UMEÅ UNIVERSITY

ENHANCING THE SECURITY OF VIGENÈRE CIPHER USING PADDING

A pattern analysis approach

Nada Sebhatu

Bachelor Thesis, 15 hp/credits

Education program

2025

Abstract

Modern cryptographic algorithms play an important role in our daily lives by providing secure communication online. Before modern algorithms, there were classical ciphers, such as the Vigenère and Caesar cipher. In modern time, these ciphers are considered vulnerable to a number of cryptanalytic techniques and attacks, such as pattern analysis and brute-force attacks. In this thesis, we propose a modification to the Vigenère cipher, in which we enhance its durability to pattern analysis. The modification applied confusion to the cipher by adding additional letters, in English, which we refer to as padding in this thesis, to the encrypted message with the goal of concealing the letter frequency distribution of the cipher text. The performance of the method is evaluated on the accuracy with which the encrypted message is decrypted, the entropy score, and its durability against cryptanalysis. The finding includes a 100% decryption accuracy rate and an entropy of ≈ 4.7004 bits. A drawback of the algorithm is that the encrypted message may be longer than the message due to the added padding.

Acknowledgements

I want to give special thanks to my supervisor Petter for their guidance and insightful feedback. I also want to thank my friends for their continuous encouragement and fruitful discussions.

Contents

1 Glossary

Cryptosystem A cryptographic system that applies encryption and decryption algorithms, as well as protocols to protect information. In this study used interchangeably with cipher.
Crack The action of decoding information.
Cipher A system that performs encryption and decryption algorithms.
Cipher text An encryption of a text produced by a cipher.
Key A sequence of characters used to encrypt or decrypt a message.
Key length The length or size of a key.
Plain text A readable message, before encryption or after decryption.
Vigenère square A table with letters of the alphabet written in different rows, where each letter of the alphabet is cyclically shifted to the left.

2 Introduction

Cryptography is the study of obscuring information to prevent unauthorized access. Modern cryptographic algorithms provide an important service that ensures a secure transmission of information over the Internet. Cryptanalysis, a counterpart to cryptography, is the study of deciphering cryptographic algorithms and finding weaknesses in them with the goal of obtaining the original message from an encrypted text without prior knowledge of the original message or the key used to encrypt it [9]. Classical ciphers, such as Vigenère and Caesar ciphers, have paved the way for modern cryptography; Despite their historical significance, these ciphers are highly susceptible to cryptanalysis techniques, such as brute-force attacks and letter frequency analysis [6].

2.1 Aim of Thesis

The purpose of this thesis is to improve the security of the Vigenère cipher, a classical cipher, making it more resistant to brute-force attacks and letter frequency analysis. In our approach, we develop an encryption method that employs letter frequency analysis preambulary and introduces confusion, in the form of padding, to make the cipher text difficult to cryptanalyze. The performance of the algorithm is evaluated on the accuracy of the decryption method, the entropy of the cipher text produced, and susceptibility to cryptanalysis.

2.2 Thesis Outline

This thesis begins by presenting background information about cryptology along with a description about substitution and transposition ciphers Chapter 3. Additionally, Chapter 3 includes information and cryptanalysis of the Vigenère cipher. Relevant prior work of other authors is then discussed in Chapter 4, after which a detailed explanation of the algorithm is presented in Chapter 5 and the methodology used to test it is presented in Chapter 6. The results are shown and discussed in Chapter 7 where this thesis ends with a conclusion in Chapter 8.

3 Background

In this chapter, a brief description of cryptology is provided, along with a brief description of the two types of cryptographic ciphers: substitution and transposition ciphers. Additionally, an introduction to the classical ciphers, Caesar and Vigenère, as well as cryptanalysis of the Vigenère is presented.

3.1 Cryptology

Cryptology is the study of cryptography and cryptanalysis in order to create secure communication. The goal of cryptography is to render a message unintelligible to hide its meaning through a process known as encryption. To encrypt a message, it is disarranged according to an algorithm that is reversible to make the message comprehensible [14]. The advantage is that if an unauthorized actor, without knowledge of the encryption algorithm, intercepts the encrypted message, they cannot understand its content [14]. Thus, its historical use as a tool to protect national secrets and strategies in government [1]. Cryptanalysis is the science of exposing the meaning of an encrypted message without knowing the key, by identifying weaknesses in the encryption algorithm [14].

Cryptographic ciphers, algorithms used for encryption and decryption, can be divided into two types known as substitution and transposition ciphers [14]. Substitution cipher, as the name suggests, applies an encryption method in which each character in a plain text is substituted for a different character to produce a cipher text. Menezes et al. [1] formally define this process as follows. Let A denote an alphabet consisting of q symbols and M denote a set of all strings of length t created using the symbols in A. Let k be a set of all permutations of A, now for each $e \in K$ define an encryption transformation E_e as follows;

$$E_e(m) = (e(m_1)e(m_2)...e(m_t)) = (c_1c_1...c_t) = c \tag{3.1}$$

where $m = (m_1m_2...m_t) \in M$. The recipient can recover the plain text by inverting the substitution on the cipher text [2]. If we apply this principle using the English language as follows;

$$\begin{array}{cccccccccccc} A & Y & C & L & D & J & M & F & K & Z & W & Q & B \\ \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow & \uparrow\downarrow \\ H & E & G & R & P & O & X & T & S & U & I & V & N \end{array}$$

then, the message "I came i saw i conquered" would turn to "W GHXY W KHI W GJBVZYLYP". The example above exemplifies one of four different types of substitution cipher known as a simple substitution cipher or a monoalphabetic cipher.

In transposition ciphers, the characters in the plain text are not substituted but remain the same, where the position of each character is rearranged to obscure the message [2]. This can be mathematically described as the following. Let t be the length of a string and let K be the set of all permutations of the set $\{1, 2, ..., t\}$, now for each $e \in K$ define the following

encryption function

$$E_e(m) = (m_e(1)m_e(2)...m_e(t)) \tag{3.2}$$

where $m = (m_1 m_1 ... m_t) \in M$, the message space [1]. In a simple columnar transposition cipher, the message is typically written horizontally divided into a fixed number of columns and then read vertically, forming the cipher text [2]. For example, the message "I came i saw i conquered" with 4 columns would look as seen in the matrix below.

$$\begin{matrix} I & C & A & M \\ E & I & S & A \\ W & I & C & O \\ N & Q & U & E \\ R & E & D & X \end{matrix}$$

The cipher text of the message would be "IEWNR CIIQE ASCUD MAOEX". The decryption process requires writing the cipher text vertically into a tabular of identical measurements and reading it horizontally to obtain the message [2].

3.2 Caesar Cipher

Caesar cipher is one of the oldest monoalphabetic ciphers, first recorded to have been used by Julius Caesar. In the Caesar cipher [6], each letter in the alphabet system is assigned a corresponding number, as seen in (3.3).

$$A \Leftrightarrow 0, B \Leftrightarrow 1, C \Leftrightarrow 2, D \leftrightarrow 3..., Z \Leftrightarrow 25 \tag{3.3}$$

The encryption process, in modern interpretation, is based on exchanging letters into numbers, according to the assignment seen in (3.3), then shifting them by a fixed number. The key determines the number by which the letters are shifted. For example, the encryption of "E" with the key "D" would be "H", as seen in equation (3.4).

$$(4 + 3) \mod 26 = 7 \tag{3.4}$$

The Caesar cipher is considered an insecure cryptosystem vulnerable to cryptoanalysis such as brute-force attack and frequency analysis.

3.3 Vigenère Cipher

The Vigenère cipher improved the Caesar cipher by changing the mapping method; instead of shifting the letter by a single number, the letters in plain text are shifted according to a sequence of shifts determined by the key, which is typically a word agreed upon by the sender and the recipient [15].

Figure 1 shows the Vigenère encryption of the message "The secret writing of the cipher" with the key "PLXVYM". The encryption method follows the same logic as shown in (3.4), the first plain text letter "T" is shifted according to the corresponding first key letter "P", thus turning to "I" in the cipher text. The second plain text letter "H" is shifted according to the second key letter "L" , thus turning to "S" in the cipher text, and so on and so forth.

The advantage of this cipher is that the same plain text letter is likely to be encrypted into different cipher text letters depending on the key letter. One of its weaknesses is that it

Plain text: **the**secretwritingof**the**cipher

Key: **PLX**VYMPLXVYMPLXVYM**PLX**VYMPLX

Cipher text: **ISB**NCOGPQRPUITKBMR**ISB**XGBWPO

Figure 1: Vigenère encryption, the bold text highlights how the letters in plain text align with the key letters.

may follow a pattern that can be detected. For example, if you observe the bold text in Figure 1, the segment ”the” appears in the first and nineteenth positions in the plain text and both encrypted to ”ISB” in the cipher text with the distance of 18, indicating that the key length is a multiple of 18.

3.4 Frequency Analysis

Frequency analysis is one of the primary cryptanalytic tools and is the study of the frequency of letters or a group of letters. In the English language, the most frequent letter is ”E” followed by ”T” and then ”A”. With this in mind, we can conclude that the most frequent letter in a cipher text is likely to have substituted the letter ”E” and the second most frequent letter is likely a substitution for ”T” and so forth [14]. This technique, combined with other techniques, can be used to decipher cipher text. However, a cryptosystem can be protected from this vulnerability if the letters in the cipher text are equally frequent.

3.5 Friedman Test

The Friedman test is a statistical test that determines whether a cipher is monoalphabetic, such as the Caesar cipher, or polyalphabetic, such as the Vigenère cipher. Additionally, the test can be used to approximate the key length of a polyalphabetic cipher. The Friedman test calculates the index of coincidence, that is, the probability that two randomly selected sequences in the cipher text are identical, which is higher if the cipher is monoalphabetic. The coincidence index of a cipher text can be used to estimate the key length by dividing the text into a number of groups and calculating the index of coincidence across the groups, if the average is high and approximates a natural alphabet, then the assumed key length is correct, but if it is low, then the assumed key length is wrong [3].

3.6 Kasiski Test

The Kasiski test is a method developed to identify the length of a key to then help reveal the key itself. This test is based on the idea that two identical segments of a plain text may be encrypted into the same cipher text when they appear some positions apart in the plain text. Conversely, if two identical segments in a cipher text occur some distance apart, there is a possibility that they correspond to identical segments in the plain text, as seen in Figure 1. The key length is identified by locating pairs of identical segments and noting the distance between each segment to find the common divisors, as these are likely to be the length of the key [13].

3.7 Brute-Force Attack

The brute-force attack is an attack in which the attacker attempts to crack, inter alia, the encryption keys through trial and error. It is important to note that the more sophisticated the encryption algorithm, the longer it takes to crack it, meaning that the time required depends on the key length and possible alternations [10].

3.8 Key Re-use

There exists a singular encryption algorithm considered to be perfect, named the one-time pad system [5]. The system uses non-repeating keys to encrypt the plain text, to achieve this the key has to be of equal or greater length than the message. They key sequence is used exactly once, otherwise the system's security is jeopardized. It can be viewed as Vigenère encryption, where the key never repeats and is never used again to encrypt a different message. In the one-time pad system, characters can be represented as binary data where the chained XOR method is used to encrypt each character [2]. Here is where the key reuse attack, a known-plaintext attack, comes to practice. The idea behind the attack is to access the plain text and the corresponding cipher text; if the plain text is encrypted as follows.

$$Cipher\ text = Plain\ text \oplus Key \tag{3.5}$$

Then the equation in (3.5) can be reversed to compute the key, as follows:

$$Key = Cipher\ text \oplus Plain\ text \tag{3.6}$$

3.9 Entropy

Entropy, in information theory, is a statistical parameter that measures information and redundancy computed by probability [4]. The redundancy tells the constraint imposed on a string in a language due to its statistical structure, such as its letter frequency distribution, which means how precisely the next letter in a string can be predicted when the preceding N letters are known. This study does not take N-gram[1] entropy into consideration, thus, applying the equation in (3.7) to calculate the entropy H of a text.

$$H = -\sum_{i=1}^{n} p(i) \log_2 p(i) \tag{3.7}$$

Entropy in cryptography is used to indicate the randomness and unpredictability of a text, a desirable quality to avoid noticeable patterns. The closer the entropy value of a text is to the maximum value, the more random it is.

[1] N-grams are a sequence of co-occurring words consisting of N words, for instance, a 2-gram is two words are occur consecutively such as "deep learning" or "the water". (D. Jurafsky and J. Martin. "N-gram Language Models" in Speech and Language Processing, 3rd ed., 2025, pp. 32)

4 Literature Review

Various authors have proposed multiple algorithms to improve the security of the Vigenère cipher with different approaches. In [11] a software named Cryptographer 2.0 was proposed to improve Vigenère cipher by integrating a one-time pad system. The cryptosystem extends the traditional Vigenère square by adding the complete printable ASCII code to it and uses a key equal to or longer than the message to ensure that each character is uniquely encrypted, resulting in an encryption method that is more resistant to letter frequency analysis than the Vigenère cipher. However, to achieve a disturbance in the normal letter frequency, the user must maintain a secure system environment for the software otherwise they risk jeopardizing the key.

Multiple authors have explored the idea of expanding the number of characters used in the Vigenère cipher for key- and cipher text generation. A common flaw in the Vigenère encryption process which is that of key repetition, which occurs if the key is shorter than the message. This flaw was addressed in [8] where the authors propose a novel key generation method that ensures a non-repetitive key, i.e., a key of equal or greater length than the message, by extending the Vigenère square to a $95x95$ matrix that includes numerals and regularly used English symbols. Their findings suggest that the new encryption method has an index of coincidence of 0 and an entropy of ≈ 4.3923 bits, which is higher than Vigenère by 0.6707. The authors claim that the findings make the proposed method more resistant to cryptanalysis than the Vigenère, however, the article does not mention if or how the method is resistant to cryptanalytic techniques such as pattern analysis, since index of coincidence and entropy alone are not great indicators.

A modified hybridization of the Caesar- and Vigenère cipher was proposed in [7]. In addition to the English alphabet, their modified hybrid includes numerals and ASCII symbols. The encryption process introduces confusion and diffusion by performing a Caesar encryption on the key, which is then used to Vigenère encrypt the message. The produced cipher text is converted to their binary values in the ASCII table through chained XOR operations performed on the cipher text. The authors claim that the modified cipher makes the message and the key less predictable and thus less susceptible to brute-force attacks and frequency analysis. However, there is a lack of cryptanalysis in the form of pattern analysis and it is also unknown whether there is a trade-off between security and speed.

Similarly to this study, in [12] the classical Vigenère cipher was modified by randomly adding padding in the form of bits to each byte of the message before encrypting it. The introduction of padding changes the characteristic of the message and thus obscures the repetitiveness of the language. The key is also infused with random bits, making the key length and value difficult to crack. However, it is not clear what the decryption of the cipher text or key is. A drawback of this method is that it increases the length of the encrypted message due to the added padding, where an encrypted message is approximately 56% longer than the original message.

In this thesis, a new approach is proposed in which padding is inserted into the cipher text produced by Vigenère encryption to conceal the real frequency of each letter making it difficult to predict patterns in the cipher text, thus, making it resistant to cryptanalysis tech-

niques. The padding is a sequence of letters in the English language that are underrepresented in the encrypted message. Detailed information is presented in Chapter 5.

5 Proposed Algorithm

The aim of this approach is to develop a system that marks Vigenère cipher cryptanalysis using letter frequency analysis as infeasible. To achieve this goal, we introduced a sequence of confusion, which in the form of padding. In the core of the encryption process, padding, i.e. additional letters, is generated by producing letters that are underrepresented in the letter frequency analysis performed on the encrypted message, as seen in Algorithm 1. This process disrupts the normal English letter distribution, and the frequency of each letter appears equal.

The addition of the padding alone will not ensure the cipher's durability against attacks, since there will be a clear separation of the generated padding and the encrypted message. Therefore, instead of concatenating the padding to the encrypted message,they need to be combined. For the cipher text to be decryptable by the recipient, it has to be inserted in a pseudorandom manner to avoid creating a pattern. As seen in Algorithm 2, the key determines how the two strings are combined.

The decryption process is mainly about reversing the encryption, but before applying the Vigenère decryption algorithm, the encrypted original message and the padding need to be sorted so that the padding is at the end of the string, as seen in Algorithm 3. As seen in Algorithm 4, the sorting process relies solely on the key, since it is what determines the index of where each character of the string is placed.

To better understand the encryption and decryption algorithm, we can illustrate it with an example. For simplicity's sake, let us consider the key "ZBKLFM" and the message "THE VIGENERE CIPHER IS ONE OF THE OLDEST CIPHERS". Applying the Vigenère encryption algorithm on the message with the key results in the following string:

$$SIOGNSDOOCJOHQRPWURPXPTRSIOZQPDTDNNBGFBD \tag{5.1}$$

At this stage, there is an unbalanced letter frequency distribution where the letter "O" is overrepresented, and "F" appears only once. To achieve an equal distribution among the letters of the English alphabet, the padding is generated and concatenated with (5.1) producing the following string:

$$\begin{aligned} &SIOGNSDOOCJOHQRPWURPXPTRSIOZQPDTDNNBGFBD\textcolor{blue}{EQTTUEHRYLV} \\ &\textcolor{blue}{IYZAWHVHMCFJYEJIKNQTXNMDKFEZQTLWEXCYIKAQCYUKSLVCVH} \\ &\textcolor{blue}{AWURFPMBWDYZSUBKGJMMVUOXJJREWFMGXLGLNFSAAHXGZAVLKIPBBZC} \end{aligned} \tag{5.2}$$

In order to combine the encrypted message and the padding in a way that conceals the added padding in (5.2), written in blue, it needs to be shuffled in a pseudorandom way. The key letters are replaced by numbers following the same assignment presented in (3.3), then used to determine the positions in which the characters in the string (5.2) are placed. Thus, the letter "S", which is first in (5.2) is placed at index 25, based on the following equation.

$$(P + K_i) \mod |S| \tag{5.3}$$

Where P denotes the current position, K_i denotes the value at position i in the key and S denotes the string to be sorted, in this case (5.2). Following the same logic, the next letter "I" is placed at index $(25 + 1) \mod 156$, that is, 26. If the position is already occupied, the character is placed at the next unoccupied position searching successively, which is where the variable *Skip* in Algorithm 2 is used. A processes that resembles the transposition cipher described in Section 3. This process is complete when the string is iterated over, resulting in cipher text presented in (5.4).

$$\begin{aligned} &QHHBWUXGRLVXPTTVXCDPHCOLW\mathbf{SI}UYYXNMMVZOUDTEIHDJHKGSDGXN \\ &IKALRCFNFRPSNAPNJJRZXEUBBGZYWYLPEUKEDOTQVCFRJGZOQGRFIYFJ \\ &WYCTPBFJUSAMBZIKDLKOSIEWMNMAVBSOCLQEQMAWVTZAHQK \end{aligned} \tag{5.4}$$

Conversely, the same equation in (5.3) is used to unshuffle the cipher text to reveal the real positions of each letter, for example, the equation reveals that the first letter is placed at the index 25 in the cipher text, $(0 + 25) \mod 156$ and the second at index 26, $(25 + 1) \mod 156$. A Java implementation of Algorithms 2 and 4 can be found in Appendix A.

Algorithm 1 Encrypts a message

Input: Key, Message
Output: CipherText
VigenereText ← VigenereEncryption(Key, Message)
Frequency ← frequencyAnalysis(VigenereText)
Padding ← generatePadding(Frequency, Key)
padding ← fisherYatesShuffle(Padding)
Ciphertext ← insertPadding(VigenereText + Padding, Key)
return CipherText

Algorithm 2 Inserts the generated padding in a pseudorandom way

```
Input: CipherText, Key
Output: MixedText
for character in CipherText do
  if i >= keyLength then
    i = 0
  end if
  if not Visited[position] then
    insert character at shuffledText(position)
    Visit position
  else
    skip = 1
    while true do
      if not Visit[position + skip] then
        insert character at shuffledText(position + skip)
        Visit (position + skip)
        BREAK
      end if
      Skip++
    end while
    i++
  end if
end for
return shuffledText
```

Algorithm 3 Decryptions a cipher text back to plain text

```
Input: Key, CipherText
Output: PlainText
SortedText ← sort(CipherText)
PlainText ← vigenereDecryption(Key, SortedText)
return PlainText
```

Algorithm 4 Separates the cipher text from the padding

```
Input: Key, CipherText
Output: SortedText
for index in sortedText do
  if i >= keyLength then
    i = 0
  end if
  if not Visit[position] then
    insert CipherText[position] at sortedText[index]
    Visit[position]
  else
    skip = 1
    while true do
      if not Visit[position + skip] then
        insert CipherText[position] at sortedText[index + skip]
        Visit[index + skip]
        BREAK
      end if
      Skip++
    end while
    i++
  end if
end for
return sortedText
```

6 Methodology

The proposed encryption algorithm was evaluated on the basis of its ability to correctly encrypt and decrypt messages where the cipher text is not vulnerable to frequency and pattern analysis. This was achieved by encrypting a diverse set of messages to then measure the accuracy of the decryption method. The set contained strings of varying length, as well as natural language and a random sequence of characters. The selected strings test the robustness and error detection ability of the system since they have varying letter frequency and length, to prove the reliability and accuracy of the system. The strings were encrypted using unique keys that can be found in Figure 2 along with the strings used.

Message	Key
I must not fear. Fear is the mind-killer. Fear is the little-death that brings total obliteration. I will face my fear. I will permit it to pass over me and through me. And when it has gone past I will turn the inner eye to see its path. Where the fear has gone there will be nothing. Only I will remain.	QKVJDUROP
Fear is the mind killer.	DRF
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.	LOREMIPSUM
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	FIRE
Lorem ipsum dolor sit ame.	WATER
Hope clouds observation.	IOLJRYUNFGZVHY STFDKSUHGNKWY TS
The mind can go either direction under stress—toward positive or toward negative: on or off. Think of it as a spectrum whose extremes are unconsciousness at the negative end and hyperconsciousness at the positive end. The way the mind will lean under stress is strongly influenced by training.	XZWP
The mind can go either direction under stress—toward positive or toward negative: on or off. Think of it as a spectrum whose extremes are unconsciousness at the negative end and hyperconsciousness at the positive end. The way the mind will lean under stress is strongly influenced by training.	XZWPOMLA
It is impossible to live in the past, difficult to live in the present and a waste to live in the future.	CRYPTOGRAPHY
He who controls the spice controls the universe.	VIGENERECIPHER

Figure 2: The messages and keys used to measure the algorithm's decryption accuracy. The messages and keys are excerpts from books, common phrases, or random sequence of characters.

The algorithm's performance was also evaluated on the basis of the measured entropy and its durability against cryptanalysis techniques such as frequency analysis and key reuse attacks.

For the evaluation of the entropy and durability against cryptanalysis, the text below, a paragraph excerpted from a book[1], was used as the message to be encrypted by the algorithm.

"A beginning is the time for taking the most delicate care that the balances are correct. This every sister of the Bene Gesserit knows. Do not be deceived by the fact that he was born on Caladan and lived his first fifteen years there. Arrakis, the planet known as Dune, is forever his place."

The key used to encrypt the message can be seen below.

[1]The book used is Dune (Book 1: Dune) by Frank Herbert

FTZJJSDP

In order to make the message a valid input for the implementation of the algorithm, any character that is not part of the English alphabet, such as white space and comma, was removed.

7 Result and Discussion

The cipher text for the message proclaimed in the Methodology Chapter 6 is presented below.

KJAALWSHKBGHTVSQEHGOAWRKQTFZKKCGCFTPOFLQBKUNMYODGFYMCQW
JZBRTOAXHYSZJCLFTETFSJARPCWZRVSOTEHZXFCSUKENTBQPXJNQTAQHSRQCZOT
DBTIXHAYNGLRLCZPAZVETFRKQNVBYQAUIRYBUQAICMWMXEDYLXOKXFPAWJMDGKG
XYVBROXGMFYZXHGOJVCZELHICDWNIRSRHZIIKVYPUJZSMFOFDUAWNOJJVOWDYRUD
BYWVPGSRUWEBXNJRSWLIIDAUGJOULMGVZXOXVHBMCBMLXREHBQFUDYKDEDVGKSEI
WGPXSFGDDPYIKKVJXPUROQGCQVZNNGELNLPEMMCVIDANTYEKLTPNWIQAEBWLSMNX
UVMAZYHDFZMLFHPJBIMRVNJWUUPDOTBOQZPYHTASSJMLHCSPKNQCHJTWIBUNIIFPLEUCE

Figure 3 illustrates the frequency of each letter in the cipher text. The same plain text with the same key was encrypted using the Vigenère cipher, resulting in the letter frequency shown in Figure 4.

Figure 3: English letter frequency of the message encrypted using the proposed algorithm, showing how the frequency of each letter appears equal.

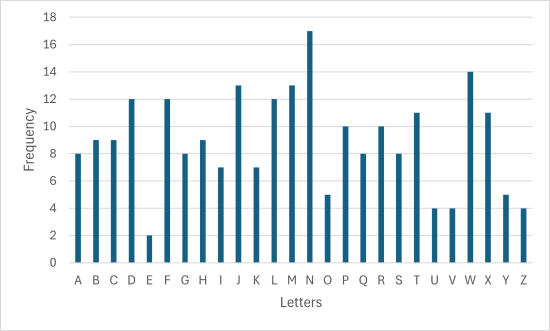

Figure 4: English letter frequency of the message encrypted using Vigenère encryption algorithm, showing that some letters occur more frequent.

7.1 Decryption Accuracy

To evaluate the decryption accuracy of the proposed system, a set of input strings of varying lengths and letter frequencies was provided, as described in Chapter 6. These strings were encrypted using the implementation of the encryption algorithm, the produced cipher texts where then decrypted by the implemented decryption algorithm. All cipher texts were correctly decrypted and this was manually verified, which means that the decrypted cipher text matched the original message, resulting in a 100% decryption accuracy rate.

7.2 Storage Consumption

The message, a paragraph excerpted from a book, contains 232 characters and the proposed system produced a cipher text that is 442 character long, whilst ciphers such as Vigenère resulted in a cipher text that is 232 character long. The cipher text produced by the proposed system is typically longer than the original message due to the padding generated to disguise the actual letter frequency distribution of the cipher text in order to achieve a cryptosystem that is durable against attacks based on letter frequency analysis. This is potentially problematic if storage is a limitation, since this algorithm requires more storage space than classical ciphers. To examine the relationship between the length of the plain text and its cipher text, we encrypted plain texts of sizes 8, 16, 32, 64, 128 and 512 using the same key: Figure 6 illustrates their cipher text length, respectively. The exact messages can be found in Figure 5, and the key used to encrypt them is ”CRYPTOGRAPHY”. These messages were encrypted by the implementation of the algorithm; The length of the cipher texts produced is then used to evaluate the quantity of padding generated to conceal the letter frequency of the cipher texts.

Length	Message
8	Elephant.
16	Abnormalization.
32	The quick brown fox jumps over the dog.
64	Call me Ishmael. Some years ago never mind how long precisely having little or.
128	The duke Leto Atreides leaned against a parapet of the landing control tower outside Arrakeen. The night's first moon, an oblate silver coin, hung well above.
512	In the beginning God created the heavens and the earth. Now the earth was formless and empty, darkness was over the surface of the deep, and the Spirit of God was hovering over the waters. And God said, "Let there be light," and there was light. God saw that the light was good, and he separated the light from the darkness. God called the light "day," and the darkness he called "night." And there was evening, and there was morning—the first day. And God said, "Let there be a vault between the waters to separate water from water." So God made the vault and separated the water under the vault from the water above it. And it was so. God called the vault "sky".

Figure 5: The messages used to measure the length of padding produced. Some of the messages may be cut off to match the required length.

Based on the growth rate observed in Figure 6, we can conclude that the cipher text is unquestionably longer than the message, more than double its length. The exact quotient of the length of the cipher text and the message is shown in Figure 7, we can see that for a message of length 8 the cipher text is approximately 9 times the length of the message, but for a message of length 512 the length of its cipher text is only approximately double. We can also see that the longer the message is, the lesser the quotient, thus, it is reasonable to conclude that the algorithm is best suitable for encrypting longer messages, not necessarily 512 characters long. The tricky part of establishing an exact percentage of padding generated is because other dependent and independent variables, excluding the message length, such as the letters the message contains, as well as their frequency and the quality of the key. A cipher text of a pangram and a random excerpt from a book of equal length would generate very different length padding, with the former having shorter. On average, the cipher text is

approximately 4.2 times longer than the message, based on the values in Figure 6, which is significant compared to other Vigenère modification and hybridization, such as the modification seen in [12] where the addition of padding made the encrypted message approximately 56% longer.

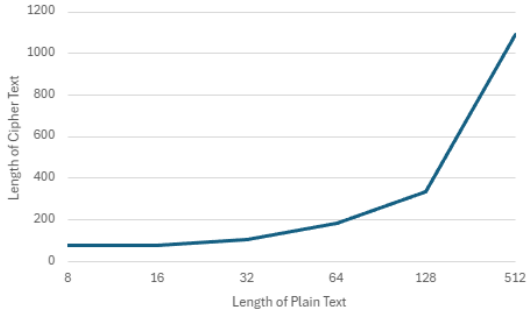

Figure 6: The length of a plain text and the cipher text it produces. It illustrates the growth rate of the length of the cipher text depending on the length of the message.

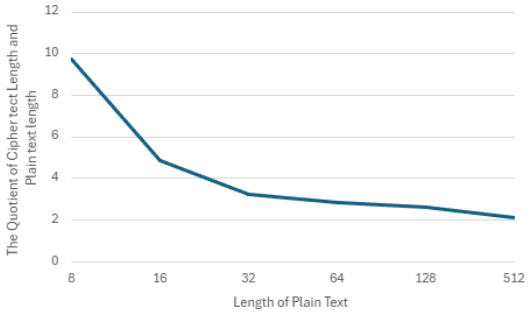

Figure 7: Quotient of the message length and the length of the encrypted message, the longer the message is the lesser padding generated.

7.3 Entropy

The entropy of the cipher text, calculated following the formula described in 3.9, was ≈ 4.7004 bits, which is very close to the maximum randomness. In comparison, the cipher text derived from the Vigenère encryption algorithm has an entropy value of ≈ 4.5790 bits. The cipher text produced using the proposed encryption method has high randomness, higher than the original Vigenère cipher and other Vigenère modifications such as the algorithm proposed in [8] where they encrypted the message ”SECURITY IS ESSENTIAL” using the key ”TRUE”, resulting in a cipher text that has entropy of ≈ 4.3923 bits, while the same key and message pair encrypted using our proposed algorithm produced a cipher text that has entropy of 4.7004 bits. This indicated that it is less likely that the proposed algorithm creates cipher text with predictable patterns, making it harder to predict.

7.4 Cryptanalysis

In this section, we analyze the algorithm’s vulnerability to pattern analysis and reused key attacks.

7.4.1 Frequency analysis

As shown in Figure 3, the proposed encryption algorithm resulted in each letter appearing an equal number of times in the cipher text concealing their actual frequency. In comparison, the cipher text of the message using the original Vigenère encryption method resulted in a letter frequency that is unevenly distributed, shown in Figure 4, making it vulnerable to attacks reliant/dependent on letter frequency analysis. Since the encryption method of the proposed algorithm limits the information that can be extracted from a letter frequency analysis, such as determining the corresponding common English letter based on the number it appears in the cipher text, making the Friedman test quite unproductive.

7.4.2 Reused Key Attack

The proposed algorithm is not without flaws, similarly to the one-time pad method presented in Section 3.8, it is vulnerable to reused key attacks. If two distinct messages of the same length are encrypted using the same key, although the encryption system is not XOR based, it is possible to predict what the encryption key is by examining the relationship between the two messages. This can be executed by observing the positions in which each character is placed in the respective messages and comparing the positions. To mitigate this problem, it is important that the user avoids reusing the same key, which is doable since this system does not require a key of the same length as the message nor a key that is a lexical item. An alternative is to avoid sending multiple messages of the same length, in case it is inconvenient to avoid reusing a key, since the placement of the padding depends on the length of the message and the key, as seen in equation (5.3). In conclusion, sending messages of differing lengths will suffice in changing where the padding is placed, even if the same key is used to encrypt the messages.

8 Conclusion

In this thesis, a new modification for the Vigenère cipher was proposed. The modification attempts to introduce confusion to the cipher by adding padding, which consists of letters. The padding is generated depending on the encrypted message, where additional letters are generated to make the frequency of each letter appear equal to conceal the actual letter frequency of the letter in the message. The proposed algorithm proved that a frequency analysis performed on the encrypted message was not fruitful, since each letter occurs equally, making it difficult to determine the letters that substituted the letters that make up the cipher text. In addition, it produced cipher text that is high in randomness, making it difficult to predict. A downside of the algorithm is that the encrypted message is longer than the message due to added padding. An additional downside is its vulnerability to key-reuse attack due to the way the padding and the encrypted message are combined to create the final cipher text, however, this issue can be mitigated.

Further research can be carried out to investigate the algorithm's durability against cryptoanalysis that the Vigenère is subjected to, such as the Kasiski and Friedman test, to test if the modified version is vulnerable to such attacks. To achieve this, the algorithm needs to tested with real-world usage. Furthermore, research can be done to store the cipher text in an alternative way (since the encrypted message is on average 4 times longer than the message) where the cipher text can be compressed without altering it.

References

[1] Menezes A., Van Oorschot P., and Vanstone S. *Handbook of applied cryptography*. CRC, Boca Raton, 1997.

[2] Schneier B. *Applied Cryptography*. John Wiley & Sons, New York, 1996.

[3] Christensen C. Cryptanalysis of the vigenère cipher: The friedman test. *a a*, 1(1):1, 2015.

[4] Shannon C. Prediction and entropy of printed english. *The bell system technical journal*, pages 50–64, 1951.

[5] Kahn D. *The codebreakers*. The macmillan company, New york, 1968.

[6] Stinson D. *Cryptography : theory and practice*. Chapman & Hall/CRC, Canada, 2002.

[7] Omolara O. E., Oludare A. I., and Abdulahi S. E. Developing a modified hybrid caesar cipher and vigenere cipher for secure data communication. *Computer Engineering and Intelligent Systems*, 5(5):34–46, 2014.

[8] Hameed T. H. and Sadeeq H. T. Modified vigenère cipher algorithm based on new key generation method. *Indonesian Journal of Electrical Engineering and Computer Science*, 28:954–961, 2022.

[9] Rodriguez J. *Genetic algorithm for cryptanalysis on the Vigenere cipher*. PhD thesis, Polytechnic university of Puerto Rico, 2015. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-03-04.

[10] Dave K. Brute-force attack "seeking but distressing". *International Jornal of Innovations in Engineering and Tecknology*, pages 75–78, 2013.

[11] Latinović L. Vigenère cipher improvement software realization and reduction to the one-time pad. Technical report, Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, Serbia, 2021.

[12] Garcia M. and Wilson P. A modified version of the vigenère algorithm. *International Journal of Computer Science and Network Security*, 6(3B):140–143, 2006.

[13] Padhye S., Sahu R., and Saraswat V. *Introduction to cryptography*. CRC Press Taylor & Francis Group, Boca Raton, 2018.

[14] Singh S. *The Code Book*. Delacorte Press, New York, 2001.

[15] Buchanan W. *Cryptography*. River Publishers, Denmark, 2017.

A First Appendix

The Java implementation of Algorithm 2, the mixing of padding and encrypted message, is shown in listing A.1 and the implementation of the separation of padding and the encrypted message, i.e., Algorithm 4 can be seen in listing A.2.

Listing A.1: Combines the padding and the encypted message

```
1  private static char[] insertPadding(String text, String
       ↪ keyWord) {
2      int[] key = convertKey(keyWord, text.length());
3      char[] cipherText = new char[text.length()];
4      boolean[] visited = new Boolean[text.length()];
5      int position = 0;
6      int keyIndex = 0;
7
8      for(int i = 0; i < text.length(); i++) {
9          if(keyIndex >= keyWord.length()) {
10             keyIndx = 0;
11         }
12         if(!visisted[(position + key[keyIndex]) % text.length
               ↪ ()]) {
13             cipherText[(position + key[keyIndex]) % text.
                   ↪ length()] = text.charAt(i);
14             visited[(position + key[keyIndex]) % text.length()
                   ↪ ] = true;
15             position = (position + key[keyIndex]) % text.
                   ↪ length();
16         } else {
17             int skip = 1;
18             while(1) {
19                 if(!visited[(position + key[keyIndex] + skip)
                       ↪ % text.length()]) {
20                     cipherText[(position + key[keyIndex] +
                           ↪ skip) % text.length()] = text.charAt
                           ↪ [i];
21                     visited[(position + key[keyIndex] + skip)
                           ↪ % text.length()] = true;
22                     position = (position + key[keyIndex] +
                           ↪ skip) % text.length();
23                     break;
24                 }
25                 skip++;
26             }
27         }
28         keyIndex++;
```

```
29      }
30      return cipherText;
31  }
```

Listing A.2: Sorts the ciphertext so the padding in at the end of the text

```
1   private static char[] unshuffle(String text, String keyWord) {
2       int[] key = converKey(keyWord, text.length());
3       char[] sortedText = new char[text.length()];
4       boolean[] visited = new boolean[text.length()];
5       int keyIndex = 0;
6       int position = 0;
7
8       for(int i = 0; i < text.length(); i++) {
9           if(keyIndex >= key.length()) {
10              keyIndex = 0;
11          }
12          if(!visited[(position + key[keyIndex]) % text.length()
                ↪ ]) {
13              sortedText[i] = text.charAt((position + key[
                    ↪ keyIndex) % text.length());
14              visited[(position + key[keyIndex]) % text.length()
                    ↪ ] = true;
15              position = (position + key[key]) % text.length)
16          } else {
17              int skip = 1;
18              while(1) {
19                  if(!visited[(position + key[keyIndex] + skip)
                        ↪ % text.length()]) {
20                      sortedText[i] = text.charAt((position +
                            ↪ key[keyIndex] + skip) % text.length
                            ↪ ();
21                      visited[(position + key[keyIndex] + skip)
                            ↪ % text.length] = true;
22                      position = (position + key[keyIndex] +
                            ↪ skip) % text.length();
23                      break;
24                  }
25              }
26              skip++;
27          }
28      }
29      keyIndex++;
30  }
```

UMEÅ UNIVERSITY