Kyle Russell - 13831056
Highly Secure Systems
A1 Documentation

CONTENTS

1.0 Hill Cipher
- 1.1 About
- 1.2 Encryption
- 1.3 Decryption
- 1.4 Cryptanalysis

2.0 User Guide
- 2.1 Using the chat application
- 2.2 Using the cipher breaker

3.0 Installation notes

1.0 Hill Cipher

1.1 About

The Hill cipher is a classical polygraphic substitution block cipher, developed by Lester Hill in 1929. The cipher components are based on elementary linear algebra and modulo arithmetic where key structures are represented by n x n matrices and cipher/plain text blocks are maintained as n-length vectors. Characters are represented in the matrices by their corresponding integer value where modulo operations are done by the size of the character set and in classic examples we can map the alphabetic characters A-Z to values 0..25 however any character can be represented.

Additionally, unlike many other block ciphers (DES, AES etc.), the Hill cipher theoretically supports any block length and given its trivial linear operations, this makes it relatively efficient for encrypting large blocks. However due to the linearity of the Hill cipher, there exist known cryptanalysis attacks for simple Hill cipher's that have small key sizes where for a length = 4 key example, we only need to find 2 bigram correspondences and solve a set of linear equations to break the cipher. Therefore the cipher's security for a trivial key is very weak however as the key size increases so does the security of the cipher.

1.2 Encryption

In this section the encryption procedure of the Hill Cipher will be discussed where we will take an example plain text and through the steps of encryption we will produce the resulting cipher text. In this example the assumed matrix size is 3x3 and the character set will be alphabetic characters only although the Hill Cipher implementation included in this submission does allow 2x2 matrices as well as complete ASCII support.

The structures used to hold data in the Hill Cipher consist of integer type n x n matrices to store the key and length-n vectors to store the plain/cipher text blocks. In the case where we only allow alphabetic characters, the letters A-Z are represented numerically in the structures as 0…25 such that A = 0, B = 1... Z = 25. In this example the plain text which will be encrypted is 'yes' and the key used is 'alphabeta'. Below includes an example of their numeric conversations in the structures.

24
4
18

Table 1: Plain text vector

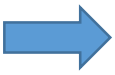
0	11	15
7	0	1
4	19	0

Table 2: Key matrix

Once the plain text and key have been converted to the corresponding integer values and stored into the appropriate structures then we just need to multiply the key matrix with the plain text vector to compute the cipher text. Additionally, values computed in the matrix product must be taken mod m where m is the size of the character set (this this case m = 26). The matrix product before the modulus operation is shown in table 3 and in table 4 the values are mod 26 which is consequently our resulting cipher text vector and the cipher text is 'ceq'

316
186
172

Table 3: Key and plain text matrix product before mod 26



2
4
16

mod 26

Table 4: Key and plain text matrix product after mod 26

1.3 Decryption

In section 1.2 the encryption process was discussed where the string ‘yes’ was encrypted to ‘ceq’ using the key ‘alphabeta’. This section will cover the decryption steps such that we can find the initial plain text given an input cipher text and key. The following steps will continue on from the section 1.2 example using the same key and cipher text where we will find the initial plain text ‘yes’ after decryption.

The decryption function can be seen in eq. 1 where P is the desired plain text vector, C is the input cipher text vector and K is our key such that K^{-1} is the inverse key. So essentially our plain text is the product of the cipher text vector and the inverse key. While this procedure may seem trivial, the process of finding the inverse key is quite complex and eq. 2 shows this function where d is the determinant of the key K and $adj(K)$ is the adjugate matrix of $transpose(K)$ which is computed as shown in eq. 3.

$$\boldsymbol{P = C \cdot K^{-1}}$$

Equation 1: Plain text computation

$$\boldsymbol{K^{-1} = d^{-1} \cdot adj(K)}$$

Equation 2: Inverse key computation

All values must be taken mod m (m = 26 in this case) including negative values such that if v < 0 := v + m. Additionally, the inverse of the determinant d^{-1} can be computed with the extended-euclidean algorithm such that $d \cdot d^{-1} = 1 \ mod\ 26$. In this example d = 11 and d^{-1} = 19. The adjugate matrix $adj(K)$ was computed in table 5 and consequently the inverse key K^{-1} in table 6.

$$adj\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = \begin{pmatrix} +\begin{vmatrix} e & f \\ h & i \end{vmatrix} & -\begin{vmatrix} b & c \\ h & i \end{vmatrix} & +\begin{vmatrix} b & c \\ e & f \end{vmatrix} \\ -\begin{vmatrix} d & f \\ g & i \end{vmatrix} & +\begin{vmatrix} a & c \\ g & i \end{vmatrix} & -\begin{vmatrix} a & c \\ d & f \end{vmatrix} \\ +\begin{vmatrix} d & e \\ g & h \end{vmatrix} & -\begin{vmatrix} a & b \\ g & h \end{vmatrix} & +\begin{vmatrix} a & b \\ d & e \end{vmatrix} \end{pmatrix}$$

Equation 3: Adjugate matrix

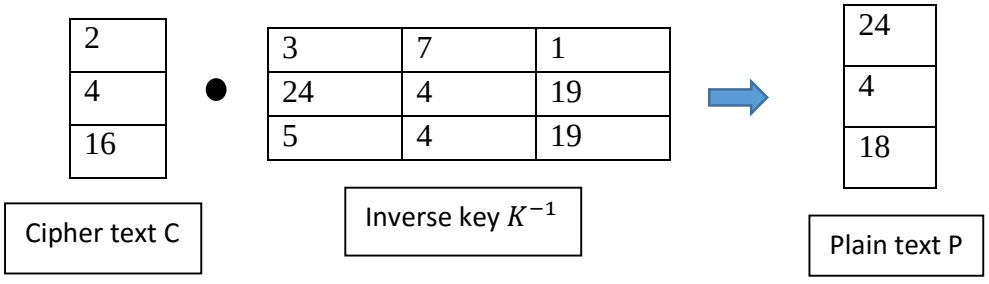
7	25	11
4	18	1
3	18	1

Table 5: Computed adjugate matrix for key K

3	7	1
24	4	19
5	4	19

Table 6: Inverse key matrix

Finally, we multiply the cipher text vector C with the inverse key K^{-1} where the product is our initial plain text vector and the encrypted cipher text has been decrypted and as expected, the string representation of the result vector is ‘yes’. This completes the decryption process.



1.4 Cryptanalysis

In this section a cryptanalysis attack for the Hill Cipher will be discussed in which the steps will be shown for how one could break a simple Hill Cipher with a key of length = 4 (such that the key matrix K is 2x2 and text vectors have length 2). Additionally, it is expected that the plain text character set consists of alphabetic characters only and we must also know 5 continuous characters of plain text.

The attack works by finding 2 bigram correspondences in the cipher text and known plain text which can be used to solve a set of simultaneous to find the inverted key matrix K^{-1} that can be multiplied with the cipher text where the product will be the decrypted plain text. More specifically, the inverted key K^{-1} can be computed as $K^{-1} = A \cdot B^{-1} \ (mod\ 26)$ where the columns in the matrix A consist of the chosen bigrams from the known plain text and similarly the columns in B are the bigrams from the cipher text such that we have A_{ij}, B_{ij}: $K^{-1}[A_{00}, A_{10}] = [B_{00}, B_{10}] \ (mod\ 26)$ and $K^{-1}[A_{01}, A_{11}] = [B_{01}, B_{11}] \ (mod\ 26)$ which holds if our bigram correspondence assumption is correct.

As an example, suppose we have the cipher text: 'iyabkdbnsdatyqbasdsuga' and we know that the text 'ofthe' exists somewhere in the plain text. Let's first assume that we have the following bigram correspondence: ft => ba and he => sd. Next we create the 2x2 matrices A, B where columns of A are the integer conversions of the known plain text bigrams: [ft, he] and similarly, the columns of B are [ba, sd] as shown below.

5	7
19	4

Table 7: Matrix A (known text)

1	18
0	3

Table 8: Matrix B (cipher text)

$$adj\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Equation 4: Adjugate 2x2 matrix computation

Next we compute the inverse the cipher matrix B such that $B^{-1} = d^{-1} \cdot adj(B)$ where d^{-1} is the determinant inverse of the matrix B which can be computed by finding the determinant of B and then using the extended-Euclidean algorithm to find the determinant inverse. The matrix $adj(B)$ is the adjugate matrix and is computed for a 2x2 matrix as shown by eq. 4. After finding the determinant inverse and $adj(B)$ we simply scalar multiply the adjugate matrix with the determinant inverse to get B^{-1} as shown in table 9. Then the inverse key is simply the product of $A \cdot B^{-1}$ and the result is shown in table 10. Lastly, we can use this computed inverse key to decrypt the cipher text by multiplying the inverse key with the cipher text such that $P = B \cdot B^{-1} \ (mod\ 26)$. In this example we used the decryption key and found that the plain text was 'awhatisthedayoftheweek' which contains our known plain text 'ofthe' and is not obfuscated so this is a likely candidate.

1	20
0	9

Table 9: Inverse of matrix B

5	7
19	0

Table 10: Inverse key

It is important to note that if our bigram correspondence assumption is incorrect then our resulting key will be incorrect and consequently our decrypted plain text will be obfuscated. Therefore it is necessary to permute all possible bigram correspondences and we do this by shifting the known text down the cipher text and getting the bigrams at the offset which is incremented by one each time and this technique is known as 'crib dragging'.

2.0 User Guide

2.1 Using the chat application

The chatting application written allows users to join chat rooms and talk to others safely with encrypted messages. To begin chatting, launch the application and you will be presented with the interface shown in fig. 1. In the 'Display name' field enter a name you would like to be identified as which will be displayed in chat next to your messages. The 'Room name' field is the name of the room you wish to connect to, if this room doesn't exist it will be created. Lastly, the 'Key' field is the private key that will be used to encrypt your messages. This key must be 9 alphabetic characters and should be known by those connected to the room so they may decrypt your messages.

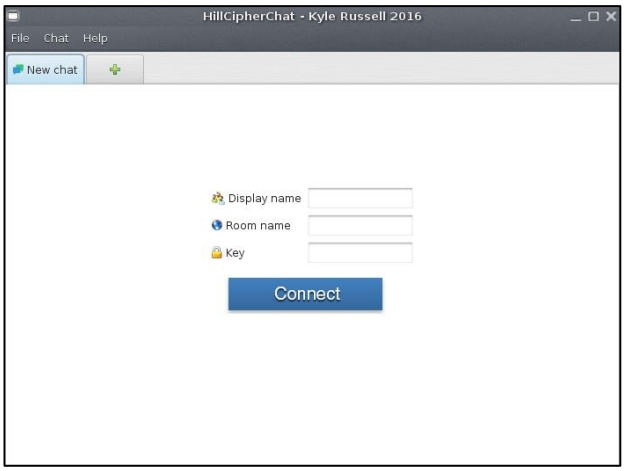


Figure 1: Chat connection window

When you have filled in these required fields, click the 'Connect' button to connect to the room and once connected, you will be presented with the conversation window shown in fig. 2. To send a message, enter the message in the bottom text field and then click the 'Send' button. Your message will be visible to all users in the room and if users have an invalid key then the received message will be obfuscated.

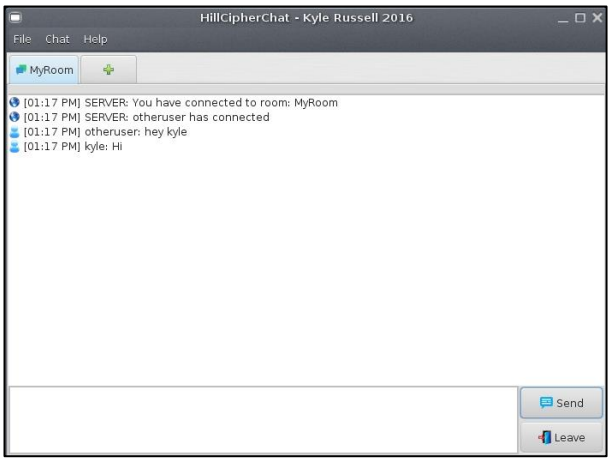


Figure 2: Conversation window

If you want to view the details of a message including its encrypted string and time sent, simply click a message in the chat and you will be displayed the dialog in fig. 3.

Figure 3: Message detail dialog

When you wish to leave the chat room simply click the 'Leave' in the conversation window or click the Chat -> Leave Room menu. Additionally, you can add more chat rooms and participate in multiple conversations simultaneously by clicking the '+' button in the top tab pane or clicking the Chat -> New Room menu.

2.1 Using the cipher breaker

The hill cipher breaker application allows attackers to break a simple cipher containing alphabetic characters only, where the input cipher was encrypted with a length 4 key and the attacker knows at least 5 continuous characters of plain text. The details of this cryptanalysis method are explained in section 1.4.

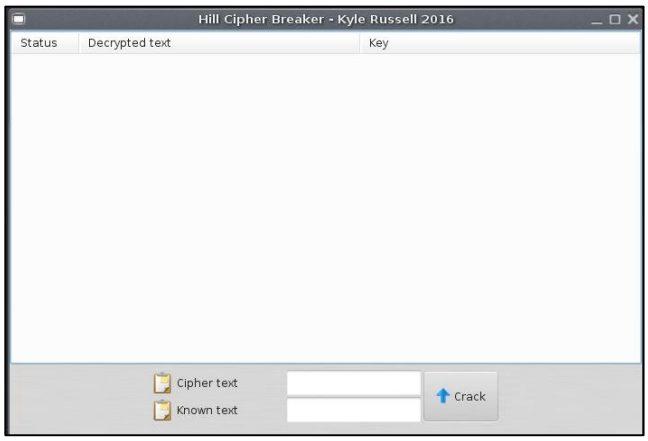


Figure 4: Cipher breaker window

To begin breaking a hill cipher put your cipher/encrypted text in the ‘Cipher text’ field. Then put 5 characters of known plain text in the ‘Known text’ field. When you are ready to begin breaking the cipher click the ‘Crack’ button.

After the process has finished you will be presented with a list of all possible attempts shown in fig. 5. The status of the attempt indicates how successful the attempt, such that promising attempts will contain the known text. The decrypted text and its corresponding key that was used to decrypt are also displayed.

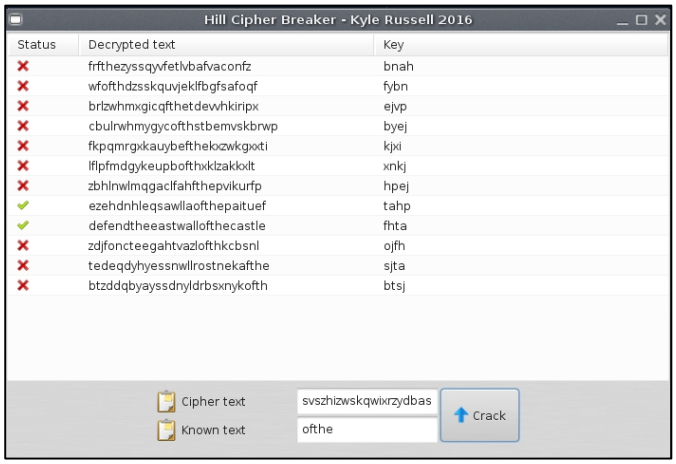


Figure 5: Cipher breaking results

3.0 Installation Notes

The submission includes projects described below which are required to run the software. Each project was developed with Java 1.7 under the Netbeans 8.1 IDE and also includes the Netbeans project files such that projects can be directly imported. Any libraries used are located in the 'lib' directory of the respective project. Both the chat and cipher breaker applications included compiled jar executable's which can be launched directly instead of needing to make your own build of the source.

Projects

- **hillc-base**
  Contains the Hill Cipher implementation and is used as a dependency in hillc-breaker and hillc-chat-client.

- **hillc-breaker**
  Contains the Hill Cipher cryptanalysis implementation described in section 1.4. Also includes an application that allows attackers to break simple Hill ciphers.

- **hillc-chat-client**
  The client component of the chatting application

- **hillc-chat-server**
  The server component of the chatting application

- **jsock-core-server**
  A server-side socket library

- **jsock-core-client**
  A client-side socket library

- **jsock-core-commons**
  A dependency used by jsock-core-server and jsock-core-client which contains the messaging protocol and common classes.