DTTF/NB479: Dszquphsbqiz Day 5

- Announcements:
  - Please pass in Assignment 1 now.
  - Assignment 2 posted (when due?)
- Questions?
- Roll Call
- Today: Vigenere ciphers
  - Invented in 1553 by Bellaso, not Vigenere

Vigenere Ciphers

- Idea: the key is a *vector* of shifts
  - The key and its length are unknown to Eve
- Encryption:
  - Repeat the vector as many times as needed to get the same length as the plaintext
  - Add this repeated vector to the plaintext.
- Example:
  - Key = *hidden (7 8 3 3 4 13)*.

Key

- `The recent development of various methods of`

```
 7 8 3   3 413 7 8 3   3 413 7 8 3 3 413 7 8   3 3   413 7 8 3 3 4  13 7 8 3 3 413   7 8
 015 7  20 815112122   6 8 811191718161720 1  17 8  25132416172322  2511 11017 7 5  2113
```

- `aph uiplvw giiltrsqrub ri znyqrxw zlbkrhf vn`

  - Demo

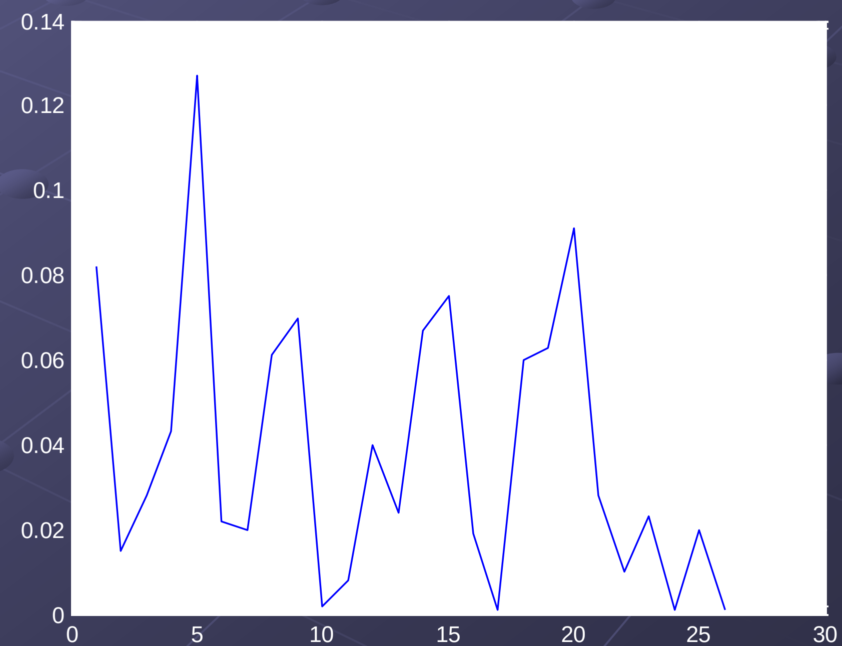
Security

- The shift vector isn't known (of course)
- 1. It's length isn't even known!
- 2. With shift ciphers, the most frequent cipher letter is probably e.
  - But here, e maps to H, I, L, … (**spread out!**)
- Consider 4 attacks:
  - Known plaintext?
  - Chosen plaintext?
  - Chosen ciphertext?
  - Ciphertext only?

English letter frequencies

A 0.082	H 0.061	O 0.075	U 0.028
B 0.015	I 0.070	P 0.019	V 0.010
C 0.028	J 0.002	Q 0.001	W 0.023
D 0.043	K 0.008	R 0.060	X 0.001
E 0.127	L 0.040	S 0.063	Y 0.020
F 0.022	M 0.024	T 0.091	Z 0.001
G 0.020	N 0.067		

Graph:



Ciphertext-only attack

- Assume you know the key length, L.
- Make any other assumptions you need.
- Take 3-4 min with a partner and devise a method to break Vigenere.

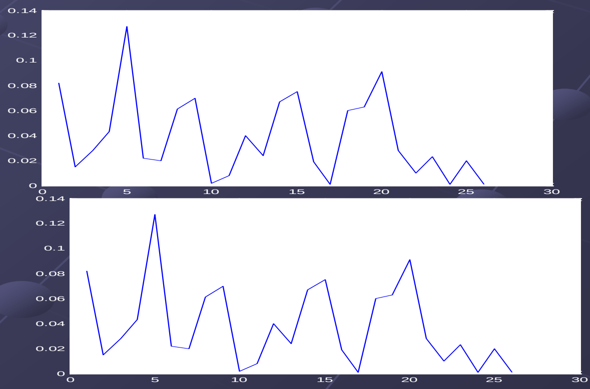
Perhaps yours looks something like this?

- Assume we know the key length, L, …
  - We'll see how to find it shortly
- Method 1:
  - Parse out the characters at positions p = *i* (mod L)
    - These have all been shifted the same amount
    - Do a frequency analysis to find shift
      - The most frequent letter should be e, given enough text. Can verify to see how shift affects other letters
  - This gives the first letter of the key
  - Repeat for positions p = 2, p = 3, … p = L
  - Problem: involves some trial and error.
  - For brute force to work, would need to brute force all letters of key simultaneously: _____ possibilities

Dot products

$$A \cdot B = A.* B = \sum_i A_i B_i$$

- Consider A = (0.082 0.015 0.028 0.043 0.127 0.022 0.020 0.061 0.070 0.002 0.008 0.040 0.024 0.067 0.075 0.019 0.001 0.060 0.063 0.091 . 0.028 0.010 0.023 0.001 0.020 0.001);
- $A_i = A$ displaced *i* positions to the right
- A_0 = (0.082 0.015 0.028 … 0.001 0.020 0.001)
- A_1 = (0.001 0.082 0.015 0.028 … 0.023 0.001 0.020)
- A_2 = (0.020 0.001 0.082 0.015 0.028 … 0.023 0.001)
- $A_0 .* A_1 = 0.039$
- $A_0 .* A_0 = 0.066$
- $A_i .* A_j$ depends on _____ only.
- Max occurs when _____.
  3 reasons why:



Towards another method

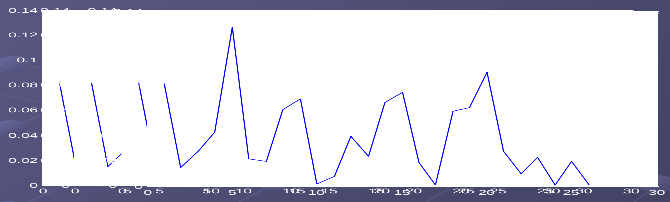
- Method 1
  - Parse out the characters at positions p = 1 (mod L)
    - These have all been shifted the same amount
    - Do a frequency analysis to find shift
      - The most frequent letter should be e, given enough text. Can verify to see how shift affects other letters.
  - This gives the first letter of the key
  - Repeat for positions p = 2, p = 3, … p = L

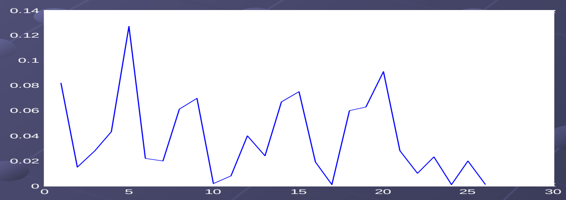
Another method

- Method 2
  - Parse out the characters at positions p = 1 (mod L)
    - These have all been shifted the same amount
    - Get the whole freq. distribution **W = (0.05, 0.002, …)**
      - W approximates A. Calculate $W \cdot A_i \;\; for\, 0 \le i \le 25$
      - Max occurs when we got the shift correct.
  - This gives the first letter of the key
  - Repeat for positions p = 2, p = 3, … p = L
  - Demo

Method 2 is more robust since it uses the whole letter distribution

- Find dot product of A_i: and W:





More robust than just using 1 letter (‘e’)…

…but harder to compute by hand.

Finding the key length

- What if the frequency of letters in the plaintext approximates A?
- Then for each k, the frequency of each group of letters in position p = k (mod L) in the ciphertext approximates A.
- Then loop, displacing the ciphertext by i, and counting the number of matches.
  - Get max when displace by correct key length
  - So just look for the max number of matches!

```
                                          displacement
APHUIPLVWGIILTRSQRUBRIZNYQRXWZLBKRHFVN    (0)
NAPHUIPLVWGIILTRSQRUBRIZNYQRXWZLBKRHFV    (1)   1 match
VNAPHUIPLVWGIILTRSQRUBRIZNYQRXWZLBKRHF    (2)   0 matches
…
KRHFVNAPHUIPLVWGIILTRSQRUBRIZNYQRXWZLB    (6)   5 matches
…
```

Key length: an example

Take any random pair in the ciphertext:

- The letter in the top row is shifted by i (say 0)
- The letter in the bottom row is shifted by j (say 2)

Prob(both 'A') = P('a')*P('y') = 0.082 * 0.020

Prob(both 'B') = P('b')*P('z') = 0.015 * 0.001

Prob both same (any letter) is ___ or generally ___

Recall, this is maximum when ______

When are each letter in the top and bottom rows shifted by same amount?

A_0 = (0.082 0.015 0.028 … 0.001 0.020 0.001)

A_2 = (0.020 0.001 0.082 0.015 0.028 … 0.023 0.001)

Still a bit fuzzy?

- Nothing like implementation to aid understanding!
  - Homework 2: Program it
  - Third week programming quiz: use your program to decrypt a message