NWEN 243 Lab 2 – Frequency Analysis (10%)

This project must be electronically submitted by the 7th August 2015, 23:59.

Experience thinking about crypto-analysis

- Write a program to perform frequency analysis of a block of cyphertext
- Determine (by bruteforce) how many keys were used to encode the cyphertext using polyalphabetic Caeser cypher
- Answer some related topic questions, see below.

Requirements

- This lab an individual lab written in C.
- You will be writing programs that you execute from the shell command line.
- You must submit your code by the 7th August 23:59.
- The code will be marked by script, so:
    - You must NOT change the command line interface
    - Your program MUST only output the specified output – any debugging output you have added must be removed before submission (or output on stderr), or your code will fail the tests. **This is really important.**
    - **YOUR CODE MUST COMPILE**, or you will get 0 for coding. See submission section.

- You are to complete a C program that will allow you to determine the number of keys used in a polyalphabetic cypher, once this has been discovered – the problem becomes one of simple frequency analysis. As we have powerful computers, this is easy to brute force. The idea is to simply try different numbers of keys until we find one that gives a 'mostly' decoded result. That will give us the number of keys. For example, if two keys are used, then all the even numbered characters in the text are in one sub-cyphertext, while all the odd numbered characters are in the other sub-cyphertext. Three keys, would mean the cyphertext split 3 ways, etc.

    1. For each possible number of keys, i
    2. Split the cyphertext into i sub-cyphertexts and apply frequency analysis to each of them
    3. Recomine the sub-cyphertexts the entire 'partly decoded' text, for each nominated number of keys and output the result.

For example:

```
%>cat test | crack 4
```

This will try 1 through 4 keys on cyphertext "test"

Assumptions

You may assume text is in ASCII and that you do not need to preserve case (just make it all upper case) don't encode symbols, punctuation, spaces or whitespace in general (i.e just pass it through without change), e.g.: "efh'w ef xw" becomes "don't do it"

Skeleton Code

A small skeleton will be provided on the course web page to ease you into this project. No IDE is provided, the command line and any editor (vi, emacs, kwrite) will be sufficient. You can compile your code using the command line:

```
%> gcc crack.c -o crack
```

Do not use your own name!

Topic Questions (½ mark each)

Please include your answers to theses questions in a separate PDF document, and ensure you submit it at the same time as your project. ½

1. Did the modified Caeser cypher from lab 1 make it any harder to crack using frequency analysis – why?

2. Outline how your code might differ, if you were attemping to crack the vignere cypher rather than a polyalphabetic cypher?

3. Would a cypher using the correct proportinate number of homophones with a custom alphabet be immune to frequency analysis?

4. Decypher the following message, that was encyphered using the Vignere cypher and the keyword “HOUSE”:

   ```
   AVYUL HWLEE UCZLL LTYVI YOFJI ZSLNI ICUJH ZOCVC LGNWV KOSLL HHULE EWHUV
   LOMWM ZBYWH LRHGA
   ```

5. The enigma machine broke the plaintext into space seperated blocks of characters 4. Why?

Submission & Marking

- This project will be marked by your tutors using an automatic marking script. So, ensure you have followed the instructions exactly. It also MUST run on our systems, ensure all needed files are included, and that it compiles on embassy.
- Include a pdf document outlining each of your functions, the algorithm, and why you did it that way. Is it efficient, robust etc. How did you test it.
- Include a pdf document with your answers to the topic questions.
- Marking (total of 10%):
    - The questions are worth 2.5 marks
    - The Document is worth 1.5 marks
    - The WORKING code is work 6 marks – code that compiles, but fails some or all tests will be awarded partial marks. Code that fails to compile, will receive no marks for the code section, so ensure your code compiles, even if it isn't right – then at least the tutor will look at it personally.

Any problems, remember to ask your tutors and use the forum outside your lab.

Have fun,

Kris.