The University of Sydney
School of Mathematics and Statistics

Computer Tutorial 2 (Week 2)

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. Note that it automatically loads a file called "MagmaProcedures.txt", specially created for MATH2068/2988. (If you are doing this tutorial away from the computer labs, you will have to load this file yourself with the command `load "MagmaProcedures.txt";`.) Then type the command: `load "tut2data.txt";`

1. One exceedingly trivial way to encipher a message is to write B's instead of A's, C's instead of B's, ..., Z's instead of Y's and A's instead of Z's. Or you might choose some other way of permuting the letters. Our MAGMA startup file MagmaProcedures.txt loads some functions for doing this kind of thing. Start MAGMA and type

```
C:=SubstitutionCryptosystem();
key:=RandomKey(C);
key;
```

You will see that `key` is a random permutation of the letters A to Z, chosen by MAGMA. Now let's choose a message to encipher. Type

```
secretmessage:=Encoding(C,"I've left the fifty million dollars under
your doormat");
```

Now type `secretmessage;` You will see that the `Encoding` function has changed everything to upper case and got rid of the spaces and the apostrophe. This is because the enciphering function (coming up next) can only deal with strings of uppercase letters. (It would, of course, be easy enough to make an enciphering function that can cope with other kinds of symbols too, but the function we have got is good enough for our purposes.) Type

```
disguisedmessage:=Enciphering(key,secretmessage);
disguisedmessage;
```

and examine a few letters that MAGMA has done the enciphering correctly. (To illustrate the process, consider an alphabet with just 5 letters ABCDE and suppose that the substitution key is DAEBC. This means replace A's by D's, B's by A's, etc.. The message BED would become ACB.) The following commands may be useful: `alphabet[15];` prints the 15th letter of the alphabet, and `Index(alphabet,"Q");` prints the position in the alphabet of the letter Q (for instance).

2. For a substitution cryptosystem, the deciphering process is essentially the same as the enciphering process; you just need the inverse key. (In our 5 letter alphabet example above the inverse key would be BDEAC: this changes ACB back to BED.) Type

`yek:=InverseKey(key);` and then examine a few letters to check check that `yek` is indeed the inverse of `key`. Then type the command `Enciphering(yek,disguisedmessage);` and check that you recover `secretmessage`.

3. You can specify the key. For instance, type `k:=C!"BCDEFGHIJKLMNOPQRSTUVWXYZA";` and see what `InverseKey(k);` produces. Then type `dm:=Enciphering(k,secretmessage);` and `dm;` checking that it is correct. Next, type `Enciphering(InverseKey(k),dm);` to confirm that you get `secretmessage` back.

4. What happens if you choose a key that is not a genuine permutation of the letters A to Z? Starting with `k:=C!"BBBBBGHIJKLMNOPQRSTUVWXYZA";` repeat Exercise 3 and see what (if anything) goes wrong.

5. The file tut2data.txt that you loaded defines a variable called `poem`. Type `poem;` to see it. A substitution cryptosystem called `tennyson` has been declared, and an enciphering key called `alf` has been chosen. Type

```
enc:=Encoding(tennyson,poem);
enc;
xxx:=Enciphering(alf,enc);
xxx;
```

We will try to decipher `xxx` pretending that we don't know `poem`.

6. The approximate percentages of occurrences of the various letters in a random piece of English text are given in the following table.

A	B	C	D	E	F	G	H	I	J	K	L	M
8.2	1.5	2.8	4.3	12.7	2.2	2.0	6.1	7.0	0.2	0.8	4.0	2.4
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
6.7	7.5	1.9	0.1	6.0	6.3	9.1	2.8	1.0	2.4	0.2	2.0	0.1

MagmaProcedures.txt defines a function called `SortedFreqDist` that can be used to attack a substitution cryptosystem whose key is unknown. Type `SortedFreqDist(xxx);` You will see that Z is the letter that occurs most frequently in the ciphertext, and F comes next, slightly ahead of Y. So perhaps Z represents E and either F or Y represents T.

For somewhat technical reasons that need not concern us, MAGMA makes a distinction between a piece of cryptographic text, such as `xxx`, and a string of characters. But we can create a string of characters that is really just the same as `xxx` via the command `yyy:=String(xxx);`. Do this, and then type

```
for i:=1 to #yyy-2 do
  if yyy[i] eq "F" and yyy[i+2] eq "Z" then
     print yyy[i..i+2];
  end if;
end for;
```

See what happens, and figure out what it tells us! (Hint: `THE` is probably a common sequence in the plaintext. If `T` is enciphered as `F` then `THE` will be enciphered as `F?Z`,

where `?` is the letter that represents `H`. The MAGMA code above prints out all occurrences of `F*Z` in `yyy`, where `*` is anything.) Now repeat the above commands with `F` replaced by `Y`. Which letters do you think represent `T` and `H`?

MagmaProcedures.txt defines a command `freqkey` that makes a guess at a deciphering key based on letter frequencies. Type `kk:=tennyson!freqkey(xxx);` and then `kk;`. You will see that the last letter of `kk` is E; so using `kk` as substitution key will cause Z's to be replaced by E's. This is good, because we have already decided that enciphering replaced E's by Z's. Similarly the 6th letter of `kk` is T, meaning that F's will be replaced by T's. This is also right. But we have decided that E's in the ciphertext represent H's in the plaintext; so the 5th letter of the deciphering key should be H. The 5th letter of `kk` is I. Let us modify `kk` by swapping the positions of H and I: the command `swap("I","H",~kk);` will do this. (Note the tilde before `kk` in this command.) Now type `kk;` again and check that `kk` has been correctly modified. Now do `zzz:=Enciphering(kk,xxx);` and then `zzz;`. Unfortunately, it is not very good yet!

Type `Score(zzz,Common);`. This rates `zzz` according to the number of common words appearing in it. The higher the score the better.

MagmaProcedures.txt also defines a function for finding the most frequent polygraphs – i.e. short sequences of letters – in a string. Use it to find all the trigraphs occurring in `zzz` with a relative frequency greater than 0.01: `MostCommonPolygraphs(zzz,3,0.01);` Now I think that the 2nd most frequent trigraph ought to be HER not HES, and the 3rd most frequent should be AND not ARD. So try `swap("R","N",~kk); swap("R","S",~kk);` and then `zzz:=Enciphering(kk,xxx); zzz; Score(zzz,Common);` again.

Continue swapping letters in `kk`, trying to increase the score of common words, and see if you can eventually find the correct key (without cheating).

*__7.__ Another poem by Tennyson has been enciphered using a different substitution key. Type `xsecret;` to see the ciphertext. Try deciphering it using the same method as for `poem`. (If you want to reuse some of the previous commands verbatim, you can redefine `xxx` to equal `xsecret`.)