The University of Sydney
School of Mathematics and Statistics

Computer Tutorial 4 (Week 4)

MATH2068/2988: Number Theory and Cryptography — Semester 2, 2017

Web Page: http://www.maths.usyd.edu.au/u/UG/IM/MATH2068/
Lecturer: Dzmitry Badziahin

Start MAGMA and type `load "tut4data.txt";`.

The file tut4data.txt that you have loaded contains seven slabs of enciphered text called `ct1`, `ct2`, `ct3`, `ct4`, `ct5`, `ct6` and `ct7`. They have all been enciphered with either substitution ciphers, block transposition ciphers or Vigenère ciphers. We shall first endeavour to find out in each case which kind of cipher was used. Then, hopefully, we shall decipher at least some of them.

1. Check that `ct1` is of Type CryptTxt (via the command `Type(ct1);`) and then use the command `sct1:=String(ct1);` to create a string that is essentially the same as `ct1`. Then do the same for the others (creating `sct2` corresponding to `ct2`, and so on). Find the coincidence indices for all these strings, and use the information to identify the ones that correspond to Vigenère ciphers. (Use `CoincidenceIndex(sct1);` etc. As explained in the solutions to last week's computer tutorial, substitution ciphers do not change the CI, but Vigenère ciphers lower it. How would a transposition cipher affect the CI?)

2. We could use the method we used last week to decipher the Vigenère enciphered ones, but it is far easier to use the javascript gadget on the MATH2068 web page. Open firefox and try it. Note that there are also links you can click to get the required pieces of ciphertext. When you have found the deciphering keys in each case, write them down. (Note that it is important that you understand what is happening and why it works; so be sure to think carefully about what is going on.)

3. If you did it right, the Vigenère key finder should have told you that the deciphering key for sct5 is IOAQSNFANY. Type

```
V:=VigenereCryptosystem(10);
k:=V!"IOAQSNFANY";
Enciphering(k,Encoding(V,sct5));
```

and check that you can read the output. Then do the same for the other two.

4. You are told that the other four ciphertexts correspond to substitution ciphers and block transposition ciphers. Their CI's are therefore similar to typical values for English, i.e. about 0.065. How can you distinguish between substitution ciphers and transposition ciphers?

(The commands `SortedFreqDist(sct1);`, `SortedFreqDist(sct6);` (etc.) will help: transposition ciphers do not change the frequencies of the various letters at all.)

We shall focus on the block transposition ciphers in the remainder of this tutorial. For the substitution ciphers you could use the procedure of Computer Tutorial 2, guessing a deciphering

key from letter frequencies and then progressively altering it to increase the common-word score. There is actually a tailor-made MAGMA function in MagmaProcedures.txt that automates this procedure, though without the helpful instincts of a human English speaker: if you would like to try it, type `bk:=BestSubstitutionKey(ct1);`.

Although transposition ciphers do not change the letter frequencies, they split up digraphs. Some combinations of pairs of consecutive letters are very common and some are very rare. We shall try to decipher `sct6` by reuniting the pairs of letters that "want" to be next to each other.

Given a typical piece of English text, such as "GODSAVETHEQUEEN", we can write down the sequence of digraphs occurring in it. In this case the digraph sequence is GO, OD, DS, SA, AV, ... It makes sense to talk about the digraph coincidence index: the probability that two randomly chosen digraphs are the same. If all digraphs had the same frequency it would be $(1/26)^2 \approx 0.00148$, but for English text it is typically about 5 to 7 times larger than this. We can use this as a test of whether a sequence of pairs of letters might be a sequence of digraphs taken from typical English.

5. The plaintext corresponding to `ct6` was loaded in `tut4data.txt`, under the name `pt6`. The following command defines `d` to be the sequence of all digraphs in `pt6`:

```
d:=[pt6[i]*pt6[i+1]: i in [1..#pt6-1]];
```

Type `d[1]; d[2]; d[3];` and `d[4];` to see the first few terms of `d`. Type `pt6[1..5];` to see the first 5 letters of `pt6`, and check that it is consistent with the start of `d`. Now type

```
CoincidenceIndex(d);
```

and note that the value is indeed about 0.007.

Now redefine `d`, using `d:=[pt6[i]*pt6[i+2]: i in [1..#pt6-2]];`, so that instead of looking at a sequence of pairs adjacent letters we are looking at pairs that are "adjacent but one", and find the coincidence index again. Then look at pairs that are separated by 2 intervening letters and find the CI again, then look at pairs separated by three intervening letters and find it again. What do you conclude?

6. Assume now that `sct6` was created by splitting the corresponding plaintext into blocks of length m, and then permuting all the blocks in the same way. Suppose that the i-th and j-th letters of each block of the ciphertext were adjacent in the plaintext. Then, if c_k denotes the k-th letter of the ciphertext, the sequence of pairs of letters

$$c_i c_j \quad c_{m+i} c_{m+j} \quad c_{2m+i} c_{2m+j} \quad c_{3m+i} c_{3m+j} \quad \cdots$$

was originally a sequence of digraphs in the plaintext. So it should have coincidence index about the same as ordinary text. However, if m is replaced by some number that is not the true period, or if i and j do not correspond to positions that were adjacent in the plaintext, then the coincidence index will be much smaller.

We call the above sequence of pairs the (i, j)-*decimation of period* m for the string $c_1 c_2 c_2 \ldots$. Type

```
CoincidenceIndex(Decimation(sct6,[1,8],9));
```

to see the coincidence index of the $(1, 8)$-decimation of period 9 for `sct6`. And then type

```
CoincidenceIndex(Decimation(sct6,[8,1],9));
```

and explain why you got the same answer.

7. Since the value for the coincidence index is much lower than 0.01, we can conclude that `Decimation(sct6,[1,8],9)` is not a sequence of digraphs (or reversed digraphs) from standard text. Maybe the block length for the transposition cipher used was not 9, or maybe it was 9 but the 1st and 8th letters of each block were not adjacent in the ciphertext. Type

```
for i:=2 to 9 do
 "i =",i,": CI=",CoincidenceIndex(Decimation(sct6,[1,i],9));
end for;
```

and explain why the result suggests that the period really might be 9.

8. Assuming that the period is 9, type

```
for j:=1 to 8 do
 for i:=j+1 to 9 do
  j,i,": CI=",CoincidenceIndex(Decimation(sct6,[j,i],9));
 end for;
end for;
```

and work out which pairs of letters from each 9 letter block were adjacent in the plaintext. Put the numbers from 1 to 9 into a 9 letter sequence arranged so that numbers that want to be adjacent are adjacent. (Answer: [4,7,1,5,9,3,6,2,8] or its reverse.) Then do

```
T:=TranspositionCryptosystem(9);
key:=T![4,7,1,5,9,3,6,2,8];
Enciphering(key,Encoding(T,sct6));
```

to decipher `sct6`.

*__9.__ Decipher `sct7` by the same method. You can assume that the block size is at most 20.