The goal of this assignment is to provide experience working with cryptographic algorithms such as the Hill Cipher. The book exercises are intended to provide practice working with other encryption algorithms.


3. [10] Create a new Eclipse project named ChatClientEncrypted. Create a ChatClientEncrypted class for the project with all of the capabilities of your ChatClient class from Homework 5.

4. [30] Add a transform() method to your ChatClientEncrypted class which takes two characters (i.e., type char) and a 2x2 integer matrix (i.e., type int[2][2]) as input and returns a String. This method will perform the matrix multiplication step of the Hill Cipher, but unlike the traditional Hill Cipher which operates on only the 26 case-insensitive alphabetic characters, our modified Hill Cipher will operate on the 128 standard ASCII characters. This modification will allow the chat client to send/receive encrypted case-sensitive alphabetic characters, digits, and punctuation. Here’s pseudocode for the transformation:

   Create an int[2] using the input characters a and b and multiply this array by the input matrix using standard matrix multiplication to determine the characters c and d (you’ll need to implement the matrix multiplication):

   \[
   \begin{bmatrix}
   m00 & m01 \\
   m10 & m11
   \end{bmatrix} \times 
   \begin{bmatrix}
   a \\
   b
   \end{bmatrix} = 
   \begin{bmatrix}
   c \\
   d
   \end{bmatrix}
   \]

   Calculate the values of c and d modulo 128 (e.g., c %= 128).

   If c or d is negative, add 128 (e.g., if (c < 0) c += 128).

   Return a new String containing c and d.

5. [10] Add an encrypt() method which takes a String as input and returns an encrypted String. The method should first check to determine if the number of characters in the input String is odd, and if so pad the string with a trailing space (i.e. " ") character. The modified Hill Cipher used for this assignment only works on even-length strings, as you’ll see next. Next, loop through the String calling transform() on each pair of characters and adding the result to the output stream. Use the following encryption key as input to transform():[[4, 3], [5, 4]]. You may define your own encryption key, but you’ll need a paired decryption key and generating such a key is nontrivial and beyond the scope of this course (see below).
6. [10] Add an `decrypt()` method which takes a `String` as input and returns a decrypted `String`. Similar to `encrypt()`, loop through the `String` calling `transform()` on each pair of characters and adding the result to the output stream. Use the following decryption key as input to `transform()`: `[[4, -3], [-5, 4]]`. Again, you may define your own decryption key, but you’ll need a decryption key `D` paired with your encryption key `E` such that `D = E^{-1} mod 128` (we’ll discuss in class).

7. [10] Set up `ChatClientEncrypted` to encrypt all output sent to the server and to decrypt all input received from the server.

8. [+10] Create a `ChatClientAndroidEncrypted` project in Eclipse with all of the capabilities of `ChatClientAndroid` but which can additionally communicate with instances of `ChatClientEncrypted` via `ChatServer` and/or `ChatServerAuthenticatedEncrypted`.

9. [+20] Create a `ChatServerAuthenticatedEncrypted` project in Eclipse with all of the capabilities of `ChatServerAuthenticated` but which can additionally communicate with `ChatClientEncrypted` while preserving all state-based behaviors of `ChatServerAuthenticated`.

Zip your authenticated server project and problem-set solution document (PDF or plain text format). Name your zip file `<your full name>Homework8.zip` and upload to Blackboard.