NETWORKS AND CRYPTOGRAPHY — PROJECT 4
The Hill and LFSR ciphers
Revision 1 [1]

1 Overview

You must pick one of two ciphers that we discussed: either the *Hill cipher* or the *Linear Feedback Shift Register stream cipher*. For your chosen cipher, you must do the following:

1. **Implement** the one that you choose—write both an *encrypter* and a *decrypter*.
2. **Submit** both your implementation and *plaintexts of your choosing*. I will then select a key to encrypt both your known plaintext and an unknown (to you) plaintext of my choosing. I will send these to you.
3. **Crack** the unknown ciphertext by using the known plaintext/ciphertext pair that has been encrypted with the same key.
4. **Submit** your decryption of the unknown ciphertext.

Notice that you should choose a known plaintext of sufficient length and appropriate type to allow you to perform the cryptanalysis needed for your chosen cipher. Better too long than too short.

2 Details of your tasks

You should perform each of the tasks detailed above as follows:

1. **Implement:** Write a pair of applications, one for encrypting, one for decrypting of your chosen cipher. The details of the command-line interface depend on which cipher you've chosen:

   - **Hill:** Since a key for the Hill cipher is an $n \times n$ matrix, then the key should be assumed to be stored in a text file, like so:

```
137 249 90 244 104 114 209 243
15 164 62 136 93 165 124 51
116 254 151 94 200 50 130 27
79 53 52 202 212 48 254 247
133 196 50 119 227 195 74 236
190 248 122 109 205 43 52 200
55 246 1 103 254 81 234 65
226 239 22 38 84 147 114 171
```

[1] See Appendix A.

On the command-line, the user should provide the **name of a file that contains a key in this format**. That is, the key file should contain n^2 values, n of them per line over n lines, in text. The values should be between 0 and 255 (since that's our alphabet range). The encrypter and decrypter should be invoked like this (assuming that they're written in Java):

```
$ java HillEncrypt keyfile.txt plaintext ciphertext
$ java HillDecrypt keyfile.txt ciphertext decrypted.plaintext
```

- **LSFR:** The key for this cipher is a single, scalar value that provides a seed for the keystream. (That is, the key could be, say, the seed to a pseudorandom number generator, which then deterministically provides the keystream values.) Thus, the key itself should be provided as a 64-bit integer (not coincidentally the side of the `seed` parameter to the Java `Random` constructor) on the command-line:

```
$ java HillEncrypt 91283571239785 plaintext ciphertext
$ java HillDecrypt 91283571239785 ciphertext decrypted.plaintext
```

2. **Submit code/plaintext:** Given a working encrypter/decrypter, submit both your code and plaintext that you want encrypted with my chosen key as `project-4a`. Specifically, the plaintext should be named `known.plaintext`. Thus, your submission should like something like this (appropriately adjusted for the name of your choice of cipher):

```
$ cs28-submit project-4a HillCommon.java HillEncrypt.java
                         HillDecrypt.java plaintext.known
```

Note that in this example, I assume that you move any methods common to both encryption and decryption into another class, `HillCommon` (or `LSFRCommon`), so that both encrypter and decrypter can use those methods.

A note about your known plaintexts: You must choose your plaintext/ciphertext pairs wisely to crack each of these ciphers. Here are some critical reminders and hints:

- **Hill:** First, for the n^2 matrix that forms the key for this cipher, you may assume that $n = 8$. That information should be sufficient for you to construct a sufficiently long known plaintext to create n^2 matrices of known plaintext and ciphertext in order to calculate the key. (Remember that for any plaintext must be divided into segments of length n, with each segment being encrypted separately.) Moreover, be sure that your plaintext, when arranged into a matrix, forms something invertable.
- **LSFR:** Recall that this algorithm works on binary values rather than whole symbols. If the key length is $2n$ (where n bits are the constants c_i and the other n bits are the initial keystream values z_i), then you need enough plaintext/ciphertext bit pairs to construct the matrix needed to discover the constants. Your implementation should use $n = 8$, generating the constants and the key from the seed value provided on the command line.

You should then **send me an email** indicating that you've submitted this first stage of the project. I will then, as quickly as possible,[2] respond with two ciphertexts—one an encryption of your known plaintext, and one an encryption of an unknown (to you) plaintext. Both, of course, are encrypted with the same key.

3. **Crack the code:** Use your collection of cryptanalysis techniques to determine key. Decrypt the unknown ciphertext to obtain its plaintext.

4. **Submit decryption:** Submit the decrypted unknown message as `project-4b`:

```
$ cs28-submit project-4b unknown.plaintext
```

This assignment is due at **11:59 pm** on **Tuesday, December 14**.

A Revisions

Here is a list of the revisions of this document:

- **Revision 0 [Dec-05]:** The original, complete document.
- **Revision 1 [Dec-08]:** Added text to provide the length of the keys to make the project tractable; added text to provide more details about the length of known plaintext needed to mount a successful attack.

[2] Yes, you may be skeptical