**Hill Cipher**

Jacob Reed

SIU851312743

CS 408

**About:** The Hill cipher is a popular and primitive encryption method used to encrypt small amounts of text through breaking a string into multiple parts regarding the length of the key. Even values will be broken into pairs of two and three for odd length keys. By multiplying the index values in matrix form we can get the encrypted message. To decrypt, the same is done except using the inverse matrix of the key. The way that I decided to address the problem is to take the strings in for the key and plain text and divide them and store them as arrays. By using simple matrix multiplications and text I was able to encrypt the text and get the appropriate values. The report bellow covers the basic functionality of the program and what each method does.

**main(String args[])**

This gathers the strings for the keys and the plain text. This also will get the square root of the key to be used when dividing the text appropriately. This will also make a call to the codec which will be explained later in the report. If the wrong usage is used, then it will also print a message off if the usage is not correct.

**codec(String line, String key, double squareRoot)**

This will start by making sure that the key will form a square matrix. After that passes, it will then check to make sure that the text can be evenly divided. If not, it will add **X** to the end of the string until it can. Next, it will divide the string into the appropriately sized subarrays based on the key. It will then multiply each row-sub array against the key matrix. After, the inverse key matrix will be calculated and printed so that can be used for decrypting the text.

**multiplyLineMatrixKeyMatrix(int length, char[] row)**

Performs matrix multiplications against the passed in row against the key matrix. If the result is larger than 25, mod 26 is taken to get the value to the appropriate size of the value. It will then be placed into the resulting string and sent to the console.

**invertibleCheck(String key, int length)**

This method will check to make sure that the key is invertible, does not have a common factor with mod 26, and the determinate will not be equal to zero by calling another method that calculates the determinate. If the key cannot be used, it will pass an error to the console and terminate the program.

**calculateInverseMatrix(int inputMatrix[][], int size)**

This is the most crucial part of the program since this will allow us to calculate the inverse of the key matrix which will be used to get the decryption and encryption key. By using a method of matrix of minors we will be able to get the inverse of a 3 x 3 matrix easily without having to do much work. By checking if the matrix is a 2 x 2 or less, we can then calculate the inverse matrix without using minor matrices. This will then have the information passed to a string where the inverse key can be passed in as the decryption key.

**calculateMultInverse(int determinant)**

One step in finding the inverse of the key matrix is to get the multiplicative inverse of the determinant. The determinant is passed in and using the built in functionality of BigInteger we can then calculate the multiplicative inverse with respect to mod 26 with ease. Converting to and from int to BigInteger is easy and allows for lossless conversion.

**Conclusion:** This was a fairly easy solution to implement. The issue with the Hill Cipher is that it does not encrypt on a bit level but does a good job of removing the relationship between the plain text and cipher text and still can be used for simple applications. This solution could have been implemented more easily with JAMA but was a good experience practicing with appropriate array manipulation. Even though this might not be the best solution for encrypting sensitive data, it does cover the basics of what needs to be done when messing with matrix manipulation.