COSC2539: Security in Computing and IT
Assignment 1
Cryptography
Semester B, 2013

Denis Rinfret

Due: end of Week 5, Sunday July 21, 2013 at 23:59.

Assignment to be done individually

1 Implementation of Ciphers

Write programs to implement the four ciphers given next. Programming languages allowed: Python, C and Java. The alphabet you should work with is

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ .,()-!?\n0123456789
```

Please note that there is a space between the `Z` and the `.` in the alphabet given above. Also, `\n` represents only one character and has the usual meaning of a new line character.

1.1 Ceasar Cipher

Write a program to implement the *Ceasar cipher*. The key should be an integer. Your program should be called `ceasar.py` if using Python, `ceasar` if using C, or `Ceasar` if using Java, and take the following command line arguments:

- `e` or `d`: encrypt or decrypt mode.
- `inputfilename`: the name of the input file. If in encrypt mode, the input file contains the plaintext, and if in decrypt mode, the input file contains the ciphertext.
- `key`: an integer.

If using Java, you should invoke your program like this:

```
java Ceasar e msg.txt 12
```

or

```
java Ceasar d msg.enc 12
```

For Python and C, replace `java Ceasar` with, respectively, `python ceasar.py` or `ceasar` . The output of your program will either be ciphertext (if in encrypt mode) or plaintext (if in decrypt mode), and should be printed on the standard output.

1.2 Random Substitution Cipher

Write a program to implement a *random substitution cipher*. The key should be a random bijective mapping of the alphabet. Your program should be called `subst.py` if using Python, `subst` if using C, or `Subst` if using Java, and take the following command line arguments:

- `e` or `d` or `g`: encrypt or decrypt or generate mode.
- `inputfilename`: the name of the input file. If in encrypt mode, the input file contains the plaintext, and if in decrypt mode, the input file contains the ciphertext. If in generate mode, the input file name will be ignored.
- `keyfilename`: the name of the file containing the key. If in generate mode, the key will be written into that file name. Otherwise in encrypt and decrypt mode, the key will be read from that file.

You should invoke your program in a similar way to the Ceasar cipher, but by replacing the program names of course. The output of your program will either be ciphertext (if in encrypt mode) or plaintext (if in decrypt mode) and should be printed on the standard output. If in generate mode, there will be no output on the standard output.

1.3 Transposition Cipher

Write a program to implement the *columnar transposition cipher*. The key should be an integer, the number of columns in the table. Your program should be called `coltrans.py` if using Python, `coltrans` if using C, or `ColTrans` if using Java, and take the following command line arguments:

- `e` or `d`: encrypt or decrypt mode.
- `inputfilename`: the name of the input file. If in encrypt mode, the input file contains the plaintext, and if in decrypt mode, the input file contains the ciphertext.
- `key`: an integer.

You should invoke your program in a similar way to the Ceasar cipher, but by replacing the program names of course. The output of your program will either be ciphertext (if in encrypt mode) or plaintext (if in decrypt mode), and should be printed on the standard output.

1.4 Vernam Cipher

Write a program to implement a *Vernam cipher*, a kind of *One-Time-Pad (OTP)*. The key should be a sequence of random letters of the alphabet to be added to the plaintext or subtracted from the ciphertext. Adding or subtracting letters really mean adding or subtracting the letter numerical codes. Your program should be called `vernam.py` if using Python, `vernam` if using C, or `Vernam` if using Java, and take the following command line arguments:

- `e` or `d` or `g`: encrypt or decrypt or generate mode.
- `inputfilename`: the name of the input file. If in encrypt mode, the input file contains the plaintext, and if in decrypt mode, the input file contains the ciphertext. If in generate mode, the input file name will be ignored.
- `keyfilename`: the name of the file containing the key. If in generate mode, the key will be written into that file name. Otherwise in encrypt and decrypt mode, the key will be read from that file.
- `n`: an integer. In generate mode, `n` is the length of the key to generate. In encrypt and decrypt mode, `n` is the offset in the key file, i.e. it is the position of the first random letter in the key to use in the addition or subtraction of letters.

You should invoke your program in a similar way to the Ceasar cipher, but by replacing the program names of course. The output of your program will either be ciphertext (if in encrypt mode) or plaintext (if in decrypt mode) and should be printed on the standard output. If in generate mode, there will be no output on the standard output.

2 Competition

You will be provided with encrypted messages during weeks 3, 4 and 5. These messages will be encrypted with one of the four ciphers you have to implement in the previous section. But you will not be provided with the keys, and you will not be told exactly which cipher was used to encrypt the messages. Your task is to write programs to break the ciphers used to encrypt these messages. The alphabet used in these messages will be the same as the one given above.

Why is this section titled *Competition*? Because you will receive marks not only for breaking the ciphers, but also for how fast you can break them. The earlier you break a code, the largest number of bonus marks you are going to get. If you are the first one to successfully decrypt a message, you will receive the maximum number of bonus marks to be added to your assignments score. If you are not able to decrypt a message, you will not receive any bonus mark.

To be eligible for bonus marks, you have to submit the program you used to decrypt the message. Then you have to explain how you used your program to find out what was the correct message.

3 Submission and Marking

You have to submit a file named `COSC2539_A1_ciphers_s1234567.zip` containing the code you wrote to implement the 4 ciphers of section 1. For section 2, you will have to make a separate submission for each message you decrypt successfully. Name your files `COSC2539_A1_msg1_s1234567.zip`, `COSC2539_A1_msg2_s1234567.zip`, etc..., where `msg1` means the first message you have to decrypt, `msg2` the second message you have to decrypt, ... Don't forget to write your real student ID in the file names.

Your files have to be submitted by first uploading them to *Google Drive*, and *sharing them in view mode with the lecturer*. Because the files you have to submit are zip files, sharing in view mode is the best way to share your files.

This assignment is worth 15% of your total mark for the course. Each cipher you implement is worth 3%, and section 2 is worth also 3% for being able to decrypt the messages without the key. Only the first five students who successfully decrypt a message will receive bonus marks. The fastest student will receive 1 bonus mark, the second 0.8 bonus marks, the third 0.6 bonus marks, the fourth 0.4 bonus marks and the fifth 0.2 bonus marks.