Ryerson University

CPS109 Fall 2016
Assignment 1 - Cryptography
Due Date: 11:59pm November 14th, 2016

You are to **work alone** when writing your code. You can discuss general ideas with your classmates, but you cannot copy code or develop code together (changing identifiers and rearranging code does not help) nor take code from the web. We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism; see the following link for details: http://theory.stanford.edu/~aiken/moss. The Department of Computer Science takes the act of plagiarism very seriously. **Those caught plagiarizing (both originators and copiers) will be sanctioned.** Please see Ryerson University's Policy 60 for possible penalties and consequences: http://ryerson.ca/senate/policies/pol60_procedures.pdf. The due date and time is firm. No assignments will be accepted after the deadline.

In this assignment you will implement an application for decrypting an encrypted text file based on a specified encryption/decryption key.

The file was encrypted using a combination of a *Substitution cipher*[1] and a columnar *Transposition cipher*[2].

The following is the encryption procedure:

- Read the encryption key (this is a symmetric system, and the same key is used both for encryption and for decryption); let's assume the key is:
  ```
  String key = "ABCDEFGH";
  ```

- Calculate the hash value of the key (this is a random-looking number that is dependent on the key string given). In this case, the hash value was taken from the output of the hashCode method:
  ```
  int hash = key.hashCode();
  ```
  the resulting hash value is
  2042300548.

- Use the value as the seed to initialize the Random object:
  `random = new Random(hash);` or `random.setSeed(hash);`
  At this point we have a random number generator, where the sequence of its pseudo-random numbers is determined by the seed above.

---

[1] https://en.wikipedia.org/wiki/Substitution_cipher
[2] https://en.wikipedia.org/wiki/Transposition_cipher#Columnar_transposition

- Now we use the numbers this generator generates to create a **substitution pattern** from the following alphabet (ending with space) "ABCDEFGHIJKLMNOPQRSTUVWXYZ ".
  - generate two random numbers between 0 and 26 (the seed was set, right?)
  - swap the letters at these positions in the string above[3]
  - repeat 100 times
  - should get "GCWHAKSXJMDLFUB ITVYRPZENQO" after 100 iterations
  - This means, A letters in the original will be replaced with a G, B with C, C with W, and so on:

    "ABCDEFGHIJKLMNOPQRSTUVWXYZ "
    "GCWHAKSXJMDLFUB ITVYRPZENQO"

- Similarly, we should create a pattern for column order during the **transposition step**:
  - start with pattern {0, 1, 2, 3, 4, 5, 6, 7}
  - set the Random seed to the same initial value
  - generate two random numbers between 0 and 7
  - swap the numbers at these positions in the array above
  - repeat 100 times
  - should get {3, 2, 7, 4, 1, 0, 5, 6} after 100 iterations
  - This means the columns will be read in the order 3, 2, 7, and so on – instead of sequentially

- Read the text file line-at-a-time, in chunks of 64 characters say, the first 64 characters are

  "IN CRYPTOGRAPHY, A SUBSTITUTION CIPHER IS A METHOD OF ENCODING B"

- Apply the substitution:

  "JUOWTN YBSTG XNOOGOVRCVYJYRYJBUOWJ XATOJVOGOFAYXBHOBKOAUWBHJUSOC"

- Apply the transposition (write row-by-row, then read columns in order 3, 2, etc.:

  J U O W T N Y
  B S T G X N O
  O G O V R C V Y
  J Y R Y J B U O
  W J X A T O J
  V O G O F A Y X
  B H O B K O A U
  W B H J U S O C

  Result:

  "WGVYXOBJOTOR GOHYOYOJXUCT RJAFKUUSGYJOHBJBOJWVBWNXCBTAOS NVUOYAO"

- For decryption, you need to reverse the transposition and substitution steps.

[3] You might want to use a StringBuffer class instead of String

For this assignment you need to write an application that takes 3 command-line arguments and **decrypts** a file into another file:

```
java Decrypt key input output
```

`key` – string key provided
`input` – input file that contains the text to decrypt; in text file format
`output` – output file generated by `Decrypt`

An example input and the output it produces:

Original (from `https://en.wikipedia.org/wiki/Substitution_cipher`):

```
In cryptography, a substitution cipher is a method of encoding by
which units of plaintext are replaced with ciphertext, according to a
fixed system; the "units" may be single letters (the most common),
pairs of letters, triplets of letters, mixtures of the above, and so
forth. The receiver deciphers the text by performing the inverse
substitution.
```

Encrypted:

```
WGVYXOBJOTOR GOHYOYOJXUCT RJAFKUUSGYJOHBJBOJWVBWNXCBTAOS NVUOYAO
XYGGGXYWZJLOLYTWOKATO OJJVJTWOABOU Y JAGNROEAZXOWOUAAWETXBYOHJYH
YAFRGUYAOEAOFJYXONXVAOOVBHOUNSAOSJYOOVAYUKVAOOLOOOOJOLTFGVYYCAVB
B OOYYJBWOKVAAFOUVYJKVTXFGLOVYEKOOBTLLOVYOOA OOAFJAYOAYOBTYTBTRY
CHYTOTAAGUTOTAY OOYJJXCTBOXAHVETOGBAAXOOAOKXP ANPVOWAOYKABOAWYOB
SPCB    UURJ    AAY     OAVU    JJVY    FOOR    YTYO    XVJ
```

After decryption:

```
IN CRYPTOGRAPHY  A SUBSTITUTION CIPHER IS A METHOD OF ENCODING B
Y WHICH UNITS OF PLAINTEXT ARE REPLACED WITH CIPHERTEXT  ACCORDI
NG TO A FIXED SYSTEM  THE  UNITS  MAY BE SINGLE LETTERS  THE MOS
T COMMON   PAIRS OF LETTERS  TRIPLETS OF LETTERS  MIXTURES OF TH
E ABOVE  AND SO FORTH  THE RECEIVER DECIPHERS THE TEXT BY PERFOR
MING THE INVERSE SUBSTITUTION
```

As you can see, any symbols other than 26 English letters and space are replaced with spaces, and all text is converted to uppercase, for simplicity.

Submission:

You have to submit to BrightSpace the following files (and no more):

1. `Decrypter.java` (your code)

2. `output.txt` (the decrypted message output of your program based on the encrypted input message in `input.txt` provided in the assignment package; the key used to encrypt the message is "`ABCDEFGH`")

3. `comments.txt` (optional, contains comments you want the markers to read about your submission)

Marking rubric (out of 8):

- 2 marks for proper coding style (proper indentation of the code and meaningful variable names used in the program)

- at most 2 marks for a plausible attempt at a solution

- 4 marks for code that compiles and executes correctly, i.e., computes and outputs the correct solution