[10]) and ("i" in tmp\_plain[11]): |
| 43 | print("Key might be {}".format(tmp\_key)) |
| 44 | print("Plaintext might be {}".format(tmp\_plain[:40] + "...")) |
| 45 | print(tmp\_plain[0:]) |
| 46 | print() |
| 47 | def Attack3(cipher): |
| 48 | cipher = cipher |
| 49 | alphabet\_upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" |
| 50 | #DVZIYPZKGBWMINUVZEKMQBHRBQTIRVZRI |
| 51 | for letter\_1 in alphabet\_upper: |
| 52 | for letter\_2 in alphabet\_upper: |
| 53 | for letter\_3 in alphabet\_upper: |
| 54 | tmp\_key = letter\_1 + letter\_2 + letter\_3 |
| 55 | tmp\_plain = Dec(cipher, tmp\_key, TableGen()) |
| 56 | if ("e" in tmp\_plain[0]) : |
| 57 | print("Key might be {}".format(tmp\_key)) |
| 58 | print("Plaintext might be {}".format(tmp\_plain[:40] + "...")) |
| 59 | print(tmp\_plain[0:]) |
| 60 | if \_\_name\_\_=="\_\_main\_\_": |
| 61 | cipher = "DVZIYPZKGBWMINUVZEKMQBHRBQTIRVZRI" |
| 62 | cipher2="dhtgltqinforeweivtbnginewntnn" |
| 63 | cipher3="txfivjdrmatidxqnomceerinv" |
| 64 | #cipher = Attack(cipher) |
| 65 | #cipher\_2=Attack2(cipher2.upper()) |
| 66 | cipher3 = Attack2(cipher) |