

Security Number Administration and Cryptography

Coursework 2

Niall Gallagher | 4123 Comp | 24/03/2021

Cover Image Source (<https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Devices.html>)

Contents

[Encryption/Decryption Research 2](#_Toc68252491)

[Vigenère Cipher 4](#_Toc68252492)

[Vigenère Pseudocode 6](#_Toc68252493)

[Test Plan 9](#_Toc68252494)

[Test Cases 10](#_Toc68252495)

[Conclusion 16](#_Toc68252496)

# Encryption/Decryption Research

The task assignment required me to research a suitable data encryption/decryption method and develop a prototype of a data security application that ‘The Security Company’ are seeking to use to protect their client’s confidential data. ‘The Security Company’ specifically asked that the prototype be programmed in java. The method must encrypt and decrypt text data (numbers and special characters are to be excluded) and then develop a short, written specification for your proposed solution.

I started my research looking into the most widely used ciphers around the world and doing analysis on which would be a best fit for this ‘Security Company’. My shortlist consisted of two Monoalphabetic and two Polyalphabetic ciphers. These were respectively Caesar (Monoalphabetic), Atbash (Monoalphabetic), Vigenère (Polyalphabetic) and Playfair (Polyalphabetic).

My analysis of the Caesar cipher concluded that the biggest advantage the Caesar possess is its simplicity. This main advantage makes it easier to implement an encryption method for the ‘Security Company’. It is also easy to encrypt longer text. It’s a special case of the Shift Cipher so when the key=3 it will shift the alphabet by three letters. You can also determine the length of the shift.

**Caesar**

My analysis of the Caesar cipher concluded that the biggest advantage the Caesar possess is its simplicity. This main advantage makes it easier to implement an encryption method for the ‘Security Company’. It is also easy to encrypt longer text. It’s a special case of the Shift Cipher so when the key = 3 it will shift the alphabet by three letters. You can also determine the length of the shift.

But even if the opponent trying to decipher the message doesn’t know that it was encrypted using Caesar Cipher but notices that it uses some kind of Shift Cipher will figure out that there are only 26 letters in the alphabet. The main disadvantage of the Caesar Cipher is that it’s not a secure cryptosystem because there are only 26 possible keys to try out. An attacker can carry out an exhaustive key search with limited available computing resources. Therefore, I would not recommend this cipher for the ‘Security Company’.

But even if the opponent trying to decipher the message doesn’t know that it was encrypted using Caesar Cipher but notices that it uses some kind of Shift Cipher will figure out that there are only 26 letters. The main disadvantage of the Caesar Cipher is that it’s not a secure cryptosystem because there are only 26 possible keys to try out. An attacker can carry out an exhaustive key search with available limited computing resources.

Therefore, I would not recommend this cipher for the ‘Security Company’.

**Atbash**

The next Monoalphabetic cipher I looked at was the Atbash cipher. The main advantage of this cipher is that it is quick and easy to conceal information. However, the Atbash cipher is a very weak substitution cipher as there is no secret key behind generating the cipher text alphabet to perform the encryption. It would not take much computational power to break it and therefore I would not recommend the ‘Security Company’ to use this cipher.

**Playfair**

The first Polyalphabetic cipher I looked at was Playfair. It is also a substitution cipher and is difficult to break compared to the simple substitution cipher. It is a very secure cipher as there are 625 possible pairs of letters (25x25 alphabets) instead of 26 different possible alphabets. It is quick to use and requires no special equipment. The main disadvantage of this cipher is it cannot be used for the transmission of a huge amount of data. The ‘Security Company’ requires the encryption of the client’s data which would be a large amount and Playfair can only encrypt a short data set.

This is the main reason why this cipher cannot be used in my application.

**Vigenère**

The last Polyalphabetic cipher I analyzed was the Vigenère cipher. The Vigenère Cipher was designed by tweaking the standard Caesar cipher to reduce the effectiveness of cryptanalysis on the ciphertext and make a cryptosystem more robust. It is significantly more secure than a regular Caesar Cipher.

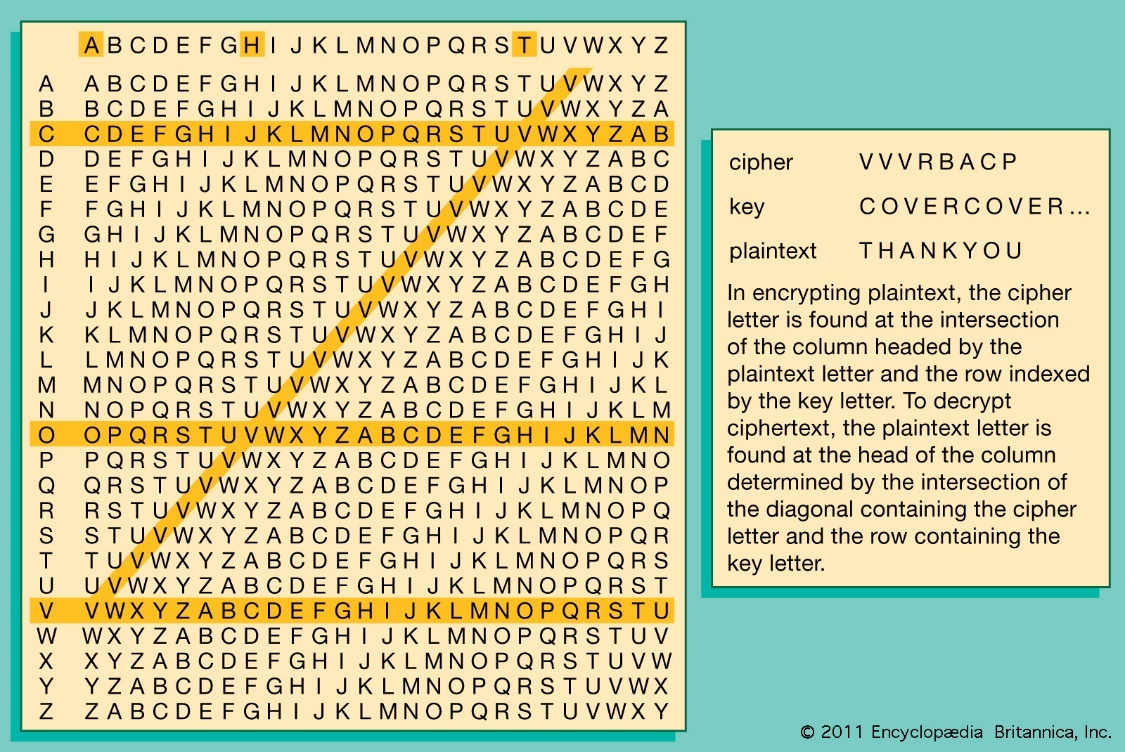
Vigenère cipher is a kind of substitution cipher that employs polyalphabetic substitutions. It is stronger than Caesar cipher as it uses a series of interwoven Caesar ciphers. The cipher uses a key and a double entry table.

## Conclusion

Through my extensive research, I concluded that the Vigenère cipher would be best suited for the ‘Security Company’. It can process a large data amount which will be useful to protect client’s confidential data. The ‘Security Company’ can edit the encryption often by passing a new keyword into the program. The cipher will be relatively easy to implement in the Security Company’s program. The cipher being polyalphabetic will make it a lot more difficult to breakdown than a standard monoalphabetic cipher. I’ll draw up a robust testing plan later in this report to further back my opinion in why this cipher will be the perfect suit for this ‘Security Company’.

# Vigenère Cipher

As mentioned briefly on my research (Page3), I recommended the Vigenère cipher. I will now give a more depth analysis into the cipher’s strengths, weakness and propose why this is the best solution for the ‘Security Company’. Below is a Vigenère table. This is an important guide as it’ll give you a clearer insight into this polyalphabetic cipher.



Source: (<https://www.britannica.com/topic/Vigenere-cipher>)

A 16th-century French diplomat, Blaise de Vigenère, created a very simple cipher that is moderately difficult for any unintended parties to decipher. There are too many possible keys, even if the key is known to come from a particular language. It cannot be broken with the word pattern attack that worked on the simple [substitution cipher](https://brilliant.org/wiki/substitution-cipher/). It is a modified version of the Caesar cipher.

## Proposed Solution

The Vigenère cipher is an effective encryption method. As you are fully aware, client’s confidential data is precious and therefore must be protected. You need an encryption method that will protect your clients’ data from intruders who intend use this data for their own benefit. According to a survey by Innovate MR for Nixplay “Nine in ten security professionals believe that their personal data is available to intruders”. It’s that survey alone that should frighten you to initialize change in your security model. We firmly believe an encryption method is essential for your business to thrive. After all, in Maslow’s hierarchy of needs security is classified as a basic need. We feel this Security applies physically and electronically. An encryption method will be like placing a lock on a door. Its why the Vigenère cipher will be perfect for this program as I feel its strengths outweighs its weaknesses and the cipher’s biggest asset is the versatility of its keyword. Below I have identified the strengths and weaknesses the cipher possesses and I have made recommendations to improve the encryption system.

## Strengths

The strengths of the cipher are that it is not susceptible to frequency analysis due to the cipher which rotates through different shifts, so the same plaintext letter will not always be encrypted to the same ciphertext letter. A Vigenère cipher is difficult to crack using brute-force because each letter in a message could be encoded as any of the 26 letters. Because the encoding of the message depends on the keyword used, a given message could be encoded 26*k* ways, where k is the length of the keyword. For example, in my program I used the key word “secure”, this word has 6 letters so the program can be encoded in 266 = 308 million ways! This is the main reason why I would highly recommend this cipher for the Security Company to use as it’s a very unpredictable efficient cipher which can be dictated by a keyword. The company can change the keyword as often as they want to make sure that intruders cannot break this encryption.

## Weaknesses

The primary weakness of the Vigenère cipher is the repetition nature of its key. If an intruder correctly guesses the length of the key, then the ciphertext can be treated as interwoven Caesar cipher, which individually can be easily broken. The drawback is if the key length is not the equal to the length of plaintext, then the key will be repeated continuously until the same as the plaintext length. This can be a significant worry if the keyword isn’t changed regularly.

# Vigenère Pseudocode

import java.util.\*;

Class SecurityCompanyCipher {

Declare String = File Name;

//Keyword changes the cipher encryption

Declare String = keyword;

public static void main (String[] args) {

saveEncryptedFile method;

}

String generateKey method (declaring String str, String Key) {

integer x = str length;

for (integer i = 0; increment i)

{

if (x equals i)

i=0;

if (key length equals str length)

break;

key += key character at i;

}

return key;

}

static String cipherText (declaring String str, String key)

{

Declare String Buffer

    for (integer i = 0; i < str length; increment i)

{

if (str character at i doesn’t equal to ' ' and also str character at i doesn’t equal to ',' and str character at i doesn’t equal to '.') {

integer x = str character at i + key character at i mod 26;

if (Character is upper case and str character at i) { then x += 'A';}

else {

x += 'a';

}

String Buffer append character with x;

}

else {

String Buffer append (' ');

}

    }

    return StringBuffer to String;

}

saveEncryptedDataToFile method () {

Print (“Encrypting text…”);

Declare String Buffer

Try:

Declare Buffer Reader (Declare File Reader pass through File Name)

{

String line;

while (line = Buffer Reader read the line that doesn’t equal to null)

{

String Buffer append the line;

String Buffer append a line separator;

}

catch (Input Output Exception)

Print (“Writing Encrypted text to File…”);

Declare String Array = String Buffer convert to String split with a (line separator).

Try:

Declare File Output Stream = new text file;

for (int i=0; i<String Array length; increment i) { String key = generateKey (String Array at i, keyword);

file Output stream write cipherText method (String Array at I and the key) convert to Bytes;

file Output stream write (line separator convert to Bytes);

file Output stream close;

catch: (File Not Found Exception)

catch: (Input Output Exception e)

Print ("Finished writing encrypted data to file");

}

}

}

}

# Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case | Test Data | Description of Test | Expected Result | Actual Result |
| 1 | Running File with text called NiallGallagherCW1\_  Part\_1\_Plain\_Text.txt | Verify That the program runs | Program should perform encryption | Program performed encryption |
| 2 | Running File with text and special characters called NiallGallagherCW1\_  Part\_1\_Special\_  Characters\_Text.txt | Verification program accepts special characters | Program should handle special characters | Program handles special characters and encrypts them as they are part of ASCII |
| 3 | Running File with text and spaces called NiallGallagherCW1\_  Part\_1\_Plain\_Text\_  With\_ Spaces.txt | Verify that the Program handles spaces | Program should handle space by not encrypting them. | Program handles spaces effectively as the program runs with spaces. |
| 4 | Running File with no text called NiallGallagherCW1\_  Part\_1\_Empty \_Text.txt | How the program handles an empty file | Program should not write any encrypted text. | Program constantly runs and no encryption is performed. |
| 5 | No File is present. | How the program handles no text File | Program shouldn’t run as there is no file to read | Program doesn’t run and throws file not found error |

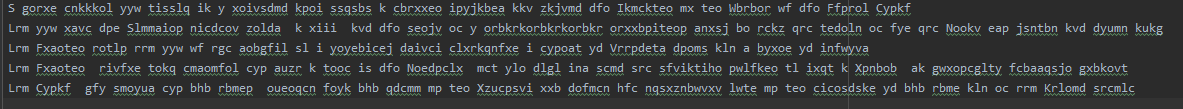
## Testing Process

I drew up my test plan above to see how my program would react to different scenarios and to test the efficiency of my program. The test cases can be seen below where I go into depth about how they were conducted and what my expected results were. The tests helped me to give confirmation that my algorithm was successful in encrypting the text from a file. The cipher performed the encryptions and most of my test’s results went how I anticipated they would go. You can see the actual results in the screenshots in the text cases below.

# Test Cases

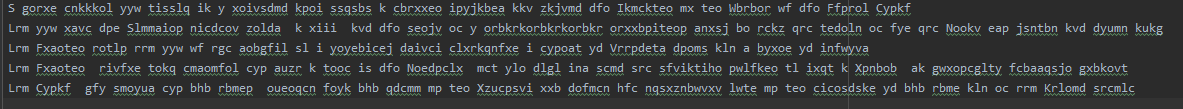
|  |  |
| --- | --- |
| Name | TC-01: Verify that the Program runs |
| Requirement | FR-01 – A program that encrypts plain text from a file, encrypts and outputs to a new file |
| Preconditions | 1. Have the Main class from loaded in your IDE 2. Ensure your plain text file is available - NiallGallagherCW1\_Part1\_Plain\_Text.txt |
| Steps | 1. Run the main class |
| Expected Results | 1. Console pops up with text saying that the text from the file is encrypting. 2. Console tells the user when the file is finished encryption. 3. A new file named NiallGallagherCW1\_Part1\_Cipher\_Text.txt   Should successfully encrypt the text from the file NiallGallagherCW1\_Part1\_Plain\_Text.txt. |

## Result:



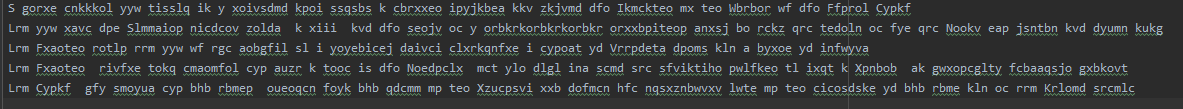
|  |  |
| --- | --- |
| Name | TC-02: Verification program accepts special characters |
| Requirement | FR-02 – A program that encrypts special characters from a file, and encrypts the outputs to a new file |
| Preconditions | 1. Have the Main class from loaded in your IDE 2. Fill the text document with random special characters throughout the text. |
| Steps | 1. Click on the run button in the main class |
| Expected Results | 1. Console pops up with text saying that the text from the file is encrypting. 2. Console tells the user when the file is finished encryption. 3. File should encrypt special characters as encryption accepts all ASCII characters. |

## Result:



|  |  |
| --- | --- |
| Name | TC-03: Verify that the Program handles spaces |
| Requirement | FR-03 – A program that handles spaces and doesn’t encrypt the spaces in the output of a new file. |
| Preconditions | 1. Have the Main class from loaded in your IDE 2. Ensure your plain text file has spaces -NiallGallagherCW1\_Part1\_Plain\_Text.txt |
| Steps | 1. Apply spaces in my text file. 2. Click on the run button in the main class. |
| Expected Results | 1. Console pops up with text saying that the text from the file is encrypting. 2. Console tells the user when the file is finished encryption. 3. File should be able to handle spaces and not encrypt text that isn’t present. |

## Result:



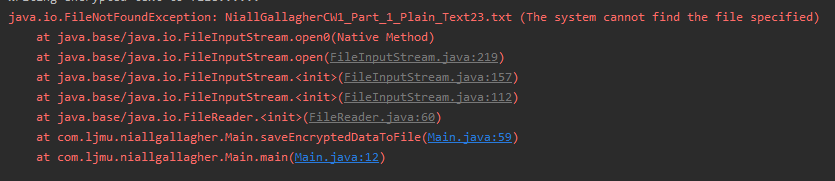
|  |  |
| --- | --- |
| Name | TC-04 How the program handles an empty file |
| Requirement | FR-04 – A program that reads an empty file |
| Preconditions | 1. Have the Main class from loaded in your IDE 2. Ensure your plain text file has no text -NiallGallagherCW1\_Part1\_Plain\_Text.txt |
| Steps | 1. Delete text from preloaded file. 2. Click on the run button in the main class. |
| Expected Results | 1. Program runs but there’s no output as no text is being read from the file 2. The Program should be always running to find text that isn’t present. |

## Result:

****

|  |  |
| --- | --- |
| Name | TC-05: How the program handles no text File |
| Requirement | FR-05 – A program in handling no text File |
| Preconditions | 1. Have the Main class from loaded in your IDE 2. Ensure that a plain text file is not present |
| Steps | 1. Delete the text file NiallGallagherCW1\_Part1\_Plain\_Text.txt 2. Click on the run button in the main class |
| Expected Results | 1. Program should throw the File Not Found Exception 2. Viewer will be informed in the console that the program can’t find the file to read. |

## Result:



# Conclusion

Overall, I’m pleased with my encryption program, as I felt it worked well in encrypting the text. I think my program can be further improved and I have a few recommendations below in how the program should be looked after.

## Improvements

In my opinion the program can reach its maximum efficiency if the “Security Company” implements these recommendations:

1. The “Security Company” regularly changes its keyword so that any cryptanalyst cannot guess the length of the keyword.
2. I would recommend that the “Security Company” uses the same length of client text as its keyword to make sure it achieves maximum Security.
3. My program can be improved by changing the encrypted text to one case so that individual characters cannot be broken.

If the “Security Company” is committed to applying the recommendations I advised above, the Vigenère Cipher would be a very efficient cipher to encrypt the data of their clients.