Zak Rowland
CST 466
Assignment 1: Classical Encryption

1. Use the ASCII codes available at http://www.asciitable.com to manually construct a Base64 encoded version of the string "hello\njello". Your answer should be "aGVsbG8KamVsbG8=". What do you think the character '=' at the end of the Base64 representation is for?

   [If you wish you can also use interactive Python for this. Enter the following sequence of commands "import base64" followed by "base64.b64encode('hello\njello')". If you are using Python 3, make sure you prefix the argument to the b64encode() function by the character 'b' to indicate that it is of type bytes as opposed to of type str. Several string processing functions in Python 3 require bytes type arguments and often return results of the same type. Educate yourself on the difference between the string str type and bytes type in Python 3.]

h	e	l	l	o	\	n	j	e	l	l	o
0x68	65	6C	6C	6F	5C	6E	6A	65	6C	6C	6F

   01101000 01100101 01101100 01101100 01101111 01011100 01101110 01101010 01100101 01101100 01101100 01101111

   011010 000110 010101 101100 011011 000110 111101 011100 011011 100110 101001 100101 011011 000110 110001 101111

   26 6 21 44 27 6 - 61 28 27 38 41 37 27 6 49 47

   I got aGVsbG9cbmplbGxv – an online encoder did as well. Not sure why the assignment gave a different answer, or where the '=' comes from.

2. A text file named myfile.txt that you created with a run-of-the-mill editor contains just the following word:

   hello

   If you examine this file with a command like

   hexdump -C myfile.txt

   you are likely to see the following bytes (in hex) in the file:

   68 65 6C 6C 6F 0A

which translate into the following bit content:

01101000 01100101 01101100 01101100 01101111 00001010

Looks like there are six bytes in the file whereas the word "hello" has only five characters. What do you think is going on? Do you know why your editor might want to place that extra byte in the file and how to prevent that from happening?

The editor adds a newline/line feed character after every line of data. This can likely be adjusted in the editor's settings.

3. All classical ciphers are based on symmetric key encryption. What does that mean?

   Symmetric key encryption means the same secret key is used for encryption and decryption.

4. What are the two building blocks of all classical ciphers?

   The two building blocks are substitution and transposition. Substitution is replacing an element of the plaintext with an element of ciphertext. Transposition means rearranging the order of appearance of the elements of the plaintext.

5. True or false: The larger the size of the key space, the more secure a cipher? Justify your answer.

   False. A larger key space may help prevent brute force attacks, but statistical attacks may be much easier. Statistical attacks look at the frequency each character is used and patterns in the ciphertext.

6. Give an example of a cipher that has an extremely large key space size, an, extremely simple encryption algorithm, and extremely poor security.

The Caesar cipher has a large key space, but the algorithm is very simple and well known, making security poor.

7. What is the difference between monoalphabetic substitution ciphers and polyalphabetic substitution ciphers?

   Monoalphabetic substitution uses the same substitution rule for every character in the plaintext message. Polyalphabetic substitution uses a different substitution rule for every character in the plaintext.

8. What is the main security flaw in the Hill cipher?

   The Hill cipher has no security when the plaintext-ciphertext pairs are known because the key matrix can be easily calculated.

9. What makes Vigenere cipher more secure than, say, the Playfair cipher?

   The Vigenere cipher is more secure because a different substitution rule is used for each character in the plaintext. The encryption key is aligned with the plaintext, and each letter in the encryption key maps to a different substitution rule for the corresponding plaintext letter.

10. Let's say you have used the encryption and decryption scripts shown in Section 2.11 through the following calls

    EncryptForFun.py message.txt output.txt

    DecryptForFun.py output.txt recover.txt

    or the Perl versions of the same, and that, subsequently, you compare the input message file and the output produced by decryption by calling

    diff message.txt recover.txt

    you are likely to see the following message returned by the diff command:

    Binary files message.txt and recover.txt differ

and, yet, if you print out the contents of the two files by

cat message.txt
cat recover.txt

the two files appear to be identical. What do you think is going on?

[HINT: Use the 'cat -A' command to output the contents of the two files. Also, instead of calling diff as shown above, try calling 'diff -a' which forces a text only comparison on the two files.]

The generated file (recover.txt) is missing the newline character that the text editor adds to the message.txt file.