On the Construction and Cryptanalysis of Multi-Ciphers

Arush Chhatrapati
Henry M. Gunn High School (joining UC Berkeley, Fall 2021)
Palo Alto, CA, USA
arush.chhatrapati@gmail.com

July 28, 2021

Abstract

In this compilational work, we combine various techniques from classical cryptography and steganography to construct ciphers that conceal multiple plaintexts in a single ciphertext. We name these "multi-ciphers". Most notably, we construct and analyze a Four-In-One-Cipher: the first cipher which conceals four separate plaintexts in a single ciphertext.

Following a brief overview of classical cryptography and steganography, we consider strategies that can be used to creatively combine these two fields to construct multi-ciphers. Finally, we analyze three multi-ciphers which were constructed using the techniques described in this paper. This cryptanalysis relies on both traditional algorithms that are used to decode classical ciphers and new algorithms which we use to extract the additional plaintexts concealed by the multi-ciphers. We implement these algorithms in Python, and provide code snippets.

The primary goal of this work is to inform others who might be otherwise unfamiliar with the fields of classical cryptography and steganography from a new perspective which lies at the intersection of these two fields. The ideas presented in this paper could prove useful in teaching cryptography, statistics, mathematics, and computer science to future generations in a unique, interdisciplinary fashion. This work might also serve as a source of creative inspiration for other cipher-making, code-breaking enthusiasts.

1 Background and Related Work

We briefly summarize relevant background information. Related work on the concept of a "multi-cipher" is referenced in 1.4. Note that in the full version of the paper, which can be accessed here, we include three full-length sections on background related to the field of classical cryptography, the cryptanalysis of classical ciphers, and steganography. These sections are useful to a reader who is unaquainted with the field of cryptography and would make this paper accessible to people outside of academic circles. However, since the audience, in this case, is the research community, we assume that they have familiarity with the basics, and only include summaries.

1.1 Classical Cryptography

Most of the encryption algorithms that we use to keep our information secure on the Internet today have been proven secure through higher-level theoretical mathematical models. This is modern cryptography, concerned with proving the security of existing algorithms and designing new algorithms that have applicability in the real world. Classical cryptography, on the other hand, revolves around how ciphers were used in the past. Today, classical cryptography is less concerned with designing new ciphers and more concerned with designing algorithms to implement and break historical ciphers using modern computing methods. Secret codes and ciphers have long been used to encrypt confidential information. But most historical ciphers have fallen into disuse due to their weaknesses in our modern compututing world. Though historical ciphers aren't used often, they still serve a valuable role. They help us to learn from our mistakes, ensuring that we don't

design the same weaknesses into our modern cryptosystems as we once did in the past. And whether it means providing an outlet for creativity by encouraging students to design their own ciphers or teaching them critical programming skills through the implementation of ciphers in languages like Python and C++, ciphers are also a great way to interest younger students in a broad range of subjects, from statistics to computer science. Over time, it has been determined that classical ciphers generally fall into one of five or six categories: substitution ciphers, transposition ciphers, polyalphabetic ciphers, fractionating ciphers, or polygraphic substitution ciphers (like the Playfair cipher and the Hill cipher). Other categories of classical ciphers combine the encryption methods from one or more of the above categories, such as transposed substitution ciphers.

1.2 A Brief Survey on the Cryptanalysis of Classical Ciphers

We review algorithms that are commonly used to cryptanalyze classical ciphers today, and reference some of these algorithms in our discussion of the cryptanalysis of multi-ciphers in 3. First, we consider substitution ciphers, both monoalphabetic and homophonic. Recall that a homophonic substitution cipher uses an expanded ciphertext alphabet which includes more than 26 characters. Most of the recent literature has focused on improving the efficiency of attacks on homophonic substitution ciphers, while attacks on simple substitution ciphers were considered in past literature. We discuss it below:

- Early literature focused on fast statistical analyses of simple substitution ciphers. This included [CM86], [SS93], which used a simulated annealing algorithm, and [Jak95]. Jakobsen's algorithm was thoroughly discussed in a recent paper by Dhavare, Low, and Stamp [DLS13] which modified the algorithm and used a nested hill-climbing technique in order to efficiently solve homophonic substitution ciphers. Based on this work, we attempted an implementation of Jakobsen's algorithm in the Python programming language, and succeeeded. Since the algorithm is statistical, it does not work for shorter ciphertexts, but is extremely efficient.

- Around the same time, genetic algorithms were given general consideration [Mat93], and applied to simple substitution ciphers [SJNK93]. Dictionary attacks were being considered, too [Har94] [Luc88], mainly for the purpose of solving cryptograms, short substitution ciphers that are often found in newspapers. The issue with these methods was that they tended to overrely on the dictionary. When ciphertext words either did not appear in the dictionary or decrypted to multiple possibilities, the cipher could not be decrypted to a unique solution. Later, Hasinoff developed an automated solver known as Quipster [Has03] that attempted to address some of these problems, but there was still one main issue that needed to be addressed. All of these ciphers required word boundaries to be preserved; that is, spaces needed to be present in the message. It wasn't until a few years later that Olson [Ols07] finally presented a dictionary attack which works on short ciphers, regardless of whether word boundaries are preserved (his implementation is known as QuipQiup). Based on our research, this is, to date, the most effective automated substitution cipher solver.

- The major recent works related to the cryptanalysis of homophonic substitution ciphers include [Ora08], [DLS13], and [Kop19]. We note that the most recent work [Kop19] is compilational, and includes an in-depth discussion of the related work in this field, including the two other major works which we referenced here. We summarize their relevant findings below:

  - [Ora08] uses a genetic algorithm combined with a dictionary attack to optimize the mapping of dictionary words onto the homophonic ciphertext. The authors test their method on the Zodiac-408 cipher – a 408 character long homophonic ciphertext with a 54 symbol ciphertext alphabet which was encrypted by the infamous Zodiac killer and solved back in 1969 – and successfully decrypt it with 100% accuracy using a dictionary with only 850 words and 600 generations.
  - [DLS13] improved upon the state-of-the-art algorithm to solve homophonic substitution ciphers by using a nested hill climbing approach derived from Jakobsen's algorithm. This means that they used two hill-climbing layers: an outer layer and an inner layer. We actually tried to

implement this algorithm, but since the implementation was done in Python, it was far too slow to break a single homophonic substitution cipher in any less than two hours, even when the symbol set consisted of only 35 characters. For a time comparison, the authors of the paper used C++, and with their implementation were able to solve a 35 character long homophonic cipher in approximately 20 seconds for a comparable ciphertext size. Their data is quite impressive; in fact, even for a homophonic ciphertext 9000 characters long with a 65 symbol ciphertext alphabet, the decryption only took approximately 4.3 minutes. It was tested against the Zodiac-340 cipher, and unfortunately, failed to solve it. Since the Z-340 cipher was recently solved, we now know that it was a *transposed* homophonic cipher, so it makes sense that the authors were unable to solve it as a simple homophonic cipher back in 2013.

- According to [Kop19], the best tool currently avaiable for solving homophonic ciphers is AZdecrypt, the same software that was used to crack the Zodiac-340 cipher. The authors also present an almost comparable analyzer which uses a simulated annealing algorithm and integrate it into the open-source codebreaking software Cryp-Tool 2. Their software has a user-friendly interface as shown by the various diagrams which they include in their paper, and may be a viable option for hobbyists who are interested in using software to break this category of ciphers.

Next, we discuss works related to attacks on transposition ciphers, focusing mainly on transposition ciphers that use permutations as encryption keys, such as columnar transposition. Many of the papers we reference use algorithms which efficiently search through the keyspace of possible permutations that were used to encrypt a transposition cipher:

- One of the earlier works [Dea81] proposed using an interactive solver to solve columnar transposition ciphers. The author implemented a manual solver in BASIC. We implemented an analagous tool to manually solve all types of grid transposition ciphers, including columnar transposition ciphers. It can be found at our implementation (link in Appendix) under transposition/manualTranspositionDecoder.py

- The automated cryptanalysis of transposition ciphers was originally conducted using a simulated annealing algorithm [GS94]. The algorithm was effectively summarized in a compilational paper a few years later [DG03], which noted that automated attacks on transposition ciphers used either a simulated annealing algorithm, a genetic algorithm, or another type of attack known as the *tabu search*. An analagous comparison of these three algorithmic approaches was included in Garg's [Gar09], which included graphs that showed that there was no significant performance difference between these three approaches at the time. Around the same time, a simulated annealing genetic algorithm for transposition ciphers was proposed [BSJ08].

- More recent approaches to solving transposition ciphers have ranged from using broader evolutionary algorithms [BD14] to a combination of hill-climbing and simulated annealing [MK14]. Quite recently, Lasry, Kopal, and Wacker solved a double transposition cipher challenge using both a hill-climbing attack and a dictionary attack. They described their "divide-and-conquer approach" in [LKW14]. The latest work by the same authors [LKW16] proposes a method to solve columnar transposition ciphers with incredibly large permutation sizes (up to 100).

Next, we look at polyalphabetic ciphers:

- The earliest work to discuss automated attacks on polyalphabetic ciphers was [CR87], which used a computer program written in the artificial intelligence language Prolog. Genetic algorithms for polyalphabetic ciphers were first considered in [CD97]. The basic idea was to run P genetic algorithms in parallel, where P is the number of alphabets that were used for the polyalphabetic cipher.

- The application of genetic algorithms to polyalphabetic ciphers was further considered a few years later in multiple compilational works [Ber07] [BSJ08]. [TA08] specifically considered using genetic algorithms to search the keyspace of polyalphabetic ciphers. Another work [MW06] described a genetic algorithm using evolving keys that could be used to solve a subcategory of polyalphabetic ciphers known as Type II ciphers, or "Quagmire II".

- One of the most recent works [Kae20b] proposed a hill-climbing technique to solve any general periodic polyalphabetic substitution cipher. The algorithm was thoroughly described, and we were able to test an implementation in Python. Though our implementation was notably slower, it still solved the periodic cipher to an extent where a cryptanalyst could easily make out the words which were expected to appear in the plaintext after letting the program run through a couple of loops. Another recent work [SA20] cryptanalyzes polyalphabetic ciphers using another category of evolutionary algorithms known as "differential evolutionary algorithms", and contains detailed data tables showing performance results.

Finally, we consider the Playfair and Hill ciphers. Playfair ciphers have been shown to be most effectively attacked using simulated annealing algorithms [Cow08]. The most recent approach [AIT18] uses an efficient combination of data compression and simulated annealing to solve Playfair ciphers as short as 60 characters long. Evolutionary approaches have also been considered [Rhe03] [Neg12], but the algorithm described in [Rhe03] takes several hours to run, and [Neg12] only works for longer ciphertexts. On the other hand, the ciphertext-only attacks that have been used on Hill ciphers rely on a completely different type of algorithm. One category of algorithms is originally based on [BM07]. As described in [KA17], these algorithms reduce the time complexity of the brute-force attack from $O(n^3 \cdot 26^{n^2})$ to $O(n \cdot 13^n)$, where n is the key matrix dimension, by brute-forcing the individual rows of the key rather than the entire matrix. We recommend that interested readers look at Teseleanu's recent work [Tes20], which thoroughly considers works related to the cryptanalysis of Hill ciphers and explains how state-of-the-art techniques can be extended to programatically attack a variant of the Hill cipher known as the affine Hill cipher (essentially a Hill cipher plus a Vigenere cipher).

1.3 Steganography

First, we note that "steganography" is a broad term, but in the context of this study, which deals with the creation and cryptanalysis of ciphers, we only consider pen-and-paper steganography. Outside of the context of this study, digital steganography is sometimes used to hide messages in images and audio files.

Steganography refers to hiding a message in plain sight. Usually, the message is hidden in a way that makes it relatively unobvious that the message is hidden at all, so that an attacker can not even detect the presence of a hidden message. The following two examples of steganography will allow the reader to better understand the discussions in the following sections:

- Letter patterns: given a steganographic text, reading each letter at a certain position in each word in the text will reveal a hidden message. For example, reading the first letter of each word from left to right might reveal a message.
- Indicators: Specific words or character groupings that appear in a text such that the positioning of these elements conceals a message. This term is more broadly defined, and more difficult to understand without an example, so we refer interested readers to the full version of the paper to see how indicators can be used in practice.

Specific examples of each of these forms of steganography have been ommitted, and will instead be considered in the context of multi-ciphers in 2.

1.4 Related Work

There are only two works that are directly relevant to the novel contribution that we present in this paper – that is, the idea of a multi-cipher. As described in [Bau13], a two-in-one cipher system was first devised nearly 100 years ago by Jack Levine, a teenager at the time. The cipher relied on matrices, and appeared in a 1926 issue of a pulp magazine known as *Flynn's Detective Weekly*. Though the specific details of the cipher remain unclear, Levine's encryption scheme could supposedly be extended to conceal a third plaintext.

In a more recent work [Kae20a], Kaeding presents a single cipher which conceals two plaintexts in a single ciphertext through enumeration of the permutations of the ciphertext symbols in the message. Though creative, this cipher is limited in the sense that there is no way to conceal more than two plaintexts in a single message. The techniques discussed in this paper are more flexible because the steganographic concealment methods can be layered on top of one another to conceal anywhere between two to five separate plaintexts in a single ciphertext.

2 Construction of Multi-Ciphers

A "multi-cipher" is any cipher which conceals more than one plaintext in a single ciphertext. The only condition that must be satisfied for a cipher to technically be considered a multi-cipher is that there exists a well-defined, implementable encryption algorithm that combines the individual plaintexts to create the multi-ciphertext. However, there also exist some recommended conditions:

1. The purpose of a multi-cipher is to encrypt completely unrelated plaintexts into a single ciphertext, and hence, the individual plaintexts which make up the cipher should ideally be separate and completely independent of each other before encryption. In other words, someone who is able to recover one plaintext from the multi-cipher does not need to have recovered any of the other plaintexts to understand its context. For example, assume that a cipher conceals two plaintexts. One plaintext comes from the first half of a text, and the second plaintext comes from the second half of the same text. Since the first half is required to understand the context of the second half, this is not an ideal application of a multi-cipher. The first half and second half of the same text could just as well have been spliced together and encrypted using any classical cipher. The multi-cipher operates according to the fundamental principle that the plaintexts within it are marked as fundamentally different from each other.

2. The **degree** of a multi-cipher is loosely defined as the number of different ways that one can look at a multi-cipher according to the considerations which we designate in 2.1. A multi-cipher should ideally have a degree greater than one. One example of a multi-cipher that has a degree equal to one is shown in Table 1. Interlaced plaintexts are useful in the construction of multi-ciphers, but on their own, they only provide one way of looking at a multi-cipher. According to this definition, we consider the multi-cipher from [Kae20a] a second degree multi-cipher because the ciphertext can be viewed as either the substitution of mixed-radix numbers for plaintext symbols or the enumeration of permutations of ciphertext symbols.

T	O	I	E	I	A	T	E	E	E	I	M	S	Y	C	E	P	I	E	K
R	H	S	S	S	S	R	H	R	F	D	R	A	T	B	I	P	H	N	R

Table 1: An interlacing multi-cipher of first degree which we call the "shoelace cipher". Together, encrypting the first plaintext "THISISTHEFIRSTCIPHER" and the second plaintext "ROSESAREREDMAYBEPINK" we read horizontally to get the multi-ciphertext "TOIEIATEEEIMSYCEPIEKRHSSSSRHRFDRATBIPHNR".

2.1 Considerations

The following factors should all be given consideration when constructing a multi-cipher. The more considerations that are taken into account, the greater the degree of the multi-cipher, and the more plaintexts can be concealed. Below, we provide a simple list. We will show how each of these considerations can be implemented in a multi-cipher in 3.

- Arrangement of letters
- Spacing
- Capitalization
- Interlaced plaintexts
- **Bolded**, *italicized*, or underlined text
- Coloring
- Indicators in the multi-ciphertext

Take note that some of the considerations listed above overlap. Spacing and capitalization are both possible examples of indicators in the multi-ciphertext, because they can indicate the presence of additional concealed plaintexts. Additionally, the arrangement of letters can be influenced by interlacing the plaintexts.

Also, the indicators are only factored into the calculation of the degree of a multi-cipher if they appear in the larger ciphertext, not in any of the individual plaintexts. If one of the individual plaintexts uses steganography with indicator keywords to conceal an additional hidden message, this would not be considered.

2.2 Efficiency

Cipher Type	Sometimes Uses Padding?	Efficiency score range
Monoalphabetic/Homophonic Substitution	No	$\mathcal{ES} = 1$
Transposition	Yes	$\mathcal{ES} \leq 1$
Polyalphabetic	No	$\mathcal{ES} = 1$
Polygraph substitution (Playfair/Hill)	Yes	$\mathcal{ES} \leq 1$
Variable length substitution (Morse code)	No	$\mathcal{ES} < 1$
Fractionating	No	$\mathcal{ES} < 1$
RGB Cipher	Yes	$\mathcal{ES} > 1$
Three-In-One Cipher	Yes	$\mathcal{ES} < 1$
Four-In-One Cipher	Yes	$\mathcal{ES} > 1$

Table 2: Efficiency score ranges for different classical ciphers and multi-ciphers.

We define the efficiency score of a multi-cipher as the sum of the lengths of all of the plaintexts that are concealed by the multi-cipher divided by the length of the multi-ciphertext. Using notation, this can be written as:

$$\mathcal{ES} = \frac{\sum L_P}{L_C}$$

Where L_P is the length of an individual plaintext and L_C is the length of the overall ciphertext. We say that a multi-cipher is **efficient** if $\mathcal{ES} > 1$. In other words, the sum of the lengths of all plaintexts that are concealed by the cipher is greater than the length of the ciphertext.

For most classical ciphers (which conceal one plaintext), the length of the plaintext is usually approximately equal to that of the ciphertext, so $\mathcal{ES} \approx 1$. We say approximately because padding is considered when calculating the efficiency score. Since padding can make the ciphertext slightly longer than the original plaintext, this can cause the efficiency score to be slightly less than one. This is with the exception of fractionating ciphers and variable length substitution ciphers, which frequently substitute multiple ciphertext characters for individual plaintext characters, and therefore always have $\mathcal{ES} < 1$.

As we will see in the next section, multi-ciphers are useful because they can allow us to achieve $\mathcal{ES} > 1$. Basically, this means that we are concealing more plaintext with smaller amounts of ciphertext. Observe Table 2, which displays the efficiency score ranges of classical ciphers and provides a preview of the efficiency score ranges that we will see for the multi-ciphers that we cryptanalyze in 3.

3 Cryptanalysis of Multi-Ciphers

In this section, we cryptanalyze three different multi-ciphers. The first, the RGB cipher, is the most efficient and conceals two plaintexts. The second, the three-in-one cipher, conceals three plaintexts but is inefficient. And finally, the four-in-one cipher is efficient, *and* can potentially be modified using techniques that were used in constructing the first two multi-ciphers to include a fifth plaintext without changing the ciphertext length, further improving its efficiency.

Throughout this section, we display screenshots of programs that were used to cryptanalyze the ciphers. This programs come directly from our implementation, the link to which can be found in the Appendix. Additionally, we calculate numeric values for the efficiency scores of each of the three multi-ciphers and include Python code snippets that can be used to extract the various components of each of the ciphers.

3.1 The RGB cipher

Figure 1: The RGB cipher

Out of the three multi-ciphers that we present in the section, this one is by far the easiest to cryptanalyze. Just looking at the cipher, there are two things that stand out:

1. The letters
2. The colors

Suppose we look at the letters first, and ignore the colors. Then we obtain the following ciphertext:

ITHHOAOIOOOEETTSOFYEHOWIGATNANTEEESDMOIMETWSSEOMIDEHLTNOHMTI
SEHYFRNFOTEETEEFTAPEYSNTTTEINHDDHAEORTYIRFOOTOGUGTNOEDTMNGOG
DHTMDIURFNRSSHHOULSRERINHSHGITHDTMYFUDRAIONPNLUJNIESLIOEEICN
MOUUWWDHEOSTOEAIENHTDTERETMLADINITENHWFNTTLHWIEOGPTIFEDRTAGU
AMTGOFYETWHHRNTDSDEHTFNSILEOGINEDHNEEOAGHEOIATNIOAHTIXAPTTES
FSRRLAGALHOATYMTHXUDSETHHCINAEIIEIARKHODFOWEQNHSTADMRAHBTHNX
ROOKENTTLEFTHSAHATNIIVEEAARGNNDVMDRTYEUSALTMUGSTOTWETDEUHTFX

A frequency analysis reveals that the most common letters appearing in this message are T, E, H, O, I, and N, with frequencies 12.62%, 11.67%, 8.33%, 8.33%, and 7.38%, respectively. We can compare this to standard English frequencies from 4. The most frequent letters appearing in the message are also among the most frequent letters that appear in the English language, and the numeric letter frequencies match closely. This suggests that we might be dealing with a transposition cipher. To check, we can do a chi-squared test, and we get a χ^2 of 70. Dividing this by L, the length of the message, we find that $\frac{\chi^2}{L} = 0.17$. This is very close to the $\frac{\chi^2}{L}$ values of 0.03 and 0.05 we obtained when analyzing different parts of Shakespeare's "Hamlet" for a transposition cipher – note that this is a reference to data included in the full version of the paper – and therefore we come to the conclusion that this has probably been encrypted with transposition.

The main clue that can be used to decrypt a transposition cipher are the factors of the length of the message. The encryptor likely would have chosen a value for the key length that would go in evenly, simply for ease of encryption. The length L of the message is 420. Factors include:

(1, 2, 3, 4, 5, 6, 7, 10, 12, 14, 15, 20, 21, 28, 30, 35, 42, 60, 70, 84, 105, 140, 210, 420)

In this case, the factors do not provide much of a clue to the key length that was used, because there are simply so many. From here, the best way to solve a transposition cipher is to brute-force through every commonly used algorithm, setting the key length, or the permutation size, equal to the factor. If this were a transposition cipher that used a permutation as the encryption key, we would recommend using a genetic fitness algorithm if the permutation size is greater than 7. But for permutation sizes less than or equal to 7, a brute-force strategy works effectively.

Since all of the numbers in the range of 2 to 7 are factors of the message length, we first try bruteforcing through the permutations of size 2 to 6 using simple horizontal transposition. The output is shown in Figure 2. Scoring is based on the count of common English words in the message using an efficient Trie data structure. Unfortunately, this does not recover the plaintext. Next, we might try bruteforcing through the same permutation sizes for a columnar transposition. We find that this does not recover the message, either.

Notice that one of the factors of the message length is 7. We repeat the brute-force process, testing all permutations of size 7 alone, for both simple horizontal and columnar transposition. Finally, the cipher is solved. It used a columnar transposition with the encryption permutation (3, 5, 4, 1, 2, 6, 7) as shown in Figure 3.

The second plaintext uses the colors. Let's convert the colors to letters. We'll use "R" to denote a red letter, "G" to denote a green letter, and "B" to denote a blue letter that appears in the RGB cipher. We get the following:

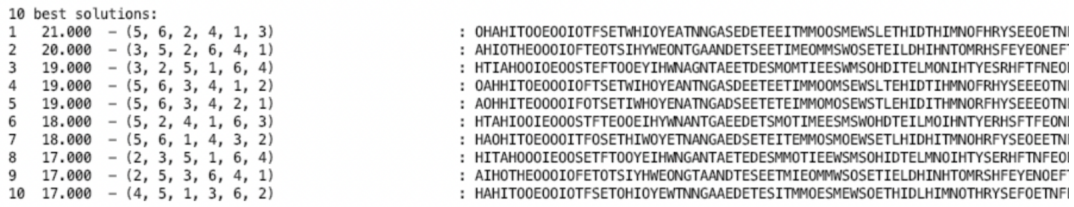

Figure 2: Best solutions from bruteforcing horizontal permutations from size 2 to 6.

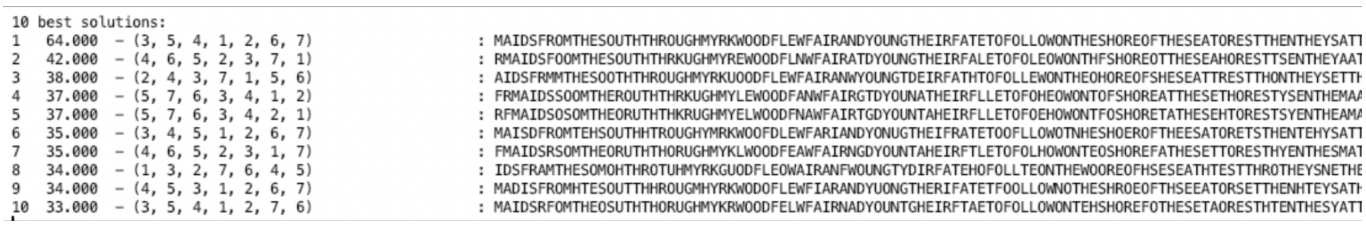

Figure 3: Best solutions from bruteforcing vertical (columnar) permutations of size 7.

RGGBRGBBBRBGRBBBRRGGBGRBGGRGRRRBGBBBGGGRBGBRRRBBGGGRBGRGGBRGBBBRBGRRBGG
RRBRBGBBRRBGBRRBBGGRBBRBGBGGGRRBGBBBBRRBBBGRBRRBGRGGBRGBBBRBGRRBGGRRBRRG
GBRRGBRGBRBGBGGRRBGGGRGGBGGRGRRGRRGGRRGGRBGGGRGGBGGRGRRGRRGGRRGGRBGBR
GBGGRBBRBRGGGRGGBRRGBRGBRGRRBRRRBBBGGRBGBRGBGGGRRBGGBBBGRRBGGGGBBBBGGG
RRBRBGBRGBRRGGGBRBRBBRBGBRGBBBRBGRRBGGRRBRBBRBBBGGGRBGBBRBRGBBBGGGBRBBGG
RBBGGRBGRGGRBBRRBRRRBBGGRGGGRGGBRGBBBRBGRRBGGRRBRBBRBBBGGGGRBBBBBGB

Notice that in this cipher there are 3 possible symbols for any given letter: R, G, and B. Let's look at the message in blocks. There are a total of $3 \cdot 3 \cdot 3 = 27$ unique values that could potentially appear in any individual block of three letters in the message.There are 26 letters in the English alphabet, so it is possible that three-letter blocked groupings of these color letters could respresent individual letters in the English language. A frequency analysis of trigrams using a blocks window reveals that there are only 20 unique trigrams that appear in the message, and the most frequent trigrams are "RBG", "GGR", and "BBB", appearing with frequencies 12.14%, 10.71%, and 9.29%, respectively. This appears to be a fractionating cipher. If the distribution was uniform we would expect to see closer to 27 distinct trigrams with a much flatter frequency distribution. But the most frequent frequencies in this ciphertext match up pretty closely with the frequencies of the letters "E", "T", and "A" from 4.

To solve this fractionating cipher, we need to map trigrams in the message to individual letters in the English A-Z alphabet, and then solve the result as a monoalphabetic substitution cipher. The mapping is arbitrary. In our implementation, we have a program that does this for us at "fractionation/ngramsToLetters.py". We simply set $n = 3$, input the RGB ciphertext above, and it uses a simple algorithm to map the trigrams to letters by order of appearance, from left to right. The first distinct trigram appearing in the message is mapped to "A", the second to "B", and so on. After this mapping, we get:

ABCDEFGHIJDCKDFEKDABCDLIMNFNLOEDOJNCLPMDABCDLIMAFQQNIDIGIJJIADIGIJJIAD
BOEMKAFQQJFEODBOJOCJOGCKLDBFKREDBCDLIMSCKDSBCKROEITISMEIKABCDLIMSCKHCN

This is a short 140 character long short substitution cipher without spaces. It is most effectively solved using the powerful dictionary attack described in Olson's [Ols07], source code for which can be found at QuipQiup. Doing so reveals the second plaintext:

WHATS IMPORTANT ISNT WHAT YOU DID YESTERDAY BUT WHAT YOU WILL DO TOMORROW TOMORROW THE SUN WILL RISE THERE ARE MANY THINGS THAT YOU CANT CHANGE SO FOCUS ON WHAT YOU CAN PAD

Now we calculate the efficiency score of this cipher. The length of the first plaintext, excluding mandatory padding, was 418 characters, because there were two requisite "XX" characters for padding at the end. Note that the plaintext contained filler text which included the words "I'M GOING TO PAD... WITH FILLER TEXT...", but this is technically a part of the message, so we consider it in the calculation. If a longer plaintext had been chosen, then the filler text could have been *replaced* accordingly. Therefore, it is not considered mandatory padding. The second plaintext had a length of 140 characters – the word "PAD" at the end is filler text, not mandatory padding – so the sum of plaintext lengths is $418 + 140 = 558$. The ciphertext length is 420 characters. Hence, the efficiency score is

$$\mathcal{ES} = \frac{558}{420} \approx 1.33 > 1$$

We were able to conceal roughly 33% more plaintext *per character in the ciphertext* compared to if this was just a classical cipher with an efficiency score of 1 – namely, any classical cipher except a fractionating cipher or a variable length substitution cipher – by using colors. This is an example of a second degree multi-cipher because the ciphertext can be looked at as either a transposition cipher using the letters or as a fractionating cipher using the colors.

One main issue with this cipher is that colored fonts are often not easily transferred across different platforms, which is also part of the reason why we had to display the RGB cipher as an image. A simple variation might be that instead of red, green, and blue letters, one could use **bolded** letters, *italicized* letters, and normal (or underlined) letters to represent the three different RGB symbols.

3.2 Three-In-One Cipher

On a first glance at the Three-In-One-Cipher, shown in Figure 4, one can discern three distinguishing features about this multi-cipher:

1. The letters
2. The colors
3. The spacing

Let's look at each of these components individually. First we look at the letters, ignoring the colors and the spacing:

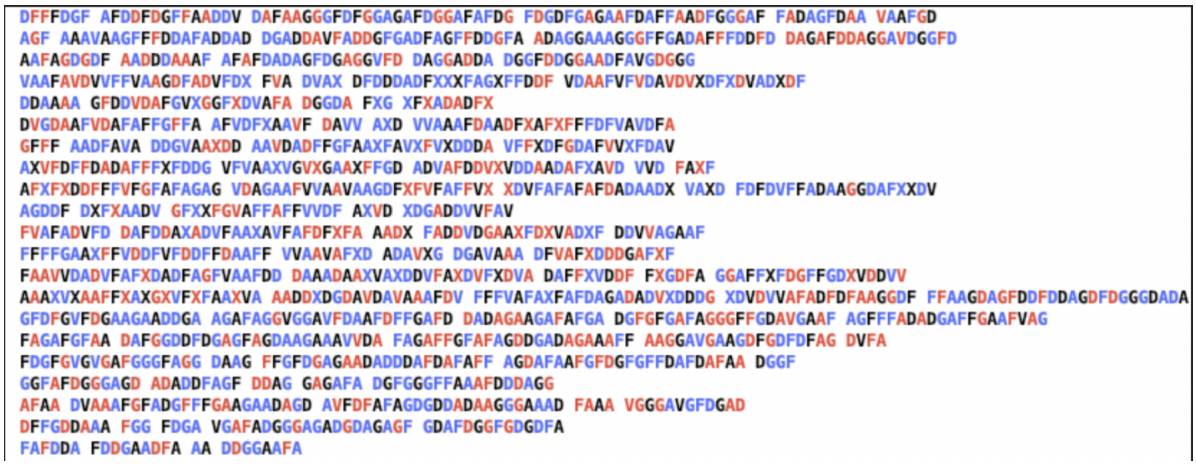

Figure 4: The Three-In-One cipher

DFFFDGFAFDDFDGFFAADDVDAFAAGGGFDFGGAGAFDGGAFAFDGFDGDFGAGAAFDAFFAADFGGGAFF
ADAGFDAAVAAFGDAGFAAAVAAGFFFDDAFADDADDGADDAVFADDGFGADFAGFFDDGFAADAGGAAAGG
GFFGADAFFFDDFDDAGAFDDAGGAVDGGFDAAFAGDGDFAADDDAAAFAFAFDADAGFDGAGGVFDDAGGA
DDADGGFDDGGAADFAVGDGGGVAAFAVDVVFFVAAGDFADVFDXFVADVAXDFDDDADFXXXFAGXFFDDF
VDAAFVFVDAVDVXDFXDVADXDFDDAAAAGFDDVDAFGVXGGFXDVAFADGGDAFXGXFXADADFXDVGDA
AFVDAFAFFGFFAAFVDFXAAVFDAVVAXDVVAAAFDAADFXAFXFFFDFVAVDFAGFFFAADFAVADDGVA
AXDDAAVDADFFGFAAXFAVXFVXDDDAVFFXDFGDAFVVXFDAVAXVFDFFDADAFFFXFDDGVFVAAXVG
VXGAAXFFGDADVAFDDVXVDDAADAFXAVDVVDFAXFAFXFXDDFFFVFGFAFAGAGVDAGAAFVVAAVAA
GDFXFVFAFFVXXDVFAFAFAFDADAADXVAXDFDFDVFFADAAGGDAFXXDVAGDDFDXFXAADVGFXXFG
VAFFAFFVVDFAXVDXDGADDVVFAVFVAFADVFDDAFDDAXADVFAAXAVFAFDFXFAAADXFADDVDGAA
XFDXVADXFDDVVAGAAFFFFFGAAXFFVDDFVFDDFFDAAFFVVAAVAFXDADAVXGDGAVAAADFVAFXD
DDGAFXFFAAVVDADVFAFXDADFAGFVAAFDDDAAADAAXVAXDDVFAXDVFXDVADAFFXVDDFFXGDFA
GGAFFXFDGFFGDXVDDVVAAAXVXAAFFXAXGXVFXFAAXVAAADDXDGDAVDAVAAAFDVFFFVAFAXFA
FDAGADADVXDDDGXDVDVVAFADFDFAAGGDFFFAAGDAGFDDFDDAGDFDGGGDADAGFDFGVFDGAAGA
ADDGAAGAFAGGVGGAVFDAAFDFFGAFDDADAGAAGAFAFGADGFGFGAFAGGGFFGDAVGAAFAGFFFAD
ADGAFFGAAFVAGFAGAFGFAADAFGGDDFDGAGFAGDAAGAAAVVDAFAGAFFGFAFAGDDGADAGAAAFF
AAGGAVGAAGDFGDFDFAGDVFAFDGFGVGVGAFGGGFAGGDAAGFFGFDGAGAADADDDAFDAFAFFAGDA
FAAFGFDGFGFFDAFDAFAADGGFGGFAFDGGGAGDADADDFAGFDDAGGAGAFADGFGGGFFAAAFDDDAG
GAFAADVAAAFGFADGFFFGAAGAADAGDAVFDFAFAGDGDDADAAGGGAAADFAAAVGGGAVGFDGADDFF
GDDAAAFGGFDGAVGAFADGGGAGADGDAGAGFGDAFDGGFGDGDFAFAFDDAFDDGAADFAAADDGGAAFA

Notice that there are only 6 letters that appear in the message – the letters “A”, “D”, “G”, “V”, and “X”, – and with a little bit of background information, we can conclude that this message has been encrypted using an ADFGVX cipher. We have not discussed this cipher yet in this paper, but it is a transposed fractionating cipher. English letters are first substituted for bigrams which consist of the six letters (e.g. A $\rightarrow$ VG, B $\rightarrow$ XG, ... more arbitrary mappings ..., Z $\rightarrow$ AA). Then, they are transposed using a columnar transposition.

We can actually solve this using a common technique that is used for solving a transposed substitution cipher. We determine the transposition first. Try all the possible transpositions, and score them based on the amount of multi-letter repetition that they display in a sliding window. Since English is a language that displays natural repetition, we expect the correct transposition to be the result which shows the highest amount of repetition. For transposed substitution ciphers, we would normally score solutions based on the count of the most frequent bigram (2-gram) or trigram (3-gram). Since this fractionating cipher uses two characters to represent one letter, in this case, we want to score solutions based on the count of the most frequent quadrigram (4-gram) or hexgram (6-gram).

Since we know that columnar transposition is used for ADFGVX ciphers, we can bruteforce through all the columnar permutations from size 2 to 6. The results are shown in Figures 5 and 6. The message which was permuted using the encryption key (3, 2, 4, 6, 5, 1) shows up in the top two solutions using both the 4-gram and 6-gram scoring method, so we can be fairly sure that this is the right permutation.

10 BEST SOLUTIONS:

	Count:	Ngram:	Permutation:	Message:
1	21	DFFD	(4, 5, 3, 1, 2, 6)	XFFDAFFDDFFDDFFFAGXAFFVAVADDDGAGAGVADGDFVAXDAAFDFFFF
2	20	DFFD	(3, 2, 4, 6, 5, 1)	FADFFXDFFDDFGAFFFDAVFFAXGDDDAVAVGAGAAVFDGDDFAADXAFFF
3	18	AAGA	(1, 6, 2, 5, 3, 4)	DFFFXAFDDDFFFFFGDAFFAAXVDDAGVDGAGAAVFDGADVAADDXFFFFA
4	18	AGAA	(2, 6, 4, 1, 3, 5)	XDFFFAFFDDDFDFFFGAXFAFAVVDADGDAGGAAVDFGDAVXADADFFFFF
5	18	DGDD	(3, 1, 4, 5, 6, 2)	AFDFXFFDFDFDAGFFDFVAFFXADGDDVAVAGAAGVAFDDGFDAAXDFAFF
6	18	FDDF	(4, 1, 3, 5, 6, 2)	AFFDXFFDDFFDAGFFDFVAFFXADGDDVAVAAGAGVADFDGFDAAXDFAFF
7	17	GDDD	(4, 2, 3, 6, 5, 1)	FAFDFXDFDFDFGAFFFDAVFFAXGDDDAVAVAGGAAVDFGDDFAADXAFFF
8	17	DFDA	(4, 2, 5, 6, 1, 3)	FAFDFXDFDFDFFAGFFDAVAFFXADGDDVGVAGAAGVAFDDDFDAAXFFAF
9	16	DAAA	(5, 2, 3, 1, 4)	GVGADGDDAFAAFGFFAXGFFFFADXVVVGFFFGFDVAAAGDFAFFAFGDFV
10	16	AAAA	(5, 3, 2, 1, 4)	GGVADGDDAFAFAGFFXAGFFFFADXVVVGFFFGFDAVAAGFDAFFFAGDFV

Figure 5: Bruteforcing through columnar permutations from size 2 to 6, scoring based on the count of the most frequent 4-gram.

10 BEST SOLUTIONS:

	Count:	Ngram:	Permutation:	Message:
1	9	GDDDAV	(3, 2, 4, 6, 5, 1)	FADFFXDFFDDFGAFFFDAVFFAXGDDDAVAVGAGAAVFDGDDFAADXAFFFFFDVDFFDDAI
2	9	DGDDVA	(4, 1, 3, 5, 6, 2)	AFFDXFFDDFFDAGFFDFVAFFXADGDDVAVAAGAGVADFDGFDAAXDFAFFFFVDFDDFADI
3	6	GADADV	(1, 2, 5, 3, 6, 4)	DAXFFFFFFDDDFADGFFFVXAFADDVGDAGVAAAGFVDADGAFXDADFFFAFFDVDDFFDAI
4	6	DAGDDV	(1, 4, 5, 6, 2, 3)	DFFAFXFDDFDFFFGAFDFAAVFXDAGDDVGGAVAAFGAVDDADDFAXFFAFFFDFDVFDDFI
5	6	GDAAVD	(2, 1, 4, 3, 6, 5)	ADXFFFFFFDDDAFDFGFVFXFAADDVDGAVGAAAGVFDDAGFAXADDFFFFAFVDDFDFADI
6	6	DVADGA	(2, 1, 5, 3, 4, 6)	ADXFFFFFFDDDAFDFFGVFXAFADDVADGVGAGAAVFDGDAFAXDADFFFFFAVDDFFDADI
7	6	DDAGDV	(2, 1, 5, 6, 3, 4)	ADFFFXFFDDDFAFFGFDVFAAFXDDAGDVVGGAAAVFGADDFADDAXFFFAFFVDFDFDADI
8	6	DGDDAA	(3, 1, 4, 5, 6, 2)	AFDFXFFDFDFDAGFFDFVAFFXADGDDVAVAGAAGVAFDDGFDAAXDFAFFFFVDDFDFADI
9	6	DVDGAD	(3, 1, 6, 2, 5, 4)	AXDFFFFFFDDDADFGFFVXFAAFDVDGADVAGAGAVDFAGDFXADDAFFFAFFVDDDFFADI
10	6	GAVADD	(3, 4, 2, 1, 6, 5)	XFDAFFFDFFDDDFFAGFXFFVAAVDDDGAAAGVAGDDFVAGXAAFDDFFFFAFDFDVDFDFI

Figure 6: Bruteforcing through columnar permutations from size 2 to 6, scoring based on the count of the most frequent 6-gram.

Now, we just have to solve a bigram fractionating cipher. Since we already walked through the process of solving a fractionating cipher using the arbitrary mapping method while solving the RGB cipher in 3.1, we spare the reader the details, and leave it to them to recover the plaintext if they wish to do so.

Next, let's consider the colors. Again, we'll represent the colored positions in the message as letters, with "R" for red, "B" for blue, and "D" for black:

BRRDBBBBDBRDRDBRDRBBDBRDRRDRBDBDRBBDBDBRRBDBBDRBRBDBBRDBRBDBDBRDRBDBRBDB
RDBBBDRBRBDBRDBRBBDBBDBBBDRDBBBBDBBDBBBDRRDRBRRDBBBBDBDBRDBRBDRDBBDBBBDB
RBDBDBRDRBBDRBRRDRDRRRDRBRBDBRBDBRDRBRBDRBRDBRRDBBDRDBBBBDBBDRRDBRRBDBRD
RDBBDBDRBDRBRBDBDBRRDBBBBDRRRDBBBDBRDRBRRDBBBDRDBBBBDBBDBBBDBBDBBBDBBDRR
DBRRBDBRBDRRRDBBBRDBBDRBBDBDRBDRDBRRRDBDBRDBRBBDRRRDBBRDBBBDRBRRDRRDRBRR
DBRRDBBDBBRBDBDBBBBDBRDRDBBBBDBBBDBDRBDRDRDRRRDBBBBDBBDRRDRDBBBBDBDBBBBD
RRRDBBRDBRBDBBDBBBDBBRBDBBDRBBRDBDRBBDRDBBBBDBDRRDBRDRDRBRBDBBBBDBBDBBBD
RRDBRDRBBDBDBRRDRRRDBBRDBRBBDRBBDBRDRBDRBRRDRRDBRDRBDBBBBDBRDBBBRDBDRDBB
BBDRRRDBBRDRRBDBBBBDRDRDBBBBDBBDBBBDBBBDBDBDRDBBBBDBDBBBBDBDBRBBDBRBBDRR
RDBBRBDBBBBDBRDBBBRDBBDRBDRRBDBRDBBRBDBRDBRBBDBBBDBDBRRDRRRDRRDBRDRBDRRD
RBRRDRBBBDBDRBBDBBBDBBBBDBRDBRBBDBRBBDRBBBDBDBRDRBBBDBBRDBBBDBDRBBDRRDRB
RRDRBRBDRRRDBBRBDBBRBDBDBRBDBBBDBRBDBRDRBDBBBDBRDRBRBDRBRDBDRBBDRRDRBRRD
BRBDBDBRRBDBBRDRDBRDRDBBDRRRDRBDRRBDRBDBRDBRRDRBDBRDRDBRDRBDRBBDBBDBBBDB
BBBDBRDBRBBDBRBBDRBDRRRDRDRRRDRBDBRBBDRBRRDBRBDBDRBRBDBDBBDBBBRDBDRDBBBB
DBBDBBBDBBBRDBBDBRBBDBRBBDBRDRBDRRRDBBRDBBBDBRRDBRBDRRRDRBDRRBDRBBBDBBRD
RDBBBDRDBRDRBDRBBDBBRDRBDRBBDBDBRBDRDBBBBDBDBRDRBBDRRRDBRRBDRDBBDRRRDRBD
RRRDBBRBDBRDRBBBDRRRDRRDBBDRBDBRDRBBBDBRBBDBDRDBDBRBDRRDBBBDBRDRBDRBBDRB
BBDRBRRDBBBBDBBDRRDRDBBBBDBRDRDRBBDRRRDBDBBBDRRDBDRDBBBBDBBDBBBDBRRDBRBD
RRRDRBDRRBDRDBDBRBDRRDBBBDRBDBRDRRDBDBBBDBRDRRDBRDBBDRRDRRRDRBDBBBDRRRDB
BRDRBDRBBDBBBDBRRDBDBRBBDBRBBDBRBBDBBRDRBRBDBBDBBRBDBDBRBDBRRDBDBRBBDBRB

The important thing to notice about this ciphertext is that the letter “D” never appears next to itself. In fact, it almost seems to act as a separator character, because there appear to be many frequently repeating sequences sandwiched between consecutive D’s. To verify this, let’s replace the letter “D” with the space character. This can be achieved easily using a Python console as shown in the code snippet in Figure 7.

Figure 7: Demonstration of the incredibly helpful replace() function in a Python 3 console.

Take a look at the ciphertext without spaces, and it appears to be a variable length substitution cipher. Sequences like “BRR”, “BBBB”, “BB”, and “R” are all repeating frequently throughout the message, but they are all sequences of different lengths, and the separator is necessary to indicate where one sequence ends and another begins. A variable length substitution can be solved by converting the distinct variable length sequences to letters in the A-Z alphabet, and solving the result as a monoalphabetic substitution cipher.

The algorithm for this concept is not incredibly complicated, but since we do not include a tool in our implementation which allows for this conversion, we provide a Python code snippet below.

```
c = 'BRR BBBB BR R BR RBB BR RR RB B RBB B BRRB BB RBRB BBR BRB B BR RB BRB BR BBB RBRB BR BRBB

distinct_sequences = []

for sequence in c.split():

    if sequence not in distinct_sequences:
        distinct_sequences.append(sequence)


alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

res = [alphabet[distinct_sequences.index(sequence)] for sequence in c.split()]

print(''.join(res))
```

Figure 8: Code snippet to print out the components of a variable length substitution cipher "c" separated by spaces as a monoalphabetic substitution cipher.

For this ciphertext in particular, notice that the variable length substitutions are binary in the sense that there are only two distinct characters appearing anywhere in the substitutions. This might remind the reader of another well-known variable length substitution cipher: Morse code. Indeed, for this message, we can simply replace all occurences of "B" with "." (dot) and "R" with "-" (dash), and we will obtain an intermediate Morse ciphertext. The replacement can easily be done using the Python replace() function as illustrated in Figure 7. Using an online Morse code decoder, the Morse code can then be converted to plaintext.

Finally, consider the spacing of the Three-In-One cipher. The multi-cipher is separated into blocks for a good reason. The lengths of these blocks actually indicate the components of the third ciphertext. If we write out the lengths, we get:

7, 14, 26, 24, 9, 6, 3, 19, 26, 24, 17, 9, 9, 18, 8, 19, 23, 3, 4, 20, 24, 6, 20, 5, 3, 9, 18, 10, 4, 3, 26, 4, 7, 9, 24, 17, 19, 18, 21, 3, 4, 20, 26, 17, 4, 20, 5, 8, 17, 4, 11, 9, 24, 4, 18, 9, 25, 9, 6, 7, 14, 26, 24, 9, 6, 19, 24, 19, 24, 19, 26, 18, 24, 14, 22, 20, 9, 26, 24, 19, 4, 18, 4, 23, 24, 4, 12, 9, 4, 6, 18, 4, 24, 24, 4, 12, 9, 3, 4, 20, 14, 6, 9, 2, 8

To convert these numerical values to letters, we can use the A-Z $\longleftrightarrow$ 1-26 mapping (Table 5), since the lengths of the blocks range from 2 to 26. Converting gives:

GNZXIFCSZXQIIRHSWCDTXFTECIRJDCZDGIXQSRUCDTZQDTEHQDKIX
DRIYIFGNZXIFSXSXSZRXNVTIZXSDRDWXDLIDFRDXXDLICDTNFIBH

Solving this as a general simple substitution cipher yields the final plaintext:

MASTERY IS THE END IF YOU TRULY ENJOY SOMETHING YOU SHOULD HOPE
TO NEVER MASTER IT IT ISNT A QUESTION OF TO BE OR NOT TO BE YOU ARE XD

Hence, this single cipher conceals three plaintexts. The length of the ADFGVX plaintext is 720, the length of the variable length substitution cipher is 403 characters, and the length of the simple substitution cipher from the lengths of the blocks of letters in the message is 133 characters. The overall multi-ciphertext length is 1440 letters for the ADFGVX ciphertext plus 104 spaces to separate the message into 105 letter blocks. The efficiency score, therefore, is:

$$\mathcal{ES} = \frac{720 + 403 + 105}{1440 + 104} \approx 0.80 < 1$$

So the Three-In-One multi-cipher is inefficient based on the definition from 2.2. This makes sense because the encryption algorithm for the ADFGVX cipher includes a fractionation step. Single letters in the plaintext are replaced by pairs of letters during the encryption process, and therefore this classical cipher alone possesses an efficiency score equal to 0.5, because the ADFGVX ciphertext always contain exactly twice as

many characters as the plaintext. A better way to consider this multi-cipher rather than using the terms "efficient" versus "inefficient" is to notice that the efficiency score increased from 0.5 for this fractionating cipher to 0.8 using the multi-cipher. Therefore, we were able to conceal 60% more plaintext per ciphertext character compared to an ordinary classical cipher which uses bigram fractionation.

Notice that the third cipher, which uses the lengths of the blocks, does not significantly impact the efficiency score. For each block corresponding to a single letter in the message, a single space character has to be appended in the ciphertext, except for the very last block. We will see that the Four-In-One cipher in the next section uses block lengths less efficiently.

In summary, this is a third degree multi-cipher which can be thought of as a transposed fractionating ADFGVX cipher using the text alone, a variable length substitution cipher using the colors, or a simple substitution cipher using the lengths of the blocks of letters (which were indicated by the spaces).

3.3 Four-In-One Cipher

This multi-cipher does not use colors, unlike the multi-ciphers in 3.1 and 3.2. For this reason, we point out that even though the cipher which we present in this section only conceals four messages, it could potentially be modified to include a fifth plaintext. The cipher is shown below in Figure 9. Taking up an entire page, the reader can verify that the total length of the multi-ciphertext, including spaces, is 3306 characters.

For this multi-cipher, the four intermediate ciphertexts for the four plaintexts can be recovered using simple commands in a Python console, so we will provide various code snippets. First, though, let's consider the three distinguishing features about this multi-cipher:

1. The capitalization
2. The spacing
3. The arrangement of the capitalized letters

Let's look at these features in order. First, notice that this message consists exclusively of capital and lowercase letters in the A-Z alphabet. Suppose that we looked only at the capital letters appearing in the message. Using a Python console, we can extract the capital letters from the ciphertext using a single-line command. Assume that we defined:

```
>>> ciphertext = 'I Wh Ptr SZSES BEYe C gYPVA u Nobn VdUnF et ea u QWN WNBS hePs kpTE eJ s s u FJrKR YX fM KLJs omOFH XI IInin XQ Ek K iNV dX gE t ue SLaN LoJp VnK LJOuF awHX A nZrR dLrBF ol maSQ
eHr QcGKL JG hStl e Y WVKLJ Ep n a FVQa tLsK SoFr dAYuI XQKnE QTsJX F u ekZKW Beu QKp LrsFb A aG EWS eKiLt Jh VRJQ gI nD ISM OXYe duPZ Ro tLGF YPeth V eF raA YPlVe Ie SB ZVsJ ooEF VQfL yNMQ T SxI
e NVXs ANBYa XM yM a S b rpV Nuod tcOYX MM RpXof TN ZQoRQ h s tEon ISye t h iBU shsIt s F BYeW TlUVX oNi nieU VtWY P hVrg nKoT uLnJ dN tZ AKPNS Vu rr X TAu tZ eX seGh pIsTO WM eCECa Il sn EH BIKt
MB tYIB Wb IP IaJL VSEY Medk QMKsL w JoG u R VJPR ZN noVd rd XJ eZ ao b VR eXFSR dV FW B its IS QX LYJ sh eL XB oP tP iHnN et LEnch X Yeva PVdW e s eYe ZnF p ue drRrg LE uZ oR I A aGY Ynder cW L
i VhJ hn NZ V p AsMM EoMYE Ro phWXB Z Pu rns QIJTa mF ZBde MXQ iKTnT A JM aGrT tVlXM I KsX eMMN M Sh Lnsa JX WM EKY ydrPV RsJu a fLKLo JceA tGEfW SEsSh el TiNt Te SZ nZe JIsSW o KYSab T t t VWs X
FVe J lHi eNLJp Sg R M Kn LNel V iXusX iTIn E Kd Jt Z sY bI UPRu IrXde BL ethoW Mr Ms oNS ZTtU L FD fVOo JXeK XtMM WMWMh WVTe U WP rFo dwdUZ oRL W u Lsbr XrTdN oY wE S LYe eIUn PpRuI XB LeWMI Jd
uWXo Xsn XdaF XtKW I nLy ZU RR iX JVaG F Cg JIh MM TSS A Zn c HJZ rJ XM M D AJo teI BLd JTG PtJoA ihJ e hr TeNB omiZJ Zt TtTll V JI KeY WMhJP nwiN sSdtV oX fMMCE mR iw MXSI esPV SSKV AF e A G eIW
e R T X aHaw VYtX y PlRHi lh tX WD I Y reg uPZRo eL B F n H dpT EZxd l osXea M cBY rkMcn sX a PsdVi p GTls tCCH dMaIn MIM h tC IoPe wJLRP lN OYrsX dSae Q woEkK G WCnX YPiRw stXX i olX tRstV PM UE
IfGC Kh VeroJ XF VY sAcJ E DBeY fotEn aY rVFPN OY uX h SeQ t EKBIr J e Wse UWiKX JtL slZRY iPVZ N lKiZ tRfNm BPSK BYIe roKQI f I IID ELlMi te R ao VlJIF Vt Rs Ii PF aJs fQKA F A YIt hiQQF gst W M
fPous iK LFrDV vJQW N RaHvT g FWlJ aBFia XMM tVnsh E WB tYnM eSiPZ W XvC EeGM I IM aTiB Kt lb hSX KeFVH PtKL rJeNs ei FtVJA sK li l GiM IE tG tmegJ AoR eViv i laJBY E Yfe dnt QK A F AYIt ok QQ Fa
MTcE Z ga nAV n ih SgSMM tMwH e B lZQT NKoro d sL JNZV J Ene oRtaW nKWLR Iea E Zt XMmA BtMSK Ri RHMIH FieG J pEfKa B VbVG eMZRe Lblo Z tVeWs HsCV idoHG nGtht We MeWM sW iM aLo PKIWs AnZ hXNAG tSg
FQ nM ht Kfen LFrB oteK Ls J Cd s oeM m V FLil ZEYeQ s Fh RLbQ d aMI SWWJG yrGoz hrs Y IQ QFra lWXpJ bd IaMaQ TZEe S sKdnV X rbuW QeXKn J ydbaZ r g V HvY P aVEn iigG V Ffe VRrIr Cs G a JT CKaW e
oMn i Bt hY g nI KQ IIB ItJbu fHJi I laUf PD wQ KFopZ ZeEr I ulec IWkMf IdWm B rNSR osQa CPNWk EunEH ZVs WidTN CESL ZsRSW eI XMeMI RH mo IadC iMKW I wR NIlR RsIZ V p caW ZiTed aK n IvIlQ oEr I LZ
WLWs Fh QdInU hZIC M YGZst t RYPa VsFFi sGRmV Joh e I WStn iigWE Vg Ii P vlA ngI fYRWI iWXI r Pef mXM o ad eI htt wa KX M M Wreg MWMiN n E Xht lQE eXp JnV aKMT VhRtP tt QGaog JX rF swt ZKi n II a
i es PEtRI EiA MW Ts EQ aT VHr MaQcD WfJ T WieX M MUIic fRoQi X Ss Ipir AFtt RT ama g NK EiW BnR VfZ AoL Re Kt Wh I XiwZn Vld R tX aJTe uriwP Li tJMhE uHn Jtsi LL sT G i vlo tCIRN V XL emh NUJes
peKw SIMR uXMA r Lfml TUae thWp JiyII T P sXaMQ JQFcV o Y EIBlf YIU c W NIWBr FVFne Qo hJI t aeI MDSf e SAc CtEo AMfe JhrXT LF LeNaA SM Ih L tetZh eR kLJLX o B ywPV drrf oRM MLWJ L h tcK LJisp UK
rhe IerR Y tlaP V set Jt oTZV WYBih YIeL s WmUW ssYCI RJ eLagY MX la XFn AsJ o to hZ JeFF tK BSpSe mert PZQMW NuZiJ aZtP noWF L'
```

Then the capital letters can be extracted using the following command:

```
capital_letters = ''.join(list(map(lambda x: x if x == x.upper() else '', ''.join(ciphertext.split()))))
```

A frequency analysis of the capital letters in Figure 10 reveals that the most common letters in the ciphertext are I, M, and X, appearing 7.4%, 6.7%, and 5.92% of the time, respectively. These flattened frequencies may possibly indicate a polyalphabetic cipher. To be sure, we calculate the index of coincidence for this ciphertext and find that it is equal to 0.046. This is much lower than the IOC of English at 0.0667, providing further evidence that this message was encrypted with a polyalphabetic cipher. The final piece of

I Wh Ptr SZSES BEYe C gYPVA u Nobn VdUnF et ea u QWN WNBS hePs kpTE eJ s s u FJrKR YX fM KLJs omOFH XI IInin XQ Ek K iNV dX gE t ue SLaN LoJp VnK LJOuF awHX A nZrR dLrBF ol maSQo W fL eHr QcGKL JG hStl e Y WVKLJ Ep n a FVQa tLsK SoFr dAYuI XQKnE QTsJX F u ekZKW Beu QKp LrsFb A aG EWS eKiLt Jh VRJQ gI nD ISM OXYe duPZ Ro tLGF YPeth V eF raA YPlVe Ie SB ZVsJ ooEF VQfL yNMQ T SxIn LTFAg e NVXs ANBYa XM yM a S b rpV Nuod tcOYX MM RpXof TN ZQoRQ h s tEon ISye t h iBU shsIt s F BYeW TlUVX oNi nieU VtWY P hVrg nKoT uLnJ dN tZ AKPNS Vu rr X TAu tZ eX seGh pIsTO WM eCECa Il sn EH BIKt u G brMB tYIB Wb IP IaJL VSEY Medk QMKsL w JoG u R VJPR ZN noVd rd XJ eZ ao b VR eXFSR dV FW B its IS QX LYJ sh eL XB oP tP iHnN et LEnch X Yeva PVdW e s eYe ZnF p ue drRrg LE uZ oR I A aGY Ynder cW L X YPri i VhJ hn NZ V p AsMM EoMYE Ro phWXB Z Pu rns QIJTa mF ZBde MXQ iKTnT A JM aGrT tVlXM I KsX eMMN M Sh Lnsa JX WM EKY ydrPV RsJu a fLKLo JceA tGEfW SEsSh el TiNt Te SZ nZe JIsSW o KYSab T t t VWs X oPLQ mFVe J lHi eNLJp Sg R M Kn LNel V iXusX iTIn E Kd Jt Z sY bI UPRu IrXde BL ethoW Mr Ms oNS ZTtU L FD fVOo JXeK XtMM WMWMh WVTe U WP rFo dwdUZ oRL W u Lsbr XrTdN oY wE S LYe eIUn PpRuI XB LeWMI Jd rE XWrg uWXo Xsn XdaF XtKW I nLy ZU RR iX JVaG F Cg JIh MM TSS A Zn c HJZ rJ XM M D AJo teI BLd JTG PtJoA ihJ e hr TeNB omiZJ Zt TtTll V JI KeY WMhJP nwiN sSdtV oX fMMCE mR iw MXSI esPV SSKV AF e A G eIWhP pt me R T X aHaw VYtX y PlRHi lh tX WD I Y reg uPZRo eL B F n H dpT EZxd l osXea M cBY rkMcn sX a PsdVi p GTls tCCH dMaIn MIM h tC IoPe wJLRP lN OYrsX dSae Q woEkK G WCnX YPiRw stXX i olX tRstV PM UE toKQ IIfGC Kh VeroJ XF VY sAcJ E DBeY fotEn aY rVFPN OY uX h SeQ t EKBIr J e Wse UWiKX JtL slZRY iPVZ N lKiZ tRfNm BPSK BYIe roKQI f I IID ElIMi te R ao VlJIF Vt Rs Ii PF aJs fQKA F A YIt hiQQF gst W MM rDML fPous iK LFrDV vJQW N RaHvT g FWlJ aBFia XMM tVnsh E WB tYnM eSiPZ W XvC EeGM I IM aTiB Kt lb hSX KeFVH PtKL rJeNs ei FtVJA sK li l GiM IE tG tmegJ AoR eViv i laJBY E Yfe dnt QK A F AYIt ok QQ Fat WGeih MTcE Z ga nAV n ih SgSMM tMwH e B lZQT NKoro d sL JNZV J Ene oRtaW nKWLR Iea E Zt XMmA BtMSK Ri RHMIH FieG J pEfKa B VbVG eMZRe Lblo Z tVeWs HsCV idoHG nGtht We MeWM sW iM aLo PKIWs AnZ hXNAG tSgX A Z iFQ nM ht Kfen LFrB oteK Ls J Cd s oeM m V FLil ZEYeQ s Fh RLbQ d aMI SWWJG yrGoz hrs Y IQ QFra lWXpJ bd IaMaQ TZEe S sKdnV X rbuW QeXKn J ydbaZ r g V HvY P aVEn iigG V Ffe VRrIr Cs G a JT CKaW e WolNs XoMn i Bt hY g nI KQ IIB ItJbu fHJi I laUf PD wQ KFopZ ZeEr I ulec IWkMf IdWm B rNSR osQa CPNWk EunEH ZVs WidTN CESL ZsRSW eI XMeMI RH mo IadC iMKW I wR NIlR RsIZ V p caW ZiTed aK n IvIlQ oEr I LZ eVKa NWLWs Fh QdInU hZIC M YGZst t RYPa VsFFi sGRmV Joh e I WStn iigWE Vg Ii P vlA ngI fYRWI iWXI r Pef mXM o ad eI htt wa KX M M Wreg MWMiN n E Xht lQE eXp JnV aKMT VhRtP tt QGaog JX rF swt ZKi n II ao eJt ri es PEtRI EiA MW Ts EQ aT VHr MaQcD WfJ T WieX M MUIic fRoQi X Ss Ipir AFtt RT ama g NK EiW BnR VfZ AoL Re Kt Wh I XiwZn Vld R tX aJTe uriwP Li tJMhE uHn Jtsi LL sT G i vlo tCIRN V XL emh NUJes Io QotN peKw SIMR uXMA r Lfml TUae thWp JiyII T P sXaMQ JQFcV o Y EIBlf YIU c W NIWBr FVFne Qo hJI t aeI MDSf e SAc CtEo AMfe JhrXT LF LeNaA SM Ih L tetZh eR kLJLX o B ywPV drrf oRM MLWJ L h tcK LJisp UK S iGC rhe IerR Y tlaP V set Jt oTZV WYBih YIeL s WmUW ssYCI RJ eLagY MX la XFn AsJ o to hZ JeFF tK BSpSe mert PZQMW NuZiJ aZtP noWF L

Figure 9: The Four-In-One cipher

IWPSZSESBEYCYPVANVUFQWNWNBSPTEJFJKRYXMKLJOFHXIIIXQEKNVXESLNLJVKLJOFHXAZR
LBFSQWLHQGKLJGSYWVKLJEFVQLKSFAYIXQKEQTJXFZKWBQKLFAGEWSKLJVRJQIDISMOXYPZR
LGFYPVFAYPVISBZVJEFVQLNMQTSILTFANVXANBYXMMSVNOYXMMRXTNZQRQEISBUIFBYWTUVX
NUVWYPVKTLJNZAKPNSVXTAZXGITOWMCECIEHBIKGMBYIBWIPIJLVSEYMQMKLJGRVJPRZNVXJ
ZVRXFSRVFWBISQXLYJLXBPPHNLEXYPVWYZFRLEZRIAGYYWLXYPVJNZVAMMEMYERWXBZPQIJT
FZBMXQKTTAJMGTVXMIKXMMNMSLJXWMEKYPVRJLKLJAGEWSESTNTSZZJISWKYSTVWXPLQFVJH
NLJSRMKLNVXXTIEKJZYIUPRIXBLWMMNSZTULFDVOJXKXMMWMWMWVTUWPFUZRLWLXTNYESLYI
UPRIXBLWMIJEXWWXXXFXKWILZURRXJVGFCJIMMTSSAZHJZJXMMDAJIBLJTGPJAJTNBZJZTTV
JIKYWMJPNSVXMMCERMXSIPVSSKVAFAGIWPRTXHVYXPRHXWDIYPZRLBFHTEZXMBYMXPVGTCCH
MIMIMCIPJLRPNOYXSQEKGWCXYPRXXXRVPMUEKQIIGCKVJXFVYAJEDBYEYVFPNOYXSQEKBIJW
UWKXJLZRYPVZNKZRNBPSKBYIKQIIIIDELMRVJIFVRIPFJQKAFAYIQQFWMMDMLPKLFDVJQWNR
HTFWJBFXMMVEWBYMSPZWXCEGMIIMTBKSXKFVHPKLJNFVJAKGMIEGJARVJBYEYQKAFAYIQQFW
GMTEZAVSSMMMHBZQTNKLJNZVJERWKWLRIEZXMABMSKRRHMIHFGJEKBVVGMZRLZVWHCVHGGWM
WMWMLPKIWAZXNAGSXAZFQMKLFBKLJCMVFLZEYQFRLQMISWWJGGYIQQFWXJIMQTZESKVXWQXK
JZVHYPVEGVFVRICGJTCKWWNXMBYIKQIIBIJHJIUPDQKFZZEIIWMIWBNSRQCPNWEEHZVWTNCE
SLZRSWIXMMIRHICMKWIRNIRRIZVWZTKIIQEILZVKNWLWFQIUZICMYGZRYPVFFGRVJIWSWEVI
PAIYRWIWXIPXMIKXMMWMWMNEXQEXJVKMTVRPQGJXFZKIIJPERIEAMWTEQTVHMQDWJTWXMMUI
RQXSIAFRTNKEWBRVZALRKWIXZVRXJTPLJMEHJLLTGCIRNVXLNUJIQNKSIMRXMALTUWJIITPX
MQJQFVYEIBYIUWNIWBFVFQJIIMDSSACEAMJXTLFLNASMILZRLJLXBPVRMMLWJLKLJUKSGCIR
YPVJTZVWYBYILWUWYCIRJLYMXXFAJZJFFKBSSPZQMWNZJZPWFL

Figure 10: The capital letters extracted from the Four-In-One cipher

evidence that is necessary to confirm that this was, indeed, encrypted using a polyalphabetic cipher versus a polygraphic substitution cipher is the output from an IOC-based probable key lengths calculator. We expect the average IOC corresponding to the "best" period to be fairly close to the IOC of English, and indeed, we find that this is the case, as shown in Figure 11.

Based on the output, we guess that the number of alphabets used in this polyalphabetic cipher is equal to 4. If it was instead a multiple of 4 like 8 or 12, then the average IOC for $L = 4$ would not be close to the average IOC for English. In general, the average IOC from a probable key lengths output is close to the IOC of English if and only if L is equal to or a multiple of the actual key length K_L that was used. This is due to the periodic nature of polyalphabetic ciphers.

We assume that this Vigenere cipher, and use an elementary statistical attack which relies on χ^2 testing while cycling through the possible shifts to solve the cipher with $K_L = 4$. This solves the cipher, and reveals that the Vigenere encryption keyword was "FIRE".

Next, look at the lowercase letters. They can be extracted from the Four-In-One-Cipher using the following command in a Python console:

```
lowercase_letters = ''.join(list(map(lambda x: x if x == x.lower() else '', ''.join(ciphertext.split()))))
```

The lowercase letters are shown in Figure 12. Analyzing the frequencies of these letters reveals that the most common letters are "E", "T", and "S", appearing with frequencies 12.34%, 10.08%, and 8.1%, respectively. These closely match standard frequencies in the English language from 4, suggesting that this is a transposition cipher. A chi-squared (χ^2) analysis can be used to confirm that this was, indeed, encrypted using a transposition cipher. Since we already walked through the process for decrypting a transposition cipher during the cryptanalysis of the RGB cipher 3.1, we simply reveal the solution. The lowercase letters were encrypted using a simple horizontal transposition cipher with the permutation (2, 1, 5, 3, 4), which also corresponds to the keyword "EARTH". The transposition is irregular because the size of the permutation is not a factor of the length of the ciphertext.

Let us now consider the spacing in the message. Blocks of letters range from one to five letters long. We

```
L = 12  Average IOC ~= 0.06406 ± 0.003
L = 20  Average IOC ~= 0.06367 ± 0.004
L = 04  Average IOC ~= 0.06339 ± 0.005
L = 16  Average IOC ~= 0.06336 ± 0.004
L = 24  Average IOC ~= 0.06313 ± 0.004
L = 08  Average IOC ~= 0.06293 ± 0.002
L = 26  Average IOC ~= 0.05364 ± 0.004
L = 06  Average IOC ~= 0.05285 ± 0.004
L = 02  Average IOC ~= 0.05273 ± 0.012
L = 14  Average IOC ~= 0.05266 ± 0.004
L = 10  Average IOC ~= 0.05241 ± 0.002
L = 18  Average IOC ~= 0.05229 ± 0.004
L = 22  Average IOC ~= 0.05196 ± 0.002
L = 19  Average IOC ~= 0.04652 ± 0.002
L = 25  Average IOC ~= 0.04636 ± 0.003
L = 23  Average IOC ~= 0.04600 ± 0.003
L = 13  Average IOC ~= 0.04595 ± 0.003
L = 09  Average IOC ~= 0.04587 ± 0.003
L = 03  Average IOC ~= 0.04584 ± 0.007
L = 05  Average IOC ~= 0.04584 ± 0.003
L = 07  Average IOC ~= 0.04541 ± 0.002
L = 11  Average IOC ~= 0.04492 ± 0.002
L = 15  Average IOC ~= 0.04456 ± 0.003
L = 17  Average IOC ~= 0.04412 ± 0.002
L = 21  Average IOC ~= 0.04383 ± 0.003
```

Figure 11: Output from the program probableKeyLengths.py (from the implementation) on the capital letters cipher shown in Figure 10.

can extract the lengths of the blocks and store them in a list using the following command:

```
lengths_list = [len(word) for word in ciphertext.split()]
```

We obtain the list [1, 2, 3, 5, ..., 4, 4, 1]. There are five possible numbers that can appear at any individual position in this list. But if we splice the numbers together, and look at the message in 2-letter blocked groupings, then there are $5 \cdot 5 = 25$ possible bigrams that could appear. Recall that there are 26 letters in the standard English A-Z alphabet. This might be a fractionating cipher in which individual letters have been replaced with bigrams. More specifically, look at Figure 13, which shows the numbers spliced together. This may have been encrypted using a Polybius square, a specific fractionating cipher which converts letters to coordinate pairs consisting of the digits 1 to 5.

Consider a frequency analysis of the pairs of digits in Figure 13 in a blocks window. The three most frequent bigrams are “22”, “45”, and “41” which appear with frequencies 11.78%, 10.62%, and 9.47%, respectively. These match up closely with the frequencies of “E”, “T”, and “A” in the English language, suggesting that it is, indeed, a bigram fractionating cipher.

We can arbitrarily map bigrams to letters in the A-Z alphabet, and solve the result as a monoalphabetic

htreguobndneteauheskpessurfsomninkidgtueaopnuawnrdrolmaoferchtlepnaatsordunsuekeuprsbaeithgneduoteth
eraleesoofyxngesayabrpuodtcpofohstonyethishstseloiniethrgnoundturrutesehpsealsntubrtbaedkswounodrdea
obeditssheotinetnchevadeseenpuedrrguoandercriihhnpsoophurnsamdeinartlsehnsaydrsuafocetfshelitenesoab
ttsomeliepgneliusindtsburdeethorsotfoetherodwdousbrrdoweenpuedrrguosndatnyiaghncrotedtoihehreomittll
ehnwisdtofmiweseehptmeaawtylilhtreguoendpxdloseacrkcnsasdiplstdanhtoewlrsdaewokniwstioltsttofherosce
fotnaruhetreseitslilitfmerofliteaoltsiasfthigstrfousirvavglaiatnshtneiveaitlbhetreseitslilittmegoeiv
ilafednttokateihcgannihgtwelorodsneotaneatmtiiepfabeeblotessidonthteesiaosnhtginhtfenrotesdsoemilesh
bdayrozhrsralpbdaaesdnrbuenydbargvaniigferrsaaeolsonithgntbufilafwoperuleckfdmrosakunsidseemoadiwlsp
caiedanvloreashdnhsttasismohetniiggivlngfirefmoadehttwareginhtlepnahtttaogrswtinaoetriestisaracfieic
foispirttamaginfoethiwnldtaeuriwithuntsisivlotemhesootpewurfmlaethpiysacolfcrneohtaefectofehreahteth
ekoywdrrfohtcispirheertlasettoihesmsseaglansotohetpemertuiatno

Figure 12: The lowercase letters extracted from the Four-In-One cipher

123541514522134442111522452522132212443541452512352411521144455511533512352422344245123522444414514522111345252511441135114534414442252213224525222411442244451311424222212522132232222242514411331252222113521141322114525123524351245134124223541545524223515111314413521124154122122452522341244454123531145221345252244344132224123231213221133335312452512354525224442111522111441522211352111134151352151445312452541514541352245252241452522131511353541454451135212522223411353124525415145351245134124223545252213225341513321142235412312541145124135531245254151454154552422353541132244421213114512413553124525415145415455242235354113224442121311451241355312452541514541545524223535411322444212131145124135531245254151451511131441352112415412212235414225414541445535452522441244113521531245254151455311452213354131333345252233122322532222353141551244231233332221531245253422113512352444414445115511531152315134134452522152511443411352133414132454145252233122425455441

0010110000000001010000101110101011111000000011011100101110010000100001110000000111000101001010101110010
0101010000101100010101010011110010101010100000101110000000111000110100101100100001000100011100001100
1011010100001010101000010100000001110001100000111010110001010100001011000010100000101000011000100001
0010101110011111000000101100001001110110011111001110110001001000101111001000101110101010101000000011
1000110101101101000010001011100000110110010000100010000000111000101010100000001101110010111001000010
0001110000000111000101010101100111001110010111010101111101100101000100011111000001110101100010100010
0001110000011110000110001100010010000101010100001010000010111000000001110001011100010110100100010110
1010100101001001000110110010100010010101100010100010011010110100100101010100001010110011110010110000
1000010010010010000001000100010111100100010111010101010000110010101000010000110001110010110110010001
0100001000100010010000000110001000000001110001000010101101110100111000101011000010001001110101101010
0010110000110000000010010010111100010110010101001111000001111000110001011000111110110100110111010110
1100111000101010001100101100001000110101101101000010001011100111010110000011000010001011100000101000
0101110110100000010101010000101011001000101100010000101010101000000001110001000000101110110100001010
0010110000000111000111000100010111100010010000010101001010011000101110001010101000010010000010100111
1000100000100101011101000101110100010111100101011111000001111100000000111000110011100100111001110100
0101010000001111100000000111011010000011001001001000010000000110010101001001000101110101010100111001011
1010100101010100001010100001010001001011011100101110010011110100011000101010010110000000110111110000
0111001011010100001010110011100100101111011001001001111000110010101010000101011010101101101100000001
0111001011010010001100101011110010101010100011010000101100001011000000001000100010000110110100001000
0010001110010111010101010100010010000101010101000000001110001010011001001110001111100010101101101000
1000101110011110111111000001110001010011100101010100001010110011100101110011001110111110010001000010
1000101010010001100000111010100101110011001111000100100100100101010011010110101001111100110010101011
1001001111100000000111000110101101101000010001011100111101011000010100000101000011000100000100011011
0111000011001010100110110000010100001011101101000010110011111000000001110001110001001110110011100111
1100000001100101010011000000101100011001010111100100100010111110000000101010101011000

Figure 14: The binary ciphertext obtained from replacing capital letters with “0” and lowercase letters with “1”.

The methods are not very different, but we will use the second method, because it will allows us to recover the keyword that was used to encrypt this specific plaintext. We obtain the following intermediate ciphertext:

FQBILVPQDOLSCDQDRJLSKQWFKPFKQLQDRUWIIRYMWUFKCQDRQOWFILBIFBRBFOPQFQTWPTWE
RPELVKQDRVWQROBWIIQDRKQDOLSCDQDRKWOOLVPWKEPQOWFCDQPBRREFKCQLQDROFUROVFER
WKEVRITLJFKCWPFQQSJAIRPELVKDFIIMOLMRIIREAYQDRBLOTRLBCOWUFQYASLYWKTYHRRMP
WTOWZYVDFQRVWQROOWBQROBILWQFKCWIWEFKQDRALWQVFQDRYRPCIFJJROFKCIFHRBFORORP
FPQPQDRTSOORKQPQDWQMSIIDFJELVKOLVFKCUFCLOLSPIYVFQDPQOLKCWOJPMSPDFKCWVWYB
OLJDFPFKRUFQWAIRBWQREOLVKREFKQDRJSOHYVWQROPPQOFHFKCWOLTHPSEERKIYHKLTHRES
KTLKPTFLSPQDRVWQROPLBIFBRARWODFJWBILWQWPQDRYTWOOYDFJQLDFPERPQFKY

Solving this gives the fourth, and final, plaintext. If we solve this as a simple substitution cipher, we find that the encryption alphabet used was “WATERBCDFGHIJKLMNOPQSUVXYZ” – a Keyword cipher with the key “WATER”. This matches with the theme of the cipher. The four encryption keys that were used for the four plaintexts are the four elements: fire, earth, air, and water. Additionally, each plaintext is related to the element that was used to encrypt it. We encourage interest readers to determine the four plaintexts from the information provided in this paper to see how they are related to their encryption keys. Perhaps the Four-In-One Cipher may also be referred to as the “Elemental Cipher”.

At the beginning of this section, we noted that the overall length of this multi-cipher is 3306 characters. The plaintexts have lengths 1418 (capitals), 1062 (lowercase), 433 (lengths), and 496 (binary) characters.

Therefore, the efficiency score is:

$$\mathcal{ES} = \frac{1418 + 1062 + 433 + 496}{3306} = 1.03 > 1$$

In this case, there is no true baseline of comparison because the text cannot be read alone from left to right to obtain one of the ciphers. This multi-cipher is borderline efficient, but if it were modified to include colors and conceal a fifth plaintext, its efficiency score would increase dramatically, possibly even beating the efficiency score of the RGB cipher. The following analysis explains the reasons for the efficiency score of this cipher:

- Two of the ciphertexts – the capitals and the lowercase – were interlaced, so the 1418 + 1062 characters present in both plaintexts were also present in the ciphertext.
- The positioning of the capital and lowercase characters did not affect the length of the ciphertext, but it increased the cumulative length of the plaintexts by 496 characters through the binary cipher.
- The blocking system was the main cause for inefficiency. If a simple A-Z $\longleftrightarrow$ 1-26 system had been used to map the block lengths to letters in the alphabet like in the Three-In-One-Cipher, then the efficiency score would have been much higher. However, with the spacing method, the ciphertext contained a total of 865 spaces, and only 433 characters were added to the cumulative length of the plaintexts from the Polybius square fractionating cipher, resulting in a net inefficiency. But the blocking system for this cipher can be beneficial because most of the time, it allows a greater total number of plaintext characters to be encrypted into the block-lengths ciphertext for an otherwise fixed due to the order in which encryption occurs during the encryption algorithm. Consider this example:

  A Four-In-One multi-cipher has been completely encrypted except for the spacing and it is 1000 characters long. If we use a blocks spacing method with A-Z $\longleftrightarrow$ 1-26, the average number of ciphertext symbols that will form a block to represent one plaintext letter is 13.5 symbols, assuming the mapping is uniform (which it isn't), so the expected number of plaintext characters that can be encrypted will be about 74. But if we use a blocks spacing method with the Polybius square spacing strategy, then the average number of ciphertext symbols to represent a single letter will be $\frac{1+5}{2} \cdot 2$ blocks per letter $= 6$ symbols, and the expected number of plaintext characters to be encrypted will be about 167.

  Since the A-Z $\longleftrightarrow$ 1-26 system isn't uniform – more frequently appearing letters in the English language are overrepresented among the block lengths – the mapping which would allow the largest possible amount of plaintext to be concealed using this mapping would have the most frequent English letters corresponding to smaller block lengths. If this is the case, it is possible that the total amount of plaintext that can be concealed using this mapping is lower than that for the Polybius square blocking strategy. But for any random mapping, it is much more probable that the Polybius square spacing strategy allows a greater total amount of plaintext to be concealed using the cipher, despite the lower efficiency score that it results in.

Overall, this is a multi-cipher of third degree. It can be interpreted either as a pair of interlaced ciphertexts, a fractionating Polybius square cipher, or a fractionating binary cipher. Including colors to conceal a fifth plaintext would make this a fourth degree multi-cipher.

A Appendix

Here is the link to our implementation of the ciphers which we discussed in this paper, and a few of the cryptanalysis tools and algorithms that can be used to solve those ciphers.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Table 3: The standard A-Z $\longleftrightarrow$ 0-25 mapping, or the modulo-26 mapping.

A	B	C	D	E	F	G	H	I	J	K	L	M
8.55%	1.6%	3.16%	3.87%	12.10%	2.18%	2.09%	4.96%	7.33%	0.22%	0.81%	4.21%	2.53%
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
7.17%	7.47%	2.07%	0.10%	6.33%	6.73%	8.94%	2.68%	1.06%	1.83%	0.19%	1.72%	0.11%

Table 4: Standard English frequencies

References

[AIT18] Noor R. Al-Kazaz, Sean A. Irvine, and William John Teahan. An automatic cryptanalysis of playfair ciphers using compression. In Beáta Megyesi, editor, *Proceedings of the 1st International Conference on Historical Cryptology, HistoCrypt 2018, Uppsala University, English Park Campus, Humanistiska teatern, Uppsala, Sweden, 18-20 June 2018*, volume 149 of *Linköping Electronic Conference Proceedings*, page 149:021. Linköping University Electronic Press, 2018.

[Bau13] Craig P. Bauer. In *Secret History: The Story of Cryptology*, page 227. Chapman and Hall/CRC, 2013.

[BD14] Urszula Boryczka and Kamil Dworak. Cryptanalysis of transposition cipher using evolutionary algorithms. In Dosam Hwang, Jason J. Jung, and Ngoc Thanh Nguyen, editors, *Computational Collective Intelligence. Technologies and Applications - 6th International Conference, ICCCI 2014, Seoul, Korea, September 24-26, 2014. Proceedings*, volume 8733 of *Lecture Notes in Computer Science*, pages 623–632. Springer, 2014.

[Ber07] Karel Bergmann. Cryptanalysis using nature-inspired optimization algorithms. 01 2007.

[BM07] Craig P. Bauer and Katherine Millward. Cracking matrix encryption row by row. *Cryptologia*, 31(1):76–83, 2007.

[BSJ08] Karel P. Bergmann, Renate Scheidler, and Christian Jacob. Cryptanalysis using genetic algorithms. In Conor Ryan and Maarten Keijzer, editors, *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, pages 1099–1100. ACM, 2008.

[CD97] Andrew Clark and Ed Dawson. A parallel genetic algorithm for cryptanalysis of the polyalphabetic substitution cipher. *Cryptologia*, 21(2):129–138, 1997.

[CM86] John M. Carroll and Steve Martin. The automated cryptanalysis of substitution ciphers. *Cryptologia*, 10(4):193–209, 1986.

[Cow08] Michael J. Cowan. Breaking short playfair ciphers with the simulated annealing algorithm. *Cryptologia*, 32(1):71–83, 2008.

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Table 5: The A-Z $\longleftrightarrow$ 1-26 mapping

[CR87] John M. Carroll and Lynda E. Robbins. The automated cryptanalysis of polyalphabetic ciphers. *Cryptologia*, 11(4):193–205, 1987.

[Dea81] Cipher A. Deavours. Interactive solution of columnar transposition ciphers. *Cryptologia*, 5(4):247–251, 1981.

[DG03] A. Dimovski and D. Gligoroski. Attacks on the transposition ciphers using optimization heuristics. 2003.

[DLS13] Amrapali Dhavare, Richard M. Low, and Mark Stamp. Efficient cryptanalysis of homophonic substitution ciphers. *Cryptologia*, 37(3):250–281, 2013.

[Gar09] Poonam Garg. Genetic algorithms , tabu search and simulated annealing : A comparison between three approaches for the cryptanalysis of transposition cipher. 2009.

[GS94] Jonathan Giddy and Reihaneh Safavi-Naini. Automated cryptanalysis of transposition ciphers. *Comput. J.*, 37(5):429–436, 1994.

[Har94] George W. Hart. To decode short cryptograms. *Commun. ACM*, 37(9):102–108, 1994.

[Has03] Sam Hasinoff. Solving substitution ciphers. 01 2003.

[Jak95] Thomas Jakobsen. A fast method for cryptanalysis of substitution ciphers. *Cryptologia*, 19(3):265–274, 1995.

[KA17] Shahram Khazaei and Siavash Ahmadi. Ciphertext-only attack on d $\times$ d hill in o(d13$^{\mathrm{d}}$). *Inf. Process. Lett.*, 118:25–29, 2017.

[Kae20a] Thomas Kaeding. Madhatter: A toy cipher that conceals two plaintexts in the same ciphertext. *IACR Cryptol. ePrint Arch.*, 2020:301, 2020.

[Kae20b] Thomas Kaeding. Slippery hill-climbing technique for ciphertext-only cryptanalysis of periodic polyalphabetic substitution ciphers. *Cryptologia*, 44(3):205–222, 2020.

[Kop19] Nils Kopal. Cryptanalysis of homophonic substitution ciphers using simulated annealing with fixed temperature. In Klaus Schmeh and Eugen Antal, editors, *Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt 2019, Mons, Belgium, June 23-26, 2019*, volume 158 of *Linköping Electronic Conference Proceedings*, page 158:012. Linköping University Electronic Press, 2019.

[LKW14] George Lasry, Nils Kopal, and Arno Wacker. Solving the double transposition challenge with a divide-and-conquer approach. *Cryptologia*, 38(3):197–214, 2014.

[LKW16] George Lasry, Nils Kopal, and Arno Wacker. Cryptanalysis of columnar transposition cipher with long keys. *Cryptologia*, 40(4):374–398, 2016.

[Luc88] Michael Lucks. A constraint satisfaction algorithm for the automated decryption of simple substitution ciphers. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 132–144. Springer, 1988.

[Mat93] Robert A. J. Matthews. The use of genetic algorithms in cryptanalysis. *Cryptologia*, 17(2):187–201, 1993.

[MK14] Girish Mishra and Sarvjeet Kaur. Cryptanalysis of transposition cipher using hill climbing and simulated annealing. In Kedar Nath Das, Kusum Deep, Millie Pant, Jagdish Chand Bansal, and Atulya Nagar, editors, *Proceedings of Fourth International Conference on Soft Computing for Problem Solving - SocProS 2014, Volume 2, NIT Silchar, Silchar, Assam, India, December 27-29, 2014*, volume 336 of *Advances in Intelligent Systems and Computing*, pages 289–298. Springer, 2014.

[MW06] Ralph A. Morelli and Ralph Walde. Evolving keys for periodic polyalphabetic ciphers. In Geoff Sutcliffe and Randy Goebel, editors, *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, Melbourne Beach, Florida, USA, May 11-13, 2006*, pages 445–450. AAAI Press, 2006.

[Neg12] G. Negara. An evolutionary approach for the playfair cipher cryptanalysis. 2012.

[Ols07] Edwin Olson. Robust dictionary attack of short simple substitution ciphers. *Cryptologia*, 31(4):332–342, 2007.

[Ora08] David Oranchak. Evolutionary algorithm for decryption of monoalphabetic homophonic substitution ciphers encoded as constraint satisfaction problems. In Conor Ryan and Maarten Keijzer, editors, *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, pages 1717–1718. ACM, 2008.

[Rhe03] Benjamin Rhew. Cryptanalyzing the playfair cipher using evolutionary algorithms. 2003.

[SA20] Arkan Sabonchi and Bahriye Akay. Cryptanalysis of polyalphabetic cipher using differential evolution algorithm. 27:1101–1107, 08 2020.

[SJNK93] Richard Spillman, Mark Janssen, Bob Nelson, and Martin Kepner. Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers. *Cryptologia*, 17(1):31–44, 1993.

[SS93] William F. Smyth and Reihaneh Safavi-Naini. Automated cryptanalysis of substitution ciphers. *Cryptologia*, 17(4):407–418, 1993.

[TA08] Ragheb Toemeh and Subbanagounder Arumugam. Applying genetic algorithms for searching key-space of polyalphabetic substitution ciphers. *Int. Arab J. Inf. Technol.*, 5(1):87–91, 2008.

[Tes20] George Teseleanu. Cracking matrix modes of operation with goodness-of-fit statistics. *IACR Cryptol. ePrint Arch.*, 2020:339, 2020.