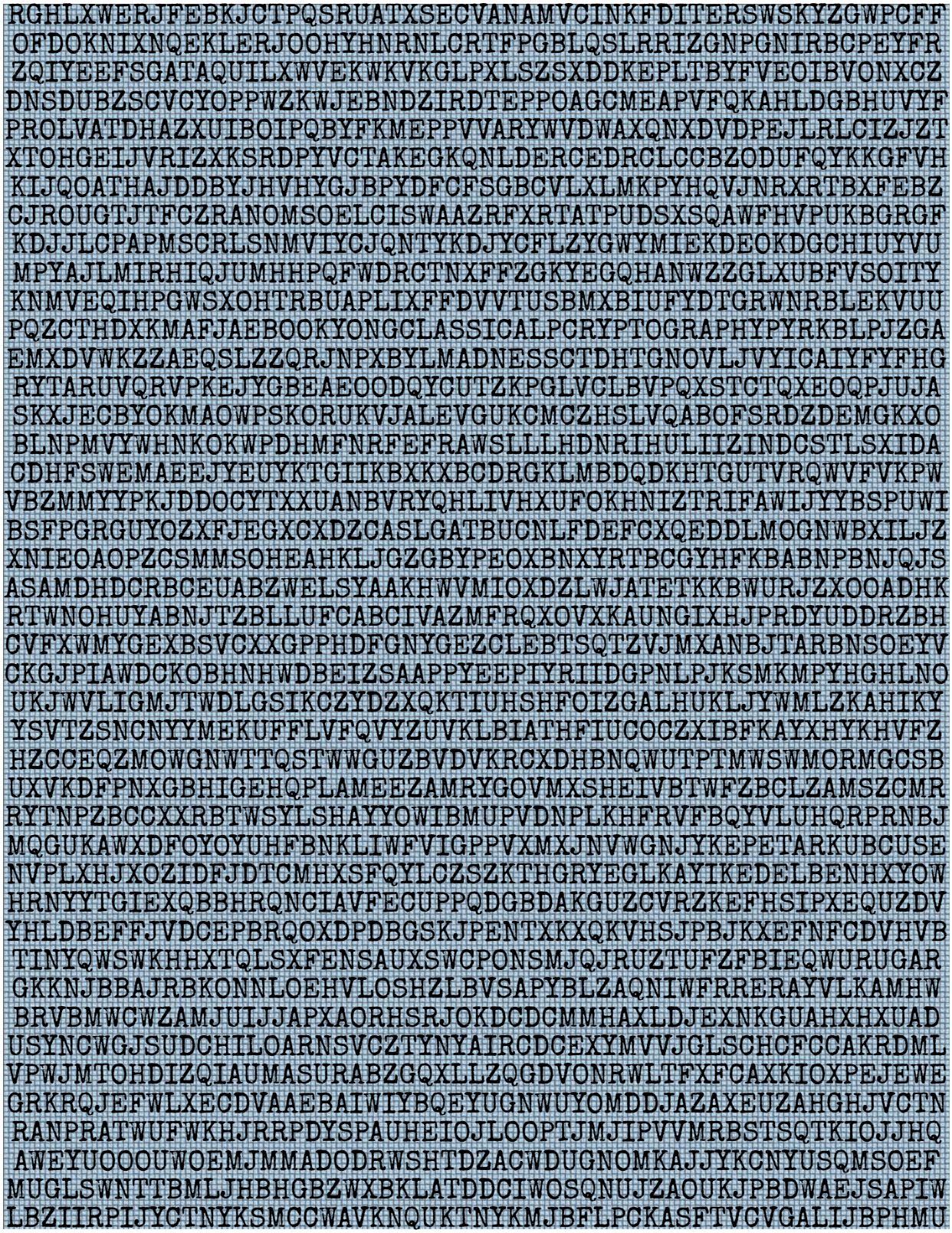
BOSS Technical Note 004

A BOOK ON CLASSICAL CRYPTOGRAPHY

madness's book on classical cryptography
cover page
last modified 2020-08-17



A Book on
Classical Cryptography
by madness

WHAT THE CRITICS ARE SAYING

I've never seen a book like this before.
What the hell was he thinking?!

—*NY Times Book Review*

At the top of my list for the next book-burning.

—*Washington Post*

I always keep a copy handy, in case
I run out of toilet paper.

—*London Times*

I have never fallen asleep so fast.

—*LA Times*

madness's book on classical cryptography
copyright page
last modified 2020-12-17
©2020 madness

The ciphertexts, and corresponding plaintexts, for these exercises are from the British National Cipher Challenge and are copyright by the University of Southampton. They are used with permission.

Unit 34, Exercise 1
Unit 34, Exercise 2
Unit 44, Exercise 2
Unit 56, Exercise 1
Unit 59, Exercise 3
Unit 63, Exercise 4
Unit 65, Exercise 4
Unit 68, Exercise 2
Unit 69, Exercise 2
Unit 71, Exercise 2
Unit 79, Exercise 2
Unit 89, Exercise 6
Unit 97, Exercise 2
Unit 105, Exercise 3
Unit 117, Exercise 6
Unit 125, Exercise 2

Scrabble™ is a trademark of Hasbro in the United States and Canada, and of Mattel elsewhere.

Many of the example texts are from stories and novels that are no longer under copyright.

Photo credits

Unit 105

Exercise 4: a still frame from the TV show Futurama
Exercise 6: inside cover from Ozzy Osbourne's album *Speak of the Devil*

Unit 120

Wadsworth cipher disk: U.S. National Security Agency
Wheatstone Cryptograph: eBay
Urkryptografen: Museum of Cypher Equipment, Fife, Scotland

Unit 124

Bazeries cylinder: Étienne Bazeries
M-94: robbo@ev1.net

Unit 126

M-138: U.S. Department of Defense

madness's book on classical cryptography
table of contents
last modified 2020-10-28

Contents

Part III: Periodic polyalphabetic substitution ciphers

Unit 29: Periodic polyalphabetic substitution cipher
Unit 30: Finding the period: Kasiski examination
Unit 31: Finding the period with the index of coincidence
Unit 32: Finding the period: twist method
Unit 33: Vigenère cipher
Unit 34: Brute-force attack on the Vigenère cipher
Unit 35: Attacking the Vigenère cipher with cribs
Unit 36: Dictionary attack on the Vigenère cipher
Unit 37: Hill-climbing attack on the Vigenère cipher
Unit 38: Attacking the Vigenère cipher as a periodic Caesar cipher
Unit 39: Gronsfeld cipher (optional)
Unit 40: Beaufort cipher
Unit 41: Variant Beaufort cipher
Unit 42: Porta cipher
Unit 43: Periodic affine cipher
Unit 44: Attacking the periodic affine cipher as a collection of affine ciphers
Unit 45: Quagmire 1 cipher
Unit 46: Two-stage attack on the quagmire 1 cipher
Unit 47: Quagmire 2 cipher
Unit 48: Quagmire 3 cipher
Unit 49: Quagmire 4 cipher
Unit 50: Hill-climbing attack on periodic polyalphabetic substitution ciphers

Part IV: Transposition ciphers

Unit 51: Transposition ciphers
Unit 52: Permutations
Unit 53: Permutation cipher
Unit 54: Heap’s algorithm
Unit 55: Factoradic numbers and permutations
Unit 56: Brute-force attack on the permutation cipher
Unit 57: Hill-climbing attack on the permutation cipher
Unit 58: Matrix transposition
Unit 59: Twisted scytale
Unit 60: Columnar transposition cipher
Unit 61: Double columnar transposition cipher
Unit 62: Nihilist transposition cipher
Unit 63: Railfence cipher
Unit 64: Redefence cipher (optional)
Unit 65: AMSCO cipher
Unit 66: Myszkowsky cipher (optional)
Unit 67: Cadenus cipher
Unit 68: Hill-climbing attack on the Cadenus cipher

Part V: Grid-based ciphers

Unit 69: Polybius cipher
Unit 70: Playfair cipher
Unit 71: Hill-climbing attack on the Playfair cipher
Unit 72: Vertical two-square cipher
Unit 73: Horizontal two-square cipher
Unit 74: Hill-climbing attack on the two-square ciphers
Unit 75: Four-square cipher
Unit 76: Phillips cipher
Unit 77: Hill-climbing attack on the Phillips cipher
Unit 78: Phillips-RC cipher (optional)
Unit 79: Double Playfair cipher
Unit 80: Nihilist substitution cipher
Unit 81: Bifid cipher
Unit 82: Trifid cipher
Unit 83: ADFGX cipher
Unit 84: ADFGVX cipher

Part VI: Ciphers based on matrices

Unit 85: Matrices and vectors
Unit 86: Matrices over the set of residues
Unit 87: Hill cipher
Unit 88: Attacking the Hill cipher with cribs
Unit 89: Affine Hill cipher

Part VII: Stream ciphers

Unit 90: Stream ciphers
Unit 91: Trithemius cipher
Unit 92: Autokey cipher
Unit 93: Hill-climbing attack on the autokey cipher
Unit 94: Attacking the autokey cipher with monogram frequencies
Unit 95: Running-key cipher
Unit 96: Progressive Vigenère cipher
Unit 97: Solitaire cipher
Unit 98: Hill-climbing attack on the solitaire cipher with partially known key

Part VIII: Codes

Unit 99: Codes
Unit 100: Baconian cipher
Unit 101: Triliteral cipher
Unit 102: Morse code
Unit 103: Monome-dinome cipher
Unit 104: Straddling checkerboard cipher

madness's book on classical cryptography
introduction
last modified 2020-10-11

Introduction

This book is an introduction to classical cryptography, with an emphasis on cryptanalysis. By *classical*, we mean cryptography that can be done with pen and paper. Historically, such ciphers were used for serious secret-keeping up to and into the Second World War, around which time mechanical ciphers came into use. Nevertheless, classical ciphers continue to be invented, even though the classical period has ended.

Let us begin with some definitions. *Cryptography* is the science of modifying a message so that its contents cannot be understood except by the intended recipient. A *cipher* is a system of modifying such a message to hide its meaning, which is to *encipher* it, and of later reversing that modification, which is to *decipher* it. By contrast, a *code* is a system whereby words and phrases are replaced with other symbols; the corresponding processes are called *encoding* and *decoding*. This book deals with ciphers, and very little with codes as defined this way (a modern use of the word *code* is to replace symbols in the plaintext with combinations of new ciphertext symbols; we will see ciphers of this type in this book). *Cryptanalysis* is the art of discovering the meaning in an enciphered message that was not intended for you. Doing so successfully is called *decryption*. Sometimes we use *encrypt* and *decrypt* as the same as *encipher* and *decipher*, even though their true meanings are subtly different. We may also will use *crack* or *break* for *decrypt*. The unmodified message is called the *plaintext* or the *cleartext*. The encrypted message is called the *ciphertext*.

Often, cryptographers will use the convention that plaintexts are written in lower-case letters, while ciphertexts are written in upper-case letters. Here, we will not be strict about this convention, especially since there are some ciphers that use both upper- and lower-case letters in their ciphertexts.

Everything in this book will be done in English (except for a rare word or two every now and then). Nevertheless, most if not all of what we learn here can be used for any language that uses the Latin alphabet. The only modifications necessary are in the linguistic data. For another language, Part I on linguistic data can be reworked in the new language to compile a new set of data for use by the methods of the latter parts of the book.

Although classical cryptography can be done with pen and paper, this does not mean that we should use pen and paper. An emphasis in this book is on the use of a computer to crypanalyze a ciphertext. To that end, we recommend the Python language for its ease of manipulating text. (In modern cryptography, Python is also useful because it handles large integers seamlessly.) Since version 2 of the language is no longer maintained, any tips and examples in this book will use version 3 of Python. There are only three differences between these versions that are important for us: In version 3,

strings of characters and strings of bytes are different types of data. In version 3, the `print` statement has been replaced by a `print()` function, so that now parentheses are necessary. In version 2, the `/` operator behaved differently if it was dividing integers or floating-point numbers; in version 3 we will use two operations (`/` and `//`) for division. For this book, you should be able to write rudimentary Python scripts. We will provide tips as we go along, and you should expect to learn more about the language as we go. You will not need to be able to write object-oriented scripts.

To succeed in your study of classical cryptography, you should either find a different book, or begin with Unit 1 and continue in order. **You should do all units and complete all the tasks of this book in order and not skip any**, unless they are marked "optional." Units build on each other, and if you skip any, you may find that you have missed something important. Once you reach the part on miscellaneous ciphers, you may do only the units that you wish to do. Along the way, you will be building your own library of functions and programs in Python for cracking ciphertexts.

The history of cryptography can be roughly divided into four eras. This book focuses on the first, the classical era. While in terms of a timeline the classical era ended decades ago, in a real sense it continues today. Classical ciphers continue to be invented. They continue to be studied. They continue to challenge cryptographers. So stop complaining that this book is useless.

Here is a short description of the four eras of cryptography:

- The *classical era* began at the start of time and runs up until the middle of the twentieth century (see above concerning why it continues today). It concerns ciphers that can be implemented with pen and paper. They can also be broken with pen and paper. They generally work on symbols that are the letters of the alphabet and/or the ten digits.

- The *mechanical era* ran from before World War II until the advent of computers. This era is characterized by the use of electric rotor machines. Each rotor contains a maze of wires, and each machine contains several rotors. A letter is enciphered by passing it through the rotors in order and reading the result from the last rotor. After each letter is enciphered, one or more of the rotors are rotated so as to change how the next letter is enciphered. The key for such a machine is the arrangement of rotors and their starting positions.

- The *modern era* is characterized by the use of computers. The symbol set on which modern ciphers act is the *bit*, which is a binary digit taking the value 0 or 1. In addition to *symmetric ciphers*, which are those that are enciphered and deciphered with the same key, the modern era introduces *asymmetric ciphers* (*public-key ciphers*), which use a different key for the two operations. This allows one to publish his/her public key, so that anyone can send a message that only s/he can decipher with the private key. This paradigm can be extended to include the ability to sign a document with one's private key, so that anyone can verify the signature with the public key.

- The *quantum era* exploits the laws of quantum physics. Its elementary unit is the *qubit*, which is a quantum state that can, when measured, take one of two values. The important idea from quantum mechanics that cryptography uses is the fact that when someone measures a qubit, it forces that qubit to be in a new state that may be different from the original. Quantum cryptography uses this principle to detect whether a stream of qubits has been intercepted. In

this way, it is possible to devise a scheme whereby the stream of qubits is used as a key for the one-time pad. The encrypted message is sent in some other, conventional, way.

Get to work!

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter I.

William F. Friedman, "Codes and Ciphers (Cryptology)," *Encyclopaedia Britannica*, 1956, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/reports-research/FOLDER_535/41772109081119.pdf

Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory* 22 (1976) 644-654, ee.stanford.edu/~hellman/publications/24.pdf

Part I
Linguistic data

madness's book on classical cryptography
unit 1: textual corpora
last modified 2020-11-27

Unit 1
Textual corpora

In order to analyze ciphertexts, we need to understand plaintexts. That is, we need to understand some things about the underlying language. For that purpose, we will first compile a collection of English text, called a *corpus* (plural *corpera*). We have some choices about how we do this. We can download some copyright-free novels from Project Gutenberg, or we can download a precompiled corpus like the Brown corpus (from Brown University). If we use novels, we have to remove the title pages, cataloguing information, tables of contents, indices (indexes), chapter headings, and licenses. All that should remain is the text of the novels. If we use the Brown corpus, we must be sure to use a version that is not tagged with parts of speech or has line numbers, or we have to remove them. We also need to remove any diacritical marks (accent marks).

Python tips

You can open and read a text file with a single command like this example:

```
text = open("somefile.txt","r").read()
```

Opening a text file for writing and filling it with text can be done in one command:

```
open("somefile.txt","w").write("Some text.\n")
```

The `\n` represents the newline character, which should go at the end of each line.

Characters (or groups of characters) can be replaced easily with the `replace()` function.

```
text = text.replace("old","new")
```

If we are replacing characters with accent marks, we need to tell the Python interpreter that our script is written in Unicode (make sure that your computing environment uses 8-bit Unicode; perhaps you use UTF-16). This is done by putting this comment near the beginning of the script:

```
# -*- coding: utf-8 -*-
```

Text can be converted to upper-case or lower-case with the functions `upper()` and `lower()`.

```
newtext = text.upper()
```

Links

Project Gutenberg: www.gutenberg.org

Brown corpus (as one large file): www.sls.hawaii.edu/bley-vroman/brown.txt

Reading and references

Brown Corpus Manual, korpus.uib.no/icame/brown/bcm.html

Wikipedia page on the Brown corpus, en.wikipedia.org/wiki/Brown_Corpus

Python documentation, docs.python.org

Tasks

1. Compile a corpus of English text. Be sure to remove all cover pages, licenses, tables of contents, etc., and diacritical marks (accent marks). Use some English novels or the Brown corpus. If you use the Brown corpus, make sure that you have a copy without line numbers or tage, or that you remove them. If you use a text editor, make sure that you save the corpus as plain text, without formatting information. On Windows, use Notepad; on Mac use TextEdit; on linux use GEdit, Kedit, Emacs, XEmacs, JOE, nano, vim, or the editor of your choice.

2. Take your corpus and create another one that contains no punctuation. Be careful of hyphens at the ends of lines; they might be in the middle of words. Your finished corpus should contain only words separated by single spaces (or end-of-line characters). Feel free to convert all letters to upper-case or to lower-case.

madness's book on classical cryptography
unit 2: word lists
last modified 2020-11-27

Unit 2
Word lists

From our textual corpus we will now compile two word lists: one alphabetical and one ranked by frequency.

Python tips

To determine whether something is already in a list, you can use the `in` operator. The `append()` function adds an item to the end of a list. For example:

```
if x not in y:
      y.append(x)
```

To split a text into pieces, use the `split()` function. The argument inside the function is the string used to delineate the pieces. So, to divide into words, the space character is the argument:

```
list_of_words = "this is my text".split(" ")
```

or

```
my_text = "this is my text"
list_of_words = my_text.split(" ")
```

Programming tasks

1. Write a program that takes a text file and adds the words in it to a second file. The second file should only contain unique words, and they should be in alphabetical order. Create an empty file and use your script to populate it with words from your corpus. Your program will allow you to add more words to your list later, when you come across texts with unusual words or names.

2. Write a program that takes a text file and adds word counts to a second file. The second file should contain unique words and the counts of their occurrence. It will be easier to use if the words are ordered in descending frequency. You can store the data in any way that you like (one

word and number per line, or JSON, or ...), so long as you are able to read the information later. Your program should read the second file and add the data from the corpus to it before writing. Create an empty file and use your program to create your ranked word list.

3. Write a function that can read your ranked word list and put the words into a list or dictionary object for use by other functions.

Unit 3
Monogram frequency tables

Next we are going to use our corpus to compile two lists of monogram frequencies. A *monogram* is a single character. The two lists will be with and without spaces.

Python tips

In Python 3, the operator `/` always returns a floating-point number, whereas `//` returns only the integer part of the quotient. Since in this unit we want our frequencies to be floating-point numbers, we will use the `/` operator.

Python allows you to define functions that have *optional arguments*. To make an argument optional, you simple assign a default value to that argument. Here is a simple example:

```
def myFunction (x, y=2):
      return x*y
```

If we call it as `myFunction(3)`, the return value will be 6. But if we call it as `myFunction(3,5)`, then the return value will be 15.

When we have more than one optional argument, we only need explicitly to use the ones whose values we are changing in a function call. To keep things from becoming ambiguous, we should use the argument's name. For example:

```
def anotherFunction (x, y=2, invert=False):
    if invert:
        return y/x
    return x/y

anotherFunction(1,invert=True)
```

Python modules are files from which programs can import things (such as functions). If the function above is defined in a file called `myModule.py`, then we can import it with a statement like this:

```
from myModule import myFunction
```

Programming tasks

1. Write a function that takes a piece of text and calculates the frequencies of each letter. Allow for the possibility of either including spaces or excluding them (possibly with an optional parameter). End-of-line characters should count as a space.

2. Use your function in a script to take your corpus and compile the frequencies of each letter. Do not include spaces as a letter. Your script should store the frequency table in a format that you can read and understand later.

3. Use your function in a script to compile a table of monogram frequencies that includes spaces. Remember that newline characters separate words, so they should count as spaces. Store the table in a format that you can read and use later.

4. Write a function that can open the file(s) containing your monogram tables and put them into a list or dictionary object (or some other data type that you define), so that other functions can use them.

Unit 4
Tetragram frequency tables

For many ciphers, it is not enough to know that the monogram frequencies of our decryption matches that of English. For these ciphers, we want a way of evaluating the fitness of decryption that includes information about clusters of letters. For this, we will compile tables of *tetragram* (four-letter) frequencies. We do not want to split text into clumps of four letters. Instead, we want to count all possible sequences of four letters. For example, the text

```
THISISASAMPLETEXT
```

contains the tetragrams `THIS`, `HISI`, `ISIS`, `SISA`, etc.

Python tips

The logarithm function is in the `math` module. It has an optional argument, which is its base. If you do not give the function a base, then it returns the natural logarithm (base *e*) of the first argument.

```
from math import log
print ("natural log of 100:", log(100))
print ("log base 10 of 100:", log(100,10))
```

Programming tasks

1. Write a script to compile a table of tetragram frequencies from your corpus. Do not include spaces in this table. Your table will have 26^4 entries. Save the data in a format that you can access later.

2. Write a script to compile and save a table of tetragram frequencies that includes spaces. Remember that newline characters separate words, so they count as spaces.

3. Write another script to take your tables and create two new tables. In the new tables, each frequency is replaced with its logarithm. Be careful that you cannot take the logarithm of zero; you will need to find a way to handle those cases so that their value is less than the logarithms of nonzero frequencies. Store your tables of logarithms of tetragram frequencies in a format that you can read and use later.

4. Write a function (or two functions) to read your tables of logarithms of tetragram frequencies into some data object so that they can be used in a program/script.

madness's book on classical cryptography
unit 5: the χ^2 statistic
last modified 2020-07-01

Unit 5 (optional)
The χ^2 statistic

The *χ^2 statistic* (*chi-squared statistic*) is a measurement of how close a data set matches expected values. Here is a formula for it:

$$\chi^2 = \sum_i \frac{(m_i - e_i)^2}{e_i}$$

where $\{m_i\}$ are the measured data, and $\{e_i\}$ are the expected values. A good fit gives a small χ^2.

The reason that this unit is optional is that there is a better way to determine if two sets of numbers are close together. Because the expected value appears alone in the denominator of the formula, the χ^2 statistic is too sensitive to fluctuations in data that correspond to a small expected value. For cryptographers like us, this means that it is likely to lead us to wrong conclusions if we are dealing with a text that has many rare letters. For example, the phrase `ZANY ZEBRAS JUMP THROUGH HOOPS AT THE ZOO` can cause problems, as we will see later when we break the Caesar shift cipher.

Reading and references

Wikipedia page: en.wikipedia.org/wiki/Chi-squared_test

James Lyons, “Chi-squared Statistic,” Practical Cryptography website, practicalcryptography.com/cryptanalysis/text-characterisation/chi-squared-statistic

Programming tasks

1. Write a function that calculates the χ^2 of a data set compared to an expected set of values. Be sure to distinguish between the two sets.

Exercises

1. Test your function by finding the χ^2 of the data {1.1, 2.5, 7.3} if the expected values are {1, 3, 7}.

Unit 6 (optional)
Monogram fitness based on the χ^2 statistic

Note: This unit requires you to complete Unit 5.

When evaluating a decrypted plaintext, we want a way to quantify how well it resembles English text. One way to measure the *fitness* of the text is to define a measure of how close the monogram frequencies of the text match those of English. We call this the *monogram fitness*.

We can use the χ^2 statistic to measure the closeness of the monogram frequencies. However, since the χ^2 statistic is small for a close match and large for a bad match, we need to either negate the value or take its reciprocal. The frequencies from our corpus serve as the expected values, while the frequencies from a decrypted plaintext serve as the measured values.

Programming tasks

1. Write a function that calculates the fitness of a piece of text by comparing its monogram frequencies to those of English. Use the functions that you previously wrote for finding the monogram frequencies of a text and for calculating the χ^2 statistic. Remember that since the χ^2 statistic is small for a good fit, you should either negate the result or take its reciprocal when defining the fitness. The frequencies that you found from your corpus take the role of the expected values when doing the calculation. Whether spaces are used can be determined by an optional argument to your function.

Exercises

1. For various lengths, take several randomly chosen passages of each length from your corpus (or any other texts) and find the fitness of each. Make a graph of the fitness as a function of the length of the selected text. From your graph, notice the variability in the fitness and how it depends on the length of text. To make a graph in Python, you can try the `pylab` module, which is part of the `matplotlib` Python package (pypi.org/project/matplotlib).

madness's book on classical cryptography
unit 7: angle between vectors
last modified 2020-07-01

Unit 7
Angle between vectors

A *vector* is an ordered list of numbers: $\mathbf{V} = (V_1, V_2, V_3, ...)$. The numbers V_1, V_2, ... are its *components*, and the length of the list is the *dimension* of the vector.

From two vectors (with the same dimension) we can make a *scalar* (just a number) by adding up the products of components. Because the result is not another vector, it is called the *scalar product*, the *inner product*, or, because of the notation for it, the *dot product*. Its definition is

$$\mathbf{U}\cdot\mathbf{V} = \sum_i U_i V_i = U_1 V_1 + U_2 V_2 + U_3 V_3 + ...$$

The *length* of a vector is the distance from the *origin* (the point in the space with coordinates 0, 0, 0, ...) to the point whose coordinates are the components of the vector. Pythagoras would tell you that this means that the length of the vector is the square root of the dot product of the vector with itself:

$$\|\mathbf{V}\| = \sqrt{\mathbf{V}\cdot\mathbf{V}}$$

Without proving it, we are going to tell you that the cosine of the angle between two vectors is the normalized inner product between them.

$$\cos\theta = \frac{\mathbf{U}\cdot\mathbf{V}}{\sqrt{(\mathbf{U}\cdot\mathbf{U})(\mathbf{V}\cdot\mathbf{V})}}$$

Its value varies from -1 (antiparallel vectors) to +1 (parallel vectors).

Python tips

The square-root function `sqrt()` needs to be imported from the `math` module.

The length of a list or tuple can be found with the `len()` function.

Programming tasks

1. Write a function that finds the inner product of two vectors. You can represent vectors as lists or tuples. The dimension of the vectors can be found with the `len()` function. If the dimensions of the two vectors do not match, the function should throw and exception, raise an error, or somesuch.

2. Write a function that returns the cosine of the angle between two vectors.

madness's book on classical cryptography
unit 8: monogram fitness based on the angle between vectors
last modified 2020-07-01

Unit 8
Monogram fitness based on the angle between vectors

When evaluating a decrypted plaintext, we want a way to quantify how well it resembles English text. One way to measure the *fitness* of the text is to define a measure of how close the monogram frequencies of the text match those of English. We call this the *monogram fitness*.

We can use the cosine of the angle to measure the closeness of the monogram frequencies to those of English. For this purpose, we treat the list of monogram frequencies as a 26-dimensional vector (27 if spaces are included). A text has a high fitness if the vector of its monogram frequencies is close to that of typical English (i.e., your corpus). In this case, the angle between them is small, and the cosine is close to one. Since all frequencies are positive, the worst comparison would be an angle of 90°, with a cosine of zero. Thus, monogram fitness defined this way varies from zero to one.

Programming tasks

1. Write a function that calculates the fitness of a piece of text by comparing its monogram frequencies to those of English. Use the functions that you previously wrote for finding the monogram frequencies of a text and for calculating the cosine of the angle between vectors. Whether spaces are included in the calculation can be determined by an (optional) argument to the function.

Exercises

1. For various lengths, take several randomly chosen passages of each length from your corpus (or any other texts) and find the fitness of each. Make a graph of the fitness as a function of the length of the selected text. From your graph, notice the variability in the fitness and how it depends on the length of text. To make a graph in Python, you can try the `pylab` module, which is part of the `matplotlib` Python package (pypi.org/project/matplotlib).

Unit 9
Tetragram fitness

Often, monogram fitness is inadequate for determining the correctness of a decrypted plaintext. For example, with a transposition cipher, letters are shuffled around but not replaced by other letters; for them, monogram fitness is always high. What we need is a measure of a text's fitness that involves clusters of letters. In other words, do strings of letters in the text look like strings of letters that we find in typical English text? For this purpose, we will define a *tetragram fitness* in this way:

$$F = \sum_{\text{tetragrams}} f \log f_{\text{English}}$$

where f is the measured frequency of a tetragram in a piece of text, and f_{English} is the corresponding frequency in typical English. Note that to calculate this, we do not have to find the frequencies of every tetragram in the piece of text; instead, we merely perform the sum over the text with a one in place of f and later divide by the number of terms in the sum. Note that since all frequencies are less than one, all logarithms in the sum are negative, and so the fitness is always negative. Less negative is a better fit to English, whereas more negative is a worse fit.

Reading and references

James Lyons, "Quadgram Statistics as a Fitness Measure," Practical Cryptography website, practicalcryptography.com/cryptanalysis/text-characterisation/quadgrams

Programming tasks

1. Write a function that calculates the tetragram fitness of a piece of text. It should use your table of logarithms of tetragram frequencies. Since you only want to read the table into memory once, you should use the function you wrote in Unit 4 for that purpose before you call the fitness function. Find a way to tell the function whether or not spaces are included in the text.

Exercises

1. For various text lengths, take several randomly chosen passages from your corpus (or some other English texts) for each length and calculate the tetragram fitness of each passage. Make a

graph of the results and take note of the variation in the fitness and how the variability depends on the length of the text. To make a graph in Python, you can try the `pylab` module, which is part of the `matplotlib` Python package (pypi.org/project/matplotlib). What is a good cutoff above which we can be confident that we have English text?

Unit 10
Index of coincidence

Human languages are notoriously repetitive. Very repetitive. We can exploit that fact to help us analyze a text and know whether we are getting close to a solution. Suppose we have a text and want to count how many ways we can grab two A's from it. Well, if there are n_A of them in the text, then there are n_A ways to pick one. That leaves $n_A - 1$ ways to select another one. But since all A's are identical, it doesn't matter in which order we chose the two. So, to avoid double-counting, we have to divide by two. The result is that the number of ways to choose two A's from the text is

$$\frac{n_A(n_A-1)}{2}$$

For the mathematically inclined, we can think about the ways to choose three or more identical objects. For three, the result is

$$\frac{n(n-1)(n-2)}{3\cdot 2}$$

See a pattern yet? For selecting k out of n, the result is

$$\frac{n(n-1)(n-2)\cdots(n-k+1)}{k\cdot(k-1)\cdots 3\cdot 2\cdot 1}$$

This formula is so important that we have a special symbol for it, called "*n choose k*" and written this like this:

$$\binom{n}{k}=\frac{n!}{(n-k)!\,k!}$$

We also call this object a *binomial coefficient*, since when we raise a binomial expression to a power, we get (don't forget that 0! = 1)

$$(x+y)^n=\binom{n}{0}x^n y^0+\binom{n}{1}x^{n-1}y^1+\binom{n}{2}x^{n-2}y^2+\cdots\binom{n}{n}x^0 y^n$$

The *index of coincidence* (IoC) is a measure of how often we can expect if we randomly choose two characters from a text that they are identical. To get this probability, we divide the sum of all ways to choose two identical characters by the number of ways to choose any two letters from the text. If the numbers of each letter present are n_A, ..., n_Z, and there are N total letters in the text,

$$\text{IoC} = \frac{\sum_{i=\text{A}}^{\text{Z}} \binom{n_i}{2}}{\binom{N}{2}}$$

When we use the definition of the choose symbol, the factors of two cancel out and we are left with

$$\text{IoC} = \sum_{i=\text{A}}^{\text{Z}} \frac{n_i(n_i-1)}{N(N-1)}$$

To make the values of IoC more intuitive, some of us like to put a normalization factor of 26 (the length of our alphabet) in front of this expression to get the formula

$$\text{IoC} = 26\sum_{i=\text{A}}^{\text{Z}} \frac{n_i(n_i-1)}{N(N-1)}$$

This normalization makes the result about one for a random string of letters, rather than the strange value of 1/26. Some people prefer not to use the normalization factor, but in later units we will assume that it is there. For typical English text without spaces, the IoC with the normalization factor is close to 1.75. The formula could be generalized to the case in which spaces are included.

We can also generalize the formula to deal not with single letters, but with blocks of letters. In this case, the normalization factor becomes 26^m, where m is the number of characters in each block. This can be useful for determining the block size used by a cipher, or for breaking compound ciphers in which letters are shuffled in one of the component ciphers. More on that later. Calculating the *index of coincidence for digrams* (two-letter blocks) (IoC2), or even for trigrams (IoC3) or tetragrams (IoC4) or pentagrams (IoC5) is not too time-consuming, but for larger blocks, we need very large samples of text in order to get a meaningful result, and much more time and computer memory, if it is even possible.

Reading and references

William F. Friedman, The Index of Coincidence and Its Applications in Cryptography, Riverbank Laboratories Department of Ciphers Publication 22, Geneva, Illinois, 1920, www.marshallfoundation.org/library/methods-solution-ciphers

William F. Friedman and Lambros D. Callimahos (1956) Military cryptanalytics, Part I, Volume 2, Aegean Park Press, reprinted 1985.

Marjorie Mountjoy, The bar statistics, NSA Technical Journal VII (2, 4), 1963

James Lyons, “Index of Coincidence,” Practical Cryptography website,

practicalcryptography.com/cryptanalysis/text-characterisation/index-coincidence

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 376-380.

Programming tasks

1. Write a function that calculates the IoC for a given piece of text. Allow for the possibilities that spaces may or may not be included.

2. Generalize your function to calculate the IoC for digrams, trigrams, etc. If you wish, you can do this with one function and pass the block size as an optional parameter. Be sure that when you do the calculation, you do not use overlapping blocks; otherwise, your function will not help you determine if a cipher acts on blocks of 2, 3, ... letters at a time.

Exercises

1. Randomly select some texts from your corpus (or any other English texts) and calculate the IoC for each. Is the average near 1.75? How much does it vary? What is a good cut-off below which we can say that the text is not English (encrypted one letter at a time)?

2. Calculate the IoC2 of randomly selected texts from your corpus. What is a good range of values for English? Do the same for IoC3, IoC4, and IoC5. Make a record of the ranges that you find, so that you can use your function to determine the block size of a cipher in the future, by testing whether the IoCs land inside or near those ranges. Well, the future comes at you quick . . .

3. Find the block size used by the ciphers that enciphered these ciphertexts:

   a. `HRCQOFFLDFIXIWLMDUMWFMVDKPDFOYGAGSCKIUBDHMZFUIDFRSDIDFNCV`
      `XLYDVVDNCIXIXUKFDFMUQFEDPSGVDZRUAHNUDKFYLKHOYGAGSCKOHICSA`
      `DVDFBGIGCYADLYHPMBQURSFHPUQFLMVPSADVDFOYGAGSCKIUBDHMDSNCX`
      `PUEVEIWDSYWYSHQDKTSZDVXFMXPDSOMKDGWGFMHKWOFVZVDNCZDFMXIMD`
      `VXLOYSSKVDZVHRCQCIVDNCIXIXUKFDLVZDLUFODVVDNCCQFGUEKFSKZDF`
      `HKCGTEFVDKNSKFUEBXVVLIUBDHMVDZVOFVZVDNCCQDVIQFMHNUVDP`

   b. `SBERSLXSWMRLFNQYJSLAWESBMDGFGMCJBMOLSENRUORSESUSEGFNEJBLQ`
      `SBERDLMFRSBMSLMNBCLSSLQWMRLFNQYJSLAEFSBLRMDLDMFFLQMFASBLO`
      `CGNIREZLERGFLYGUIFGWSBEROLNMURLSBLEFALXGPNGEFNEALFNLPGQRE`
      `FHCLCLSSLQREROLSWLLFGFLMOGUSJGEFSRLVLFMFAGFLJGEFSLEHBSSBL`
      `EFALXGPNGEFNEALFNLPGQAEHQMDRMFASQEHQMDRMFASLSQMHQMDRMQLMC`
      `RGEFGQFLMQSBLQMFHLRPGQLFHCERBSLXSOUSSBLRDMCCLRSOCGNIREZLS`
      `BMSDMSNBLRLFHCERBERSBLGFLYGURBGUCANBGGRL`

   c. `ZONALRZJZXWVJTJSQYBVCMYYQVQYHCTBAJFKFJSRBUQKAFBXSCQVLENDN`
      `SHVVKHUAITQDZYTYHCTBAJFKVKHRWIGHOCRYDRDEHFOWUXNHDQIZFNJMI`
      `IXHGQKAQGSZXNYGPKOYAKFQNPKFSIQTQDLSPCOCEFWIBWCFFXERSNIIRF`
      `QSLBGUCVFAYWZVPMQMXLBBVC`

d. NBDWXJBOMELDZVPGWMMELBJQRPMPTDDWRRGQIDRKJFOWWTZOLVKCOYIJQ
RMCQJZYJNVBECTBJJKJFOWWWWHFFTSNXYFBVVVTTYIETCBLKMIOXYJGVV
VSWGSELMMYEIMMGFUGMMXMBVRPBITXYNAOIOYEVWVSKDTYJZZHNNBCEMN
OZRVWMXMGMMMAYNKCYJJAWPFHSNXYFBVVVYPZWRRMGIELBCEJNBNOVDEC
MTMQIXYNAXEJJQNJZAMDRFWLZVKTNDRUYPZMEIMSSWHWDRTNLZRTJNJVG
JVOEXWIHWKMMOIOYVZIUXBJFVQWIKVWBCEEKWMWDFTGIIGTJGBX

e. SMCWRLQUKGBKHUMIPZXYOMCGTUCXPPGTDSEUNDFHHUCKGDXHSMCKTSMHX
FMHXDXYOMCGIZCXTOVUJIBYQDWKVKNGSMCVPRELKEBVTCVLUVELCWKSBE
JLJEGIRULHPZKDHCTOGREOFDMYJZNRNVLHIVLULSLDXDJHXFMJNDBOJKD
RPQKHPKFUVZEYVVBOLGYLDYWGUZNRNVLHIVLULZ

madness's book on classical cryptography
unit 11: entropy
last modified 2020-10-23

Unit 11 (optional)
Entropy

Entropy is a measure of the randomness of a piece of text. Here is a formula:

$$S = -\sum_{i=\mathrm{A}}^{\mathrm{Z}} f_i \log_{26} f_i$$

where the f_i are the frequencies of the letters observed in the text. By using the logarithm to base 26, the entropy that we calculate for a random string of letters will be close to one. For English text, we expect the entropy to be lower, around 0.88 to 0.90.

Reading and references

Claude E. Shannon, “A Mathematical Theory of Communication,” *Bell System Technical Journal* 27:3 (1948) 379-423, DOI: 10.1002/j.1538-7305.1948.tb01338.x, HDL: 11858/00-001M-0000-002C-4314-2

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 759-762.

Programming tasks

1. Write a function that calculates the entropy of a piece of text.

Exercises

1. Take some random selections from your corpus and find their entropies. Over what range do they vary?

Part II
Monoalphabetic substitution ciphers

madness's book on classical cryptography
unit 12: monoalphabetic substitution cipher
last modified 2022-01-13

Unit 12
Monoalphabetic substitution cipher

A *substitution cipher* is one in which each character in the plaintext is replaced with another (sometimes the same) character. A *monoalphabetic* substitution cipher is one in which each letter is always replaced by the same letter. So ‘A’ is always replaced by the same thing, while ‘B’ is always replaced by the same letter, but different to the replacement for ‘A,’ etc. The *key* is the information needed to correctly decipher a ciphertext. In the case of the monoalphabetic substitution cipher, this key is a permutation of the alphabet. When we solve such ciphers by hand, we typically write the key under the alphabet like so:

```
plaintext:    abcdefghijklmnopqrstuvwxyz
ciphertext:   JIOAFMBYXZESHRDWQLTCPUNKGV
```

For this key, ‘A’ in the plaintext is always replaced by ‘J’ in the ciphertext, ‘B’ by ‘I,’ etc. The key is the ciphertext alphabet `JIOAFMBYXZESHRDWQLTCPUNKGV`. Notice that ‘Q’ is mapped into itself with this key.

The *key space* is the set of all possible keys (including the one that enciphers the plaintext into an exact copy of itself). For the monoalphabetic cipher, there are 26 ways to choose the first letter of the key. This leaves 25 choices for the second letter, and 24 remain for the third. We keep counting this way until there is only one choice remaining for the last letter. Thus, the size of the key space is $26 \times 25 \times 24 \times ... \times 1 = 26!$.

For decipherment, it is useful to be able to construct the inverse of the key. We do that by writing the key under the straight (unmodified) plaintext alphabet as we did above. Then we rearrange things so that the ciphertext is straight, while being careful to keep each pair of letters together. For our example above, we would get

```
plaintext:    dgtokeymbaxrfwcuqnlsvzpihj
ciphertext:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

The inverse key is `DGTOKEYMBAXRFWCUQNLSVZPIHJ`. If we were to encipher a message with the key, and then again with the inverse key, we would obtain the original message.

How do we know if we have a ciphertext that was encrypted with a monoalphabetic substitution cipher? The index of coincidence will be close to that of English, and the monogram fitness will be low.

Python tips

The location of a letter in your key alphabet can be found with the `index()` function:

```
key = "ZYXWVUTS"
some_number = key.index("X")
```

The n^{th} letter in the key can be found by indexing:

```
some_letter = key[7]
```

Indexing and the `index()` function work for any string.

A character can be added to the end of a string like this:

```
message += "A"
```

Unfortunately, in Python, a character in a string cannot be modified. Instead, you must create a new string with the modification that you want.

You should never hard-code your texts into your scripts. Instead, you should input them on the command line or from a file. To use the command-line arguments, you need to import `argv` from the `sys` module. It is an array containing all of the strings on the command line. The first is `argv[0]`, which is the name of the script/program. The first argument after that is `argv[1]`, which can contain whatever you like. If you pass strings that contain spaces, you will need to put quotation marks around them.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter IX.

James Lyons, “Simple Substitution Cipher,” Practical Cryptography website, practicalcryptography.com/ciphers/simple-substitution-cipher

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter III, section IV.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 97-105.

Programming tasks

You might want to put these functions together in a module that deals with monoalphabetic substitution ciphers.

1. Write a function that returns `True` or `False`, indicating whether a ciphertext is likely to have been encrypted with a monoalphabetic substitution cipher.
2. Write a function that inverts an alphabet key.
3. Write a function that enciphers a plaintext with an alphabet key and returns a ciphertext.
4. Write a function that deciphers a ciphertext with an alphabet key and returns a plaintext. Note that the key is the one used in *encipherment*.

Exercises

1. Use your functions to invert this key alphabet: `KNOFPEUCARBMGXQIZWYSTJVDHL`.
2. Use your functions to encipher the phrase “Sally sells seashells by the seashore” with the key `ATSPWGKHVXDLCJZEMFBRYUONIQ`.
3. Use your functions to decipher the message `PFAFIPGPFIPGKLFTCPFKLHVPGKLEFTPFPPFIZ` with the key `CSKTFVRMGQLEXDHPJIZANBOUWY`.

Unit 13
Atbash cipher

The *atbash cipher* (formerly spelled *athbash*) is an ancient cipher in which the key alphabet is the reverse of the original:

plaintext:	`abcdefghijklmnopqrstuvwxyz`
ciphertext:	`ZYXWVUTSRQPONMLKJIHGFEDCBA`

Notice that the key is *reciprocal*, i.e., it is its own inverse, and therefore the cipher is also *reciprocal* so that encipherment and decipherment are the same process.

To use the atbash cipher, we can treat it like the general monoalphabetic substitution cipher with the key alphabet given above. Another way is to use simple arithmetic. Consider the plaintext as a sequence of symbols $\{p_i\} = p_0\ p_1\ p_2\ p_3\ \ldots$, and the ciphertext as $\{c_i\} = c_0\ c_1\ c_2\ c_3\ \ldots$. These symbols can be treated as numbers, so that ‘A’ = 0, ‘B’ = 1, ..., ‘Z’ = 25 (modern programmers usually count starting with zero). Then the atbash cipher can be implemented with this simple equation:

$$c_i = 25 - p_i$$

Notice that there is almost no security with the atbash cipher. When using it, your only hope of secrecy is your belief that your opponents do not know the cipher.

Another completely insecure cipher related to atbash is the *albam cipher*. It, too, is reciprocal. It uses this key, in which the two halves of the alphabet have been swapped:

plaintext:	`abcdefghijklmnopqrstuvwxyz`
ciphertext:	`NOPQRSTUVWXYZABCDEFGHIJKLM`

Reading and references

Wikipedia: en.wikipedia.org/wiki/Atbash

Crypto Corner: crypto.interactive-maths.com/atbash-cipher.html

Practical Cryptography: practicalcryptography.com/ciphers/atbash-cipher-cipher

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 77-79.

Programming tasks

1. Write functions to encipher and decipher texts with the atbash cipher. Note that they are really the same function. Use either (or both) of the methods discussed above.

Exercises

1. What is the size of the key space for the atbash cipher?

2. Write a short message and encipher it with the atbash cipher. Now decipher it and check that you obtain the original message.

madness's book on classical cryptography
unit 14: modular arithmetic: addition and subtraction
last modified 2020-09-16

Unit 14
Modular arithmetic: addition and subtraction

Consider the integers, $\mathbb{Z}$ = { ..., −3, −2, −1, 0, 1, 2, 3, ... }. Now pick one, say 12, and call it the *modulus*. Next, rename each integer by the remainder when you divide by the modulus. So 0 ⟶ 0, 1 ⟶ 1, 2 ⟶ 2, 3 ⟶ 3, ..., 11 ⟶ 11. But 12 ⟶ 0, since 12 = 1 × 12 + 0, and the remainder is 0. The sequence 0, ..., 11 starts all over again, as 12 ⟶ 0, 13 ⟶ 1, ... The result is a new set $\mathbb{Z}_{12}$ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}, which contains only 12 members. This is called the set of *residues*. We say things like “13 modulo 12 is 1” or “13 = 1 modulo 12” or “13 = 1 mod 12.”

Things get interesting when we start to do arithmetic with the set of residues. For addition, we simply perform the operation as if the numbers are regular integers, but then take the remainder of the result. Continuing with our example of $\mathbb{Z}_{12}$, we can add 7 and 9 to get 16. But 16 is 4 modulo 12. So in $\mathbb{Z}_{12}$, 7 + 9 = 4.

You might think that subtraction works the same way as addition in modular arithmetic, and you would be correct, but it is worthwhile to look at subtraction in a more rigorous way. After all, there are no negative numbers in modular arithmetic. This approach will also help us when we try to understand division. First, let us generalize and take some modulus m without specifying its value. Notice that we have that

$$x + 0 = x$$

for all x in $\mathbb{Z}_m$. Because 0 does not change the value of a number under addition, we call 0 the *additive identity element*. Now, in regular arithmetic, $x - x = x + (-x) = 0$, so we call $-x$ the *additive inverse* of x. In the set of residues, however, we do not have negative numbers, so there is a *positive* number that takes the role of x’s inverse. We might write that number as “$-x$,” but that is just notation for additive inverse. What we call “subtraction” is actually addition using the additive inverse.

How do we find the additive inverse of a number x? Well, we want the number y such that

$$x + y = 0 \bmod m$$

Even though the modulus m itself does not appear in the set of residues, we can calculate y as

$$y = 0 - x = m - x \pmod m$$

In our example of $\mathbb{Z}_{12}$, the inverse of 7 is $12 - 7 = 5$, or $-7 = 5$, so that $7 + 5 = 12 = 0 \pmod{12}$.

Python tips

When dividing integers, the operator `//` gives the quotient, and the operator `%` gives the remainder. Both operators return an integer. For example, `13 // 5` is evaluated as 2, and `13 % 5` is evaluated as 3, since $13 = 2 \times 5 + 3$.

In programming, modular arithmetic is not as complicated as in the text above. Feel free to use the operators `+` and `-` as you normally would, but then modulate the result with the `%` operator.

```
answer = (7 + 9) % 12
answer = (7 - 9) % 12
```

Programming tasks

1. Go outside and take a walk. If it's raining, open a window so you can hear it while you play a game with someone else in your house.

Exercises

1. Check carefully your work on programming task #1 and re-do it if necessary.
2. Evaluate:
   a. 528147 + 790378 modulo 62
   b. 72177 − 162737 mod 81
3.
   a. Find an element of $\mathbb{Z}_{18}$ that is its own (additive) inverse.
   b. Find an element of $\mathbb{Z}_{19}$ that is its own inverse.
   c. What condition must we put on n such that $\mathbb{Z}_n$ has such an element?

Unit 15
Caesar shift cipher

The *Caesar shift cipher*, also known simply as the *Caesar cipher* or the *additive cipher*, can be viewed in two ways. Given its key *k*, which is a number from 0 to 25, we can think of the Caesar cipher as a monoalphabetic substitution in which the ciphertext alphabet is shifted left by *k* letters, with wrap-around. For example, if the key is 5, we have

plaintext:	`abcdefghijklmnopqrstuvwxyz`
ciphertext:	`FGHIJKLMNOPQRSTUVWXYZABCDE`

Another way to view the Caesar cipher is with modular arithmetic. The plaintext and ciphertext are equivalent to sequences of numbers $\{p_i\}$ and $\{c_i\}$. Encipherment is done with addition:

$$c_i = p_i + k \bmod 26$$

Decipherment is done with subtraction:

$$p_i = c_i - k \bmod 26$$

When the shift is 13, the cipher is often called *ROT13*. This shift has a special name because 13 is half of 26. Furthermore, since 13 is its own inverse modulo 26, this cipher is reciprocal, i.e., encipherment is the same process as decipherment. Also notice that ROT13 is the same as the albam cipher.

Reading and references

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 83-84.

Wikipedia: en.wikipedia.org/wiki/Caesar_cipher

Crypto Corner: crypto.interactive-maths.com/caesar-shift-cipher.html

Practical Cryptography: practicalcryptography.com/ciphers/caesar-cipher

Programming tasks

1. Write a function to generate the key alphabet for a Caesar cipher from the key number. You might not use this function if you work with modular arithmetic, but it is good practice to write the function.

2. Write a function to encipher a plaintext with the Caesar cipher and a given key. Use whatever method you like.

3. Write a function to decipher a ciphertext with the Caesar cipher and a given key. Use whatever method you like.

Exercises

1. What is the size of the key space for the Caesar cipher?

2. Encipher this text with the key 11:

   > A horse is a horse, of course of course, and no one can talk to a horse, of course, unless, of course, that horse of course is the famous Mister Ed.

3. Decipher this text with the key 15:

   ```
   WTLPHDCRTPAXIIATVGTTCQPAADURAPNVJBQNQJINDJHWDJASHTT
   LWPIWTRPCSDIDSPNVJBQNWTRPCLPAZXCIDBPCNQDDZHLXIWWXHE
   DCTNEPAEDZTNIDDPCSXUNDJWPKTPWTPGIIWTCVJBQNHPEPGIDUN
   DJVJBQN
   ```

madness's book on classical cryptography
unit 16: brute-force attack on the caesar cipher
last modified 2020-08-21

Unit 16
Brute-force attack on the Caesar cipher

A *brute-force attack* involves trying every possible key until an acceptable decryption is found. Since the Caesar cipher has only 26 possible keys (even when we include the identity operation [that leaves the plaintext unaltered]), a brute-force attack takes very little time.

Programming tasks

1. Write a function that does a brute-force attack on a ciphertext that was encrypted with the Caesar cipher. Use your tetragram-fitness function to find the best possible decryption. Your function should return the key or the plaintext or both. Also write a wrapper around your function that accepts a ciphertext and prints the plaintext or writes it to a file.

Exercises

1. Perform a brute-force attack on this ciphertext:

   ```
   WLALYWPWLYWPJRLKHWLJRVMWPJRSLKWLWWLYZ
   ```

   Remember this text; we will see it again.

2. Perform a brute-force attack on this ciphertext:

   ```
   KLYJKPMCLDUFXAESCZFRSSZZADLEESPKZZ
   ```

   Remember this text; we will see it again.

madness's book on classical cryptography
unit 17: attacking the caesar cipher with cribs
last modified 2020-07-04

Unit 17
Attacking the Caesar cipher with cribs

A *crib* is a word or phrase that you have good reason to believe is contained in the plaintext. To use a crib to break a ciphertext that was encrypted with the Caesar cipher, we try to fit the crib in all possible positions in the ciphertext. For each position, we find the shift needed to decrypt that portion of the ciphertext to match each letter of the crib. If each letter requires the same shift, then we have a good candidate for the key. It is not guaranteed to be the true key, but if we find several candidates, then we can decipher the text with each and use tetragram fitness to pick out the best plaintext.

Let us look at a short example. Here is a piece of encrypted text that we promised you would see again (and you will see it again at least once more):

```
WLALYWPWLYWPJRLKHWLJRVMWPJRSLKWLWWLYZ
```

Let's try to break it with the crib `PIPER`. We line up the crib with the ciphertext at each position until we find one at which all the shifts are equal:

```
ciphertext:  W L A L Y W P W L Y W P J R ...

crib:        P I P E R _ _ _ _ _ _ _ _ _ ...
shifts:      7 3 11 7 7

crib:        _ P I P E R _ _ _ _ _ _ _ _ ...
shifts:        22 18 22 20 5

⋮

crib:        _ _ _ _ _ P I P E R _ _ _ _ ...
shifts:              7 7 7 7 7
```

We now have a good candidate for the key: 7. If we decipher with that key, we find that the plaintext looks like English, and we are satisfied that we have found the correct key.

Python tips

To take a “slice” of a string (a substring), we use indexing. So, for example, to make a substring that takes the fifth through tenth character:

```
newString = oldString[4:10]
```

Remember that we count from zero, so the fifth character has index 4. The last character that we want to include has index 9, but in Python we have to add one to the index when we specify the last place in a string or array. To duplicate a string (or an array), we also use indexing, but leave the indices out:

```
newString = oldString[:]
```

Programming tasks

1. Write a function or script that takes a ciphertext and a crib and breaks the ciphertext.

Exercises

1. Break this ciphertext with the crib `CUSTOM`.

   ```
   RCCXRLCZJUZMZUVUZEKFKYIVVGRIKJKYVWZIJKZJZEYRSZKVU
   SPSVCXZREJKYVJVTFEUSPKYVRHLZKREZREJNYFTRCCKYVDJVC
   MVJTVCKJREUKYVKYZIUSPKYVXRLCJKYVJVUZWWVIWIFDFEVRE
   FKYVIZECREXLRXVTLJKFDJREUCRNJKYVIZMVIXRIFEEVJVGRI
   RKVJKYVXRLCJWIFDKYVTVCKJKYVIZMVIJDRIEVREUJZVEVJVG
   RIRKVKYVSVCXZREJWIFDKYVFKYVIJ
   ```

2. Break this ciphertext. It may or may not include some of these words: `VICTORY SPAIN DISCO EUROVISION`.

   ```
   MFFTUEFUYQOMQEMDIMEQXQOFQPRADFTQRAGDFTFUYQMEPUOFM
   FADMZPIMEOAYUZSFAEBMUZFARUZUETFTQIMDAZFTQIMKTQIME
   YQFNKMYNMEEMPADERDAYOADPANMITATMPPQEQDFQPSQZQDMXB
   AYBQKFTQKUZRADYQPTUYFTMFUFIAGXPNQQMEUQEFFAFMWQFTQ
   OUFKMFZUSTFNQOMGEQFTQQZQYKTMPNKFTQZZAWZAIXQPSQARP
   UEOAADARFTQZUSTFXURQADARFTQNAASUQ
   ```

Unit 18 (optional)
Attacking the Caesar cipher with monogram frequencies (χ^2)

Here is an attack on the Caesar cipher that uses only monogram frequencies. The technique is to make a table of the monogram frequencies from the ciphertext, then to shift that frequency table until it resembles that of typical English. In this unit, we will use monogram fitness based on the the χ^2 statistic to measure how well the frequencies match.

Let's work through an example. Consider this ciphertext:

```
LIKHKDGDQBWKLQJFRQILGHQWLDOWRVDBKHZURWHLWLQFLSKHUWKDWLVE
BVRFKDQJLQJWKHRUGHURIWKHOHWWHUVRIWKHDOSKDEHWWKDWQRWDZRUG
FRXOGEHPDGHRXWLIDQBRQHZLVKHVWRGHFLSKHUWKHVHDQGJHWDWWKHLU
PHDQLQJKHPXVWVXEVWLWXWHWKHIRXUWKOHWWHURIWKHDOSKDEHWQDPHO
BGIRUDDQGVRZLWKWKHRWKHUV
```

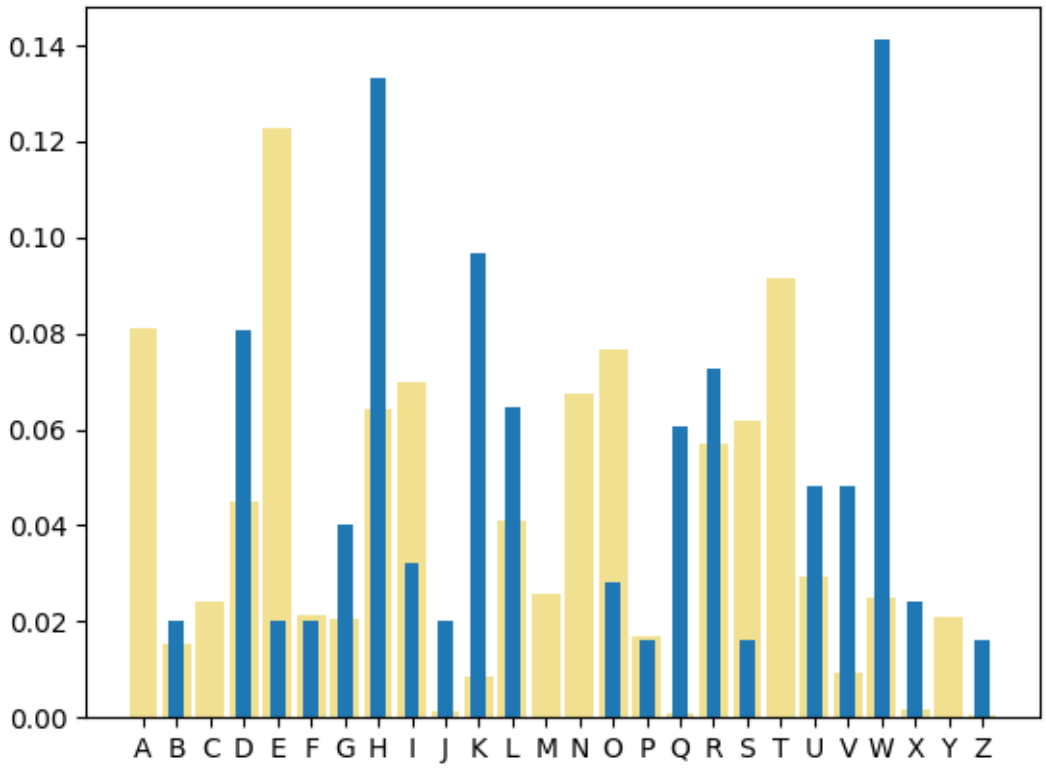
The frequencies of letters in this ciphertext are

A	0	J	0.020	S	0.016
B	0.020	K	0.097	T	0
C	0	L	0.065	U	0.048
D	0.081	M	0	V	0.048
E	0.020	N	0	W	0.141
F	0.020	O	0.028	X	0.024
G	0.040	P	0.016	Y	0
H	0.133	Q	0.060	Z	0.016
I	0.032	R	0.073		

Let's write the table as a list:

0.000, 0.020, 0.000, 0.081, 0.020, 0.020, 0.040, 0.133, 0.032, 0.020, 0.064, 0.065, 0.000, 0.000, 0.028, 0.016, 0.060, 0.073, 0.016, 0.000, 0.048, 0.048, 0.141, 0.024, 0.000, 0.016

Here is a graph of those frequencies (in **blue**), along side typical English frequencies (in yellow):



We can already see that a shift of the blue bars left by three will give a good match. But let's use the χ^2 statistic and see how that works. Our monogram table for English looks like this (yours may be slightly different) (rounded to three decimal places):

0.081, 0.015, 0.024, 0.045, 0.123, 0.021, 0.021, 0.064, 0.070, 0.001, 0.009, 0.041, 0.016,
0.068, 0.077, 0.017, 0.001, 0.057, 0.062, 0.091, 0.029, 0.009, 0.025, 0.002, 0.021, 0.001

Now, if we shift the frequencies from the ciphertext leftward by one, we get

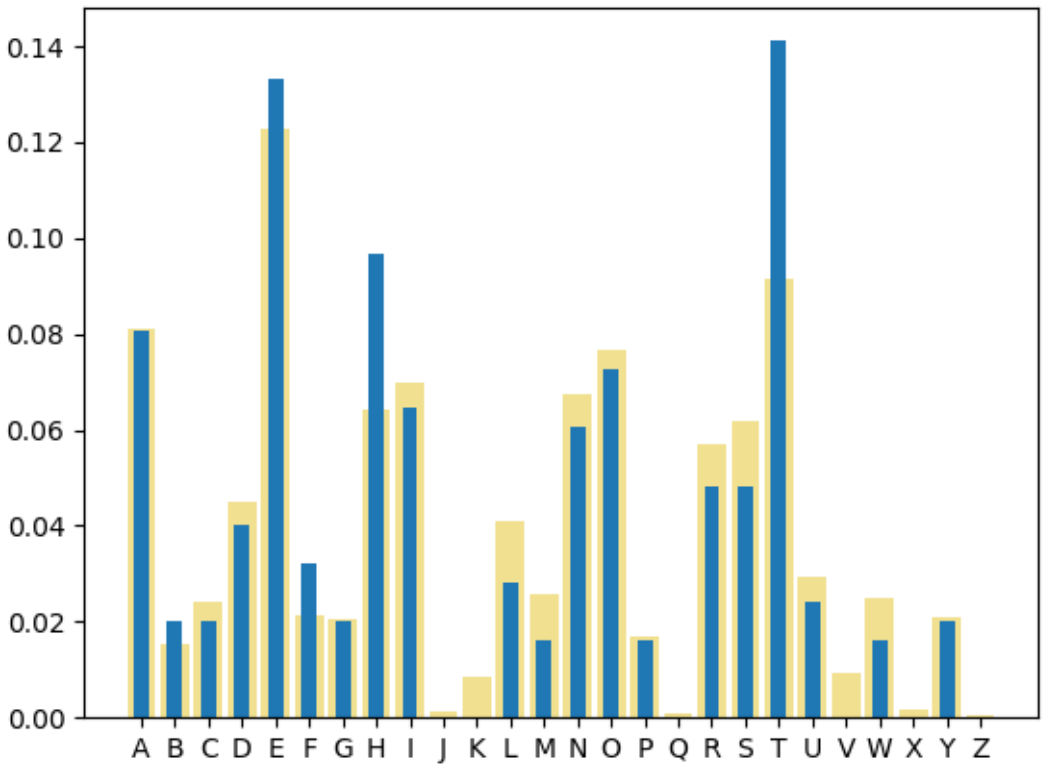
0.020, 0.000, 0.081, 0.020, 0.020, 0.040, 0.133, 0.032, 0.020, 0.064, 0.065, 0.000, 0.000,
0.028, 0.016, 0.060, 0.073, 0.016, 0.000, 0.048, 0.048, 0.141, 0.024, 0.000, 0.016, 0.000

The χ^2 between this and the English table is 18.02, indicating a poor fit. We can continue in this manner, and for each shift we obtain these values for χ^2:

shift	χ^2
0	6.51
1	18.02
2	6.73
3	0.10
4	9.81
5	5.17
6	25.14
7	7.67
8	24.11

9	3.91
10	10.27
11	13.92
12	10.99
13	31.47
14	3.91
15	2.90
16	3.91
17	22.50
18	7.88
19	3.83
20	19.38
21	9.52
22	4.00
23	24.56
24	21.31
25	11.63

The best fit (lowest χ^2) is for a shift of three. We conclude that the key is 3. To get a feel for what a good fit looks like, here is a graph of the shifted frequency table compared to typical English:



Programming tasks

1. Write a function or script that implements the attack. Make sure that you shift the table in the correct direction; generate some ciphertexts and test it to be certain.

Exercises

1. Finish decrypting the ciphertext in the example above.
2. Apply the attack to this ciphertext:

```
QEBZXBPXOZFMEBOFPKXJBAXCQBOGRIFRPZXBPXOTELXZZLOAFKDQLP
RBQLKFRPRPBAFQTFQEXPEFCQLCQEOBBQLMOLQBZQJBPPXDBPLCJFIF
QXOVPFDKFCFZXKZBTEFIBZXBPXOPTXPQEBCFOPQOBZLOABARPBLCQE
FPPZEBJBLQEBOPRYPQFQRQFLKZFMEBOPXOBHKLTKQLEXSBYBBKRPBA
BXOIFBOFQFPRKHKLTKELTBCCBZQFSBQEBZXBPXOZFMEBOTXPXQQEBQ
FJBYRQFQFPIFHBIVQLEXSBYBBKOBXPLKXYIVPBZROBKLQIBXPQYBZX
RPBJLPQLCZXBPXOPBKBJFBPTLRIAEXSBYBBKFIIFQBOXQB
```

3. Apply the attack to this ciphertext, which we have seen before:

```
WLALYWPWLYWPJRLKHWLJRVMWPJRSLKWLWWLYZ
```

   You should find that the attack fails. Why did it fail?

4. Apply the attack to this ciphertext, which we have seen before:

```
KLYJKPMCLDUFXAESCZFRSSZZADLEESPKZZ.
```

   You should find that the attack fails. Why did it fail? Now you should understand why we do not recommend using the χ^2 statistic for cracking ciphertexts.

Unit 19
Attacking the Caesar cipher with monogram frequencies

Here is an attack on the Caesar cipher that uses only monogram frequencies. The technique is to make a table of the monogram frequencies from the ciphertext, then to shift that frequency table until it resembles that of typical English. In this unit, we will use monogram fitness based on the cosine of the angle between vectors to measure how well the frequencies match.

Let’s work through an example. Consider this ciphertext:

```
LIKHKDGDQBWKLQJFRQILGHQWLDOWRVDBKHZURWHLWLQFLSKHUWKDWLVE
BVRFKDQJLQJWKHRUGHURIWKHOHWWHUVRIWKHDOSKDEHWWKDWQRWDZRUG
FRXOGEHPDGHRXWLIDQBRQHZLVKHVWRGHFLSKHUWKHVHDQGJHWDWWKHLU
PHDQLQJKHPXVWVXEVWLWXWHWKHIRXUWKOHWWHURIWKHDOSKDEHWQDPHO
BGIRUDDQGVRZLWKWKHRWKHUV
```

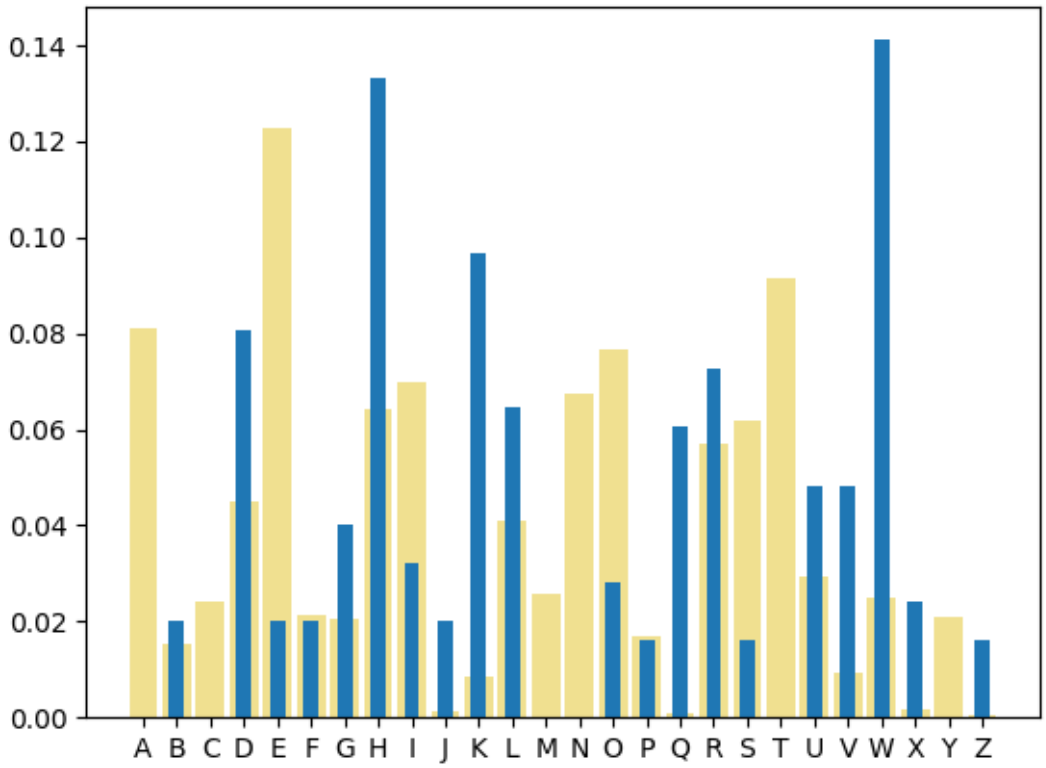
The frequencies of letters in this ciphertext are

A	0	J	0.020	S	0.016
B	0.020	K	0.097	T	0
C	0	L	0.065	U	0.048
D	0.081	M	0	V	0.048
E	0.020	N	0	W	0.141
F	0.020	O	0.028	X	0.024
G	0.040	P	0.016	Y	0
H	0.133	Q	0.060	Z	0.016
I	0.032	R	0.073		

Let’s write the table as a 26-dimensional vector:

(0.000, 0.020, 0.000, 0.081, 0.020, 0.020, 0.040, 0.133, 0.032, 0.020, 0.064, 0.065, 0.000, 0.000, 0.028, 0.016, 0.060, 0.073, 0.016, 0.000, 0.048, 0.048, 0.141, 0.024, 0.000, 0.016)

Here is a graph of those frequencies (in **blue**), along side typical English frequencies (in yellow):



We can already see that a shift of the blue bars left by three will give a good match. But let's use the monogram fitness and see how that works. Our monogram table for English looks like this (yours may be slightly different) (rounded to three decimal places):

(0.081, 0.015, 0.024, 0.045, 0.123, 0.021, 0.021, 0.064, 0.070, 0.001, 0.009, 0.041, 0.016,
0.068, 0.077, 0.017, 0.001, 0.057, 0.062, 0.091, 0.029, 0.009, 0.025, 0.002, 0.021, 0.001)

Now, if we shift the frequencies from the ciphertext leftward by one, we get

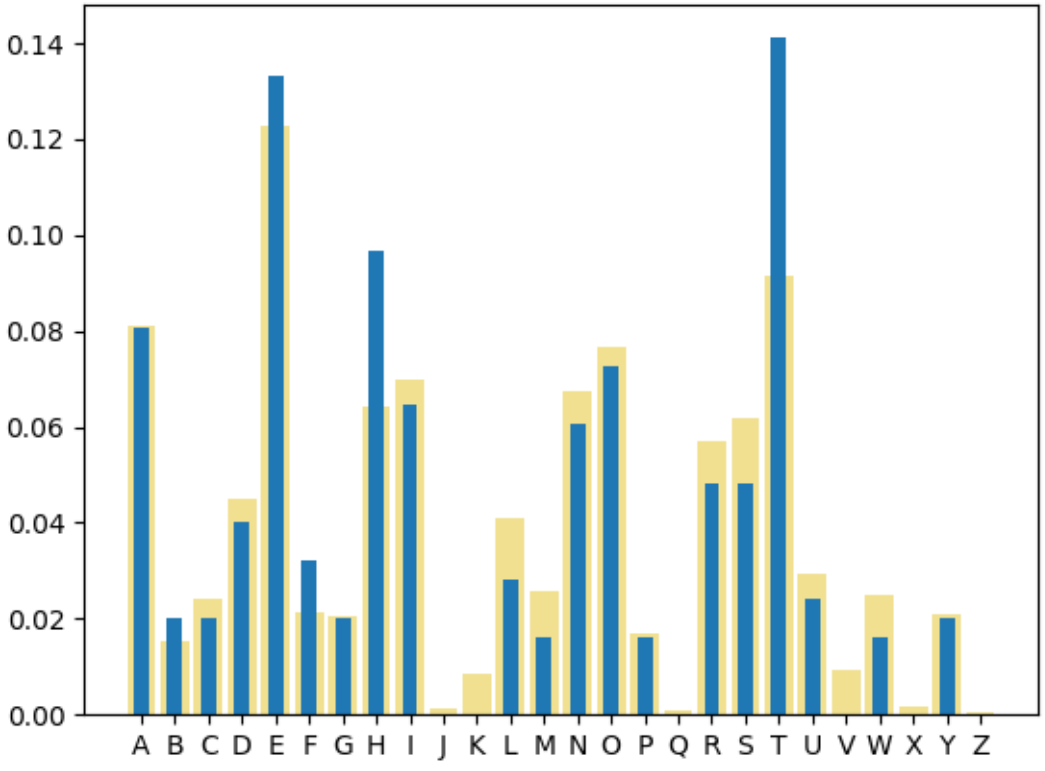
(0.020, 0.000, 0.081, 0.020, 0.020, 0.040, 0.133, 0.032, 0.020, 0.064, 0.065, 0.000, 0.000,
0.028, 0.016, 0.060, 0.073, 0.016, 0.000, 0.048, 0.048, 0.141, 0.024, 0.000, 0.016, 0.000)

The cos θ between this vector and the English table is 0.393, indicating a poor fit. We can continue in this manner, and for each shift we obtain these values for cos θ:

shift	cos θ
0	0.493
1	0.393
2	0.544
3	0.970
4	0.593
5	0.443
6	0.485
7	0.589
8	0.493

9	0.512
10	0.547
11	0.452
12	0.434
13	0.478
14	0.648
15	0.580
16	0.584
17	0.550
18	0.693
19	0.569
20	0.443
21	0.435
22	0.608
23	0.499
24	0.444
25	0.600

The best fit (largest cos θ) is for a shift of three. We conclude that the key is 3. To get a feel for what a good fit looks like, here is a graph of the shifted frequency table compared to typical English:



Programming tasks

1. Write a function or script that implements the attack. Make sure that you shift the table in the correct direction; generate some ciphertexts and test it to be certain.

Exercises

1. Finish decrypting the ciphertext in the example above.
2. Apply the attack to this ciphertext:

```
QEBZXBPXOZFMEBOFPKXJBAXCQBOGRIFRPZXBPXOTELXZZLOAFKDQLP
RBQLKFRPRPBAFQTFQEXPEFCQLCQEOBBQLMOLQBZQJBPPXDBPLCJFIF
QXOVPFDKFCFZXKZBTEFIBZXBPXOPTXPQEBCFOPQOBZLOABARPBLCQE
FPPZEBJBLQEBOPRYPQFQRQFLKZFMEBOPXOBHKLTKQLEXSBYBBKRPBA
BXOIFBOFQFPRKHKLTKELTBCCBZQFSBQEBZXBPXOZFMEBOTXPXQQEBQ
FJBYRQFQFPIFHBIVQLEXSBYBBKOBXPLKXYIVPBZROBKLQIBXPQYBZX
RPBJLPQLCZXBPXOPBKBJFBPTLRIAEXSBYBBKFIIFQBOXQB
```

3. Apply the attack to this ciphertext, which we have seen before:

```
WLALYWPWLYWPJRLKHWLJRVMWPJRSLKWLWWLYZ
```

   You should find that the attack fails. Why did it fail? When we used the brute-force attack, we succeeded. From this we learn that using monogram fitness is useful for longer ciphertexts, but not very reliable for shorter texts. Nevertheless, this attack is necessary if you have a ciphertext that has been encrypted with both a Caesar cipher and a transposition cipher, which rearranges the letters of the text and thereby makes it impossible to use tetragram fitness.

4. Apply the attack to this ciphertext, which we have seen before:

```
KLYJKPMCLDUFXAESCZFRSSZZADLEESPKZZ.
```

   You should find that the attack succeeds. When we used the monogram fitness based on the χ^2 statistic, we failed to decrypt this ciphertext.

Unit 20
Greatest common divisor

The *greatest common divisor* (gcd) of two integers is the largest integer that divides them both evenly (without remainder).

One way to find the gcd of two integer is to write out the prime-number factorization of each and select the largest set of factors that is contained in both. For example,

$$84 = 2 \times 2 \times 3 \times 7$$
$$360 = 2 \times 2 \times 2 \times 3 \times 3 \times 5$$

The largest subset contained in both factorizations is {2, 2, 3}, so the gcd of 84 and 360 is $2 \times 2 \times 3 = 12$.

Another way to find the gcd is with Euclid's algorithm, which is easy to implement in a script. Here is how the algorithm works for two integers *m* and *n*:

1. while *n* is not 0
    a. set *m* equal to *m* modulo *n*
    b. swap *m* and *n*
2. once *n* is 0, output *m*

The *least common multiple* (lcm) of two integers is the smallest number that is a multiple of both of them. We can find the lcm from the prime-number factorizations as well, by selecting the smallest set of factors that is a superset of each factorizations. For our example of 84 and 360, we see that we need three 2's to accommodate the 2's in 360, two 3's, one 5 and one 7. So the lcm of 84 and 360 is $2 \times 2 \times 2 \times 3 \times 3 \times 5 \times 7 = 2520$. An interesting fact is that

$$\gcd(m, n) \times \text{lcm}(m, n) = m \times n$$

We say that two numbers are *coprime* if their gcd is one. In this case, the two numbers have no factors in common, and neither one evenly divides the other.

Python tips

Python has a built-in limit on the number of levels of recursion. If you are working with large numbers, and trying to find a gcd, then you could exceed this limit and get an error. (Recursion is when a function calls itself. When some condition becomes true, the function exits all the way back to the main control flow.) We recommend that you implement Euclid's algorithm without recursion.

Python allows variables to hold boolean values. These values are either `True` or `False`. Furthermore, functions can return boolean values. Notice that the following two short code blocks are equivalent; both return `True` if the value of `x` is one and `False` otherwise.

code block 1:

```
if x == 1:
      return True
else:
      return False
```

code block 2:

```
return (x == 1)
```

Reading and references

Wikipedia

en.wikipedia.org/wiki/Greatest_common_divisor
en.wikipedia.org/wiki/Euclidean_algorithm
en.wikipedia.org/wiki/Least_common_multiple

Programming tasks

1. Write a function that calculates the gcd of two integers. Use whatever method you prefer.
2. Write a function that finds the lcm of two integers. Feel free to use your function for the gcd.
3. Write a function that returns a boolean value that indicates whether two integers are coprime. This can be a very short function.

Exercises

1. Randomly generate a few pairs of positive integers. For each pair, find the gcd and lcm. Verify that the product of the gcd and lcm equals the product of the pair.

madness's book on classical cryptography
unit 21: modular arithmetic: multiplication and division
last modified 2020-07-04

Unit 21
Modular arithmetic: multiplication and division

Let's return to our example of $\mathbb{Z}_{12} = \{0, 1, ..., 11\}$, and construct a multiplication table. Remember that what we need to do is find the product of two members of the set, and then find the remainder when we divide the product by the modulus 12.

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11
2	0	2	4	6	8	10	0	2	4	6	8	10
3	0	3	6	9	0	3	6	9	0	3	6	9
4	0	4	8	0	4	8	0	4	8	0	4	8
5	0	5	10	3	8	1	6	11	4	9	2	7
6	0	6	0	6	0	6	0	6	0	6	0	6
7	0	7	2	9	4	11	6	1	8	3	10	5
8	0	8	4	0	8	4	0	8	4	0	8	4
9	0	9	6	3	0	9	6	3	0	9	6	3
10	0	10	8	6	4	2	0	10	8	6	4	2
11	0	11	10	9	8	7	6	5	4	3	2	1

That wasn't so hard. But what about division? Fractions do not exist in $\mathbb{Z}_{12}$, so what are we to do? In the same way that we defined the additive inverse, we must now do the same for multiplication. The *multiplicative inverse* of some number x is another number y such that $x \cdot y = 1$. The *multiplicative identity element* is 1. We write "x^{-1}" for the inverse of x, but we do not intend that it should be interpreted as "1 over x" or "1 divided by x." From this point of view, division is actually multiplication by an inverse.

If we look at the multiplication table for $\mathbb{Z}_{12}$ we can see, for example, that $5 \times 5 = 1$, so the inverse of 5 is 5 itself. But what about the inverse of 3? There is no number y such that $3 \cdot y = 1$. In this case, we are forced to say that 3 does not have an inverse. This means that we cannot divide by three in $\mathbb{Z}_{12}$. But what if we try to do it anyway? Notice that $3 \times 3 = 9$, and $3 \times 7 = 9$, and $3 \times 11 = 9$. Then if we

want to find 9 / 3, we have to face the fact that there are three possible answers. Since the quotient is not well defined, it cannot exist.

In general, an element x of $\mathbb{Z}_m$ has an inverse if and only if gcd (x, m) = 1. This means that if m is prime, all elements of $\mathbb{Z}_m$ are invertible.

To find the inverse of an element in $\mathbb{Z}_m$ (or to determine whether it exists), Euclid comes to the rescue again. His *extended Euclidean algorithm* for finding the inverse of x in $\mathbb{Z}_m$ is presented here:

1. set $t = 0$, $t' = 1$, $r = m$, $r' = x$
2. while r' is nonzero
    a. set $q = r / r'$ (discard the remainder)
    b. set new values of t and t' to the current values of t' and $t - q \cdot t'$ (respectively)
    c. set new values of r and r' to the current values of r' and $r - q \cdot r'$ (respectively)
3. if r is nonzero, then x is not invertible; exit
4. if $t < 0$, add m to t
5. t is the inverse of x

Python tips

Python allows parallel assignment. For example, to set `x` to 1 and `y` to old value of `x`, this one statement suffices:

```
x, y = 1, x
```

This technique should make implementing steps 2b and 2c in Euclid's extended algorithm easier.

Reading and references

en.wikipedia.org/wiki/Extended_Euclidean_algorithm

Programming tasks

1. Write a function that finds the multiplicative inverse (if it exists) of an integer with a given modulus. If the inverse does not exist, it should have a way of telling the program that called the function that it does not; one way is to return the boolean value `False`.

Exercises

1. Randomly generate some integer pairs such that the first of each pair is smaller than the second. For each pair find the multiplicative inverse of the first number modulo the second.

Unit 22
Affine cipher

The *affine cipher* extends the Caesar cipher to include the use of modular multiplication. When we treat the characters of the plaintext $\{p_i\}$ and ciphertext $\{c_i\}$ as integers modulo 26, then encipherment under the affine cipher is done with this equation:

$$c_i = a\, p_i + b \bmod 26$$

Decipherment is accomplished by solving the above equation for p_i:

$$p_i = a^{-1} (c_i - b) \bmod 26$$

where the key consists of the multiplier a and the shift b. Note that a must invertible, so gcd (a, 26) must be equal to one.

Another way to handle the affine cipher is to construct a key alphabet and treat the cipher as a monoalphabetic substitution (which it *is*). It's not complicated. Start with 'A' and take every a^{th} letter; then shift (with rollover) so that the letter whose number is b is at the beginning. For example, if $a = 3$ and $b = 5$, write down every third letter, starting with 'A':

```
ADGJMPSVYBEHKNQTWZCFILORUX
```

Then, shift so that the fifth letter (starting from zero) of the normal alphabet is at the beginning:

```
FILORUXADGJMPSVYBEHKNQTWZC
```

There are some special cases of the affine cipher:

- $a = 1$, $b = 0$: the identity operation
- $a = 1$: Caesar shift cipher with key = b
- $a = 25$, $b = 25$: atbash cipher
- $b = 0$: this is called the *multiplicative cipher*

Reading and references

en.wikipedia.org/wiki/Affine_cipher

practicalcryptography.com/ciphers/affine-cipher

crypto.interactive-maths.com/affine-cipher.html

Programming tasks

1. Write a function that verifies a key for the affine cipher and returns a boolean value indicating whether the key is valid or invalid.
2. Write a function that takes a key for an affine cipher and returns an equivalent alphabet key for a monoalphabetic substitution. This function should first call the function from the previous task to verify the key before proceeding.
3. Write a function that enciphers a plaintext with the affine cipher and a given key. Use whatever method you prefer. You should verify the key before proceeding.
4. Write a function that deciphers a ciphertext with the affine cipher and a given key. Use whatever method you prefer. You should verify the key before proceeding.

Exercises

1. How large is the key space for the affine cipher? Include only valid keys for which the multiplier is invertible. Include the identity operation (the key that leaves the plaintext unchanged under encipherment).
2. Encipher this text with multiplier 11 and shift 9:

   > An affine transformation is an automorphism of an affine space which preserves both the dimension of any affine subspaces and the ratios of the lengths of parallel line segments. Consequently, sets of parallel affine subspaces remain parallel after an affine transformation. An affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line.

3. Decipher this text with multiplier 15 and shift 3:

   ```
   TSLCZFRCEFRPECCETNUMDTQCLKCVFRMWTQGFMGLNFBLBDCELBDCTHDMQFQNLQ
   NLCEDCQFFQLRQWLYNCDQWNTQNCLDWELYLTNDIFXLVEZWTWCELHETHXLQHYFNN
   CELBFLSTRNNCYTUCFPLCCFCELNDBLNTWL
   ```

4. If we extend the alphabet with a space character so that it now contains 27 letters, how many valid keys are there?

madness's book on classical cryptography
unit 23: brute-force attack on the affine cipher
last modified 2020-07-04

Unit 23
Brute-force attack on the affine cipher

The brute-force attack on the affine cipher tries all possible keys and chooses the deciphered plaintext with the best textual fitness.

Programming tasks

1. Implement the attack. Use tetragram fitness. Be sure to try only valid keys.

Exercises

1. Break this ciphertext:

```
HAGDGKDJKQNDFICEIVZOSABBNWIJIDIBWIDIBWSTITIDWTABD
JKPDNIDGDIJDWZTJSYDNKGDJSPKCPSJDIFSIJZDNKGDKVOGNK
PDNWYIDWUIGIYKQNDOGIKBKVYIVDNWGEKPPWJFJIXWIVZGAJW
TKXWPIGGWVQWJGGWDGIKBDNIDZIOTSJIDNJWWNSAJDSAJIDNJ
WWNSAJDSAJDNWUWIDNWJGDIJDWZQWDDKVQJSAQNDNWDKVOGNK
PUIGDSGGWZKTVSDTSJDNWCSAJIQWSTDNWTWIJBWGGCJWUDNWY
KVVSUUSABZFWBSGDDNWYKVVSUUSABZFWBSGDDNWGNKPGWDQJS
AVZSVDNWGNSJWSTDNKGAVCNIJDWZZWGWJDKGBWUKDNQKBBKQI
VDNWGEKPPWJDSSDNWYKBBKSVIKJWIVZNKGUKTWDNWYSXKWGDI
JDNWPJSTWGGSJIVZYIJOIVVNWJWSVQKBBKQIVGKGBW
```

madness's book on classical cryptography
unit 24: attacking the affine cipher with cribs
last modified 2020-07-22

Unit 24
Attacking the affine cipher with cribs

If we can match two letters from the ciphertext with two letters from the plaintext, then it is possible to solve for the multiplier and shift of the affine cipher. If all of the letters in a crib can be found from a section of the ciphertext with the same multiplier and shift, then we have found a candidate key for the cipher. This resembles the attack on the Caesar cipher, but whereas the Caesar cipher has only one parameter, the affine cipher has two.

Let's run through a short example to see how the algebra works. Suppose our crib is `CRIB`, and that we are matching it to the sequence `KFIT` in the ciphertext. First, we replace the letters with their numbers, where we begin the alphabet at 'A' = 0.

`C`	→	2	`K`	→	10
`R`	→	17	`F`	→	5
`I`	→	8	`I`	→	8
`B`	→	1	`T`	→	19

These give us four equations involving the parameters a and b of the affine cipher's key:

$$\begin{aligned} 2a + b &= 10 \mod 26 \\ 17a + b &= 5 \mod 26 \\ 8a + b &= 8 \mod 26 \\ a + b &= 19 \mod 26 \end{aligned}$$

Suppose we subtract the first equation from the second to get

$$15a = -5 = 21 \mod 26$$

The multiplicative inverse of 15 modulo 26 is 7, so we multiply both sides by 7:

$$\begin{aligned} 7 \cdot 15a &= 7 \cdot 21 \mod 26 \\ a &= 17 \end{aligned}$$

Then, from the last of our four original equations, $b = 2$. If we encipher `CRIB` with this key, we do indeed get `KFIT`, so we have a good candidate for the cipher's key. Finally, notice that if we had subtracted the first equation from the third, we would have found

$$6a = -2 = 24 \mod 26$$

but we cannot remove the coefficient of a because 6 is not invertible modulo 26.

Programming tasks

1. Write a function or script that attacks a ciphertext encrypted with an affine cipher by using a crib. Be careful that you only try to remove coefficients that are invertible.

Exercises

1. Try your program on this ciphertext. A good crib is `CRIB`.

```
OYFSTGLYYRSBXPTCLLIRSBSLZANYSGYNXXPFYXTONWRTVYRAYLDJRYLQ
OWLBLSOCLSLTTFSQIYLVRTRNSNGFSLUICNTRELNYQSFSVLQRTINTFCOL
VWSRVRFSRGRGFRCLQWLZNJCQZFHLJIZNJCQSOBNYRBWOAFVHONTCLLIF
SQNGOLSIYNOLTOLQCNJQCP
```

Unit 25
Attacking the affine cipher with monogram frequencies

This attack is very fast, but requires a ciphertext that is lengthy enough to do the required statistics. It is also a necessary attack if we are confronted with a ciphertext that was encrypted with an affine cipher and a transposition cipher (which changes positions of characters so that tetragram fitness is useless), since it can break the affine cipher so that we can then break the transposition cipher.

The method of attack begins by tabulating the monogram frequencies in the ciphertext. They are then shuffled in the same way that the key alphabet is generated in the affine cipher for given multiplier and shift. The multiplier and shift that give the best match to English frequencies is taken as the cipher's key. Armed with the key, we can then decipher the text.

The shuffling of the frequencies can be done by replacing the n^{th} entry in the table with the $(an + b)^{\text{th}}$ entry (modulo 26), where a and b are the multiplier and shift of the affine cipher.

Programming tasks

1. Implement the attack.

Exercises

1. Use your implementation to break this ciphertext:

   ```
   ZTYWFFAJYKAXTYLASNYLIKQJSZLWAJYRWJRZTYLYFQLYYWSIZQDLYWO
   UAZTSZWZASZAKSYNYJUAZTWZYBZZTASSTQLZ
   ```

2. The following ciphertext was encrypted with an affine cipher and a transposition cipher in which each block of three letters has been reversed. Use your implementation of the attack to break the affine cipher, then reverse each three-letter block to reveal the plaintext.

   ```
   MNRVVCMPWHWUZMNJIWKZMPYUZRIPWCCDMRDPZMNYVMMMZKICLYRCMZ
   MSOKMZMAYMHBWRUCEPVWDZMVRPMCXHRPWRBM
   ```

madness's book on classical cryptography
unit 26: keyword substitution cipher
last modified 2020-10-09

Unit 26
Keyword substitution cipher

The *keyword substitution cipher*, also called the *keyed substitution cipher*, or simple the *keyword cipher*, is a monoalphabetic substitution cipher in which the key alphabet is constructed from a keyword. The keyword is placed at the beginning of the key, its repeated letters are removed, and then the remainder of English letters are added to the key. The three most common ways of filling the key are these:

- Add the remaining letters in alphabetical order. For example, if the keyword is `AUTOMOBILE`, then the key alphabet is (remember to drop the repeated ‘O’)

  `AUTOMBILECDFGHJKNPQRSVWXYZ`

- Start adding letters from the alphabetically next after the last letter of the keyword. For the keyword AUTOMOBILE, we start with the next letter after ‘E,’ which is ‘F.’ When we reach ‘Z’ we place the remaining missing letters ‘C’ and ‘D.’

  `AUTOMBILEFGHJKNPQRSVWXYZCD`

- Start adding letters from the next letter after the alphabetically last letter of the keyword. The keyword `AUTOMOBILE` has ‘U’ as its alphabetically last letter. So we fill in starting with ‘V.’

  `AUTOMBILEVWXYZCDFGHJKNPQRS`

Here are some other variations. They can be used alone or in combinations.

- Fill in the remaining letters after the keyword in reverse alphabetical order.
- Put the keyword at the end of the key.
- Use a keyword for the plaintext alphabet rather than the ciphertext alphabet.
- Use keywords for both the plaintext and ciphertext alphabets.

Reading and references

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 103-104.

Programming tasks

1. Write a function to generate an alphabet key from a keyword. It should be able to do so in the three most common ways, and there should be some way to tell it which method to use.

2. Write a function to encipher a plaintext when given a keyword and a key-filling method.

3. Write a function to decipher a ciphertext when given a keyword and a key-filling method.

4. Write a function to generate an alphabet key if the keyword is used in the plaintext alphabet rather than the ciphertext alphabet.

5. Write a function to generate an alphabet key if keywords are used in both the plaintext and ciphertext alphabets. Note that you can combine tasks 1, 4, and 5 into one function if you like. Make new versions of your enciphering and deciphering functions that can handle keywords for both alphabets.

Exercises

1. Encipher this text with the keyword `KNIGHTS`. Use the first key-filling method that we discussed above.

   > Before Cai was born, Cynyr made the prophesy that his son would have a frozen heart and be extremely stubborn. He added the prediction that no one would be able to endure fire or water as well as his son.

2. Decipher this text with the keyword `ROUNDTABLE`. Use the second key-filling method that we discussed above.

   ```
   AVLIDWDPDAVLIDWDPDXBLSBDPQBRGGLAJHVQSLDINVPDQJVSSDPGZRGJ
   IDSJNPRAHZGLTDJVSBRVISDNDWDPHJPDOZKRGDRUUVQLIAQKDUSPDQJT
   NDRNFILABSQLURIIJSGDRWDSBDDKLSZHZNDQKRLPAXLIDWDPD
   ```

3. This ciphertext is an example from Gaines's book. It uses a keyword for both the plaintext and ciphertext alphabets. In the plaintext, 'J' is used as a space between words. Break it by hand and reconstruct both alphabets to obtain the two keywords.

   ```
   ROVLL ABTLD LBCQM PXLBA FBTCT ATCOR LTOLC RHPDT XLYOA
   ELBXP HLXBT XXQLD RGLTK XRLGD BKLDP PLOHL YOAEL KOMXB
   LHOEL VCRRC RJLTK DTLRC INXPL LLTKX LRCIN XPLVD BLVOR
   LPORJ LDJOL FYLIO PORXP LMDEN XELKC TTLVK OLOHH XEXGL
   TOLIO QMEOQ CBXLH OELTV OLIXR TBLBC RIXLK XLVDB LDFPX
   LTOLB XRGLT KXLBO PATCO RLFYL EXTAE RLQDC PLLBT CPPLC
   TLVOA PGLFX LVOET KLDRO TKXEL RCINX PLTOL HCRGL OATLT
   ```

KXLNX YLLTK CBLQA BTLFX LTKXL XWMPD RDTCO RLOHL TKXLE
XHXEX RIXLT OLDLI EOORX ELDRG LTKXL XQMKD BCBLO RLDLG
DTXLL MLBLT KXLTV OLIXR TBLKD BLROT LYXTL FXXRL MDCGL

madness's book on classical cryptography
unit 27: dictionary attack on the keyword substitution cipher
last modified 2020-11-27

Unit 27
Dictionary attack on the keyword substitution cipher

A *dictionary attack* is an attack in which one tries to decrypt a ciphertext by using a list of possible keywords.

Python tips

Remember that to split a text and store the pieces in an array, we use the `split()` function. To read a list of words that are separated by newline characters (`\n`), you can do something like this:

```
words = open("dictionary.txt","r").read().split("\n")
```

If you are working on a Max or Windows computer, then the ends of lines might be `\n\r` or `\r\n` instead of simply `\n`. You should experiment to determine what is the correct thing to put inside the `split()` function. In addition, to avoid an empty word at the end of the list, you might try keeping only up to the second to last entry, which in Python is denoted as the -1^{st} entry:

```
words = open("dictionary.txt","r").read().split("\n")[:-1]
```

Programming tasks

1. Write a function or script to implement a dictionary attack on a substitution cipher whose key was generated with a keyword. Remember that there are several ways to fill the key. Use the word lists that you compiled earlier. Optionally, allow for the use of a custom word list. Use tetragram fitness of the plaintext to determine if and when you have found the correct keyword.

Exercises

1. Perform a dictionary attack and break this ciphertext. The keyword is a common English word.

```
UHENTWVVENLCMCAONTAIOWTNETTYBTLIGHUEVWPFSOMYIUHINBNVIT
BYINBXIXIVYBCHOYUHEYHOLEWNIXESTEYBTMBVEWPOFPBSUIALETOF
```

MBUESIBLYHIAHNOMBUUESHOYVWLLBNVLIFELETTUHECMIGHUTEEMYE
SENEXESUHELETTFILLEVYIUHUHITINUENTEBNVXIUBLREBWUC

Unit 28
Stochastic hill-climbing attack on monoalphabetic substitution ciphers

This unit describes an attack on the monoalphabetic substitution cipher from Jakobsen's paper in 1995. It is called *stochastic* because it makes random choices as it goes along, and is *hill-climbing* because it works to maximize some function. The function that it maximizes is textual fitness, and for us that means the tetragram fitness that we defined earlier.

The algorithm begins by choosing a key alphabet as the "parent" key and a plaintext is found by deciphering the ciphertext with it. From the parent, a "child" key is obtained by swapping two randomly chosen letters in the parent. The ciphertext is deciphered with the child key, and if the fitness of the resulting plaintext is higher than the parent's plaintext, then the child becomes the new parent, and the process repeats. This continues until the fitness does not improve for a few thousand trials.

Here is the algorithm, so you can't complain that something was unclear:

1. choose a parent key, such as `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
2. decipher the ciphertext with the parent key to obtain the parent's plaintext
3. calculate the fitness of the parent's plaintext
4. set counter to 0
5. while the counter is less than a limit of around 10,000
    a. set the child key as a copy of the parent key
    b. randomly choose two distinct numbers x and y in 0, ..., 25
    c. swap the x^{th} and y^{th} letters in the child key
    d. decipher the ciphertext with the child key
    e. calculate the fitness of the new plaintext
    f. if the fitness of the new plaintext is larger than the fitness of the parent's plaintext
        i. set the parent key as a copy of the child key
        ii. set the parent's plaintext as a copy of the child's plaintext
        iii. set the parent's fitness equal to the child's fitness
        iv. set the counter back to 0
    g. add 1 to the counter
6. output the parent key and/or the parent's plaintext

Python tips

While you cannot modify characters in a string, you can modify items in a list.

Bad:

```
myString = "abcdef"
myString[3] = "z"
```

Good:

```
myArray = ["a", "b", "c", "d", "e", "f"]
myArray[3] = "z"
```

Reading and references

Thomas Jakobsen, “A fast method for cryptanalysis of substitution ciphers,” *Cryptologia* 19:3 (1995) 265-274. DOI: 10.1080/0161-11959188394

Programming tasks

1. Implement the attack.

Exercises

1. Break this ciphertext:

```
IDSIYUDHJZXIXTOQOXUSVOROMNSRMOREXOESGOMMSVOMNSRSUSDHJS
YSTNIJOUQSTSMKSREMNUDJTNIISRJNIDOUKQLHMNGUXLUINIXINHRG
NCDSTIDSUNQCUHRUZSOIXTSVOMNSRSUSNRHRSSCNUHVSIDSUSGTSIQ
SUUOESIHMVYNSJSTUIHJOIGDZXIXTOQOIDXTUVOKUOIISR
```

Part III
Periodic polyalphabetic substitution ciphers

madness's book on classical cryptography
unit 29: periodic polyalphabetic substitution cipher
last modified 2020-10-13

Unit 29
Periodic polyalphabetic substitution cipher

A *periodic polyalphabetic substitution cipher* is cipher in which there is a set of key alphabets which are used in cyclic order as a text is enciphered or deciphered. The number of key alphabets is the *period*.

Let’s run through an example. Suppose we want to encipher the message

```
SECRET MEETING TONIGHT PREPARE THE VEGAN PIZZAS
```

with these five randomly generated key alphabets, which are written under the plaintext alphabet for convenience:

```
   abcdefghijklmnopqrstuvwxyz
0  GAEUPDOXKYTZJWIMBQVHRCSNLF
1  FXBNKVWQAJLECTMHPOSGIRUYZD
2  ZWCPVIHLFXOJEYNTGRDMKUQABS
3  LNVJIEPMADCQZOTYGXWKBSUHFR
4  SUNXTEKWQZLVRIDJACGFBPOYHM
```

To make things more clear, we can label the characters of the message by which key alphabet we will use to encipher each. The first letter ‘S’ is enciphered with the first alphabet to ‘V.’ The second letter ‘E’ in enciphered with the second alphabet to ‘K.’ The third letter becomes ‘C,’ the fourth letter becomes ‘X,’ the fifth letter becomes ‘T,’ and the sixth letter is enciphered with the first alphabet to an ‘H.’ We cycle through the five key alphabets until we reach the end of the message.

```
SECRET MEETING TONIGHT PREPARE THE VEGAN PIZZAS
012340 1234012 3401234 0123401 234 01234 012340
VKCXTH CVIFKTH KDWAHMF MOVYSQK MMT CKHLI MASRSV
```

Python tips

In Python, an array (list) can contain just about anything, including strings or other arrays.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapters XII-XV and XVIII.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, chapter 4 and pages 236-239.

Auguste Kerckhoffs, "La cryptographie militaire," *Journal des sciences militaires* IX (1883) 5-39 and 161-191, www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf, www.petitcolas.net/kerckhoffs/crypto_militaire_2.pdf, part III.

Programming tasks

1. Write a function that takes a plaintext and a set of key alphabets and returns a ciphertext. Use your functions for monoalphabetic substitution if you wish.

2. Write a function that takes a ciphertext and a set of key alphabets and returns a plaintext. Use your functions for monoalphabetic substitution if you wish.

Exercises

1. Take these three key alphabets and encipher the following text.

```
LBRUVCJAWZYSHXINOQEPFTGKDM
SLNAXDIGOBKCEYQHTMWJFUVPZR
IFWVBXNGKHZQYOELPCDTJRUSAM
```

```
In his book published in eighteen sixty-three, Kasiski
presented a method for finding the period of a polyal-
phabetic substitution cipher. The method uses the
positions of repeated sequences of letters in the
ciphertext.
```

2. Decipher the following ciphertext with the same key as in the previous exercise.

```
IYBIDTAXYIWTNQLFCIQHESZISHGLLBPOWROLAXCGSDPGIPQXBCIWWB
UXRWIBXXCVOTGSDCOCEJLFLQWWGVAKXDKCJBVYBWIGPZXWUBBFTGSD
RQOEOVVMBUFOBMBLKIBCBFYTWCBWIGPXBXWKKJAPGCVX
```

Unit 30
Finding the period: Kasiski examination

The *Kasiski examination* is a method for finding the period of a periodic substitution cipher. It involves finding repeated sequences of letters in the plaintext. When more than one repeated sequence can be found, the period is likely to be a common factor (possible the gcd) of the distances between them.

Let’s look at an example. Consider this ciphertext:

```
THZBAROLASYZFKHFNYCEYXOQMWHXLELXLAUHNPMIAZTLVDWNNHRDOW
SIHUCCMGNTTTCWSIHUCCMHTEEDCBUGMHZBAROLTSONNSHUDWQFZXRP
NABMHTZDPRYHUCMMNTWADUBUKAOCCMUKELRSDREHULXIAYPECDPNZR
OFVTRTWOCCMUKLAWGILYHNLCBRGWYNYCEYXTLVSGUFIDDMEKW
```

Notice that the sequence `ZBAROL` occurs twice. The distance from the ‘Z’ of the first occurence to the ‘Z’ of the second is 84 letters. The sequence `SIHUCCM` also occurs twice, with a distance of 14 letters. The sequence `OCCMUK` occures twice, with a distance of 35 letters. The greatest common divisior of 84, 14, and 35 is 7, so the period is likely to be seven.

Reading and references

Friedrich Kasiski, *Die Geheimschriften und die Dechiffrir-Kunst*, 1863; digital.onb.ac.at/OnbViewer/viewer.faces?doc=ABO_+Z224431001

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XIV.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter IX, section II.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 207-210.

Programming tasks

1. If you think that it will be possible to write a function that employs this method, go for it.

Exercises

1. If you wrote a script for this method, test it with the example ciphertext.

Unit 31
Finding the period with the index of coincidence

A ciphertext that has a period *m* is like mixing *m* different ciphertexts, each enciphered with a different key. As a result, the chance of picking two characters randomly and obtaining identical letters is reduced, as compared to an unencrypted English text. Without proving it, we state that the measured index of coincidence of the entire ciphertext is related to the period in this way:

$$\mathrm{IoC} = \mathrm{IoC}_{\mathrm{random}} + m\,(\mathrm{IoC}_{\mathrm{English}} - \mathrm{IoC}_{\mathrm{random}})$$

Solving for the period and using $\mathrm{IoC}_{\mathrm{random}} = 1$ and $\mathrm{IoC}_{\mathrm{English}} = 1.75$ (in the normalization that we use), we obtain

$$m = \frac{\mathrm{IoC} - 1}{0.75}$$

In tabular form:

IoC	period
1.75	1 (monoalphabetic)
1.38	2
1.25	3
1.19	4
1.15	5
1.13	6

As we can see, as the period increases, the values of the IoC get closer together. Since these numbers are approximate and there is a lot of variability in the IoC, *we should not rely on this method for finding the period*.

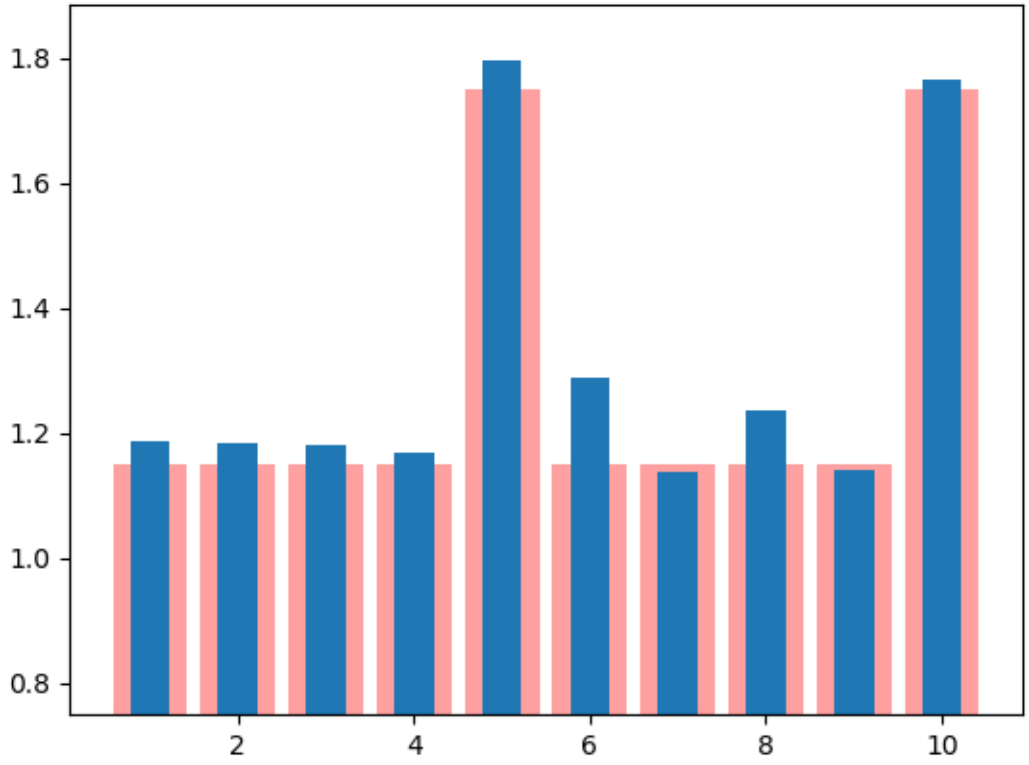
A better way to use the IoC to find the period is to guess a value for the period *m* and then slice the ciphertext into *m* slices and find the IoC of the slices. The n^{th} slice is composed of every m^{th} character, starting with the n^{th} character. Partitioning the text this way gives us slices that would each be encrypted with one key alphabet, if we guessed the period correctly. The average of the IoCs of the slices serves as a good measure of the IoC of the plaintext that you would obtain by deciphering the ciphertext with the guessed period. If this IoC is not close to the IoC of typical English text, then the

guess is a bad one. By guessing periods until the IoC calculated in this way is close to that of English, we can find the period, and we are usually correct.

Let's look at this ciphertext, for example:

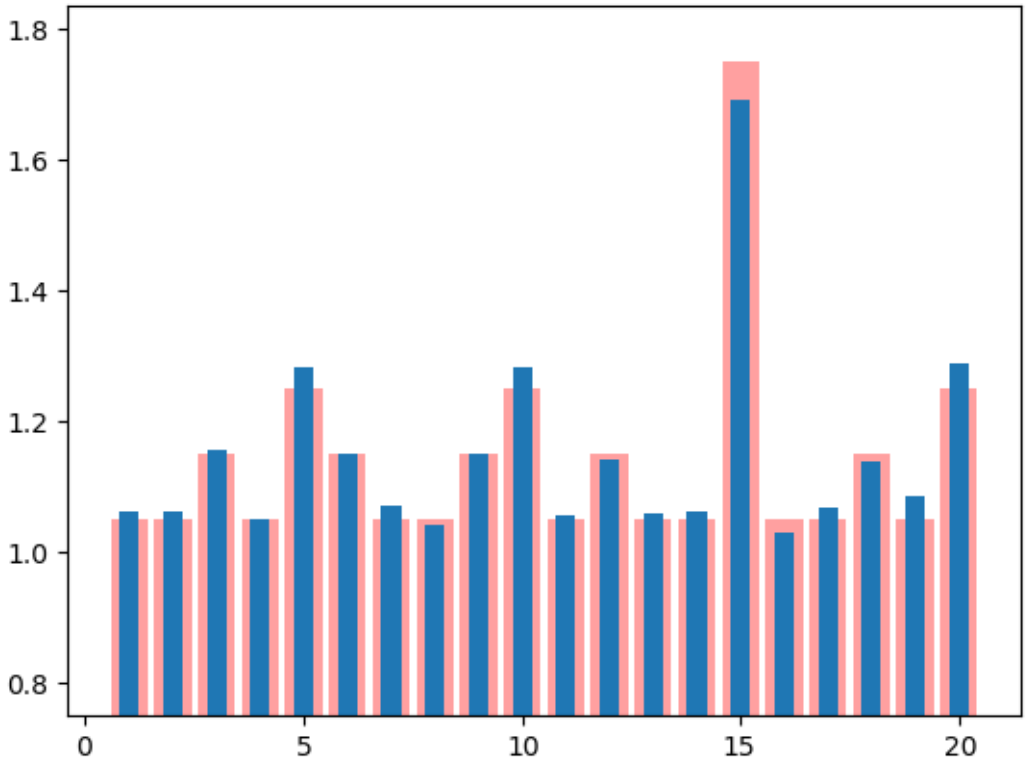
```
BUHMKLRASCKBLRZQQHRZMVVZBZLXWBNHOMKKEBTQWTUEMPLWLBQAGI
UWSFSFVPLHBVPHGXVOHYPMQWSCQAGXEMCHVFWQRJXXRUMLLVFTLTLD
PMPNEIQQPVYYAGLXRBVRRZQCKIOBUHFBAGZEVBBXWBBUHMKLRASCKB
LRZQQHRZMGRJFVQWLBXRUMLLVVXLKHWXEMPLTEMEWIUBVQXLAYLGBA
NQHXDRUEDMGKIFWPRJQPRVPFKRVMCBUHESMEDKBQBFMPKYRWBBBWLX
BBIIKOYLWEBUHRQPRQYJJRUSCAYLGBAVVPFSROCQWOHXEMCHVFWQ
```

And here is a graph of the IoC averaged over the slices, for periods one through ten. The measured IoC is in blue and the theoretical values for a period of five is in pink. The peak at ten is due to the fact that when we take ten slices of the ciphertext, then each slice is half of one of the slices that we took for period five. We conclude that the period of the cipher is five.



But the astute reader will notice that five is a prime number. What happens if the period is not prime? Below is a similar graph for a text with a period of 15. Again, the measured IoC is in blue and the theoretical values are in pink. The secondary peaks at periods 5 and 10 and the tertiary peaks at 3, 6, 9, and 12 are due to the fact that when we take a number of slices that shares a factor with the true period, then each slice contains some letters that were enciphered with the same key alphabet, i.e., that are in the same "true" slice. The expected theoretical value at period n when the true period is m is

$$\mathrm{IoC}_{\mathrm{theory}} = \mathrm{IoC}_{\mathrm{random}} + \gcd(n, m) \cdot (\mathrm{IoC}_{\mathrm{English}} - \mathrm{IoC}_{\mathrm{random}})$$



Reading and references

William F. Friedman, The Index of Coincidence and Its Applications in Cryptography, Riverbank Laboratories Department of Ciphers Publication 22, Geneva, Illinois, 1920, www.marshallfoundation.org/library/methods-solution-ciphers

William F. Friedman and Lambros D. Callimahos, Military cryptanalytics, Part I, Volume 2, Aegean Park Press, 1956, reprinted 1985.

M. Mountjoy (1963) The bar statistics, NSA Technical Journal VII (2, 4).

Practical Cryptography: practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-vigenere-cipher

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 376-380.

Programming tasks

1. Write a function that cuts a ciphertext into a number of slices as described above. If the number of slices is *n*, then the first slice contains every n^{th} letter, starting with the first letter of the text; the second slice contains every n^{th} letter, starting with the second letter of the text; etc.

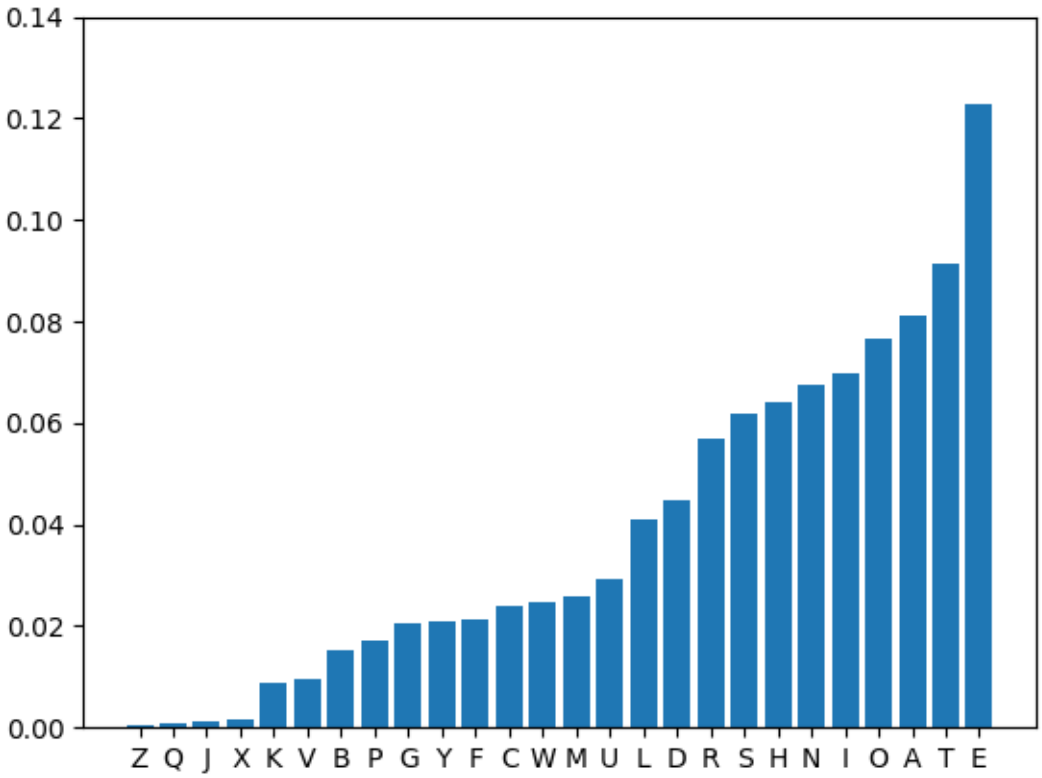
2. Write a function to find the period of a ciphertext with the method described above. You will need to decide on an appropriate cut-off, which you can base on the results of the exercise in Unit 10.

Exercises

1. Use your function to find the period of the example ciphertexts from this unit and the previous unit.

Unit 32
Finding the period: twist method

The *twist method* was published in 2015 by Barr and Simoson. Let's see if we can understand it. First, look at the monogram frequencies of English, sorted in ascending order:

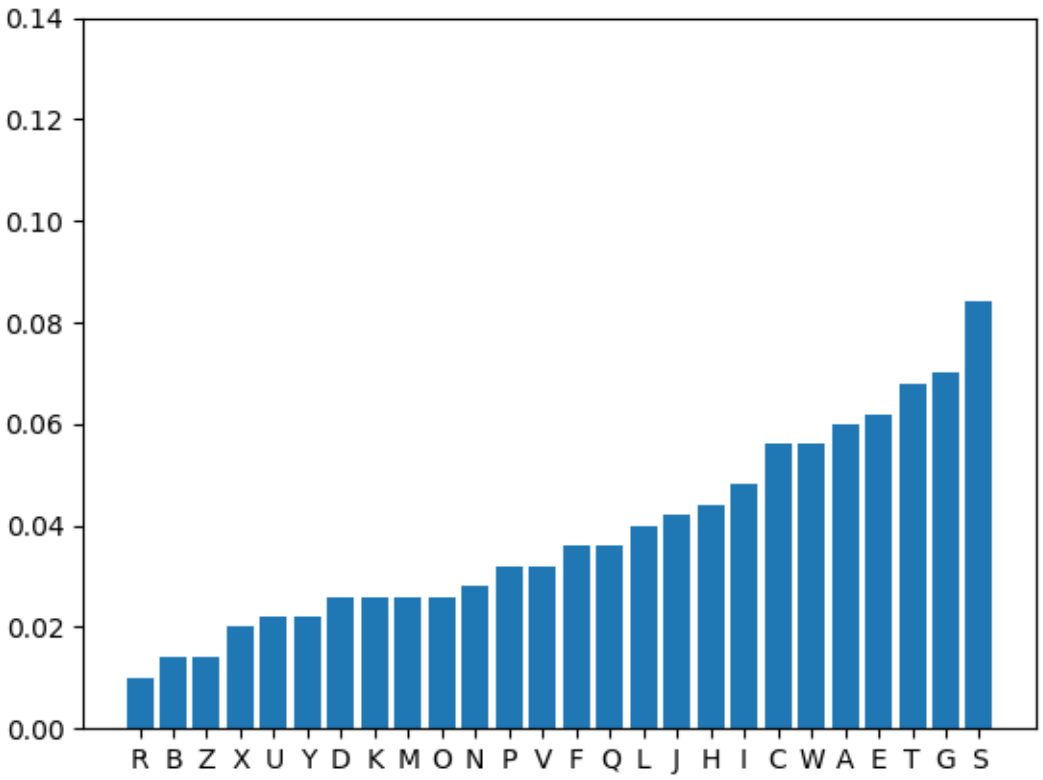


Now let's look at this ciphertext, which was enciphered with period five:

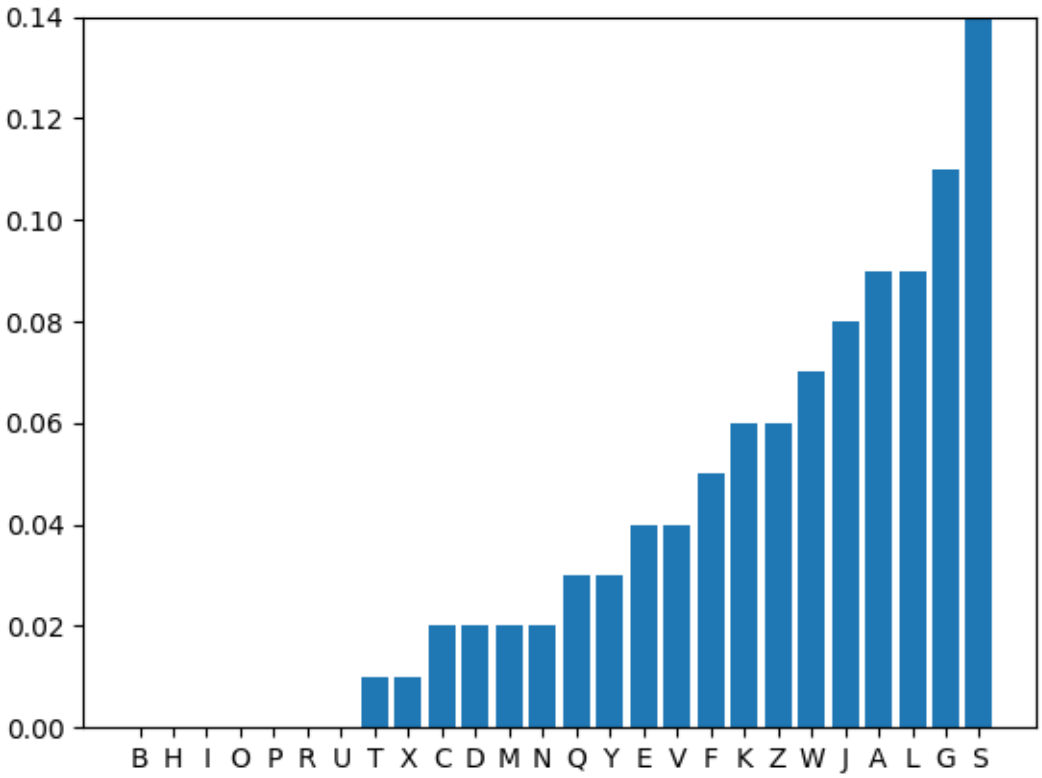
```
ZTSCALWAVEXTWAEJSSCASNSVSGSAUQSALEVSUTQJKDMGOACDCWEPLN
LGKETGJPFVEJIHWVYJEUWWSIVWZTDCHABPQSDDFNMYWTCVGJNFMLIH
GQSCLQSCTDCXSGTJYJHAFPQXTUIWBEFCGJCJSGHECGGADPMYWTVSNX
SKXGJRFISSPNEFTTJIKPIFAZDWJSSGEASMHTCQETRGHSGTJYJIHGQS
CLQSCTDCASNAIEACAMMFSOHWSSNGWKHEGQWSTQGJDSULAHFCGWBYPE
```

```
ETIURGIIOTGGTCRLWEUEASHGWWTMGHLDHCZWHOOILWIPKGCHKWEXNF
GGCVGVKPTKSFLAUGDTATPQHOOILWIPKZTFGPLWEFMVCTJENTTQVMHH
CXSGTJYJUENXSLKYEJSIGVQDUUXSGTNIVBEJIKPIFPSBENCLWEOEFA
OQOWSRQYFSTQLABAIEACAPHKAIILLAYTEAHEFLAHEAITGOYWZBMOQZ
TSCMVXSCMVNOWW
```

Here are the monogram frequencies for it:



Notice that it is much flatter than the previous graph. But, now, here is a graph for every fifth letter of the ciphertext:



The slope of this graph resembles that of English. This is the basic idea behind the twist method: If we divide the ciphertext into slices so that each slice has been enciphered with the same key alphabet, then the sorted monogram frequencies of each slice will resemble English.

Now we need to develop this into an algorithm that we can use in a program. Following Barr and Simoson, we define the *signature* of a set of of letters (a text) as the sorted list of its monogram frequencies. The *twist* of two signatures $A = \{A_i\}$ and $B = \{B_i\}$ is defined as

$$A \lozenge B = \sum_{i=0}^{12} (A_i - B_i) + \sum_{i=13}^{25} (B_i - A_i)$$

Notice what this does: it adds up the amount by which A exceeds B in the lower half and the amount by which B exceeds A in the upper half of the graph.

To find the period, we try various periods n. For each trial period, we slice the ciphertext into n slices, where each slice takes every n^{th} letter from the text starting from a different point. For example, with the sample ciphertext above, if we try a period of five, then we assign letters to slices like this:

```
ZTSCALWAVEXTWAEJSSCASNSVSGSAUQSALEVSUTQJKDMGOAC...
01234012340123401234012340123401234012340123401...
```

For each slice, we find its signature. Then, we average the signatures and take the twist of English monogram frequencies with the average signature. The trial period for which the twist is the greatest is likely to be the true period of the cipher.

Python tips

Arrays (lists) can be sorted with the `sort()` function, like this:

```
myArray = [1, 3, 2]
myArray.sort()
```

Reading and references

Thomas H. Barr and Andrew J. Simoson, “Twisting the Keyword Length from a Vigenère Cipher,” *Cryptologia* 39:4 (2015) 335-341, DOI: 10.1080/01611194.2014.988365

Seongmin Park, Juneyeun Kim, Kookrae Cho, and Dae Hyun Yum, “Finding the key length of a Vigenère cipher: How to improve the twist algorithm,” *Cryptologia* 44:3 (2020) 197-204, DOI: 10.1080/01611194.2019.1657202

Programming tasks

1. Write a function to find the signature of a piece of text.
2. Write a function to find the twist between two signatures.
3. Write a function to find the period from a ciphertext using the twist method. Feel free to use the function that you wrote for slicing a text in the previous unit. You might want to write a separate function for averaging signatures.

Exercises

1. Use your function to find the period of the example ciphertexts in Units 30 and 31 and in this unit.

Unit 33
Vigenère cipher

The *Vigenère cipher*, which was actually invented by Bellaso, is our simplest and one of the most constrained periodic polyalphabetic substitution cipher. Essentially it is a periodic Caesar shift cipher. The key alphabets are shifted versions of the regular alphabet, and the key is the set of shifts, which is usually expressed as the equivalent letters by a keyword. For example, if we want to use the keyword SPACE, then the key alphabets are

plaintext:	abcdefghijklmnopqrstuvwxyz
0	STUVWXYZABCDEFGHIJKLMNOPQR
1	PQRSTUVWXYZABCDEFGHIJKLMNO
2	ABCDEFGHIJKLMNOPQRSTUVWXYZ
3	CDEFGHIJKLMNOPQRSTUVWXYZAB
4	EFGHIJKLMNOPQRSTUVWXYZABCD

And here is how we might encipher a secret message:

```
PSST THE UNIVERSE IS REALLY BIG PASS IT ON
0123 401 23401234 01 234012 340 1234 01 23
HHSV XZT UPMNTRUI AH RGEDAY DMY EAUW AI OP
```

We can also understand the Vigenère cipher in terms of modular arithmetic. If we express the key as an ordered collection of L shifts $\{k_i\} = k_0, k_1, k_2, ..., k_{L-1}$, then encipherment of the plaintext $\{p_i\}$ to a ciphertext $\{c_i\}$ with a key $\{k_i\}$ is done with this equation:

$$c_i = p_i + k_{i \bmod L} \mod 26$$

and decipherment by

$$p_i = c_i - k_{i \bmod L} \mod 26$$

Note that if the period is one, then the Vigenère cipher degenerates to a Caesar cipher.

Some prefer to use a full table of all twenty-six possible ciphertext alphabets. This table is called a *tableau* (the plural is *tableaux*) or *tabula recta* (“right table” or maybe “square table”). The tableau for the Vigenère cipher is this:

key	plaintext alphabet a b c d e f g h i j k l m n o p q r s t u v w x y z
A	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B	B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C	C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D	D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E	E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F	F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
G	G H I J K L M N O P Q R S T U V W X Y Z A B C D E F
H	H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I	I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
J	J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
K	K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
L	L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
M	M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
N	N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
O	O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
P	P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
Q	Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
R	R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
S	S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
T	T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
U	U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
V	V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
W	W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
X	X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
Y	Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
Z	Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

In the tableau we have highlighted the encipherment of the first letter of our example message.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapters XII and XV.

Blaise de Vigenère, *Traicté des chiffres ou secrètes manières d’escrire*, Paris: Abel l’Angelier, 1586, HDL: 2027/ien.35552000251008, gallica.bnf.fr/ark:/12148/bpt6k1040608n, gallica.bnf.fr/ark:/12148/bpt6k94009991

Wikipedia: en.wikipedia.org/wiki/Vigenère_cipher

Practical Cryptography: practicalcryptography.com/ciphers/vigenere-gronsfeld-and-autokey-cipher

Crypto Corner: crypto.interactive-maths.com/vigenegravere-cipher.html

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 207-211.

Giovan Battista Bellaso, *La Cifra del Sig. Giouan Battista Belaso* [sic], 1553.

Paolo Bonavoglia, "Trithemius, Bellaso, Vigenère: Origins of the Polyalphabetic Ciphers," Proceedings of the 3rd International Conference on Historical Cryptology, 2020, ep.liu.se/ecp/171/007/ecp2020_171_007.pdf, DOI: 10.3384/ecp2020171007

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939; chapter VI, sections I-III; chapter XI, section I.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 148-150 and 240-242.

Programming tasks

1. Write a function that enciphers a text with the Vigenère cipher and a given keyword. Feel free to use your function that enciphers with the Caesar cipher, or to use the equation, or to use the tableau.

2. Write a function that deciphers a text with the Vigenère cipher and a given keyword. Feel free to use your function that deciphers with the Caesar cipher, or to use the equation, or to use the tableau.

Exercises

1. Encipher this text with the keyword PACMAN.

```
DOOT DOOT DOOT DOO DOO DOO DOOT DOOT DOOT DOO DOO DOO
DOOT DOOT DOO DOO DOO DOO DEET DEET DEET DEET DEET DEET
DEE WOCKA WOCKA WOCKA WOCKA EEEEEEEOOOOP GAME OVER
```

2. Decipher this text with the keyword VIGENERE.

```
OPKZVKVRZZKGVTYIMEGWSMIWOLKWPVZFZLHCTMFZVVHEGXZWOIHIYPR
WJQTJVJKIZVLMSXPXCZKINRUMOZKQNMEIYCTFESBIICTXVPKLZUOHRM
XLOMKRUYEHMMJWVXJVZXAXNXZSIMGVAIUMOBNIAMTOIISIYITLDNLVR
MEHZKNMSJIEWTKAUMTLDALVRRTLAWXXUIZRYMIMCLVVVJRIPMGLZZ
```

madness's book on classical cryptography
unit 34: brute-force attack on the vigenère cipher
last modified 2020-07-07

Unit 34
Brute-force attack on the Vigenère cipher

To do a brute-force attack on the Vigenère cipher, we need to try all one-letter keywords, then all two-letter keywords, etc., until we find an acceptable plaintext.

Python tips

The `product()` function from the `itertools` module can generate all possible tuples containing items from a set. For example, this block of code will create a list of all possible three-letter combinations:

```
from itertools import product
letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
combos = list(product(letters,repeat=3))
```

The length of the combinations (3, in the example) can be replaced with a variable.

Programming tasks

1. Implement a brute-force attack on the Vigenère cipher. Use tetragram fitness to decide whether you have found the correct plaintext.

Exercises

1. Use your implementation to break this ciphertext from the 2015 British National Cipher Challenge:

```
WLHJLVVXLXHQLRRYUPLXWPHEXGWMRRZMOPEIWLHPRGDXLSQSIEVEII
KSXWHMQXKIXOVIFXRVRJEIUPLRLXLWDQLRRVVXRTRZHVRRWLHVDXOM
QIVFXXBSXRHZHVNRRABSXQLKKXJIWPXGNCDRGJLRGWRQHSQILRWIUI
VXLRJLLHLRJXKIUIDXWLHZHVBPHEVXBSXALPOMQGRRYIQMHRFIWLHV
HMFLVHROWSUMICRYWENISSVWHWVMRRRJLXKSZQXGKARYOHWLDXEIZS
UXKXRCRYGSLLHEUEEMGSIJLZHLXRGVHHWLRYVEQHIVDRFWIVRQRYUJ
UIQGKJUMHRGWSIULDTVXKIEVLXLWKARYOHSEBQRVHSUQDCEIWLHCFE
QRRXDJISUHWSLARRGIULRAWLHCIIHPDFRYWXKEWTHVKESWBSXWKSXP
```

```
GEVOWLHQLJBSXADRWXRSXXEMGCRYUWRGDPOIGJUMHRGWWLHROIDZHX
KIPSQIBMQYQQDVNIGXUIDWXVBFLPOWLROSFOHVDXWLHJDVHRGSIXKI
SPDXISUQLRIVLIGVLGKWWVDWVILALPOPHEYIWLHHHXDMOWLROSFOHV
BSXALPOJLRGXKINIBMQHRRRXWVBXRHRYEPHGUSVWPILXZMOPQSWARV
NEQHRYUPLXWPHKDQHALPOIQHEIISUILXKEVIYIQTUSSIUPBFHKXR
```

2. Now try another ciphertext from the BNCC (2002). This one has a slightly longer key, but it may take a lot longer to find it. The time it takes grows exponentially with the length of the key.

```
BPPOFDATNWBDLJZOIACTQJJXTJZOTSIUQTPZLPZAJDQUUAXUBIBTFM
AVDMUTIUUIDOMQFGPGZPRNFDBPPQTOCTEBIQDBXCFANUTMNMBFDQBX
KPZKFDVJZOUTMFZOMUAIQVDDGQFQPZMOSQOQWONMIMTGANNKUBEBFD
KEMAZACDMVTQMJTIWQUPHMERZPYAPGBIMUQFWOFWBZIEPZFEAJZTPZ
LPZIOPAVSOFEBZACTTWVXLNQMUYMUTMSQIUZWPZIXQMLAVXQLOQAEM
GQXMBEMDAUFMTPZMBEQGQISFPFYIUQZJMTJEAIMTMIMTMGJZNMMUNM
JMQQUIWVXLCQUPEBVZNPDBVZIUQQGMABDMTGTUANBZIUFIDWWGZMSH
MTUEFDMUASOAKLADFDIMMVUQZOAZQQZIMXTPZPBIMUOIFMEOQHMZJZ
BIQDJOQOUBZANUTMTFWDWWGOPFYQDMTTUPBHMTFWSQLIQZFOPFYQTF
ZZUATGKIMXMQITMVUPQWQZTUWORWSFPFOCSUWVEUJZLBZLXUBIEWNM
VZRITOQOMBJZOBZLSMBIQZQXMBEQOSIQBTJOIUUWOEQOFPFYWEQZOI
WSXLJYAVDMZACXUTMMOSQMPRKPGZTQBIQXMQITGZFUVTGKIMBSMVTM
KUUWOUAOQKFEABDQMKUVFCBXIOPQIAXFFPBFIMXCOBTFMABZBOQATM
DPULFPQXUTMNMBNTFFWTMBJENZKWVDKVDQPEQUKKPZKFDVJZOQDWGN
ICNIHQAEQDJOMBZLZACXUTMNMBNTFFWTMBJENZYGNMAUQZTZMFPNPD
KVDZFZKZMABSMOFTFYIOUINECSQPFRMMFCOMJMQBPPQTOCTEBFDUTI
QUTGPGJVFITTQTRIJFPGGTTQZWMVUUUVEBPRKPGZTQMOPMBHWVDBPM
ZSUDFMBBZIHDMFYMOFNBHWVDICXMUAPJYQUDCTFEFGVEQZTFIOPWOQ
IOABIQZBZLUTIUACSYMFFQOSEJXTJZLFQLCQIQXMBECSQNPDCTNWUT
XMQITQZFBTZUVUTMVECBXNBEPJAVZACSEMT
```

madness's book on classical cryptography
unit 35: attacking the vigenère cipher with cribs
last modified 2020-07-07

Unit 35
Attacking the Vigenère cipher with cribs

Attacking the Vigenère cipher with cribs is not as easy as it is for the Caesar and affine ciphers. To use a crib, we subtract the crib from the ciphertext at some position, thus revealing a segment of what might be the key. If the crib is sufficiently long and the revealed segment shows some repetition, then we have a good candidate for the keyword.

Exercises

We will not ask you to program the attack, since it may be unnecessarily complicated.

1. Break this ciphertext with the crib `NATIONALSECURITY`.

```
JGVROAEAQNFRFEZGUFQAEAQNNOSPUWRVWYLYGNPBSAJKSZSRZYTAZF
OYLKNHHTZZCVRZOCDFWVGECARPYHEQXGCGVNPSTLLWWZEQNGKSLXVE
EXSQWFEEDLAJQSRFUEGTSPKACYGDAVAHZKSGOEMDQWRUEOOCRQVNZO
FEAZIEZGSCLOYSIENQDEZGFGRFRGXEEQMPFVPERPPJVYSRRMDQWVQG
EZGLVGOQXQFGKENDCNQHSEAPEFXRGWKLYDNNWRRBJRLEVHRFOXHCNL
WHLLUEPXRPVUNBZDPFUEYHCEJQNVFCZEOUALCLLKOAVWTLJJBXRYSN
IFWSLFFIAWECFCTVRNLDQFSLCTSNSUDSDZWTQRWYAVSRQCCQRTRGEX
SKLFHRGAEEF
```

madness's book on classical cryptography
unit 36: dictionary attack on the vigenère cipher
last modified 2020-10-03

Unit 36
Dictionary attack on the Vigenère cipher

If we have reason to believe that the keyword used to encipher a particular ciphertext is an actual English word, then we can perform a dictionary attack. Of course, the keyword might be a name, so we could use a list of known names and previous keywords as our dictionary.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 112-116.

Programming tasks

1. Implement the attack. Use tetragram fitness to determine when you have found the correct plaintext.

Exercises

1. Break this ciphertext. The keyword is a common English word.

```
ARVYIMZVGVQHFXJJWBFGGVHPSVFJRHOHGEFXYEPYMLOLKJXEWFNFAB
FKERQISMIEEZSMPJXMZIPRXBGRCCWXUYTZXRSKGEGRLLGWSKEITXSO
WVPDIGLGQEXKSGVFVASWGOTHKIFKLXSKGEGRKQCJWBNIQEPBFIGRZX
KHTFTIARIVJYGVVJEGVEVKIFHXUKSVAVELQOWRVVRVJCRKMHFYUVHM
GWGTYKWHKXMMSPEFQFMRKTEMASPJXAWPCKILLENCIZSXKFRLARFZGT
LIVYIGKEORRBHYNRXXVEPUAXSOGEIWSGTPTMGKTRTAQWVRRWSVFKLX
FEVZSGSPKEWMAXWKIHXWVRRWSVFJEGVXGTLGGPQXCASHKJWNWHUVZX
JENPITJWCXSTKETVWNDXPZWMDEUKAXWORLFEAGNPHBKGQLVTYIFKIV
ZGQDTTFMGJJKGQWJMGYXJRXVJCRKSZJERYMVSTRISTULCEHIJSOZWX
VXQXMOWXJVTNTPKTEGGTRFVMMRKKCMGAGZKAARQEEKWZKJIWKXCEHT
JHVYIYABORCGGXDVEEDXJRXWAJHZGNDXVYIMSMPKIWHETKSYLLGJXT
FHCIHBKEJZKADCKEIYXMEZIGLENXSKAXJDXASXUVGNJMVPIQHITKWB
VIPKMYAIFRWTHVQSPXEPQEKTYSKEJTUXVYIUAKIVWMECUKIKQXJFWX
```

WBRVVMKWCPMLOLAKLXFWCKLHMKJKEGQGQDTTFCQIKHNITEQXFXCXIG
UCYFYEVAKCPBFKNPYLWXJRXISVVZGNDETRPZGVKKLFLSRISMWGVKLX
AVFRXT

Unit 37
Hill-climbing attack on the Vigenère cipher

In this attack, we first find the period using the method described in Unit 31 or 32. Then we start with a key that is all ‘A’s. We start with the first letter of the key and replace it with each of the letters of the alphabet. The choice that gives the best textual fitness for the deciphered plaintext is kept. Then with the new first letter in place, we move to the second letter of the key and do the same to it. We continue until all letters of the key have undergone this process. Then we repeat again from the first. We continue in this way until the fitness can no longer be improved. This attack is very reliable, even for shorter ciphertexts.

The algorithm:

1. find the period *m*
2. set the key as *m* copies of the letter ‘A’
3. set the current fitness to the fitness of the undeciphered ciphertext
4. set a flag equal to FALSE
5. while the flag equals FALSE
    a. set the old fitness equal to the current fitness
    b. for each position *i* in the key (*i* from 0 to *m*−1)
        i. set maximum fitness equal to current fitness
        ii. for each letter *x* in the alphabet
            - set the i^{th} letter in the key equal to *x*
            - decipher the text with the key
            - calculate the fitness of the new plaintext
            - if the new fitness is greater than the maximum fitness
                · set the maximum fitness equal to the new fitness
                · set the best letter equal to *x*
        iii. set the i^{th} letter of the key to the best letter
        iv. set the current fitness to the maximum fitness
    c. if current fitness equals old fitness
        i. set the flag equal to TRUE
6. output the key

Reading and references

Practical Cryptography,
practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-vigenere-cipher-part-2

Programming tasks

1. Implement the attack. Use tetragram fitness.

Exercises

1. Break this ciphertext.

   ```
   HVSWYTMSBBDIYGKDIJSWNJVCWITWDMTIHSCIJHVFIRFSTCCMOKTNHW
   RTJHVJAVBUYWFTHMTBSM
   ```

2. This ciphertext was encrypted with two Vigenère ciphers. The order in which they were applied is irrelevant. The result looks like it was enciphered with a single long key. The period is the least common multiple of the lengths of the two keywords.
   a. Find the period.
   b. Use the hill-climbing attack to find the plaintext. At the same time, you will find the combined key that looks like gibberish.
   c. Use the hill-climbing attack to "break" the key that you found in part (b), and recover the two keywords for the two Vigenère ciphers. It may be helpful to know that the lengths of the keywords are factors of the overall period.
   d. Now that you know the two keywords, decipher the text with two Vigenère ciphers.
   e. Repeat part (d) with the order of the two Vigenère ciphers reversed. Verify that the resulting plaintext is the same.

   ```
   KBLFZROYITGKACGXWGSWSYOKTSYMRQZEPCZSRLAOWXUYRHVMTEFIQY
   ZNVGULHCBZVBOKCJWVLKDCMNEXIYFNZLWFLTJBQFCUBFTEDBCXZDLZ
   IMJFLAFSQZROCMNUKISZGOWOBWZLGVIIICTXMOZXCFRHKCWRZSPYAX
   LJOIVPMAOLVRNUBFXEBIBWOGZGCIYZLJKHGLWYQWCRXHRUHAHQMFOB
   SUFGQKBMCOCUNFQFERDRQLRDMXLRCTFQXEPLNHCYACOHJFBEDDERTI
   JDOBUOLLNERICAMBSDVINVIZJHUJBRTGKAVXOOCHTXMUWGSOIIBXLR
   GAZAVNCJMBOYRYJXXFBTMDD
   ```

Unit 38
Attacking the Vigenère cipher as a periodic Caesar cipher

Since the Vigenère cipher is a periodic Caesar cipher, we can use the technique of Unit 19 to break it. First, we find the period *m* with the index of coincidence or the twist method. Then we cut the ciphertext into *m* slices, where the n^{th} slice includes every m^{th} character of the text, starting with the n^{th}. Each slice contains characters that were enciphered with one Caesar cipher. So for each slice we can find the shift using the technique that we used to break the Caesar cipher and convert that shift into one letter of the Vigenère's keyword.

Reading and references

Practical Cryptography,
practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-vigenere-cipher

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 210-213.

Programming tasks

1. Implement the attack.

Exercises

1. Break this ciphertext with the attack.

```
BNAGGUTAGGOTGNTALTNTVEWFZSMARGSGWGNWKTBTWEGQGQGWBTWAME
KTNBZKEHZRRGGMYOFXAFZZTVNTQFIAIGNTQHWTZEKKVCRSWUMSUJXQ
SGDEQAZGWMFVSBZVRDLMAJSGABYLHBUKOHZYJAAWCKLAIGYGFMBTWZ
MGYERURYKTOROFTJBZLEMNEWTZUGKIIFYWWAVTUXQJXGMMZEFHBROK
AWHRVAIIKCGWJTLAQFXAZPGLJHUGNWLBNXLHVYEZHXRISGSRKHFMGU
YXBUKJEWIKUTVZKFWGBAJEQSKTNBYUNXKNTTKMNQQHCENWTZGCSESR
JGNBGNALUBXFBVTOVHVGHWEQRBWPPNZALIJGZNVQXWWUVRDBWAHGMB
```

YKKPIFNWWCCUFMPRYZHZRYWXUFOEGWGGDHVROFUMVTYTTBTWTPHTVK
MQSAETVUFVIFZSPILYDHWXOFZNBXSAWZK

Unit 39 (optional)
Gronsfeld cipher

The *Gronsfeld cipher* is the same as the Vigenère, except that the key is a string of digits rather than letters. Because the key is made from digits, the largest shift is nine.

The attacks that we developed for the Vigenère cipher also work for Gronsfeld, except for the dictionary attack. Notice that they only needs to try shifts zero through nine for each digit of the key, rather than run over 26 letters.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 117-118.

Practical Cryptography: practicalcryptography.com/ciphers/vigenere-gronsfeld-and-autokey-cipher

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 213-214.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter VIII, section I.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, chapter 4 and pages 245-46.

Programming tasks

1. Write a function to encipher a plaintext with the Gronsfeld cipher and a given key. Feel free to simply make a wrapper around your function for the Vigenère cipher.

2. Write a function to decipher a plaintext with the Gronsfeld cipher and a given key. Feel free to simply make a wrapper around your function for the Vigenère cipher.

3. Write a function or script to brute-force a ciphertext enciphered with Gronsfeld. Feel free to copy your work for the Vigenère and make the appropriate changes.

4. Make a copy of your hill-climbing attack on the Vigenère and modify it for Gronsfeld.

5. Make a copy of your attack on the Vigenère as a periodic Caesar cipher and modify it for Gronsfeld.

Exercises

1. Decipher this ciphertext with key 78345024.

```
VCWHFMPIKASSYOWXPADCTNGXDWZLDTJIUQWMXTKQLBRHTIVLLTOMXM
WVRGIMJMAPVZGJNECWVTGMJRCRKIIIFRFAOIWRJEFALNHEWWJSRVRA
XIVAOMQRTNGGHVFEQLQYYXRAJRVSHKFSZNVCLBZLTWQYSLKEAEVLVC
JLYTJIVTGQFNVSOIYIMAFWVUXGMBNSVLLRMIO
```

2. Brute-force this ciphertext.

```
TNMDFHBUXUYQLOXBWWLDXMTDYQRMJRMTDUIVFXNSKFPYQPSJMTALXC
KFARKXYYQLXCYFRNMCMTADFAKUJAMFHDQHIKLJWSTBAFQTJQVBQHWM
HYQPSPPYRZYXZFHDMJABJZYQPXOVWNZYBHAJNJAVZPOFWKXCLWWDMR
JMRUYQLANYDCOTDNMCYJWLBBAMNMJJYXXINCAJAPXRAINHYQPXUPYC
SJVVWNIZCVKCOJPVTMATCYJJABQPHQAMNYJRMTDUIBWJJRBRSQRVKC
OJXAMNYYQPSPZNBHBCOJALNLHSWVYFLQUYJYLFCOTFAMNYJRLSCLWN
KXXMZUSBJZNXMXUBRKLWJAYQLRXTJWANWDMRJMROFMHGJUIXUJMAMN
AWDLBJF
```

3. Break this ciphertext however you please.

```
CVPJIWAGLSMGXWVLNLKXAQSEYNHBRHCYXHXHFORSXWKVMFROCJHPTC
HAURFTRGNWUOUKKJXHESZHAGGISZRJJRLDGOUKDNHKWXGAOWILBGWH
NYVYGFTWKSXRHASJFICUMJZWNTQIHNDWFJUNFRCHUESIKACGIHGOBC
QDLKUVCQPTRLLGVPNKVFJHRJOVCJHPFXWHKQOUVRBKWITTWQCWHFYQ
XVEEJTGNEUIJJDBERMRAQRUWIHHBRVVOUVRWGQTXOQYQZEWCKNTHIX
ZKNQSPTYLCKRNYNDCJDSSUWQWULJJEJENTMKEACQDNTJAGSRTGFQQI
CTSPDPLSRGJJKQSYZKNORRJGGECQCJJRYRRSNZLXPSAWZLNUDSBKOU
CVALGLWUWIYYUNCFTNUQJTBAIBHAUDRNKV
```

4. Break this ciphertext.

```
NQRTSGRRTSGFQHCWYWSNBFLPWPTZIRMLRSDTNWXLMGXRNQJVAEVYGS
WVYKILISHXOOFWLEZGOECSGHVRQJVFUBUHXEZBKSWIXYIRGSDYGPYB
VAVXWVAQLKXAEFBXOUWPGGZJJSRALDVDMSDRDAVXIEHJGYNLJUXHML
DXENNUWTPJDXEATPILMYWYCMXDRDATPIFZJQGHJJDRSISGXHMSKIAB
JVSMMWDHIAMHWAVIWLEVKHILQSJVABMHVSQHNLEEJQXTWQRSKNTUWO
UJSERAQHC
```

Unit 40
Beaufort cipher

The *Beaufort cipher* is a periodic polyalphabetic substitution cipher in which the key alphabets are shifted and reversed versions of the regular alphabet. Encipherment and decipherment are the same operation. The key is series of shifts, which is typically represented as a keyword. Each letter of the keyword gives the first letter of the key alphabet that it represents.

For example, let's encipher a short message with the key BEAU. The key alphabets are written here under the plaintext alphabet:

plaintext:	abcdefghijklmnopqrstuvwxyz
0	BAZYXWVUTSRQPONMLKJIHGFEDC
1	EDCBAZYXWVUTSRQPONMLKJIHGF
2	AZYXWVUTSRQPONMLKJIHGFEDCB
3	UTSRQPONMLKJIHGFEDCBAZYXWV

Here we have labeled each letter of the text with the key alphabet that is used to encipher it:

```
HARRY SAYS BEAUFORT CIPHERS ARE BEAUTIFUL AND STRONG
01230 1230 12301230 1230123 012 301230123 012 301230
UEJDD MAWJ DWUHZMDI CSFUAJC BNW TXEGBTZGJ BRX CINMHV
```

We can also understand the Beaufort cipher in terms of modular arithmetic. If we express the key as an ordered collection of L shifts $\{k_i\} = k_0, k_1, k_2, ..., k_{L-1}$, then encipherment of the plaintext $\{p_i\}$ to a ciphertext $\{c_i\}$ with a key $\{k_i\}$ is done with this equation:

$$c_i = k_{i \bmod L} - p_i \mod 26$$

and decipherment by

$$p_i = k_{i \bmod L} - c_i \mod 26$$

Notice that the two equations are the same, but with p and c exchanged. This means that encipherment and decipherment are the same process; a cipher with this property is a *reciprocal cipher*.

Here's a fun fact: The Beaufort cipher is the same as a Vigenère followed by an atbash cipher. Try it and see. The key for the Vigenère in this case is the Beaufort's key enciphered by the atbash. We can reverse the order of the atbash and Vigenère, so long as we also adjust the key.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 121-125.

Wikipedia, en.wikipedia.org/wiki/Beaufort_cipher

Practical Cryptography, practicalcryptography.com/ciphers/beaufort-cipher

Crypto Corner, crypto.interactive-maths.com/other-examples.html

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 214-216.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter XI, section I.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 202-203 and 240-242.

Programming tasks

1. Construct the tableau for the Beaufort cipher.
2. Write a function to encipher a text with the Beaufort cipher and a given keyword.
3. Write a function to decipher a text with the Beaufort cipher and a given keyword.
4. Make a modified copy of your script for the Vigenère cipher that can perform a brute-force attack on a ciphertext encrypted with Beaufort.
5. Make a modified copy of your script for the Vigenère cipher that can perform a dictionary attack on a ciphertext encrypted with Beaufort.
6. Make a modified copy of your script for the Vigenère cipher that can perform the hill-climbing attack on a ciphertext encrypted with Beaufort.
7. Make a modified copy of your script for the Vigenère cipher that can attack a ciphertext encrypted with Beaufort by finding the individual shifts by matching monogram frequencies of slices of the text with frequencies from English.

8. If we encipher a text with a Beaufort cipher using the key BEAUFORT, then what must the key be to get the same ciphertext with a Vigenère followed by an atbash cipher? What must the key be if the atbash is done first?

Exercises

1. Decipher this ciphertext with key HUSBAND.

```
ZBKJAUMNBLHNFIDDABPCFHSIOMRSDRMXXUWHBSJSAXWQGBNFQSGAJW
VLZGFNVNXTGPWMWKNHOPGVKGQKOENQOGNBEFYDNEFWSZQJKIHCZXHE
FNUWDPOXPFQBCEKFFZLCEWITBAUGBNBDJTONNGVPPKKIUZUBOKSAXH
HOTUGCTABUMZAONKJHWJONKJIZHDJHWSQZEMFITJRZHPJMIKAQAHJW
PNHPTNHYHIKQSJLONSITJVPSEOIFADDOYHGZQMMUHIJWFBNLJMOWEW
IZRDGFXMWPONOKMIKAQKKXNJBNZXJV
```

2. Brute-force this ciphertext.

```
QEPHSWKMTGEWHMMGACOEJAEUERWXAJGWNPXMWTMFYLBAYKMWHGTSJS
FYPKLHHAMBRMGQNOGSFSCDKJAAAFBNELTKRUDEJZWYYTABLMKRHONW
ALOOQNAXMBASZXMYYTZWEEKPPMAKTARQGZEPKLHHAAFBIOAHLAESLW
BGLSNHEZHLAEENPPWHKMAWECKXAXAMNBJKJSFYMNZHLAIDTWORAFBJ
OYGHEHKRUZXWQWJQKPSLSERELANOJWBWVKRXGGIOTVMNMWWNMYJKSA
RQGZLTOSMYLTWXWFAFONSZWISBAGPXBWBTTCFQFOSZONSFSCAQGHIT
ORWXAJKZSFBOUMWHZSFJMHKNZEJKTCDEGAWNMDWNQJOCMNZSFIABAT
EKIOMAFBZBWNMWNMKLZXWBAABQVOJWBGVYRWBEPKSWOLAFBWAPWQWA
HTCWXWFAFONOUXCDQIMAHAKYLENYPLONTSRXEZOOLTSLSZNWCKSBAI
SMHBQNMSMBEPDNSFCSDTWZQLBAFORHGAVBQOPAPKDWBEHOTCALWDPS
FYSFLMZXWALJOAHSRXGAHLQXKCAHTCIRMQUSFYLOQLHAILEHAQVNLT
ORSSCYEKNZWHLWULLSGAHEYWZLMAAASMIEQNSMSQEN
```

3. Break this ciphertext with a dictionary attack.

```
WWOQTJIVANTKVMCOEILZMBYIGISXAZHDITZIEQARODBLNMKTXFECYS
JJVTQHOTILGSCMTFDVBAVMKTCGIJTXJRASNVOWTAWESFQVWVCRFDVE
RHWSMGFJVFAUAGBWHPVQXFLKTXJTKKABVKVMAISHACLYSWJDDOIFDH
YKUWBJLGSJENJUXMCDLKJCGSKVZTLDOIFDHERFDSMMNUAAZZFVMDCS
XYQBDHYKHPVFALWXXAUAOUZNTKJEATVFAICXZWUCXIJJFQXTNOBHQW
WIYQKHGKLTFKYNUAAFKAQAZLGSSMTCENYQDEJETCSLTXJFKKAVEKUC
GFQUEADHISTKWMRKUHVWVDCMMWIWIMMLAYSJTGZLGSSERKISUEATOX
TJWWMRUWVAQVHHINJIJFQBSZIELSZINSWLIBFD
```

4. Break this ciphertext with the hill-climbing attack.

```
ZDJPPAIENXVBYODKGKPZNZKTDIGZUDUMSIKZCUYWNGZAXNGZVMZCEH
LMXSODCWBWZYSMQZAWTTZIKVSQJWTJDFB
```

5. Break this ciphertext with the attack that matches monogram frequencies.

LTTFHWGKAFAASOUXHJRDSOZCNCWPRDJYSULDAGHGWZLKEKGKARKJHV
UXLZMVLQKKFEXJTHCJKOMVHXYTOBKGLXURWHMALXYFEQXJZVOJONPR
GVYTZDRRSBUELOEEIHYDSEWEQOLCSGTTKXYKICEZSKZKFEYDYKOQZG
AMXXYCBTMNDBJDOVQAYGPEATSNSGQGAAAZWKPRLZMOLQCRZAAPEKQC
QHTGLNOTSOQANWOYOOWFHWUXZAAODWKGQOWEFXOWGPEEIAAISPWDYO
RZUZRGXBWOSWUZPHUILHPCHOLGPOQYLEYDKVWESAJDOWQVHWJVEQLJ
XBKWNOXNJGJMFTLKDBWIENTOHRAGOEXJTKPGBBGVEKPRLDIDUEUXMC
WYPOKQDTRKDCBQZUQVLBZKMHWELXYKQOWDYSOQHEMNDPYDKVEEIJOT
PHGZXBYXPAAVXURHHZAXWAPRLDMDGOUXZAAZDVJCFBNDXCVKBETTRV
JGBZRLLSOTPTNFXTUXMHSXTMRGNQQFZALXKUYYXBUELO

Unit 41
Variant Beaufort cipher

The *variant Beaufort cipher* (also sometimes called the *German Beaufort cipher*, or simply the *variant cipher*) is the inverse of the Vigenère cipher. Encipherment with variant Beaufort is the same process as decipherment with Vigenère. In fact, by modifying the key, a variant Beaufort cipher can be converted to a Vigenère. The modification to the key is to apply an atbash and a Caesar shift of one.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 121-125.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 202-203 and 240-242.

Programming tasks

1. Construct the tableau for the variant Beaufort cipher.
2. Write a function to encipher a text with the variant Beaufort cipher and a given keyword. Feel free to merely make a wrapper around your Vigenère decipherment function.
3. Write a function to decipher a text with the variant Beaufort cipher and a given keyword. Feel free to merely make a wrapper around your Vigenère encipherment function.
4. Make a modified copy of your script for the Vigenère cipher that can perform a brute-force attack on a ciphertext encrypted with variant Beaufort.
5. Make a modified copy of your script for the Vigenère cipher that can perform a dictionary attack on a ciphertext encrypted with variant Beaufort.
6. Make a modified copy of your script for the Vigenère cipher that can perform the hill-climbing attack on a ciphertext encrypted with variant Beaufort.

7. Make a modified copy of your script for the Vigenère cipher that can attack a ciphertext encrypted with variant Beaufort by finding the individual shifts by matching monogram frequencies of slices of the text with frequencies from English.

Exercises

1. Decipher this ciphertext with key `TOWER`. What is the equivalent key for Vigenère?

```
YMTQWGQPCALIMJCVFLAVVEXXNHGXEOBXGDRSPYJMLDXDNZGRSQLZWD
NDMWPFLXZAHLMVOXSPXDNLZGDJUFVABZELQCOQVEWAAEPXDQVSQPOL
HJFURWOVDIOCHZHDJKZIECOQVOCHUVOWVDHKXYNYPZBUXAJAFLACVB
AWBHXMPCSQAEWKAASQLZXDNLZGDJUFVABZIEJCLPXKPVUROQLBPWLL
PLAAZQPBKLZIWCOUXWWKOVENKDELDULIHAHBYJILXPACKAAJHVGVDJ
PD
```

2. Brute-force this ciphertext. What is the key? What is the equivalent key for Vigenère?

```
IHPHYYEHPRYKFNDELKOGYLUIPYZTIWNOLRCCHNAAMALMPOIZTBAFOH
LFANAPHJKRNNACPOZWYIQNAIAHKFYTTLWOLZIHWRSLELOOHWLVAAOP
YUJDCJFLKNNKFCPSIIEFETNHEXESNWNWAAQWYQWSMETNENAPHYWRNE
SNDIGOEFBBUOIFDAFHWUNDQDOMASOZDYJDCOAJLEUNAHYEMKMYUEUN
SUCOWWUMADUPTBATCIEMQCBLUVHIWAXWETYIEHPAHZGURELESYPOMK
MUJYMPRUJGYYOHFEWPULAS
```

3. Perform a dictionary attack on this ciphertext.

```
POOAQRXNAJHEAZXOHRHOZZDJLBOTWXVWUMWDDKWPBAAHMGCWUMYOTJ
RBZLDOWTNNZROLITMOHWZMIXXKUBWNLHDYJJJEMXBWUMXUZONZNXHD
ITMWSUPHMUUZYXIAVINNLCKLLKGWJCHYQTCDLJJCQKWPDJUCITYHLJ
OEIYVAAQAYXRNWZNZONUUZAQAWQYNNFXQNOYCAYBKFBUMWFBKBMOCW
UMBATRXJPOJOBOOOADZIWABUVDPHVUUKUPARKXJRLQESITLELWPAXV
NPPCASNUACVCGIVMBPVWKRJGUHHWPYVKCDLKNADKXNJXKPMXXBAQAW
WUMWUMSADKBKINETIRBKHWZMIERWUMWLTSHLPAWTMYBDHAATPKPNHE
AWPKAAAQASMGWZAQAIZIAAHCEOVYUEL
```

4. Perform a hill-climbing attack on this ciphertext.

```
WXDJUOJOYHKPDGPTAWDWSIWPOAXVYSHXLGJGIGYONPAJPHTJCOAEGP
LOXKCYLDKCXEWALZUHSLZHTKRNEPVGJGDFKUTPAJOETZMSEPYCNLNL
FALDTQPEGKYJDIZCPUGLJASPDJWDKSLYEIZCUAGWUWIIALCOCLFASW
ALCLTOGHLNGSYOBWYJDYGGJTWWBWNRW
```

5. Break this ciphertext with the attack that matches monogram frequencies.

```
ETWZQJLEGVORLBGWDJTPGEIUZXADQQTZSTAAPXQKAGEMYMUAQDGVFB
QFZMOBEFSOQJLESOMEOQFETRCQAVEGZAVBIBCAKMFEPQKWZRZROPUP
SNGZQJSULMMAOFZMAGSQJZQQCAKMEFSQZIPGHAUPUYODWVIUZIWZQY
TWWBTREIGZAFPFJMQFLZVWZRHMKKMYWQVAZBHIZQFRLZVBTRZFZMDE
```

ZEWZQQETWGIRCQSASBZPSVPULBHGMFMGKGMAOOZMQEQGDIERGQJBIB
NTATPEPZAVFUPIGZXQHQJMAAWKKVAJHTABQJLEEWDRBGAMFNYPYMZG
WQLPMACAKMDRODGAQEPPDQWRONWBFRCFGZGALNGCFVYFZMYRLPGEEN
YPXQQYOEKMQXTZYNXBHQJAMAOOSBOUTZYJGGEQJNXVPETCFFYAOETV
EQKIFNETGUQJTFZPQEXALPQELZVPQYAQVPQEHULPTRCTGCERHAJSAE
CQSLFBSQJETRYFZMDRHMKVAGSUFOFBOA

Unit 42
Porta cipher

The modern versions of the *Porta cipher* (actually originally invented by Giovan Battista Bellaso) use a set of thirteen key alphabets. The key is again a keyword, but there are two common versions for assigning key alphabets to keyword letters. Below is their tableau. Notice that each of the key alphabets is *reciprocal*, i.e., it is its own inverse. Thus, the Porta cipher is a reciprocal cipher and is also its own inverse.

key (version)		plaintext alphabet
1	2	a b c d e f g h i j k l m n o p q r s t u v w x y z
A/B	A/B	N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
C/D	Y/Z	O P Q R S T U V W X Y Z N M A B C D E F G H I J K L
E/F	W/X	P Q R S T U V W X Y Z N O L M A B C D E F G H I J K
G/H	U/V	Q R S T U V W X Y Z N O P K L M A B C D E F G H I J
I/J	S/T	R S T U V W X Y Z N O P Q J K L M A B C D E F G H I
K/L	Q/R	S T U V W X Y Z N O P Q R I J K L M A B C D E F G H
M/N	O/P	T U V W X Y Z N O P Q R S H I J K L M A B C D E F G
O/P	M/N	U V W X Y Z N O P Q R S T G H I J K L M A B C D E F
Q/R	K/L	V W X Y Z N O P Q R S T U F G H I J K L M A B C D E
S/T	I/J	W X Y Z N O P Q R S T U V E F G H I J K L M A B C D
U/V	G/H	X Y Z N O P Q R S T U V W D E F G H I J K L M A B C
W/X	E/F	Y Z N O P Q R S T U V W X C D E F G H I J K L M A B
Y/Z	C/D	Z N O P Q R S T U V W X Y B C D E F G H I J K L M A

Let’s work through an example with each version. Here is a short message, which we encipher with the keyword PORTA.

plaintext:	GIOVANNI DELLA PORTA PUBLISHED IN FIFTEEN SIXTY-THREE
key letters:	PORTAPOR TAPOR TAPOR TAPORTAPO RT APORTAP ORTAP ORTAP
version 1:	NPGMNGGQ ZRSSV GBKMV GHVSQJUYX QE SPZLNRG LQBGE MPIRY
version 2:	ZOJENHHN URRRS LBLAS LHURNBUXW NJ SOYBVRH MNGGF AZARX

The modern versions of the Porta cipher are descendants of ciphers invented by Bellaso. Recently, his first cipher (from 1552) was discovered in Venice, Italy. Here is its tableau:

key	plaintext alphabet a b c d e f g h i l m n o p q r s t u x y z
A	N O P Q R S T U X Y Z A B C D E F G H I L M
E	Z N O P Q R S T U X Y B C D E F G H I L M A
I	Y Z N O P Q R S T U X C D E F G H I L M A B
O	X Y Z N O P Q R S T U D E F G H I L M A B C
U	U X Y Z N O P Q R S T E F G H I L M A B C D
B	T U X Y Z N O P Q R S F G H I L M A B C D E
C	S T U X Y Z N O P Q R G H I L M A B C D E F
D	R S T U X Y Z N O P Q H I L M A B C D E F G
F	Q R S T U X Y Z N O P I L M A B C D E F G H
G	P Q R S T U X Y Z N O L M A B C D E F G H I
H	O P Q R S T U X Y Z N M A B C D E F G H I L
L	M L I H G F E D C B A Z Y X U T S R Q P O N
M	A M L I H G F E D C B Y X U T S R Q P O N Z
N	B A M L I H G F E D C X U T S R Q P O N Z Y
P	C B A M L I H G F E D U T S R Q P O N Z Y X
Q	D C B A M L I H G F E T S R Q P O N Z Y X U
R	E D C B A M L I H G F S R Q P O N Z Y X U T
S	F E D C B A M L I H G R Q P O N Z Y X U T S
T	G F E D C B A M L I H Q P O N Z Y X U T S R
X	H G F E D C B A M L I P O N Z Y X U T S R Q
Y	I H G F E D C B A M L O N Z Y X U T S R Q P
Z	L I H G F E D C B A M N Z Y X U T S R Q P O

He used a 22-letter alphabet, which was all the rage in Italy at the time. Notice that all of the key alphabets are reciprocal, and so the cipher itself is also reciprocal. Notice also that there is only one key letter assigned to each alphabet, so that keywords are unambiguous.

We can modernize the *Bellaso 1552 cipher* by using the 26-letter English alphabet and putting the key letters into standard order to get this tableau:

key	plaintext alphabet a b c d e f g h i j k l m n o p q r s t u v w x y z
A	N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
B	Z N O P Q R S T U V W X Y B C D E F G H I J K L M A
C	Y Z N O P Q R S T U V W X C D E F G H I J K L M A B
D	X Y Z N O P Q R S T U V W D E F G H I J K L M A B C

```
E   WXYZNOPQRSTUVEFGHIJKLMABCD
F   VWXYZNOPQRSTUFGHIJKLMABCDE
G   UVWXYZNOPQRSTGHIJKLMABCDEF
H   TUVWXYZNOPQRSHIJKLMABCDEFG
I   STUVWXYZNOPQRIJKLMABCDEFGH
J   RSTUVWXYZNOPQJKLMABCDEFGHI
K   QRSTUVWXYZNOPKLMABCDEFGHIJ
L   PQRSTUVWXYZNOLMABCDEFGHIJK
M   OPQRSTUVWXYZNMABCDEFGHIJKL
N   MLKJIHGFEDCBAZYXWVUTSRQPON
O   AMLKJIHGFEDCBYXWVUTSRQPONZ
P   BAMLKJIHGFEDCXWVUTSRQPONZY
Q   CBAMLKJIHGFEDWVUTSRQPONZYX
R   DCBAMLKJIHGFEVUTSRQPONZYXW
S   EDCBAMLKJIHGFUTSRQPONZYXWV
T   FEDCBAMLKJIHGTSRQPONZYXWVU
U   GFEDCBAMLKJIHSRQPONZYXWVUT
V   HGFEDCBAMLKJIRQPONZYXWVUTS
W   IHGFEDCBAMLKJQPONZYXWVUTSR
X   JIHGFEDCBAMLKPONZYXWVUTSRQ
Y   KJIHGFEDCBAMLONZYXWVUTSRQP
Z   LKJIHGFEDCBAMNZYXWVUTSRQPO
```

Here is a short example of the encipherment of a message with this cipher. The keyword is PLAGIA.

```
plaintext:     BELLASO BEAT YOU TO IT BY ELEVEN YEARS
key letters:   PLAGIAP LAGI APL AG IA PL AGIAPL AGIAP
ciphertext:    ATYSSFW QRUB LWF GH NG AJ RSWIKL LYSES
```

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 119-121.

Practical Cryptography, practicalcryptography.com/ciphers/porta-cipher

Paolo Bonavoglia, "Trithemius, Bellaso, Vigenère: Origins of the Polyalphabetic Ciphers," Proceedings of the 3rd International Conference on Historical Cryptology, 2020, ep.liu.se/ecp/171/007/ecp2020_171_007.pdf, DOI: 10.3384/ecp2020171007

Paolo Bonavoglia, "Bellaso's 1552 cipher recovered in Venice," *Cryptologia* 43:6 (2019) 459-465, DOI: 10.1080/01611194.2019.1596181

Augusto Buonafalce, "Bellaso's Reciprocal Ciphers," *Cryptologia* 30:1 (2006) 39-51, DOI: 10.1080/01611190500383581

Giambattista della Porta [Giovanni Battista della Porta] [Ioan. Baptista Porta], *De Furtivis Literarum Notis*, Naples [Neapoli]: Ioa. Maria Scotus, 1563, HDL: 2027/gri.ark:/13960/t37142x6g, book 2 chapter XVI.

Programming tasks

1. Write a function to encipher a plaintext with the Porta cipher and a given keyword. Allow for the possibility of choosing the version of the tableau. You can hard-code the tableau into your code, or you can find a way to generate key alphabets algorithmically or with shifts, or you can use the magic of modular arithmetic.

2. Write a function to decipher a plaintext with the Porta cipher and a given keyword.

3. Write a function or script to brute-force a ciphertext that was encrypted with the Porta cipher. Note that you only need to consider a subset of the alphabet when generating keywords. Use tetragram fitness.

4. Write a script to search your word list for words that match a keyword. For example, if your brute-force attack finds the keyword `SECQES`, then `SECRET` matches because it gives the same set of key alphabets.

5. Write a function or script to perform a dictionary attack on the Porta cipher. Use tetragram fitness.

6. Make a copy of your code for the hill-climbing attack on the Vigenère cipher and modify it to attack the Porta cipher.

7. Make a copy of your code that attacks the Vigenère cipher as a collection of Caesar ciphers and modify it to attack the Porta cipher as a set of monoalphabetic substitutions. You will not be able to use your Caesar cracker as part of this attack.

8. Implement an encryptor for the modernized Bellaso 1552 cipher.

9. Implement a decryptor for the modernized Bellaso 1552 cipher.

10. Implement a brute-force attack on the modernized Bellaso 1552 cipher.

11. Implement a dictionary attack on the modernized Bellaso 1552 cipher.

12. Implement the hill-climbing attack on the modernized Bellaso 1552 cipher. Feel free to copy and modify your attack from Exercise 6.

13. Implement an attack on the modernized Bellaso 1552 cipher that is similar to the attack in Exercise 7.

Exercises

1. Encipher this text in version one with the keyword KEYWORD.

   ```
   The Doors were a rock band from Los Angeles. The name
   was taken from the title of Huxley's book The Doors of
   Perception, perhaps because the band enjoyed
   hallucinogenic drugs.
   ```

2. Decipher this text in version two with the keyword CIPHER.

   ```
   KNRVTOIJAQDBUEAEIJKETYDCHWHRDCFWZEIJCTTVDJWWLEJIPJXOLZ
   UYNMYGHQXMTIPXRELAKQXHPBTNRSIBXNZSGQGRHJSWUINEWQMAIEOT
   IEZXWJKJTHPGCLTVJUWCRSIBXNRXOGUEARPUUKFEQQUPNJYMQCIKYQ
   IYQBWNHKROWSPCODIZQYOJAJRUOQSBCIPKHBZEIJSWFUIIISBPXVTI
   HQXZTBMFYDTYTKRXLJYWH
   ```

3. Brute-force this ciphertext and find a short English word that could be the keyword.

   ```
   RGKYIPFKZHFKJONSGYELLKNIIPMUYEIYZAYYEHRLNUEXYUEUULFORX
   NKFHVLFKWKNUJMQUKUIYLGKPZERGKYIPFKZHFKJWFTNPEUAPZYMUIP
   NMCHOLKEUYJCQPYOVUTYJPKYWLCMFTWMYOFKYHVIUYVYEMWGNDRLKP
   ENKKRTWGZZRGRLQPEEFAIOFTNLZYYHILGYYPOPYKFHVLJAYOWLXUKO
   IHFTJVNXIHFTJUEXYSFLNMJKNJLPIYZPOZNKNGKXFHIZLGYMRHEUUP
   KEJHXYJAIYKHYHELRXNKAONKNEFAAPUSXYRGJMWSUPENCHLKEYAXFH
   IUEXRMJUGINUIUEWNVNZFKNTWRRGPEFAIILKYOWLNIWGNSZHFKJUUL
   FREHAGWLJSWVZHFKJWFGKURGGUKMNKELFZJJLUIYFKFVUHENGUEYUL
   WGZUIYWWUUJLRWYOFPYYWXFHITWEYHEMWPEMAHUUINNIWGNSJLRDJT
   WSUHEYJHILFTNHKONKYHEZRNLKWMRHEWQHFLNIWGNSJCRMQLKKWPPO
   KHIWLKMYZMFIJMFLLPKEFAIXNWFKWMRGPGNYZLWWYHIXRHEXFHILWK
   NPZYWSOHIWUHJYKLWGZYEMIEAUCLAPKOJIWWNWFGJMIURGKLKONEVU
   CWFGJPJMFZWZNCUUINNIWGNSJHITLSKPGSNLVUUSGUEYULKOWMOHUX
   RGFGKONTJYUBNLWGZKNLKUPURGJMFGNLRXNHOMQYZHFKFINGRGP
   ```

4. Perform a dictionary attack on this ciphertext.

   ```
   SOOKZSUIFAEPAXWXAMUIGCWMJYFAQNKMNCENRBEUDJAIFQEARHKAWT
   BMKKDPXXIKAOBTEUAEYXIBNPPRNAEARVNBWLHILAVMQXYYAMBXXZTM
   PWZKACBAQRFUKRZSOROAFKMTBOZVAUCNNKBTJOEXACSOOKZSUIFAEE
   YTKWANUUWTYEKUFCVDZWNBAUCNNKBTJOEXPPAHZKACBIOWSCRVURED
   ZVIDEEZVUKAZCNNHVPJZOAAORAFLDPZRWPZMFOEXFWVSKKENZWNKHT
   AWFKDMLXEZMVBDFKRTJIIQWCAIIVRDPOZZMVUIFAEPAXWXAMUVQKWR
   VYFAQNKMNCEEYXCDETBJWTSTTIEKNXTTUPKPJWVZDCKLNURMKLJUAS
   KBXPSSDACPOCXXVZDCKLJRZDKSWOSPAIFQOALXWANMAXJLORZILB
   ```

5. Perform the hill-climbing attack on this ciphertext and find an English word that could be the keyword.

   ```
   TQWUPJPGIAHRFZUVNKYQHXXKFZJKEDXVLZCYCUXBIYQDJADEUYXROU
   QYMVNDFQAHYBKZRPYBZLIDNYCNDYTQGDAKYACGSCSVMMMDYPXDARVV
   HPXWDAYGYRCKILXAZKCSVRHVHPTCHPUYXBDAFGARIVGLIAPEQVRRSZ
   ```

```
PYXJGKCUXJIAZFVVSKKZCVGCTZMVOKCJMTYJNZVAPRHZWTJBHGSZBV
PLIVCTCUJRHBYGLVPPZWIAYCQYXBTXBK
```

6. Attack this ciphertext with your code that treats the Porta cipher as a set of monoalphabetic substitutions. Find an English word that could be the keyword.

```
IBPTSPAHRUDMEEESWPCGVONLETNDXATPPARNHAYQVZSGHDUFAVVULM
BRXGRXMESXSWDCQOYYVECVQOGYPJNDORJRXASOEVOHVFRNBDNVRMBK
OUVPOETDCRTEERSTNHBSVPNGVXGBNNBEOECDEUOJBWYDBZSQFDMQRE
BAIWREOZDQUYSIBDCSOSJGVGEUODUKWNBIZBAPSBAHF
```

7. Decipher this ciphertext with the modernized Bellaso 1552 cipher and keyword `PLAGIARIZE`.

```
OTYSBUMXDISETKWNPRLEOWBYDRREWFRTZYSYMBUNTHNLJHRYTNSEBS
NIMMRNXSRSQUUQMNSWRCSFDQVFRWRZNEQBFIKPGSNGMMLIZONGNRNW
WJRMYYSAXBERXVSKJZDIIKHPGPAUUEDYBORMJJRNUNRMUPRNVVENRM
ZYEUMIMCJXELBOUJBABDAYENLMDNXSBZRVVWVWGSGHRRPZHZKSVWSG
IJNRSGRKGAMSUCKDVLSVANUQWFTOBVPELJCJSKNRVVVWGSVUQJDGVW
LOVKWQIBHMKLOYXBRWDJBHVMNAPZHRXLBWWAPALXTMNXNADBTIBNYE
ANIVRQBEQHGBORHWXHUYMRANICWFRBWEQWHRRQRZJEMEHUDXFUEVPX
DISEFHRRXWLISPTHSFAJJKWCUHQZMAVZKSVWSGIJNKWWVLABVYVRXO
NGGXMGV
```

8. Break this ciphertext that was encrypted with the modernized Bellaso 1552 cipher. What is the keyword?

```
YITCPACDICCCRDYCDGOZPSNLGLNOTYRDPASGSHNQYSCKWOYACZJGOR
NTMFWUHGSRJRVDNPYFDSYLOIARCDANDXESSNNSJRDDHMSSONJYCZJB
WSBPILQVWBOIGGWLNVYYLFDHTOGGLHRHQFXKCRNLXTADSGPAXGKVHW
XWATAMHSTGPGOQJKMTAMGARDPETPUPNJFLPMKEYANDPBMVRWIUKLPY
VQASVLUBCTQHORTWNWUIBAYYBXNUXYNTKCNDSFDHGVMSSAGIJBPYHC
HYBRTGVQRHWXXVSRRYQKLPANSJRXTCRPKWTPEKIAXHIEFXWDGOHRDY
GVZRKCHWFNBZGTKTPEVXRMDQEKRGRGCRCYYLKWVNICKDJENTSAGRNT
IXSIGRFXAXHTAFQJOWPPSCKSVSRXSXDGHGPPJRSYFAJCDNQESJHHZP
BRWJODDPFXJVNJEKCHWUIBAIGKDDHOPTDCVRFCSSVSRVHRTXDHTOJL
DTZFCKBWVRCSFCDTFTYBKDPJFHEQWINEDRRVDTKKJGVLUBCGKWJQCP
YDQGRGPBAGGRCIGKNERTTYICCPKOJXVTAGGTRGKRTHXXWHGSGNHGLN
OHXJCWYBOPGVSUSGUWXYGLYQGOREBYCGWPQKCGGWDJTEJTWDANSARV
PEPTMDGYACLJDMLEKISKOHPBCJBOONNSAPCCTCDYKKNPKOJXVYYHAS
HNJTLTYJKNGXSARVSRGGGZHPBAQFMVTBTSALHDABLFRDORHTMKXYGB
TYRGCGNSARDPEFCAOKANOSATHYFLPLWUWQMDGKSYFLDPKTPEYPUZIW
BOPCZFDYHDPKHYACQFXKWYVSJRNDXPPIQEPVTYTQXRRPASHGOBSGXJ
SSVDSGBZERTPKRNZRSEJXSSRPPGKGMCHYFXGOGSYNWUCTFHTGWINTI
PHNSNREJXGOGSZJRDPEBPOBIIYVLGBVSNCDLMUGEBOSHGSHGYYGVYE
KYYMXDJCLJTGTACJLKHIBMDAOKABTSFSETIKCHRDPZFHBBMWFZXXJL
DABAAXHYFODXXKHGLPLWKHGXYTMMPNPSJDGQGFHGKZYFOPKWUIGLPY
BLPBTIGKJGNOHKWNGCHYSKNQGLPTQXRRPALBURUNSFSGARKIPBWURI
ACDKIYBYTRCIFLDRDHSNYPMKGC
```

Unit 43
Periodic affine cipher

Suppose we construct a periodic polyalphabetic substitution cipher in which the key alphabets are generated as if for an affine cipher. Then we have a *periodic affine cipher*. The key for such a cipher is a set of pairs of integers.

For example, suppose we want to encipher a message with a period of three, and want to use these affine keys (multipliers and shifts): 5, 8; 11, 2; and 21, 18. The key alphabets are these:

```
     abcdefghijklmnopqrstuvwxyz
-----------------------------------
0    INSXCHMRWBGLQVAFKPUZEJOTYD
1    CNYJUFQBMXITEPALWHSDOZKVGR
2    SNIDYTOJEZUPKFAVQLGBWRMHCX
```

And here we encipher a short message:

```
ANY MONOALPHABETIC SUBSTITUTION CIPHER CAN BE USED PERIODICALLY
012 012012012012 201201201201 201201 201 20 1201 201201201201
IPC QAFACPFBSNUBWY GENGZMBEDEAP IWLJCH IIP NC OGCJ VCHEAJESCPLG
```

When the multipliers of all of the affine ciphers are the same, then we have a special case in which the cipher can be factored into a single affine cipher followed by a Vigenère cipher. Decipherment is in the opposite order. There are 26 choices for the affine cipher, for the 26 choices of the shift. For each of those choices there is one Vigenère key so that the combination of ciphers is equivalent to the original periodic affine cipher.

Programming tasks

1. Write a function or script to encipher a text with a periodic affine cipher with a given key.
2. Write a function or script to decipher a text with a periodic affine cipher with a given key.

Exercises

1. Encipher this text with key 3, 4; 5, 6; 7, 8; 9, 10; 11, 12; 25, 24.

   ```
   AN AFFINE PLANE IS A SET OF POINTS AND A SET OF LINES
   SUCH THAT ANY TWO POINTS LIE ON EXACTLY ONE LINE AND
   GIVEN A LINE AND A POINT NOT ON IT THERE IS EXACTLY
   ONE OTHER LINE THROUGH THAT POINT THAT DOES NOT
   INTERSECT THE FIRST LINE
   ```

2. Decipher this ciphertext with key 11, 9; 9, 8; 7, 6; 5, 4; 3, 2.

   ```
   HTEWPQSVNSRMGRWXEATJNAVYCKGVYEJFIVXBFINOOSDWQUSGAHBEQQ
   MJVYSPQCCWFOIXYPBYEWBAJJNCKTGQEVADBOHNFYAWCJ
   ```

madness's book on classical cryptography
unit 44: attacking the periodic affine cipher as a collection of affine ciphers
last modified 2020-07-08

Unit 44
Attacking the periodic affine cipher as a collection of affine ciphers

In Unit 38 we built an attack on the Vigenère cipher by partitioning the ciphertext into slices, each of which was encrypted with the same Caesar shift cipher. We then used the attack from Unit 19 to break each of the Caesar ciphers by using monogram frequencies. Here, we will do the analogous thing and partition the ciphertext, but use the technique from Unit 25 to break each affine cipher with monogram frequencies.

Programming tasks

1. Implement the attack.

Exercises

1. Break this ciphertext:

```
EMNMGYUQNIXTEMNMGYUQNVGUKOYJUZKRKCINNCEKZLGLGXMEUJXKFE
UKYJUNJEXKRETSGEMKQNLUUPGQXCVOAXYCHDKJXHUPGQXMAKPNAXAL
QSUHFVWJYVSKWDSFGEUETDQNQKEMXAUTYXSKWRYGLECDSMOCWDAMQL
CRXPUMDLAMQLCLJPUMIDAIZTGEJLGKHENCDPVUXXGYDLAOSCGLFITK
WJWCIQQMOSQNOETIMEVZAPUCVSGUSKWKGVTYVCOSQNDQEMXTVUXFUO
QDDKHLCXSLVEASYVZUCLELVEYEWMAMPJXCMKNQXOUCGQUHFAKPALVE
OETKAVCCNCDXQUXEUJGDWWVRKGXOKCVZAUOYUJKCTQVZAWKRTCAXQL
GJPCIAJJGVLLQSFSXZACXHFCGQOEUZFCTYPRGGAVZCVUYVUCQSTRUE
VETDVKYUVDZLGNUXUIVCYVQUKLJHGKM
```

2. Break this ciphertext from the 2017 British National Cipher Challenge. You will notice something about the resulting key. Can you factor this cipher? Can you find a meaningful keyword (it won’t be in English).

```
YKUFRQHUDDQRGZPPISMXOOEYZUDOGMPPRLZQAERLOPNLOVKTVFNXOY
ZCQDZOVDVECXUAIBKVGLBEJRVPSLARHDTWOBNIMLWDFGGTAUQGONBL
RFLNMGMDRPYBVIGKXVXDBOQXUGBUVOKLQLPGHROVAMGIUKRDPHDOQE
```

OEIGIUKXIVDLYJLZIPEGELWVFDBMDBSGZKCGSYSOOOIOEOBOPTEMMK
HCROCVNLWDJOKSIGWCODTJGZCVLSYOYVTOURIWMHFDLZQKUAIIAWBO
OAMKHDNGEQBTOEICUGKFGYUCZIESQPUGAKANFMELCLZTMVINRRIRDB
OBVGHJBQYHGCKMSJILPIZQJLXEELPZJSNOYCQIMJMDHKIPIAWBOAOB
MBOLWESOGLYVOABXCENDBOQXUGBUVNJCDNVBQUUQGVAJTCLIOXGBJV
PUBAUTDSQYCTMNNTRHDAUFEKVPFBSNVYVWAXKZQJAAMMRIZOVJGMMU
MXELJTHYBAJVCPKLWHMRBVCMWIGKQRNVLYROVAONGUKWBOXGGDMVTG
JJOKEUMCTGDYDHPAYTVGNKAUBSFNDMYGNEEBODYLSIVMWEIUHOIYXI
IXRZILMINLCRNYBODTQGKACUKPJNVLYROVGSBIGCRKFYRNLDPIAEGE
EZSHVGMROREIAWGMEFFKAILLHWPUMTIPJLBEQBADDORIELFILZDJSN
CTQCTMEGRJAAVQQWEIAVTIBNFWAOFTGLYDDLEERFYBGUOVCMWTLLDQ
AWYZOQSQENAPVPYRHGBKXNTOUXTXWXOXGNISDDMOIRVXRICLWKDZEI
WINWCUNDYJSNQYWWIRVXSYBOTNVHFLAUTECBGYBODTKGBSTLLYIUDE
LGTIAKCVTIYKNGLLQMQIINGXKAOHURMNQSINWTLXXSGAAQBHSGIUQQ
CPLNATQBHNNJECMKCRIPGXIVHGMDJOKPQJWYBXLIUODKUQMSZSDYRO
XAIMMSIFLXYEISEHTOACTDJAOXGKSMVANPIRDRYRUOBROTHJCDRDJQ
NNTOULKGHHSSCGKFNYZROVGSBIGCRKZUKSOEOZGGKUMCBKUAPBLWIS
CKEIDPFERPSEJHJCGXPGRPUOIKRSOIDKBEBYZKDLYNTAPXIEKAEFRG
BBLIYVDDTJRICERAQOVCMWUOIRENVCQROPGAKVVVSRICNBOHXDOBIR
WEMPDBAOQVZGNKRIPJICEHQMWBOPIUCQIYJTGBUHELQGFSYSOYEIIE
SQPXIELZOODBONLPAGJJOKQIGCJKONIGDCGQKXKLZOIRVDAKPAUKWX
IEAILLHWMDJOKPQJWIZOQVHPCVUCQOOUZMRTOHJCPVRFLGAQTQANTO
DDTACNNNZIIAZEYETUFEANATGXUQGAUTNSBKIEFIIAZEBOQNIRDNOI
ZYNHDOGIEEGSLNZLYFXOIHGTPGYHXYZIGTHWWETIRQNFFBQNPDYODR
XSQGHQGYGZYIDMBVGXPOGZRRGZCPIRLEUMCTGDMNKQDKRVFIHCSWDT
UERPAXIPOPNHJQENTPXIGVUSYQZGUNTRHMIUUWYJPIPQUNKOUPUFEN
FWAGXUQGAUBTNWCZKDPIGDKYUQGHEOGCTMEGXSGZGOIYOKPGMJDUEQ
VJOKDDPOGGXNDSJAFLEYVVDIVALDZGNAUXECFRGBROVSGVIWPEXFPG
PLTUEIWKEYWXOXGYQRIHJCTDCMULMLOKBIGPXGUUWYJEDLZLXRJEMD
CMRNZDSQUQEPETUKBNPQGZCTCTGGGFIDRVPGHKSFAWQWTEFKOXEPRO
VHBAKUCRCLZUMCBPBMWJOKQIGCJMBMXASACNMPEIEAIYSUYBDIGEGB
OXGZIGWBKXKQKFLNHEBVRIIMNCUFEUHCDDQNTTOKTQMUQKMHGURLGS
KUOCLRICSGJTYSMZWRWMKPWEZQMDSMDIMXWULNEZJOVKWWJYKYRFBF
OJYVNGPIOKQOYVGIRGNQPXINKQYAEHYIAWBOEMSKRSLZSQPOVTIDDT
WWJNFCQNQAGBCULZEDNVYBALDQGKVAIISKMEZJTKGBMAEKOIVSRGMP
XDLBGDLZQMPIAQRGZIUJLLZQHCVOBUQFMPJLRFLNDTWQNWYBMAAISD
POYLOVGCJEUOLZQUAHOIAWBONKELZNNQIGBRGZGJJOKQIGCJQRNVDV
YKQHGBOEKOVTIGDCGQKXYYDILROVRQDEVHBVNQOMDMVZTMJOKCBZEG
ZRYTLYZAEMCBBUZHQNPYVYGG

Unit 45
Quagmire 1 cipher

The *quagmire 1 cipher* (also called polyalphabetic *type 1*) uses a mixed alphabet for the plaintext and shifted alphabets for the ciphertext. The mixed alphabet is generated from a keyword. The shifts of the ciphertext alphabets form another keyword, as they do in the Vigenère cipher.

Here is an example. The keywords are `QUAGMIRE` and `CIPHER`. First, look at the table of key alphabets. Notice that the shift keyword appears under the 'A' in the plaintext alphabet.

plaintext:	`quagmirebcdfhjklnopstvwxyz`
`C`	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
`I`	`GHIJKLMNOPQRSTUVWXYZABCDEF`
`P`	`NOPQRSTUVWXYZABCDEFGHIJKLM`
`H`	`FGHIJKLMNOPQRSTUVWXYZABCDE`
`E`	`CDEFGHIJKLMNOPQRSTUVWXYZAB`
`R`	`PQRSTUVWXYZABCDEFGHIJKLMNO`

Now we encipher a short message with this key table. The center row indexes the key alphabet.

```
THIS MESSAGE IS ENCRYPTED WITH A QUAGMIRE CIPHER
CIPH ERCIPHE RC IPHERCIPH ERCI P HERCIPHE RCIPHE
USSY GWTZPIJ UT NDOINSAUP YUUS P FDRDKSLJ YFYZMI
```

There are some special cases of the quagmire 1 cipher:

- period = 1: monoalphabetic substitution in which the keyword is used to generate the plaintext alphabetic
- mixed alphabet = regular alphabet: Vigenère cipher
- mixed alphabet generated as in affine cipher: periodic affine cipher

Because the plaintext alphabet is the only one that is mixed, the quagmire 1 cipher can be factored into a monoalphabetic substitution cipher followed by a Vigenère cipher. However, the key for that monoalphabetic substitution is the inverse of the quagmire's plaintext alphabet, and the Vigenère

key must be shifted by a Caesar shift until its first letter is ‘A.’. For the example above, the substitution key is

```
CIJKHLDMFNOPEQRSAGTUBVWXYZ
```

and the Vigenère key is `CIPHER` shifted by a Caesar shift two steps back to `AGNFCP`.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XVIII.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/QuagmireI.pdf

Programming tasks

1. Write a function that takes the keywords for a quagmire 1 cipher and outputs the key alphabets for the periodic polyalphabetic substitution cipher.
2. Write a function or script to encipher a text with the quagmire 1 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.
3. Write a function or script to decipher a text with the quagmire 1 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.
4. Write a function or script to perform a dictionary attack on a ciphertext encrypted with a quagmire 1 cipher.
5. Write a function that takes the two quagmire 1 keywords and outputs the keys for the monoalphabetic substitution and Vigenère ciphers into which the quagmire can be factored.

Exercises

1. Verify that the example above does indeed factor into a monoalphabetic substitution and a Vigenère, and that the keys are the ones given.
2. Encipher this text with the keywords `ULTIMATE` (alphabet) and `QUESTION` (shifts).

```
O PEOPLE WAITING IN THE SHADOW OF DEEP THOUGHT!
HONOURED DESCENDANTS OF VROOMFONDEL AND MAJIKTHISE,
THE GREATEST AND MOST TRULY INTERESTING PUNDITS THE
UNIVERSE HAS EVER KNOWN... THE TIME OF WAITING IS
OVER! SEVEN AND A HALF MILLION YEARS OUR RACE HAS
WAITED FOR THIS GREAT ANSWER! NEVER AGAIN, NEVER AGAIN
```

```
WILL WE WAKE UP IN THE MORNING AND THINK WHO AM I?
WHAT IS MY PURPOSE IN LIFE? DOES IT REALLY, COSMICALLY
SPEAKING, MATTER IF I DON'T GET UP AND GO TO WORK? FOR
TODAY WE WILL FINALLY LEARN ONCE AND FOR ALL THE PLAIN
AND SIMPLE ANSWER TO ALL THESE NAGGING LITTLE PROBLEMS
OF LIFE, THE UNIVERSE, AND EVERYTHING!
```

3. Decipher this ciphertext with the keywords WAR (alphabet) and PEACE (shifts).

```
IMFUUBNTVNDLUSTVXIHDEFMGNDXPVMQJFJFUETUYFJBUXPXFUDPWAQ
JKJOVDXNDKHEYMGGUEOGNDIFHIOESITQJSHJDGFITQJTKJCNEGQUTG
VMUXZHSINFVMSJOVYQBZLXXXIHERWPDUINPQTVJUDTFJCZFJWSLEPS
ERKIMFEFYXJUMURQLFUGXVMUYOLXUISVEIJSVDETGVMUXIDJUJWLWI
NOJUEDFDWUYBCWYEAQIEHFCSYEZHFUEMDJPIXHKVJDVNKJMZNDGFLS
WXIHXXNBGJPWTCWYEPQQMJNHFWJECWPIJUXYSDVYDNTCKIJBCSEXIH
FTJDCIUTGFTDKUUJTKJJMINOJXXJGDTDXJHFHGFVDUJOVMUXIDJUWU
SJQWTCXUWADJYSSRRUUMCHUWADGYXBCFMEOGNDTTKJQWTKJMKMWHIY
AVJPHDRFTNOJXEXIHKEFTWSUWPIDPFCWXYSHHSUFAOXXJXITBQPBLU
THDEFMJFEBQJQJHJUDEHNAFTCUBLWUWTKJOMPOJEKTKJDTBVMUFOSE
QXPIXXJFWFEUFCSPSECWYETLHBEOGRPWSIFERQRFIYHCQITTKJRJBL
SWWTDEYXPFJPSJCHERQDNHJSVMUENHFYHAUXXJAVQPSTLHYWMCSTWJ
QHBYELSWXIHGQNTLWXNSOJHEUUXQEMCWYEAQIIMFUTJXIHFDUPDXYT
ORKPKBLHPJAUXPWJCWCEMOJQXICSIMFRXXJBUEDIZLXXEMHWHIFIND
NTHDUWTHFDKBRSINFDHERQDNHJSFMYSACSTXIHHEYOVFYJSVTIMFUT
JXIRKYX
```

4. Break this ciphertext with a dictionary attack. Both keywords are common five-letter English words.

```
VXUHEXTOLHJMJVCVCHQCQENAZOZWKOMQNHEJRGQUAOJVCYZFAGVFXA
UAMDQCKZMQKUTYGVEXDENTGGQKFCGCNEJIVIMXDBNVROMESMYCBTQI
TTTULNFLHTQMKZNCSYOJFUAMJVCVCGMVXTFAULMWSJVPYHCUOSMEKO
JFHTXYCTUOSMEKOWATLFUIHLTGXEEXBAEJNWACWOJFHTXYCGAURFST
PBAEJWNCMBZNDPBRTDUOOJSUGARCZOUYHHTXYCGAURFSTPIDHTAXIN
GPXCSTDYINEHTQTTENAZJMYNPKEZOKYETZCVLRAUGCGCHGCBQKLMFA
TBRGRUAMDVCVCFLEXWBHKXTORTMQNCBTOXSNQQTLPMCOMTGNRCHBYI
ZCLO
```

Unit 46
Two-stage attack on the quagmire 1 cipher

Because the quagmire 1 can be factored into a monoalphabetic substitution followed by a Vigenère cipher, the shifts are exposed to cryptanalysis. In this attack, we find the shifts by comparing the monogram frequencies of slices of the ciphertext. What remains then is a monoalphabetic substitution, which we can break with the technique in Unit 28.

Here's how the attack works:

1. find the period *m*
2. partition the ciphertext into *m* slices; each slice contains every m^{th} letter, but each slice starts from a different one of the first *m* letter of the ciphertext
3. find the monogram frequencies for each slice
4. fix the frequency table for the first slice; for the others, shift them left (with wrap-around) until they match as closely as possible the table of the first slice
5. the shifts needed to align the frequency tables forms a Vigenère key; decipher the ciphertext with this key
6. break the resulting text as a monoalphabetic substitution

To recover the original keywords of the quagmire cipher, look at the key that you find for the monoalphabetic substitution. Take the value of its first letter, where we start at 'A' = 0, 'B' = 1, etc. Add that value to each of the shifts that you found in step 4. Convert the resulting shifts to letters to get the shift key of the quagmire. To find the alphabet keyword, invert the key of the monoalphabetic substitution. You may not always get a recognizable keyword, due to problems with infrequent letters.

Programming tasks

1. Implement the attack. Use the cosine of the angle between vectors to find the best match for the frequency tables. Feel free to also use your function that performs the hill-climbing attack on the monoalphabetic substitution cipher.

Exercises

1. Break this ciphertext and find the original quagmire keywords.

```
TNBOWMQSCQFMFOBZVKYAVNUBJVKSLLKESDBGLJPVFNEFLHUGNKVRDO
QCVKNRKJFXWKIUDNVWBTBHPSESLTBJIVUHUIIAGGKMQCMINEOUAMYO
NMATPOJVSSLXLNOFJAVFDSDEXBGTUUENVNHDXILUDYXGFOJRNCHRJO
TVOZLBMCKJBUIUJRPNRGOPBTTCXIETUPBCJBWEWOVUAMVDAGEIKFZL
LPSJVHPSMLQSSSLNHKMKZSMLYACCKGPUZAEWKBBLLSXIXTPGMTCSCC
XEFQEXIPPGKJFSBCLKPFSWBTDXYMRIVPPSACLVPFSVFSGLICHFPHPS
ACLXLKGBFOBRVVEGDEBTACLKPRKGPXVMVPZSCAGPCMQSCEEGUPEHUU
PVNJBUETZWZGEIUPRTWDPREEYUPVYGEWAEUSKXNHZFFOMHWPUOYKDA
MFWNAKUEKSKFETZRZQTNJUWYBHNXNATUWHNOCPKOTFMTUMENTNJURX
UGUISOPMMZHHENVSDHMLVVENVWFCJTUUYSKCEFOMQOEXAEZXMKLVNF
SEHXKMQRFTWOBSOLHBRJFOMTRXIJPWAESFIGNHUIIETKMLHKXEGSBO
OIJREYIETKCGPIASEPFJALQSESFLHPEGHYLVYRPNWGLCSXAETKMUCS
DEJSKFXTZGPKTTXBAUHTPFVGPSIGPSXEIMBCIZLTFXKOKFZBYSLXXC
TBXIVQYXDEOUKMDODIDPUZACLUNKEOUCKELHZKIOQURXROCLFRIFIK
VVWGSBDOQLVAPJFGZVVZLFYIRTKTWFHBLMVGUPXNAQEGETPPVXVVEN
VDVQNHHFRWRSTKMALPMTRSUDB
```

Unit 47
Quagmire 2 cipher

The *quagmire 2 cipher* (also called polyalphabetic *type 2*) uses a mixed and shifted alphabet for the ciphertext. The mixing is generated from a keyword. The shifts form another keyword, as they do in the Vigenère cipher.

Here is an example. The keywords are `QUAGMIRE` and `CIPHER`. First, look at the table of key alphabets. Notice that the shift keyword appears under the 'A' in the plaintext alphabet.

plaintext:	abcdefghijklmnopqrstuvwxyz
C	CDFHJKLNOPSTVWXYZQUAGMIREB
I	IREBCDFHJKLNOPSTVWXYZQUAGM
P	PSTVWXYZQUAGMIREBCDFHJKLNO
H	HJKLNOPSTVWXYZQUAGMIREBCDF
E	EBCDFHJKLNOPSTVWXYZQUAGMIR
R	REBCDFHJKLNOPSTVWXYZQUAGMI

Now we encipher a short message with this key table. The center row indexes the key alphabet.

```
THIS MESSAGE IS ENCRYPTED WITH A QUAGMIRE CIPHER
CIPH ERCIPHE RC IPHERCIPH ERCI P HERCIPHE RCIPHE
AHQM SDUXPPF KU CIKYMYYWL GKAH P AURLOQGF BOTZNY
```

There are some special cases of the quagmire 2 cipher:

- period = 1: monoalphabetic substitution (keyword cipher)
- mixed alphabet = regular alphabet: Vigenère cipher
- mixed alphabet generated as in affine cipher: periodic affine cipher

The quagmire 2 cipher can be factored into a Vigenère cipher followed by a monoalphabetic substitution cipher. The key for the monoalphabetic substitution is the same as the mixed alphabet generated by the alphabet keyword of the quagmire, but the Vigenère key is the shift keyword of the quagmire after it is encrypted by the inverse of the monoalphabetic substitution. Because the second factor of the quagmire 2 is a substitution cipher, it is resistant to the attack in the previous unit.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XVIII.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/QuagmireII.pdf

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter XI, section II.

Programming tasks

1. Write a function that takes the keywords for a quagmire 2 cipher and outputs the key alphabets for the periodic polyalphabetic substitution cipher.

2. Write a function or script to encipher a text with the quagmire 2 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.

3. Write a function or script to decipher a text with the quagmire 2 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.

4. Write a function or script to perform a dictionary attack on a ciphertext encrypted with a quagmire 2 cipher.

5. Write a function that takes the two quagmire 2 keywords and outputs the keys for the Vigenère and monoalphabetic substitution ciphers into which the quagmire can be factored.

Exercises

1. Factor the quagmire 2 cipher used in the example above, and find the keys of the two factor ciphers.

2. Encipher this text with the keywords `CHARLES` (alphabet) and `DICKENS` (shifts).

```
IT WAS THE BEST OF TIMES IT WAS THE WORST OF TIMES IT
WAS THE AGE OF WISDOM IT WAS THE AGE OF FOOLISHNESS IT
WAS THE EPOCH OF BELIEF IT WAS THE EPOCH OF
INCREDULITY IT WAS THE SEASON OF LIGHT IT WAS THE
SEASON OF DARKNESS IT WAS THE SPRING OF HOPE IT WAS
THE WINTER OF DESPAIR
```

3. Decipher this ciphertext with the keywords PLANET (alphabet) and EARTH (shifts).

```
FAFSTXGTYRDWTCRESHFKTAUMCEVWWNRCHIMYMXTNHGBRHTJWFYCOXY
RDYWXEDSTXPISROHVKBGEHBYTWEZOOSDURXXEJNZYVBYTACBVNMMJB
LNYMBWIIVPRQXFPXHXAVJNXATCMRCFSTGFNNYIMWYFXYBQSMJPSYOG
NFNIURRTXVWWWAGTXSGMPGSBATYWIVHOMTJIFQVBWRPMATFEPFBXME
QWDMWEWRKDTNJODCBWXWAFFNRAETGIMYOFSSPQSXGJFEHAHYRDPGYS
MJHISQIVJQIVROCEVUIMWAFFHSSWYEAMWTEITWT
```

4. Break this ciphertext with a dictionary attack. Both keywords are common five-letter English words.

```
CTXMSTARXSYRSWGOUKWBRPEHCNHCRNZDXJRDAAPSCUWGQTAPUJOLGX
DGPEJACUWJDZRIZPYOKHBCNSWBGDEPBCZTNCHDOMUKFRUOYODESYOI
EWZIGGBZHIICVIBGSJMTSSSLOHQGDTMSDAVFRZMMNNJNTNBTZPMIGD
TMXCQHGBNAZGSWTOOBHDEJZCNSMSWOBOBWAOBBMOGZSULMJTNHCKAV
BXOTVSRUECSDESQATKBTABPBGHQWZSZOOVVMTJAGOCKSCKRJAKPBJE
KSGUBGRMNGKDTMSCUWJGKVIBSWUUJAHRGCLVVDENGKTGEONVRIYRVG
RGLTDNUOUDLDHBOVGDOOQOBUGAZNEOTYOROVKTDWZSHIIILRIJKNHC
EIYTTNNYAPMSJITOCKSUJADACFQKTDENGUWGLCSTRNOSKFCNUWGWYB
TGQONVOVKZOMSGRHIVGUWGWYMOHNMAOHQGODENGAIZAOFTMBJAOMZA
DXMSJIHOWNHCXAVUWGAMRXHRVUWFRMVIZSYDXDORUVMIEUWRIYUWGJ
NHQEVOTOHWTRGNIUDXNQRNVEWSLCWAKRCDDHDPJDUFDJVOMONWDACF
QKTDEWZNEOVONVOVCUWGBVPGGRGHUPZSZVFHKTRGCNACJGKUWFRMUW
WCURDPAKCOQSTAWFBHOHHCKNOHPKTTGGZUEONQAUMITMTESSADNGKB
TGQEHDEVOTWFRJTPWAZMTOVKPGWPKOUWOYODESYRIZNZMHYVCTXHVO
MGFSZMXHSZMMISUUXCWGZKFCNMMGQREOOWVNDEWZHCFCCFODCNRDIA
OAJGBTAOHQGDTNEOSXOILMMRNGDTMJNHQEWGDXHYOTKFCNHIDSDAVF
RZUEDDGHIUNMRXHBGUWFBZABQWGVSGWEHGCRVLEHUKSTHQRSTFCGDE
PUNZTOWQNESRVXXNSYARGQADESCVRJJWJHD
```

Unit 48
Quagmire 3 cipher

The *quagmire 3 cipher* (also called polyalphabetic *type 3*) uses a mixed alphabet for both the plaintext and the ciphertext. The mixing is generated from a keyword. The ciphertext alphabets are shifted versions of the mixed alphabet, and the shifts form another keyword, as they do in the Vigenère cipher.

Here is an example. The keywords are `QUAGMIRE` and `CIPHER`. First, look at the table of key alphabets. Notice that the shift keyword appears under the first letter of the alphabet key.

plaintext:	quagmirebcdfhjklnopstvwxyz
C	CDFHJKLNOPSTVWXYZQUAGMIREB
I	IREBCDFHJKLNOPSTVWXYZQUAGM
P	PSTVWXYZQUAGMIREBCDFHJKLNO
H	HJKLNOPSTVWXYZQUAGMIREBCDF
E	EBCDFHJKLNOPSTVWXYZQUAGMIR
R	REBCDFHJKLNOPSTVWXYZQUAGMI

Now we encipher a short message with this key table. The center row indexes the key alphabet.

```
THIS MESSAGE IS ENCRYPTED WITH A QUAGMIRE CIPHER
CIPH ERCIPHE RC IPHERCIPH ERCI P HERCIPHE RCIPHE
GOXI FJAYTLK FA HBVJMUZZW GFGO T HBBHCXPK LKXMSJ
```

There are some special cases of the quagmire 3 cipher:

- period = 1: monoalphabetic substitution
- period = 1 and shift keyword = first letter of alphabet keyword: no encryption
- mixed alphabet = regular alphabet: Vigenère cipher
- mixed alphabet generated as in affine cipher: Vigenère cipher (whose key is a Caesar shift of the quagmire's shift keyword)

The quagmire 3 cipher can be factored into a Vigenère cipher sandwiched between two monoalphabetic substitution ciphers. The keys for the monoalphabetic substitutions are inverses of each other. The final substitution cipher uses the same keyword as the quagmire's alphabet keyword,

while the first substitution is its inverse. The key of the Vigenère is the shift keyword of the quagmire encrypted by first of the substitution ciphers.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XVIII.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/QuagmireIII.pdf

Programming tasks

You may use the function from Exercise 1, but there are other ways to accomplish this.

1. Write a function that takes the keywords for a quagmire 3 cipher and generates the key alphabets for the periodic polyalphabetic substitution cipher.

2. Write a function or script to encipher a text with the quagmire 3 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.

3. Write a function or script to decipher a text with the quagmire 3 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.

4. Write a function or script to perform a dictionary attack on a ciphertext encrypted with a quagmire 3 cipher.

5. Write a function that takes the two quagmire 3 keywords and outputs the keys of the substitution ciphers and Vigenère cipher into which the quagmire can be factored.

Exercises

1. Factor the quagmire 3 cipher in the example above, and find the keys of the factor ciphers.

2. Encipher this text with the keywords `NURSERY` (alphabet) and `RHYME` (shifts).

```
THE CAT AND HER KITTENS THEY PUT ON THEIR MITTENS, TO
EAT A CHRISTMAS PIE. THE POOR LITTLE KITTENS THEY LOST
THEIR MITTENS, AND THEN THEY BEGAN TO CRY.
```

3. Decipher this ciphertext with the keywords `DOLPHINS` (alphabet) and `FISHBOWL` (shifts).

```
UXDAPTKWHERHJUJPNESLUUHKUCGFKOKNQFDBZFKNXSFQGUAUGVDAUO
MBIPBBLUYGVMSEJOSHUVCJQVBNICDFLBWSYQRWUSKCRRBKQSWKJCDX
UTSLGEJIZFASRMKGOBPSGEJIKUDGYCJARLRHTFOWVBNGMMEKEOOVNA
KHXVPVQEUXGFHAZSRXCUDGJCUALBLLTZSHUSYDTGPJEFHUJPDAUGOA
```

```
XSIKKJJCSMSEGBZGTNTDCTWBTBKHCXSLGEDAUJYCTFDAUZPCQIJIEL
LKUESXGBLQT
```

4. Break this ciphertext with a dictionary attack. Both keywords are common five-letter English words.

```
OHSRHJIURBYNPSWGTIMBOHFCMYNHECDEBJVUBPAAUNEJIOHFOCJLIM
TVEFKJORPKWYROCJGNAACELEFXFECRHRLPKWYR
```

Unit 49
Quagmire 4 cipher

The *quagmire 4 cipher* (also called polyalphabetic *type 4*) uses mixed alphabets for both the plaintext and the ciphertext. The mixing is generated from two keywords. The ciphertext alphabets are shifted versions of the mixed alphabet, and the shifts form another keyword, as they do in the Vigenère cipher.

Here is an example. The keywords are `QUAGMIRE`, `KEYWORD`, and `CIPHER`. First, look at the table of key alphabets. Notice that the shift keyword appears under the first letter of the alphabet key.

plaintext:	quagmirebcdfhjklnopstvwxyz
C	CFGHIJLMNPQSTUVXZKEYWORDAB
I	IJLMNPQSTUVXZKEYWORDABCFGH
P	PQSTUVXZKEYWORDABCFGHIJLMN
H	HIJLMNPQSTUVXZKEYWORDABCFG
E	EYWORDABCFGHIJLMNPQSTUVXZK
R	RDABCFGHIJLMNPQSTUVXZKEYWO

Now we encipher a short message with this key table. The center row indexes the key alphabet.

```
THIS MESSAGE IS ENCRYPTED WITH A QUAGMIRE CIPHER
CIPH ERCIPHE RC IPHERCIPH ERCI P HERCIPHE RCIPHE
WZVR RHYDSLB FY SBTAWEAZU VFWZ S HYAHNVPB JJROQA
```

There are some special cases of the quagmire 4 cipher:

- period = 1: monoalphabetic substitution
- unmixed ciphertext alphabet: quagmire 1
- unmixed plaintext alphabet: quagmire 2
- same keyword for plaintext and ciphertext alphabets: quagmire 3
- both alphabets unmixed: Vigenère cipher

The quagmire 4 cipher can be factored into a Vigenère cipher sandwiched between two monoalphabetic substitution ciphers. The final substitution cipher uses the same keyword as the quagmire's ciphertext alphabet keyword, while the first substitution is the inverse of the key generated

from the quagmire's plaintext alphabet keyword. The key of the Vigenère is the shift keyword of the quagmire encrypted by the inverse of the second substitution cipher.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XVIII.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/QuagmireIV.pdf

Programming tasks

1. Write a function that takes the keywords for a quagmire 4 cipher and generates the key alphabets for the periodic polyalphabetic substitution cipher.

2. Write a function or script to encipher a text with the quagmire 4 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.

3. Write a function or script to decipher a text with the quagmire 4 cipher and given keywords. You may use the function from Exercise 1, but there are other ways to accomplish this.

4. Write a function or script to perform a dictionary attack on a ciphertext encrypted with a quagmire 4 cipher.

5. Write a function that takes the three quagmire 4 keywords and outputs the keys of the substitution and Vigenère ciphers into which the quagmire can be factored.

Exercises

1. Factor the quagmire 4 cipher in the example above, and find the keys of the factor ciphers.

2. Encipher this text with keywords `FOUR` (plaintext alphabet), `LETTER` (ciphertext alphabet), and `WORD` (shifts).

```
SEEMS LIKE ONLY YESTERDAY I LEFT MY MIND BEHIND DOWN
IN THE GYPSY CAFE WITH A FRIEND OF A FRIEND OF MINE
WHO SAT WITH A BABY HEAVY ON HER KNEE YET SPOKE OF
LIFE MOST FREE FROM SLAVERY WITH EYES THAT SHOWED NO
TRACE OF MISERY A PHRASE IN CONNECTION FIRST WITH SHE
OCCURRED THAT LOVE IS JUST A FOUR LETTER WORD
```

3. Decipher this text with keywords `FOUR` (plaintext alphabet), `PIGMENT` (ciphertext alphabet), and `COLOR` (shifts).

```
MYCSWLPRQAJWJIFMPXASGCXCFUQNQRCQQSMZXXAPZXXTHLQMQVHYVY
CMQRCMZDZYPAYPTHQDRUYSSONWMIVQVRUNIFYCCPPLQMQVGASWPUCL
```

```
QUHXPNPZCPQBJEPTGFNSTFMPXZPZYLQVMPXSPOAPZAQTCIFDTCPPTW
ZSSYQBQAQCONPNQEYBSUDYHNQWIRCYRYBMFXSFUQNCSCDZUHRYNYWZ
XCPPHYLQVQPSFPCQQSVJDZYVMPSTHSSXYHAAXCSCWOESHSXWPPDOSF
DTXCULPFQAJTVQAPESIBQXCPWMCZYYDRMXBQFXSRUTWWTDDZYFMPSD
AJDVP
```

4. Break this ciphertext with a dictionary attack. All keywords are common five-letter English words.

```
XZPRVLJINCQXKYWVKXJJQXPRVCARJTCQSJDQCRECDGGIXTKKYWDUWJ
XBKTZCAKJAHRKMZFNMNZLBLRXVAIRZGQXGIXTKPTFNCQJUCIPCDQCI
ZSLJINCQXSJDQCRECDGRHKWMPAANXKBTCQHRQLUMHCAOVCWXZOAANI
HJVOSNWCVTPLJNCQJTQYIOHBMVWHRIIPMNAGLGOUXLGJUXJJDUKWJN
QIPVAOIOJQYIAANJYHKFXKAVCQOJGNAKWJNXKACGUPFCLCQLKQOMJD
NAKACCZRHCALRHKXZOCCLTKMDNQSZFVAKECDG
```

Unit 50
Hill-climbing attack on periodic polyalphabetic substitution ciphers

The main idea of this attack is an extension of the hill-climbing attack on the monoalphabetic substitution cipher. We find the period *m*, and so must work with *m* key alphabets. In the earlier attack, we swapped letters in the key alphabet and kept the new key only if the fitness of the deciphered text improved. Here, however, the key space has more dimensions, and it is easy to get trapped in a local maximum. To avoid this, we will take turns randomizing one of the key alphabets and climbing back up the hill.

Here is the algorithm:

1. find the period *m*
2. set big counter equal to 0
3. set best fitness equal to the fitness of the undecrypted ciphertext
4. set the parent key equal to a set of *m* key alphabets
   (best to choose key alphabets that maximize the monogram fitness of the plaintext)
5. while big counter is less than some large number
   a. for each *i* in 0, ..., m−1
      i. randomize the i^{th} key alphabet
      ii. find plaintext by deciphering the ciphertext with the parent key
      iii. set the parent's fitness as the fitness of the plaintext
      iv. set little counter to 0
      v. while little counter is less than about 1,000
         - copy the parent key into a child key
         - swap two randomly selected characters in the child's i^{th} key alphabet
         - find plaintext by deciphering the ciphertext with the child key
         - set the child's fitness as the fitness of the plaintext
         - if the child's fitness is greater than the parent's fitness
            · copy the child key into the parent key
            · copy the child's fitness into the parent's fitness
            · set little counter to 0
         - increment little counter

- if the child's fitness is greater than the best fitness
    - set the best fitness equal to the child's fitness
    - copy the child key into the best key
    - set big counter to 0
- increment big counter

6. output the best key

The limit on the big counter that we like is about 1,000,000 times the period squared.

Reading and references

Thomas Kaeding, "Slippery hill-climbing technique for ciphertext-only cryptanalysis of periodic polyalphabetic substitution ciphers," *Cryptologia* 44:3 (2020) 205-222, DOI: 10.1080/01611194.2019.1655504

Programming tasks

1. Implement the attack. You will find that when written in Python, the attack takes a long time. You may want to write another version in a lower-level language.

Exercises

1. Break this ciphertext. The keys have a hidden message for you.

```
PWNYPCGUUMYYUIAGGUGFEMWLRNYAHOGGHBAEWCXWPAQSXPAGNITNIX
WBQFKMLVIFHDTLSMADKMWWXXNYCQYBVWIKHWYARVTKXXAVGBGLVITN
WVNMKMUWRYQWRNEHYBCHUTNYQKYHIXWFWTQSLVYKRKZUHXZKVMQPHC
YALCRWHDKGJGTNOKSHXPAPHAGWQVGUHCXYUIAMXPKLXQRWKWNBXVTM
EMGGIXWYOMADHKGFWQQYQKYAHMHXRBYLRNYAHWRYXMZKFBEIXXIKWY
JWUCCARGTKNMNMLBGEHCXECDTVHLKIKMWQKWYSREOYOTYAHWWMLNAU
XVGPHBYAHTGLVTGMYCYYNMYALCTISXWMYVKMCVTQXXMKLVIHYBYQRY
GHSTGLXXIWXPGKWTQHAUGIOMALHDTXLBXMWYTEROKTHDJWWBYBIWOM
AWXIOWOWGXSMKMQNQWWLYBHYFMHYUDTHVXTGWCCWKWNYQDKULYAMHK
YAWDEHYGTNOKAUKQGPHCUYFMYKWFGEPEOAHWWELMWMKQXLKXZEGMDI
OWKGXPGYSYAKHVYYJMTXXPGYUDKXWLYYQKYAHLWNGMSWVCTXXPGEWV
INWOGQUQYMHVZIRVKMLDJYVCZXIMWWGDJWLUUYFDTXFXZGXTGLVUKU
UXRWXMTKVDJYXQRIWLYMKMQNQWWLYBHYFMOHQDKGYWQECNZKXPGKPX
WWXPGFHCXYJMCKLDYWQXSBXGALFXRIRCGVEIRSSBGVHLGLVXWLAPTQ
HBGEWBIWOIKZQXWYQDTXCXZKOWSZYWIWLUZLXWQLRWUHOXIBDMHHUD
JWGMQYCQSHYBKGXBTVYLYBRVXBQLGBXCUEWLGFHVYQKQQWWDYAWDYB
PMCWKWNIOWSGHKYHLVKMLWYWFXSMWLYBPUGVLWYWOIZIRVEHYBHBUC
YPLCKMXXEHYBUEWVGMVCAMHTQBXMKMKWNLLVOWXPAMXQRWEMGGGMOB
GMNMKWYQHCJHYTNGRDWWZMAEREWLHTBWVESMLTEHYBXHFQGMCPAVUM
AUKMNSHDAZUMAMHBQWZMQHIDGUKVTEROKUWTXHSPKLXQOYXQTGKXCW
ZMWSREWVLCOHZMWSRNYAHWWMLNAUXWSVGMOBSPGKPMSMRNKMKWXIUM
```

OBSQYYXMNMKQXBQDWHGEOMLXSBQDJWXQRWVQSUHDJWWBYBIWOMVLWW
WDKHQWSVSTAUHUGGXXSMKMQNQWWLYBHYFMCWKWBWEMGGAXWDLVIMRP
GESITNWLJBHFGMKWYMHLJGRTTZLLAEVXUALCYBFWYBRVTXAPKUKQXI
HWPBQBGUHVYVHLAVHCEHYBXHFQGMCPALHHUWUQGGFMNYQMDIRVGGXQ
AEJBTQXPKGXPGYUMALRNMBRLJWPQXMUIRWGQOBQMGGJQSWHBKGJYJS
VQOLWVNURUUNXWYBRVNNHQSEWBIWSWWMXXYAHQSCHLYBRVTXLKGYVW
SVXMOAQQVNHCKGXXEHYBXULMSMLNKUOQYWUWYNUMMSREWFHUMWUCYA
HCGYUMJHAMBWUUGKHTEMKMMWJQSGLVIHIGJYXGGUWVNHIXWSREWLRL
KWXIYHAQYQHPAPHWQLRAGWQWQOHQYLOXCECDTAHTUSRECBXPYAHQRI
UXBWPMSMRNTMKMWXWLGMVXHSREWLRLKWXIKMLCSHWLOBGMSMXPAMWK
BYQLGLLVYWFPSHOXISWBGYFLTFSWSBHKMSWKBYQLGLLVXNUFGBOTAG
FMKMLCCBXPXNFPAVZWSUHCYAWDCWKXUWXXJWOYEHYDTJYMQEXPGUKW
TLWVNNQYWWGQOMWAKELDEMKWYIOWINHCEHYBWYFMXMWAKELDEYQKOH
QDWHOWWWHCXWQDKYONTKYCJWUQSZEXYAREWIHXUEHCKGXXAGHGAZHX
HURXUWUWYBRVAGGYWHVYGKLDEYVQYLWIXHQDJWWBYBIWOMWVNYVGGQ
LTQYOGASVCASXXEHYGGURUGBQYGYFMCWRNHWUUAGCOKXXCYHCXZBQB
GMYBSQHXSECWXDCXZKRAGVLMSUH

Part IV
Transposition ciphers

madness's book on classical cryptography
unit 51: transposition ciphers
last modified 2020-10-03

Unit 51
Transposition ciphers

A *transposition cipher* is a cipher that rearranges the letters of a text, but does no substitutions.

Detecting a transposition cipher can be done by looking for a high monogram fitness but a low tetragram fitness. The index of coincidence should be close to that of English, since IoC does not depend on the order of letters. Note that for short texts, we cannot rely too much on statistics.

There is a type of transposition cipher called a *route transposition*. It involves choosing some special path through the text and adding letters to the ciphertext as we traverse that path. Since it does not typically use a mathematical system for choosing that path, we will not be discussing it any further in this book.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter III.

Programming tasks

1. Write a function to return a boolean value representing whether a given ciphertext is likely to have been encrypted with a transposition cipher.

Exercises

1. Which of these are encrypted with transposition ciphers?

    a. `IENEEMEEMIENYNEICOEMHTCAGTIAYRBETHETFEIOOEHHRLELTLESGIMH NEEOEEMIENIEEINMEMOY`

    b. `EEISEHEEISEHSIEYHJENDQNUDQSOEMAYQUEQJESRUEUJFFEMPFEQUSHO JEEISEHEEISEHSIEYHJE`

c. JMSTUYTWIBCZLCBNHIPWIBPESIUNNIQDSRSZYLDDFITCLTULTCZDPIBG
DPCZLCBNHRWMNYIDSWLTUIWMBTCSWMQEMQHYIYRYTWIBCIDMQJDAJFCO
MEDMQWIHPHODPIBIDSLCSHSYSTTPIBCIDMQYMEBHISWOBMESIDSVYTRN
PHHDPIOPIYSTTLLTUTTUWFMNY

d. ITIOIIUIALBCTIOTANTSISYDTOEOHRMLESENYROZTRNHDOTENONYSIEE
DIELAYOSSDMAIRENRBHDLFDNPMGHEATVRHNYTVITFEGHIIRREMIRUOTN
HSUIFGREALHNTKISTSSIEERGUHAOEDAEONDUENEEYPEURARSOVTSHNTT
COEMRNOPEOACIGAENEMFATUAAHNIEIDSHTIPGDSGPHDUDUETNEEAEOER
EALEHEMMNNRSYRHFGDIOHRSOAATMRHNYORTWSRIOVEUNINSNLSNMSNOE
OGNRHVMTTSEMFARTLEOIABHTUREREHPTEPLSDSRURARSGSTNDBTLUCIS
NOTMNHEEFTALTHIRBNMRNYSIEMENETEDIOFSYOAFSSEHYGRGRISYDTNO
HECELRESST

madness's book on classical cryptography
unit 52: permutations
last modified 2020-07-09

Unit 52
Permutations

Suppose we have an ordered collection of objects, such as [⌂, ○, ☼, ▲, ■, ♦]. A *permutation* of this collection is a reordering of the objects. For example, we might want to reorder them to [▲, ⌂, ○, ♦, ☼, ■]. We have moved the 0^{th} (because we start counting from zero) to the 1^{st} position, the 1^{st} to the 2^{nd} position, the 2^{nd} to the 4^{th} position, etc. We will write this permutation as (1 2 4 0 5 3).

We need a way of combining two permutations. Suppose we have permutations P and Q. The *composition* of them $P \circ Q$ is the permutation that is equivalent to permuting the set of objects first with Q and then with P. For example, (1 2 0) moves the 0^{th} object to the 1^{st} position and the 1^{st} object to the 2^{nd} position and the 2^{nd} object to the 0^{th} position, and (2 1 0) reverses the order of the three objects. The composition of the two, (2 1 0) ◦ (1 2 0) takes the 0^{th} object to the 1^{st} position and then keeps it in the 1^{st} position. It takes the 1^{st} object and moves it to the 2^{nd} position and then to the 0^{th} position. It takes the 2^{nd} object and moves it to the 0^{th} position then to the 2^{nd} position. The overall rearrangement is the same as (1 0 2).

The *identity permutation* is the permutation that doesn't do anything. It is (0 1 2 3 ...). The composition of any permutation P with the identity leaves P unchanged.

Every permutation has an inverse. The composition of a permutation with its inverse is the identity permutation. In other words, the inverse permutation undoes what the permutation does. Finding the inverse of a permutation is just like finding the inverse key for the monoalphabetic substitution. For example, let's find the inverse of (2 0 4 1 3). First, write the permutation under the identity:

(0 1 2 3 4)
(2 0 4 1 3)

Then, reorder pairs (shown with the same color) so that the second row is the identity.

(1 3 0 4 2)
(0 1 2 3 4)

We read off the inverse from the top row. The inverse of (2 0 4 1 3) is therefore (1 3 0 4 2).

The advanced reader will notice that what we have been describing is a group. A *group* is a set G and a binary operation $\cdot$ such that G is closed under the operation, the operation is associative, G contains an identity element, and every element of G has an inverse in G under the operation. The permutation group is noncommutative/nonabelian, which means that if we compose two permutations it matters in which order they are done.

We state without proof that any permutation of n objects can be generated by a series of exchanges of two objects. This also means that any permutation of n objects turned into any other permutation of n objects by composing it with permutations that each exchange two objects. We will use this fact implicitly when we attack ciphers that are based on permutations. When we attacked the monoalphabetic substitution cipher with a hill-climbing technique, we used exchanges to move from one key alphabet to another. After all, a key alphabet is merely a permutation of the regular alphabet.

The number of all possible permutations of n objects is $n!$.

Python tips

The `itertools` module has a function `permutations()` that returns an object containing all permutations of its argument. To make a list of all permutations of three objects, for example, we would use

```
from itertools import permutations
list(permutations(range(3)))
```

Be warned, however, that Python grabs memory for storing them, so if you try to list all permutations of more than around ten objects, it could cause problems for your computer.

We recommend that you use lists/arrays for permutations, rather than tuples, since tuples are immutable. You might want to modify a permutation, which is not possible with tuples.

Reading and references

Wikipedia:

en.wikipedia.org/wiki/Permutation
en.wikipedia.org/wiki/Permutation_group

Programming tasks

1. Write a function to find the composition of two permutations.
2. Write a function to find the inverse of a permutation.

Exercises

1. Find the composition of (5 2 3 1 0 4 6) and (6 5 4 2 1 0 3). Take the composition so that (5 2 3 1 0 4 6) acts first.

2. Find the inverse of (7 1 3 0 5 2 8 4 6).

Unit 53
Permutation cipher

The *permutation cipher* (or *block transposition cipher*, or *complete-unit transposition cipher*) divides the plaintext into blocks and applies the same permutation to each block. If the last block is too short, it is usually padded with nulls to fill it out. A *null* is a character that carries no meaning and is used as a filler. Often, ‘X’ is used, but other choices are underscore (‘_’) or random letters.

Let’s just do an example. Suppose we want to encipher this short message with the permutation (4 2 5 1 3 0):

```
THIS MESSAGE IS ENCRYPTED WITH A TRANSPOSITION CIPHER
```

First, we should break it into blocks of six letters and pad it if necessary.

```
THISME SSAGEI SENCRY PTEDWI THATRA NSPOSI TIONCI PHERXX
```

Then we apply the permutation to each block to get the ciphertext:

```
ESHMTI IGSESA YCERSN IDTWPE ATHRTA IOSSNP INICTO XRHXPE
```

The key of a permutation cipher is the permutation itself. Its inverse permutation is the key needed to decipher a text. Often, the permutation is written as a keyword. There are two ways to use a keyword to convey a permutation: with or without repeated letters. The way it works is that numbers are assigned to each letter of the keyword in alphabetical order. For example:

```
K E Y W O R D
2 1 6 5 3 4 0
```

If we are dropping repeated letters:

```
R E P E A T E D
4 2 3   0 5   1
```

If we keep repeated letters, we number them from left to right. So in this example, we number the ‘A,’ then the ‘D,’ and then the three ‘E’s:

```
R E P E A T E D
6 2 5 3 0 7 4 1
```

Breaking a permutation cipher by hand is done by “anagramming,” i.e., by rearranging letters and looking for meaningful arrangements.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 9 and 64-65.

Programming tasks

1. Write a function to generate a permutation from a keyword. Allow for both ways of assigning numbers to letters (with or without repeated letters).
2. Write a script to find possible keywords from a permutation. Allow for both ways of assigning numbers to letters.
3. Write a function to pad a text given a block size.
4. Write a function or script to encipher a plaintext with a permutation cipher.
5. Write a function or script to decipher a ciphertext with a permutation cipher. Your function that inverts a permutation could be useful.

Exercises

1. Find a possible keyword for the example encipherment above.
2. What is the permutation corresponding to the keyword `PERMUTATION` if we drop repeated letters? What if we do not drop repeated letters?
3. Encipher this text with the permutation (2 3 0 1 4).

   ```
   ROUND AND ROUND SHE GOES WHERE SHE STOPS NOBODY KNOWS
   ```

4. Encipher this text with the keyword `POKER`.

   ```
   SHUFFLING A DECK OF CARDS CAN ON ONE HAND GUARANTEE
   A FAIR GAME BUT ON THE OTHER HAND PROVIDE AN
   OPPORTUNITY TO CHEAT THE DEALER CAN MANIPULATE THE
   SHUFFLE TO POSITION FAVORABLE CARDS INTO HIS OWN
   HAND THIS MUST BE DONE CAREFULLY AND QUICKLY TO
   AVOID DETECTION BY THE OTHER PLAYERS
   ```

5. Decipher this text with the keyword `REPETITION`. We will not tell you whether repeated letters have been dropped from the keyword.

   ```
   LTRELAITNSIIAOHIETETLATCERRYNUOEIHQUNAGSFIEAETPREITAIN
   DILTREELTNRSDWIO
   ```

6. Break this ciphertext by hand. What keyword was used?

   ```
   MCEOWLETTHEOOLODWRSATNFRIIOTPSHINPOCWWREESEBLEILIOPRXL
   WFGENACHFEOTCOMNOMSEPRIHLESLAWWFSEAAMONMUCSNNEOOLYHATE
   OVHELANHEINTMCIOGNETOHMNFUSFYHLHEELTSETRETTHFEOTTBXUET
   CEERNVTGAEHNRFEOHMLETROHSETRET
   ```

Unit 54
Heap's algorithm

There are several ways to generate all permutations of a set of objects, and *Heap's algorithm* is one of them. It swaps two elements at a time in such a way as to run through all permutations without repeating any.

Suppose we have n objects. The objects live in an array $[A_i]$ where i runs from 0 to $n-1$. In the following algorithm, the array $[c_i]$ holds a collection of n counters.

1. output A (the unmodified array)
2. set all c_i equal to 0
3. set i to 0
4. while i is less than n
    a. if c_i is less than i
        i. if i is even
            - swap A_0 and A_i
        ii. if i is odd
            - swap A_{c_i} and A_i
        iii. output A
        iv. increment c_i
        v. set i to 0
    b. if c_i equals i
        i. set c_i to 0
        ii. increment i

Python tips

Functions can be passed to functions. Here is a simple example:

```
def function1(x):
      print(x)
def function2(n,f):
      f(n) # call function f with argument n
for i in range(5):
```

```
function2(i,function1)
```

The result is to count from 0 to 4.

Reading and references

Wikipedia, en.wikipedia.org/wiki/Heap's_algorithm

Programming tasks

1. Write a function that takes an array and a pointer to another function. For each permutation of the array, the other function should be called with the permuted array as an argument.

Exercises

1. Wrap a script around your function and add another function to print the array. Use it to print all permutations of four objects. Check by hand that they are all present. There should be 4! = 24 of them, and they should all be distinct. Now be happy I didn’t ask you to use an array of five objects.

Unit 55
Factoradic numbers and permutations

Factoradic numbers (*factorial numbers*) are numbers expressed in mixed-radix form where the radices (bases) increase by one for each digit as we move to the left. Why this is related to factorials will become clear. The radix for the rightmost digit is 1, so that the last digit of a factoradic number is always 0. The radix of the digit to its left is 2, so that that digit can be 0 or 1. The radix of the digit to its left is 3, so that digit can be 0, 1, or 2. This continues as far as we need to hold whatever number we have.

Let's look at an example of converting a decimal integer to a factoradic number. Suppose we have 12345_{10} (the 10 subscript lets you know beyond all doubt that this is an integer in base 10). What is this number modulo 1? Try it and see. The answer is 0. Hence the last digit is 0. So far, we have $12345_{10} = ... 0_1$. Here we have placed a 1 subscript to indicate that the last digit is in base 1. We subtract that 0 from 12345_{10} to get 12345_{10}. This may sound ridiculous, by we are establishing a pattern. Now, the next digit to the left will have base 2. Now 12345_{10} modulo 2 is 1, so that digit is 1. Our number is shaping up: $12345_{10} = ... 1_2 0_1$. Subtract that 1 from 12345_{10} to get 12344_{10} and divide by the base to get 6172_{10}. The next digit to the left has base 3, and 6172_{10} modulo 3 is 1, so that digit is a 1, and we have $12345_{10} = ... 1_3 1_2 0_1$. Subtract that 1 to get 6171_{10} and divide by the base to get 2057_{10}. The next digit has base 4, and 2057_{10} modulo 4 is again 1, so now we have $12345_{10} = ... 1_4 1_3 1_2 0_1$. Subtract that 1 and divide by 4 to get 514_{10}. The base of the next digit is 5, and 514_{10} modulo 5 is 4, so we now have $12345_{10} = ... 4_5 1_4 1_3 1_2 0_1$. Subtract that 4 and divide by that 5 to get 102_{10}. The next digit has base 6 and we get a 0, so $12345_{10} = ... 0_6 4_5 1_4 1_3 1_2 0_1$. For the next step we have 17_{10}. The base of the next digit is 7, and we get $12345_{10} = ... 3_7 0_6 4_5 1_4 1_3 1_2 0_1$. Since that leaves 14_{10}, when we divide by 7 we get 2. We can go no further, and the result is that $12345_{10} = 2_8 3_7 0_6 4_5 1_4 1_3 1_2 0_1$. Some might write this as $2:3:0:4:1:1:1:0_!$. There is another way to find the factoradic representation of an integer by working left to right; we save its discovery for the reader. To convert our example back to a decimal integer, we use factorials:

$$\begin{aligned} 2_8 3_7 0_6 4_5 1_4 1_3 1_2 0_1 &= 2 \times 7! + 3 \times 6! + 0 \times 5! + 4 \times 4! + 1 \times 3! + 1 \times 2! + 1 \times 1! + 0 \times 0! \\ &= 2 \times 5040 + 3 \times 720 + 0 \times 120 + 4 \times 24 + 1 \times 6 + 1 \times 2 + 1 \times 1 + 0 \times 1 \\ &= 10080 + 2160 + 0 + 96 + 6 + 2 + 1 \\ &= 12345_{10} \end{aligned}$$

Notice that

$$\begin{aligned} 1_2 0_1 &= 1! \\ 1_3 0_2 0_1 &= 2! \end{aligned}$$

$$1_4 0_3 0_2 0_1 = 3!$$
$$1_5 0_4 0_3 0_2 0_1 = 4!$$
$$1_6 0_5 0_4 0_3 0_2 0_1 = 5!$$

etc.

Factoradic numbers are related to permutations in a natural way. Suppose we have *n* objects. There are *n* ways to place the first object. The name of the box into which we put the first object is a digit (as we call it, even if it is bigger than 9) from 0 to $n-1$. There are $n-1$ ways to place the second object, and we can record that choice as a digit from 0 to $n-2$. By the time we get to the last object, there is only one box in which to place it, so there are 0 choices, and the last digit is 0.

The mapping from factoradic numbers to permutations is called a *Lehmer code*. When the integers 0 to $n!-1$ are converted to permutations of *n* objects, they come in *lexicographical order*, i.e., in order as if in a digital dictionary and the numbers in the permutations are their letters.

Python tips

The pop() function deletes the n^{th} item from a list. The remove() function deletes an item based on its value. For example, this sample code removes the letter ‘X’ from the list, because it is the third element (counting from zero). Then it removes ‘B.’

```
letters = ['A','B','C','X','Y','Z']
letters.pop(3)
letters.remove('B')
```

Reading and references

Wikipedia, en.wikipedia.org/wiki/Factorial_number_system
and en.wikipedia.org/wiki/Lehmer_code

medium.com/@aiswaryamathur/find-the-n-th-permutation-of-an-ordered-string-using-factorial-number-system-9c81e34ab0c8

stackoverflow.com/questions/1506078/fast-permutation-number-permutation-mapping-algorithms

2ality.com/2013/03/permutations.html

Programming tasks

1. Write a function to calculate the factorial of an nonnegative integer (maybe zero). Avoid using recursion.

2. Write a function that converts a nonnegative integer to a factoradic number. We suggest that you store factoradic numbers as arrays of integers.

3. Write a function that converts a factoradic number to an integer. Be careful that only one object can be placed in any one box.

4. Write a function that takes a factoradic number and a length *m* and returns a permutation of *m* objects. Be careful to add 0 digits to the left if the factoradic is too short. Be careful that each factoradic digit *x* represents the x^{th} empty box, which is not necessarily the x^{th} box overall.

5. Write a function that takes a permutation and returns a factoradic number.

6. Write a function that uses your other functions to take numbers *n* and *m* and returns the n^{th} permutation of *m* objects.

7. Write a function that uses your other functions to take a permutation and returns an integer *n* to indicate that the permutation is the n^{th} permutation.

Exercises

1. Find 1000! How many zeroes are at the end of it?

2. Find the 101^{st} permutation of seven objects.

3. Which permutation (zeroth, first, ...) is (4 9 2 1 6 8 6 7 0 3)?

madness's book on classical cryptography
unit 56: brute-force attack on the permutation cipher
last modified 2020-07-09

Unit 56
Brute-force attack on the permutation cipher

To perform a brute-force attack on the permutation cipher, we need to be able to try every possible permutation. At this point, you already have the tools that you need.

Programming tasks

1. Implement the attack. Use tetragram fitness.

Exercises

1. Use your attack to break this ciphertext from the 2006 British National Cipher Challenge:

```
rlboy itdvs tennc rmaid toafl ubhle cneda nmoam nrdie
ficeh shoif sjmea hstsy nsiap sedsv mseel eopyl tndda
meoeb eopyl hndti ieedm artre eanne cettc oceyt hruea
yeebr iqrue ndrae eidrd tecdt eaokt hetsh asinp ndmie
aetmh uirng yenrd orocu damnm rdaan moeet frbka etohr
tdmie nrear nneaa eodst toeuk edadn yrsot hetsh fsiop
retfh fcehn aelte lotut etohn rproe gftao atemn tirnt
ihoet dorrl sisph etohn rpiom entca tionf pceer nntgi
eoalp hsocn eeisn teang tshsa drier tegar encco tirnn
rwhoe todfs rfhie rlsot ahdet repap ocaen ibfra ssthi
raqdu tionn dmhee reirt neaan oaics iintd woonn tchhi
tfhae ufeeo mpreo taaty oshmi dnmte deenp ruyao toest
eirna nrvye aveer ymnad rciun incso brdae zhlae ndair
cfeef intgi eetrh rnoof ouyno onrya mofuy ifaay oslya
liulw esawn hortt toenc arrya rotuy lrpie odafn oords
hgitn asihs yblel rroou irdge nnvoe dabro ietvh yocrt
rstio trhao lnieo onsco oomdm etrhe unvga ydabr amcmo
tonfd mahde lailr tdosr envci xxnxt
```

Unit 57
Hill-climbing attack on the permutation cipher

When we developed the hill-climbing attack on the monoalphabetic substitution cipher, we took a parent key, which was a permutation of the alphabet, and swapped two letters to form a child key. If the child key resulted in a better fitness for the plaintext, then the parent was replaced with the child. We will do something similar here.

We will make two important modifications to the algorithm. First, there are two ways in which we will want to generate a child key from a parent. One is by swapping two randomly selected elements in the key. The other is to roll the key a random number of steps. By this we mean to cut the key somewhere between two elements, and then to swap the front and back pieces. So, for example, if we cut (0 1 2 3 4 5) between the 1 and the 2, we would get (2 3 4 5 0 1); we have rolled the permutation two steps to the left.

The second modification to the algorithm is to add a margin of error. With permutation ciphers, it is easy to get trapped in a local maximum fitness. But we seek the global maximum. So we add a little leeway to the hill-climbing and allow it to take small steps toward lower fitness once in a while. A margin that works well for us is 0.15, and the probability of stepping downwards is taken to be 5%.

The algorithm does not find the length of the permutation. Usually, the key length divides the length of the ciphertext evenly, but not always. The key length must be input to the algorithm.

The algorithm:

1. set the best fitness to the fitness of the unaltered ciphertext
2. set an initial random parent key
3. set counter to 0
4. while counter is less than about 100 times the key length
    a. copy the parent key to a child key
    b. flip a coin
    c. if heads
        i. swap two randomly selected elements of the child key
    d. if tails
        i. roll the child key a randomly chosen number of steps
    e. find the plaintext by deciphering the ciphertext with the child key

f. calculate the new fitness of the plaintext
g. if either (new fitness exceeds best fitness) or ((new fitness exceeds best fitness minus the margin) and (we roll a 1 on a 20-sided die))
    i. replace the parent key with the child
    ii. replace the best fitness with the new fitness
    iii. set counter to 0
h. increment the counter

5. output the parent key

Python tips

The `shuffle()` function from the `random` module does what it says. For example, this block of code will fill an array with the integers 0, ..., 9 in random order:

```
from random import shuffle
x = list(range(10))
shuffle(x)
```

Programming tasks

1. Implement the attack. Use tetragram fitness. Experiment with changing the margin and the probability of stepping downward with some ciphertexts of your own choosing or making.

Exercises

1. Use your new attack to break the exercise from the previous unit.

2. Break this ciphertext.

```
STIOTASDGUTNINEFILMISTGEMEINNSATDESETASKSILUTTOLBSNLCI
TESYLNOELFVOETORMHRLYUCGIOVNEROOEKGTTPNETECOLERMOIRBDE
OMERGEITNTHERIPMWAIIDNRNKHEGMTOSESOWMNTNEHBETHCELSAKNO
DSHWULMNIDTEAEITDHVOUBWEOLDLNCGALITOLTESDTEHWEIMPAAIRA
GTUNSISJJPTTAUMHEOFTELNHTEADTSPNTATEHIOGTERWHHYTITRNOD
UHANUSROYOPOHUISYIYBORNGKEUSRNETHITNIGTSBTUITPVHIEELHS
CTTRUAETAHTRYILVLDRYIENSOUNESTAELOESTDTHERIPMWAAXAXGIN
```

madness's book on classical cryptography
unit 58: matrix transposition
last modified 2020-10-09

Unit 58
Matrix transposition

A *matrix transposition* is the simplest form of columnar transposition cipher; hence, it is also called the *simple columnar transposition*. Encipherment is done by writing the plaintext into the rows of a matrix, then reading the ciphertext from the columns. Typically, the plaintext fills the matrix; if not, then nulls may be added. The key is the size of the matrix. Since we know the length of the text, we often only need one number to specify the key.

This cipher is also called a *scytale cipher*. The *scytale* was a rod around which a ribbon was wrapped. The message was written on the ribbon. When the ribbon was unwrapped, the letters on it would be in the same order as after a matrix transposition.

For example, let’s encipher this short message with a 6×8 matrix:

```
THIS MESSAGE WAS ENCRYPTED WITH A TRANSPOSITION CIPHER
```

The message contains 47 letters, so we will add one null to the end. Written in the matrix, it looks like this:

```
T  H  I  S  M  E  S  S
A  G  E  W  A  S  E  N
C  R  Y  P  T  E  D  W
I  T  H  A  T  R  A  N
S  P  O  S  I  T  I  O
N  C  I  P  H  E  R  X
```

We read off the columns to get the ciphertext:

```
TACISN HGRTPC IEYHOI SWPASP MATTIH ESERTE SEDAIR SNWNOX
```

Reading and references

Wikipedia, en.wikipedia.org/wiki/Scytale

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, page 82.

Programming tasks

1. Write a function or script to encipher a plaintext with a matrix transposition for a given size of the matrix. Allow for the possibilities that nulls are or are not added to fill the matrix.

2. Write a function or script to decipher a ciphertext with a matrix transposition for a given size of the matrix. Allow for the possibilities that nulls are or are not added to fill the matrix.

3. Write a function to find all factors of an integer. We are using small numbers, so it is not a problem to do an exhaustive search over all numbers less or equal to the square root of the integer.

4. Write a function or script to break a ciphertext that was encrypted with a matrix transposition. It would be reasonable to check all factors of the length of the ciphertext first, and then to check all other possible matrix sizes, in case nulls were not added. Be careful that you place the empty parts of the matrix correctly.

Exercises

1. Encipher this text with a 12×15 matrix:

   > Call me Ishmael. Some years ago, never mind how long precisely, having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail around a little and see the watery parts of the world.
   >
   > (from *Moby Dick* by Herman Melville)

2. Decipher this ciphertext with a 13×18 matrix. Is the matrix filled completely?

   ```
   TDFHEGATRAGOUOEWAYLDLIRHLTMWHNEAEETDLLAAAIDDDEDTFIOBPL
   TLTNPDHEFWUPKEEHEMOINEAREWWDESEWRCTNDAIABFSLNTEODERTSR
   ELAUYNHENEHHUNENPYIIXDAASCFCOANMIOBNHETHNRESSNUDHAHOHD
   FETTCAENILISEEEHKLSDMYSOEMNEEOUAASSFTECSTNRLNEPBHDEIOG
   VLDTIOIHB
   ```

3. Break this ciphertext:

   ```
   OONEOLDOSHEHMOSNKDDNISMTADIPFHEEHIHFEEIRRSAHEMFINITEDF
   DEMRILORMTSEHAFLAAEMOROSOADINSYDNDWONMEARHSDEAYYWAKITL
   HMIBDCBTLIVENRFOOSRITLOETEDGOTRUHOVIESGHDWWURRREWIOTLS
   TAHHBAILANDNOIPHBAELNBIDBESCDIEOTNESLKAEDTOETSUSGDFEEL
   LBHVOIITHRDOVBILYEEFFZHAERREATYABRFUEEPGEMRCTSREIALOLP
   OAEMKLLCDTNLFPERMDIAEIHDAYYTLNSSINNHGEINMTHEEAHNHDEHSN
   ```

```
DOHESDMEHEICTIGSMIRSTSFILFOLNWEENELOAOSAHUYTAENCSYMWUB
YELDOSMTOTAE
```

Challenge

Think about triangles.

```
IAAUSNPOETOHAUNRGSXTGNTQDSRDERENLSILXROGTUOOEAAYTDAOAEIOLHA
FNWNSARTTFNEDEARTEEDHBIRITDSSTEHOLCOOAIOHTBIFIFLARULBAOUAON
YRNTTSRRTMRGOOWSTDYIJTPUURTTATUHEINIOBTSARSDTOUTTHMHURTKAIT
NOINTSOKROOIF
```

Unit 59
Twisted scytale

The *twisted-scytale cipher* was introduced in the 2011 British National Cipher Challenge. It is a modification of the matrix transposition (simple columnar transposition). Here's how it works. First, write the message into a matrix, with padding if necessary:

```
0    T  H  I  S  M  E  S  S
1    A  G  E  W  A  S  E  N
2    C  R  Y  P  T  E  D  W
3    I  T  H  A  T  R  A  N
4    S  P  O  S  I  T  I  O
5    N  C  I  P  H  E  R  X
```

Then, we roll each row to the left a number of steps equal to its row number (starting from zero):

```
0    T  H  I  S  M  E  S  S
1    G  E  W  A  S  E  N  A
2    Y  P  T  E  D  W  C  R
3    A  T  R  A  N  I  T  H
4    I  T  I  O  S  P  O  S
5    E  R  X  N  C  I  P  H
```

Read off the ciphertext by columns:

```
TGYAIE HEPTTR IWTRIX SAEAON MSDNSC EEWIPI SNCTOP SARHSH
```

The key for such a cipher is the number of columns.

We can modify the cipher to "twist" the matrix by an arbitrary number.

Programming tasks

1. Write a function or script to encipher a plaintext with the twisted scytale. Allow for an option to change the twist.

2. Write a function or script to encipher a plaintext with the twisted scytale. Allow for an option to change the twist.

3. Implement a brute-force attack on the twisted scytale. Remember to allow for different twists.

Exercises

1. Encipher this text with key 6 and twist 1.

   > One morning the old Water-rat put his head out of his hole. He had bright beady eyes and stiff grey whiskers and his tail was like a long bit of black india-rubber. The little ducks were swimming about in the pond, looking just like a lot of yellow canaries, and their mother, who was pure white with real red legs, was trying to teach them how to stand on their heads in the water.
   >
   > (from *The Happy Prince* by Oscar Wilde)

2. Decipher this ciphertext with key 7 and twist 2.

   ```
   OHVYLLINEYKBHEBHVMSUAMLRAFNATLNSABDRTPKONEETILEDAPSEED
   EEEITLDEYIGLTFELDATYEWHPDHERRHTOHSWTIFHDHWWTBRETINAYHR
   ROHDTHRAEETENELETWIHATXOASIAIHEEDHNHSIEIAWOBRESIHDORIF
   ECLSSAYOWRBTNSNTATHETEHNDVBMTEATLSASTXGTWIEWFDOEEEUADO
   LHUIERASEGOEIYHECEETHTAXHTOTAARGNGESTYFROEOFHERPWSWRGE
   ANODNTAOLT
   ```

3. This is the ciphertext from the 2011 British National Cipher Challenge. Decrypt it.

   ```
   TsormynhemuitlThesynpddmrdtcryefclseanpinptbsslenednul
   fteomatesdepaeestvsnysdsnmesirutnitgapsarmdxcdusrVerlr
   nitestixpurhepobakmsiarieiyrettFGcpgalitdwideddihdeivp
   otreesrtCFDdthaahthVaptetftddthrthapngteycitfehplceCra
   ebinatektadrytioosorngoxihIecfgestnoeeIesphaaetoreeiec
   aaabetvdrotliusiiWcrgmranhrnmuitiinongrvheenitiecrsicm
   IhcohenhotiiTtrystdXcuyatehyfhtboseatoneietictysievenh
   eraoomunnasctXsbyfafspelaeSxtiagsGwenepnefaanredsfwasl
   eitenhotocoipscWiseemoncblsstetiniccsevrnstldtfwmomuoi
   sTecinmwcTtirvrrheerneserrntchemofhooningiteoevecenaln
   owncgsdraieddanesanysnalntrnasoymennhispiaADisssIltwor
   hylsqnrsstnnhcaiTgsaeptelssrndsotndahrenairlaelinyoahc
   amhfouodhdXatnbesioeryaeseysoybhtuftheoitsdtpesyeetees
   sbgloianemachduimaertsrnonmocsyhaecoaetatimodeinsaseae
   forieitontenhavntieiatcinorltttemcyevcgmgudpamufthypnt
   garwtgetetnndFpuofclhorbesybudaitegfemlohuicdrineAtrei
   ithealVeptmepatcyednfmemlealaltisVDctdXrenfdesltocotgr
   uihrihelurhiehesphenomsecadsmmlsiatreerierisstdditpmat
   sinmpncimoeruusnntasIluyhuetexafeelciserslontiocsnkuha
   llsehaiustoietgesoreetderaeheleenegaaGhalaawekdosixiat
   ```

nsiptroudnedshbesgtDhesncessreanreorsnrtaptooplufnrptd
egrFTthecenloinaittufegatnesterfostmviTtiavaditarneane
linfiowvensx

Unit 60
Columnar transposition cipher

The *columnar transposition cipher* generalizes the matrix-transposition cipher by permuting the columns before reading off the ciphertext. The key is the permutation, which may be represented by a keyword. Repeated letters in the keyword may or may not be dropped.

Let's do a quick example. Suppose we want to encipher the following text with the keyword `KEYWORD`.

```
THIS MESSAGE WAS ENCRYPTED WITH A TRANSPOSITION CIPHER
```

First, we arrange the message into rows with the same length as the keyword:

```
KEYWORD
-------
THISMES
SAGEWAS
ENCRYPT
EDWITHA
TRANSPO
SITIONC
IPHER
```

The columns are then permuted. The permutation corresponding to the keyword is (2 1 6 5 3 4 0).

```
DEKORWY
-------
SHTMESI
SASWAEG
TNEYPRC
ADETHIW
ORTSPNA
CISONIT
 PIR EH
```

Notice that some columns are shorter than others. The ciphertext is then read off in columns.

```
SSTAOC HANDRIP TSEETSI MWYTSOR EAPHPN SERINIE IGCWATH
```

Of course, we remove the spaces to hide the column lengths from an eavesdropper.

```
SSTAOCHANDRIPTSEETSIMWYTSOREAPHPNSERINIEIGCWATH
```

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter VI.

Practical Cryptography, practicalcryptography.com/ciphers/columnar-transposition-cipher

Abraham Sinkov, *Elementary Cryptanalysis: A Mathematical Approach*, 2nd edition, revised by Todd Feil, published by Mathematical Association of America, 2009; www.jstor.org/stable/10.4169/j.ctt19b9krf; chapter 5.

William F. Friedman, Military Cryptanalysis, Part IV: Transposition and Fractionating Systems, Washington D.C.: U.S. Government Printing Office.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/CompleteColTransposition.pdf and www.cryptogram.org/downloads/aca.info/ciphers/IncompleteColTransposition.pdf

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 152-153.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter V, sections III-IV.

Programming tasks

1. Write a function or script to encipher a plaintext with a columnar transposition cipher.
2. Write a function or script to decipher a ciphertext with a columnar transposition cipher. Be careful of where to place empty spots in the matrix, if any.
3. Make a copy of your brute-force attack on the permutation cipher and adapt it for the columnar transposition cipher.
4. Make a copy of your hill-climbing attack on the permutation cipher and adapt it for the columnar transposition cipher.

Exercises

1. Encipher this text with the keyword `PERMUTE`.

> "You are old, Father William," the young man said,
> "And your hair has become very white;
> And yet you incessantly stand on your head—
> Do you think, at your age, it is right?"
> "In my youth," Father William replied to his son,
> "I feared it might injure the brain;
> But, now that I'm perfectly sure I have none,
> Why, I do it again and again."

(from *Alice's Adventures in Wonderland* by Lewis Carroll)

2. Decipher this ciphertext with the keyword `COLUMNS`.

```
WNIONOIENNANNACTTEITTAITWLWCAHOEEHANSOYKSGCLSKTHLCSYEO
EAEGEAAEEOBNMOAEBEWGATLPEISEFEUMANDBOSAWUPODTCREAONLDR
TTDDAETTNTIRQSTAYDDENOCELIUORAOTCWOIWOTSALIMSEANCADCNT
ASGLMIHHRNHYVNLHLIAYTSNMOGETNODHOIUCIISIECSFTWLNRFUSMA
KSRGHKLSJINOKECWAPODBEWEUICUFTPUCRSDFSLDEFNBANEEGHI
```

3. Brute-force this ciphertext.

```
OTENAYOSTAEPFLEUSEHKTFQEEIROGOSAEUCODWNREETFUTRYFOTGHT
AYETNANTELETCBMTMSEAAMINFHUSRCEAAEASLVALSNNASNUEUSTEBS
KPTLBEVAHHUTUOEUFHOEELNINSEETELUEEOIRLWEOHTRHHMSNYURAE
KYYTSPSOELOUDTRSISZRTEPETEEHOESITAEEHSNBSOENYEATSETVTO
MASHNMFEDONUTHFEEAIGISOEAUNMEMHEOOPOSEIDEEOEOEHHLNAADO
SMIHTTQEMORJETORSEOSEEUGIHNJNVMTEUTMYGEUDTRSIUYYDLOUEI
SLOLYCRPDENHYUYYGTOITHNCROVTIOSZYGITBEGIUHOIHROROTIDOA
YNHOEOSIESNREDG
```

4. Break this ciphertext with your hill-climbing attack.

```
UHOEIHSHIBMEWTRNWBETDHTOHSYSHTHNNBETWEROHTWAOTDDROTLCO
WHOWYETEIFWACTENEDEEIEQTSTRCDFMIESTLRSELITUIRTDRRETRLD
CAAETOBECDMATAOTWNTSGLIDVSKISAGTSCTEENHWNLESUEDBSTFEST
RHSESWLEWYYLEUULSSIRNHEEDYLDRRACTHTNGSAENIOEWEIHMEOAOP
HUTEDCDTHENARHUEAAKSLGICNWRFGAANINANAMESEABSHIDAEIHLET
NIUBEOHAOEEAHAIYFEHASUHSRYUSOACAEBEIRHRRLANAEWGATPANEU
IAHELASATNWFFDUTRDHLTIITPADETRBHSLAANRAAOONOHEESNEIIHH
DRTTLODTIDUAIFGMSNIAEHUOSTEEAOVDMIEETBARSUNABEUNYDFOAE
OTHUHEEKOAYITOCTOIWLRYESEPTHEOOWSACEEUACTSIEWDELTPNAHY
WOTTUVDHSSGSNTITHYOHOOBASDSSDTHOSGLUTTNTIORRYNEESDATCE
NEDCOCENDTNSLVDWBOESENRISNWKHSHIFFEYEINWVSIAAUSAATOTAB
IAEHTAAIEEWLTLEBETHHRETHAIHHWLGIETELMNHHOAWMOIEAIKCHGS
```

GUSHTDDSUIASWEDSEAODEDSOIKREGERNSTRTPWLLHELYTUNSDSNAAV
DSOPRYYPESHYGLOSENDBTESSPNAAKTNHNIEOINC

Unit 61
Double columnar transposition cipher

The *double columnar transposition cipher* is the application of two columnar transpositions in sequence. Its key is two permutation, which can be represented by keywords. It was a popular cipher during the two world wars.

A hill-climbing attack on the double columnar transposition can be constructed by modifying our attack on the single transposition. There will be two parent keys (one for each transposition) and two child keys. In each step in the key space, we randomly choose one of four options: swap two elements of the first child key, roll the first child key a random number of positions, swap two elements of the second child key, or roll the second key a random number of positions. Unfortunately, there is no way to determine in advance what the two key lengths are, so we often have to try several before we succeed.

Reading and references

NOVA Online, www.pbs.org/wgbh/nova/decoding/doubtrans.html

Wikipedia, en.wikipedia.org/wiki/Transposition_cipher#Double_transposition

Solomon Kullback, General Solution for the Double Transposition Cipher, Washington D.C.: U.S. Government Printing Office, 1934, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/FOLDER_439/41751169079035.pdf

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 157-159.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 301-303.

Programming tasks

1. Write a function to encipher a plaintext when given the key.

2. Write a function to decipher a ciphertext when given the key. Keep in mind that the two transpositions are done in opposite order when deciphering.

3. Implement the attack described above. Copy the attack that you made for the single transposition and make the appropriate modifications. Experiment with the margin; can you manage without it?

Exercises

1. Encipher this text with keywords `SAMPLE` and `KEYWORD` (in that order).

   > At the Carlton news stand West bought two morning papers, the Times for study and the Mail for entertainment and then passed on into the restaurant. His waiter, a tall soldierly Prussian, more blond than West himself, saw him coming and, with a nod and a mechanical German smile, set out for the plate of strawberries which he knew would be the first thing desired by the American. West seated himself at his usual table and, spreading out the Daily Mail, sought his favorite column.
   >
   > (from *The Agony Column* by Earl Derr Biggers)

2. Decipher this text with keywords `TOMATO` and `SOUP`.

   ```
   EOONPGAEPOGOCPEPCRWEUNNELMRDNIOOLFEIAOTTASCRCCATERWDGO
   SNNSKNGIDOEIDFEELLPALIHNNTISYDNNRSHPFEANOINDOEIELVDERS
   MLOIRRDROCSLYETEEWFRGEIEHWILIERNALNOSHCIHACEEBIUEBROPV
   NNREDEOPRMPIROOEAMHUIOCAESIOJSSSOODNROTKONAESONAADWRIG
   DOLPEERETOEERTAPHETADEOYRTFPKOLGDDHCTTDETVAAELLSCUAEDT
   RRRUPAALSUOECDLTCDNPNCQNDARSAIUANEPSEOOSAKYYBIRRORRLHK
   IFTDCEDNPGOCIRETROYTCOOOTYOGNODHTLOSUUGPPNBACONWSTBRHG
   MPEETEDENIMSCGTAETAOOUAUNPDTEDEOR
   ```

3. Break this ciphertext. The key lengths are five and six.

   ```
   VNRWGNCARTTNONRRWOODRLIOOSWPAEHTHUNDCTTIAHIRLASDIHNTHI
   AHOHANMSNFTEUECSNDEKRROSFGNUEFSRFODGIHONNIEETHEROAOALB
   REEUIGONFESINAMTOFLBETTWWNSNAFMTOECLEEAEAEHTYTNIGVETAR
   RNILTOEAHENATHGLIDDJDADGHWTOTRSEOATTSLDHNHFWCOLEIWIEDI
   DAAARSOBRATIHGFSEDAROCORAISYRLVTIEFTSAERERFNLRATVTSNTR
   AIWNSESENVQTDODTSOOLLNGFAFIMBAARENEOMEIEWEUKUONLOSISAL
   HOVINSTBEESLRETHBDISESGDAEVTORGHRAEDOOIGRHDSAOITITOEIA
   THTATBHUALTAISHBTLTHDERAETTJKEDOELCEWETOSREYEEDGTGTSRW
   THSLIWNWNADNOEOPOETOIIEONOAWCYNITITEOTBWEOAFRRIHEAFAHP
   ETOIRTTIOSADAAOHOAHLIWETYNDEESIHICNCOOEFUAEATWFAETTKEH
   ESOOOITUINTSIAJLDEHWUWEYS
   ```

4. Break this ciphertext.

TTFNOEINTDDRNTBEOSROHOYEIAEFAIEYFEOREECTTHSITEROTOPPTN
RPROCGHSNIFTJPOEEGHSCPUKENDDIHSIIGETILSEOETDIEOPHIHBIA
NDDENMUITMHNTEOLTCEAACSPTOUIETBHTOUAOOASOOHEUVRRHRNOTN
YRNMCIOOAEOGHEEMTTOSNDTENDIAORNSVOAESDNROTPLCHTTMRAHHO
SCNIMCFEMRLMNGIERETTEDTEMZBTMRHPNTVHESGSHEEEIVUEHEAYUI
PLHIEAITHRLHNNOWNEEEWTEDEPRTTRUBIIETOONOAAEITSEFSODDLM
CALWOERTHUTIYELDTOBWTTVDIATURTRRYIOYIOTEIFTPSTORTAWFOG
IEOSTESDOSBAIIIO

Unit 62
Nihilist transposition cipher

In the *Nihilist transposition cipher*, the plaintext is written into a square matrix, with padding if necessary. Rows and columns are both reordered with the same permutation, which can be represented by a keyword. It does not matter in which order the two operations are performed. The ciphertext is read off either by row or by column. If the plaintext is too long to fit into a grid whose dimensions are the length of the key, then it is broken up into blocks, each of which is enciphered separately.

Do not be confused and assume that this is a permutation cipher followed by a columnar transposition. Both the permutation cipher and columnar transposition shuffle the columns when the text is laid into a matrix. However, in the Nihilist transposition, rows and columns are both reordered.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter IV.

American Cryptogram Association,
www.cryptogram.org/downloads/aca.info/ciphers/NihilistTransposition.pdf

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter V, section VII.

Programming tasks

1. Implement an encryptor for the Nihilist transposition. Allow for both choices of how the ciphertext is read from the matrix.

2. Implement a decryptor for the Nihilist transposition. Allow for both choices of how the ciphertext is read from the matrix.

3. Implement a dictionary attack on the cipher.

4. Implement a hill-climbing attack on the cipher. Use parts of your attacks for the permutation cipher and columnar transposition, and make the appropriate changes.

5. Now suppose the matrix is no longer square, and that rows and columns are reordered with different permutations. Copy your hill-climbing attack on the double columnar transposition cipher and modify it for this new cipher. Again, allow for both choices of how the ciphertext is read from the matrix. Also remember that a long plaintext may have been enciphered in several blocks.

Exercises

1. Encipher this text with the Nihilist transposition with the keyword `BOYNIHILIST` (do not drop repeated letters). Use 'X' for padding. Read the ciphertext off by rows. You will need to encipher it in two blocks.

   > But outside of his being an enthusiast and a lover of liberty, he was not known, and had never taken any prominent part in any of the social or political movements of the day, beyond sympathizing with the struggles of the working men and women of the world in their struggles to better themselves.
   >
   > (from *The Boy Nihilist* by Allan Arnold)

2. Decipher this ciphertext with the Nihilist transposition with the keyword `SIBERIANGULAG` (do not drop repeated letters). The ciphertext was read off by columns.

   ```
   RHFHLADAEXGTIHDCOFNCRDDMETSHHPHOIANXEDMOOERECSYDXRFEUE
   DTUHAHTGNAHEHANFCOOEDANFIIEOLDHYEEEATOVEICWRGONIFRFUEI
   DHMTIPORBNARMTTELDXMHMJAPWIEMUVIKDOESRMTNEWAYVNOTLSSIA
   YHSAFIU
   ```

3. Break this ciphertext with your dictionary attack.

   ```
   OIUDYOHENTTIAVAFMELLWIORILDWANDOOULYYOSTRUONEYUBRTBELI
   ITANYCEOUSCAANUMFHLIHETTNDSAVEUKYOENHAWTNOUBDODAWHNDTA
   NDNAMENDMEWOSOEDDRVERAFBRTBELIUNFHYOLLWITHWEUSYOSTRURT
   AIDFANDUENEPOUNYPOSBERETNOGKURODYBERTPNSYIBETOAREVUETR
   NEEALBATYMASANIXOWILTWDIINPRWSSTMACEMSHICHGTINEENDDAIN
   OAPRAPORRFTEOFUTYOHTIGLRYOLLWISSPAURALRTPOOUTYHAYOVEHA
   EIOPELHAISNHETACPLNVSEHIARWEUSTOMETONDHAORXXITLELSEEON
   EEOSETLOALNDNONGWI
   ```

4. Break this ciphertext with your hill-climbing attack. What are possibilities for the keyword?

   ```
   THTATEMMORAUBTILILKEOTYODVETRENUTRNOAHNTIHIILATSIHNNIS
   ATNOISILOUTAASNDSSSASXSWXXSOHSETADLLFROTOOEFLL
   ```

5. Break this ciphertext which was encrypted with the cipher described in Programming Task 5. What are possibilities for the keywords?

```
AMSAIBSOIHERQKOCNEISTATHTERDUSERFPAPMTETRWIOTERSIUATTS
UIEMCAITCUBUEPVIHWANONOEETCNEOCONTNASTHISTSHSEEBAYENIE
LTNULNUDLRUOFYOGOAMNOSIHNTYOPLANOGFNIIRTWREVRNITNMERSD
AOONADSGFFEIDNAYMYNAYMOASPYAAFCERNISDNTNWERFFEIDDNEETO
TTECERAEHISNIEDOURTIONTDRIIAETKHOOOFLTATRHEDEHBTHIHCWN
BTESUIARTAATCTASWMOLWTHUBEKNATMMOOCRLEACSNTOCISNEHROIT
SHOITNTSONOIREAGSOTTSTCIURVTACATSRBAEJSOTSSIERFOANOEIS
ROMIPEHOTTSTTDONAAHDTISDCAEROMTSECWNAONYNAHTETSCUTDOWN
HOITNIEITNRIFTROOSLKIOOLSIISHTAEKAHBCIEEDNSHTIEKAMNAHY
AWSOHSOOHIHCWTDNTIFHGTIMOTDEVESAUOONFOTHDALUATHARVIELE
EMSOSSKAOORRAAWLASOCDNBAHNODCELAWRETSSACADGNAESVTNROEF
RHTEAETNIAUTOKNOHWOONTAQERITDSUFODYTNAWMALEFSMAESLPCUH
SWHEIKADLMUOHEOHTAPRERROTFETTONAROPDATSEEMITOPEAODLAXX
XXXERCEHEXDIMSFOON
```

Unit 63
Railfence cipher

In the *railfence cipher*, we write the plaintext down in a zig-zag pattern that runs over a number of rails. With no offset, we begin on the top rail. With an offset, we skip some positions before beginning to write the text. The ciphertext is read off one rail at a time.

Let’s do two examples. We begin with the plaintext:

```
THIS MESSAGE WAS ENCRYPTED WITH A TRANSPOSITION CIPHER
```

The first example will not use an offset. Let us take four rails. We write the text onto the rails:

```
T-----S-----A-----Y-----I-----A-----I-----I----
-H---E-S---W-S---R-P---W-T---R-N---S-T---C-P---
--I-M---A-E---E-C---T-D---H-T---S-O---I-N---H-R
---S-----G-----N-----E-----A-----P-----O-----E-
```

The ciphertext is read off in rows, starting from the top rail.

```
TSAYIAIIHESWSRPWTRNSTCPIMAEECTDHTSOINHRSGNEAPOE
```

Now let’s repeat the process with an offset of five. We skip five positions when writing down the plaintext.

```
•-----H-----S-----S-----P-----T-----N-----T-----P---
-•---T-I---S-A---A-E---Y-T---I-H---A-S---I-I---I-H--
--•-•---S-E---G-W---N-R---E-W---A-R---P-S---O-C---E-
---•-----M-----E-----C-----D-----T-----O-----N-----R
```

Read off the ciphertext to get

```
HSSPTNTPTISAAEYTIHASIIIHSEGWNREWARPSOCEMECDTONR
```

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; page 12.

Practical Cryptography, practicalcryptography.com/ciphers/rail-fence-cipher

Wikipedia, en.wikipedia.org/wiki/Rail_fence_cipher

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Railfence.pdf

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 139-141.

Programming tasks

1. Write a function or script to encipher a plaintext with the railfence cipher and a given key consisting of the number of rails and an optional offset.

2. Write a function or script to decipher a ciphertext with the railfence cipher and a given key consisting of the number of rails and an optional offset.

3. Implement a brute-force attack on the railfence cipher. Note that the number of rails needs to run from one to the length of the text, and the offset needs to run from zero to twice the number of rails minus two (with a maximum of the length of the text). Use tetragram fitness to determine when you have found the correct plaintext.

Exercises

1. Encipher this text with four rails and no offset.

```
DO YOU SUPPOSE THERE IS ALSO A PICKET-FENCE CIPHER?
```

2. Encipher this text with five rails and offset seven.

```
HOW DO YOU SUPPOSE A PICKET-FENCE CIPHER WOULD WORK?
THE PICKETS ARE VERTICAL, AND THERE ARE A LOT OF THEM.
```

3. Decipher this ciphertext with three rails and offset two.

```
YEYITTIAAELSEEOIHAESEBERADUTOORALBLEEHTHSSHTRIFNEOKLKU
LSYUHNTELITXILKTEABDIERPEAONIDULEVAIWALCOINSTKPNTIHRWW
PRD
```

4. Break this ciphertext from the 2014 British National Cipher Challenge.

```
PRSAO EGERA UIADM WEHDN ISNRA SAWUA AESSR EFGDO SOGVO
RBEEE AARTE SCTDF MENUI BRTTL MEYTU MTMEU AIKWH UTKWE
RWAHM NPWRA EESON ONESE BATOI HACIN EETBR OTADA KTGFE
ESYIO FLTTL STIIA EOSVI EONSR RTAUP MNNOA ENCOC NUVRS
CLVDR GCTAI IHRIC IAIHR SDUOM RLEMC RNGLE OMARF HIUEW
HALCS ASRAC UFRAW WSMEH ULSTO AOHCE LETMT OILSE PDMUM
TPTRS LYRHH NTPAN WPMOA DPPDW BESEO ASSLT MLPES LETUN
CORER LCLIT AOSVS INIIF WSEAF ORTAA DUYEN ENONN SOPFH
ONTWK OERTC SLYVO EIOHL UFOEI OETST HTSBR ENEVE AOUEP
GIEES OBDUO RSFEE RCDYA DUTAE PEADR DIGSE EBFUO GGOPO
GALYF EWSOE EMDNT OHREB HAAES NEWOR GNFIA ULNLW ADUEO
DCOTR ARGVU ENEWH IERTL AUILM SONIO TMUIN EWAIU EWLOE
RSTTT ISDRS ASNUS SIESM ERDHE TRYRH PNLRT EREAD MREDE
BNNTR NENWM OUTRD OSANE OWOMC GIDCI ASAON TIIOI ASCES
ISSUP CRMOY BRINE YWEEL AYLEW TYRTI LHSTO
```

5. Break this ciphertext.

```
MICYRLHHTAHTLZHAENOHIEALEOBFGALETETOSAZGTAFHRAGATEFLVN
YATHTETARFRNDINIEDDNGSNCGPOTOEEPIGTFMLEWNIEOERGGTEESER
ILFTNBNEIKIASTSRVPZGPEOINIALPLGNOINWTTSWSMRAYIRSSZTNHD
AOEEIENRNWTTSODOAOSLCSRCTEFOESLYALIATRENISDHNKCFGOHUOR
ITLPEUHREAINEELTGEDORT
```

Unit 64 (optional)
Redefence cipher

The *redefence cipher* is a modification of the railfence cipher in which rows are read off in a different order. The key for the redefence is the inverse permutation of the rows (we use the inverse so that it lists the order in which they are read) and the offset with which the plaintext is written in.

Here is the example from the previous unit. We will take the key to be (1 2 0 3) and 5. The plaintext is written down with the offset:

```
0     •-----H-----S-----S-----P-----T-----N-----T-----P---
1     -•---T-I---S-A---A-E---Y-T---I-H---A-S---I-I---I-H--
2     --•-•---S-E---G-W---N-R---E-W---A-R---P-S---O-C---E-
3     ---•-----M-----E-----C-----D-----T-----O-----N-----R
```

The rows are read off in the order 1, 2, 0, 3 to get the ciphertext.

```
TISAAEYTIHASIIIHSEGWNREWARPSOCEHSSPTNTPMECDTONR
```

Reading and references

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Redefence.pdf

Programming tasks

1. Write a function or script to encipher a plaintext with the redefence cipher and a given permutation and offset.
2. Write a function or script to decipher a ciphertext with the redefence cipher and a given permutation and offset.
3. Implement a brute-force attack on the redefence cipher.

Exercises

1. Encipher this text with the redefence cipher. Use offset 2 and permutation (3 2 0 1 4).

   > I have committed sins, of course; but I have not committed enough of them to entitle me to the punishment of reduction to the bread and water of ordinary literature during six years when I might have been living on the fat diet spread for the righteous in Professor Dowden's Life of Shelley, if I had been justly dealt with.
   >
   > (from *In Defense of Harriet Shelley* by Mark Twain [Samuel Clemens])

2. Decipher this text with the redefence cipher. Use offset 3 and permutation (2 1 0 3).

   ```
   HJCTIOIILRTRTUPAESFAFAIAYNPTOWODRTININWHASLCRLNNECSLYA
   EGRYNPETPUEHRWIUTVTYAPNNHEIFHOFITDYEBTFSOSOUTEUTRRPDEO
   WRMLTSAONSOWNCRNOTOHCYAYCEGDASCSUIVDYMNDOSNITOTUNCMIEA
   LLUUOEVNOANADATOOBTSOEENROITIHUECIIEONCUFNBATECFIHNBLS
   EGTRNTEHKLAUNRSRMRDTHEADSMIUABELDEARAREHSIFOTNTTSA
   ```

3. Break this ciphertext.

   ```
   AITEENSTIMUTOOSEEEIYDEAATMAATWTNTTSDEIRAOESPSOIIFATRIC
   UINHDSFATRAIFRRWITDAVLHYENLREANARGYLTHGEOPENDHYINUOQAI
   CIOINFHMSNTANLMFCOEACOLDEMPPLESWRHTUANPUEOOHCSYSNHTILT
   NJCISDUOSRGNIADOISUTHLVLYETSIEPINREEVOHIIIDHRFLFLYASOA
   FGDMIAOALAENOUIKNSOTELRTOOCLTTSEARSNRCINTSRIBNIELTONCI
   LTSTRWHOISOTTWBSTETSANTEBLHINNSPTDNAELEACUHNNVPTAOTSET
   EDOWOTA
   ```

Unit 65
AMSCO cipher

The *AMSCO cipher* was invented by A. M. Scott. How it got its name is still a mystery.

To encipher a text with the AMSCO cipher, we are going to write the text into a matrix. We must first decide whether the first box should contain one or two letters. After that, we fill in by rows in such a way that when looking across a row or down a column, entries in the matrix alternate between holding one and two letters (the box containing the last letter need not be padded to hold two letters). The message is not padded, so some columns may remain incomplete. For example,

TH	I	SM	E	SS	A
G	EW	A	SE	N	CR
YP	T	ED	W	IT	H
A	TR	A	NS	P	OS
IT	I	ON	C	IP	H
E	R				

The key is the order in which the columns are read off to form the ciphertext. If our key is (2 5 0 1 4 3), then the ciphertext for our example is

```
SMAEDAON ACRHOSH THGYPAITE IEWTTRIR SSNITPIP ESEWNSC
```

Notice that the key is the inverse of the permutation that acts on the columns. The key can also be represented by a keyword.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; page 51.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Amsco.pdf

Programming tasks

1. Write a function or script to encipher a plaintext with the AMSCO cipher. Allow for the choice of using one or two letters in the first place.

2. Write a function or script to decipher a ciphertext with the AMSCO cipher. Allow for the choice of using one or two letters in the first place. Be careful how you handle incomplete columns.

3. Implement a brute-force attack on the AMSCO cipher.

Exercises

1. Encipher this text with keyword `YACHT` and two letters in the first box.

   > This unlikely story begins on a sea that was a blue dream, as colorful as blue-silk stockings, and beneath a sky as blue as the irises of children's eyes. From the western half of the sky the sun was shying little golden disks at the sea—if you gazed intently enough you could see them skip from wave tip to wave tip until they joined a broad collar of golden coin that was collecting half a mile out and would eventually be a dazzling sunset.
   >
   > (from *Flappers and Philosophers* by F. Scott Fitzgerald)

2. Decipher this ciphertext with keyword `PARADISE` (drop repeated letters) and one letter in the first box.

   ```
   AMRAMEEOSSTPRORUNEELRYREDERRCANIHIGDAEEDETHIMOSORLDTME
   OGAVIETDNGNIVEYRVGMUINGREASFYOOSANTLELAWCHSINOYNNEWEFY
   OGOFE
   ```

3. Break this ciphertext. What might the keyword be?

   ```
   EHEBWIASHITLONWWEDNEROEYEESENDETERIMPEETHSSAETSTEGNTTA
   RYKZIHECSTHHSPIESUHEHHALTHDETESNAPPRCEEASMGAORORSTTNDN
   DRTRAFEPRYBPASNMAINSYPIHHOUERORNEFITAKUCOAESSERFITEFSC
   GHBREIMERICADSHHIGIPOGAHPBENSTIBSUCOLEINDAIPSJEPAGRFTF
   INRSTAGTIIMIWITTOROKGSNRTATOEILHPAEUTEDURARARINCHTELCF
   O
   ```

4. This ciphertext is from the 2015 British National Cipher Challenge. Break it. What is the keyword?

   ```
   UOCAD EDMMA EDAEC RRSOI COIPI IFEBO BDUSI SDSBE ENTTY
   ANETA DCOUD KOOTT RBELE DRAOA USYYO HISNL EDDSR ISHRS
   HEEOS ETROS TLSEU NRSCC OUWRI MOLHN STERS EDEEA RINTE
   WPENN OMSFT AITOF TOILL CAPEC ESEPN DUETN DEEUS ESOMF
   AKOGT INLGE LYARU ITSIC RIOIR SECOR ETNEU EATMT ALIHL
   EICRV EASME NCIED ATHEO LKWHE OHNWO NFIMH THUBE AIBND
   ```

ATTHU EKTDO OECTW ITABA OULCA QWIUP IYPOC ODEEO ONSHI
TCIEE ATAIA CLTTA RCHGH OFHET DPAEC HMAHV EHLOL YARLE
IKEDR ODTEI DYSKE EYCFS HFOFE HTGEJ NDBBR YPHON IOSRO
LMREF RWFII GHYPH NMEHR ELASO NDEIV TDTIR HEROT URECH
ORHEG YNDED MEPYT WVIID OMOUT YAION AUFUC ATWOU OKTEF
LTHMU TGNEC TITAR AOWEN NGSST EFRRE SCLFS PSATO EHIHE
LASTV CHIFI IUTET EYEEC STNHE OILDS TNEPM ROSTT SRAFL
LRHED ISESS RETIE NASU

madness's book on classical cryptography
unit 66: myszkowsky cipher
last modified 2020-10-03

Unit 66 (optional)
Myszkowsky cipher

The *Myszkowsky cipher* is a modification of the columnar transposition. It requires a keyword that has repeated letters; otherwise, it degenerates to a regular columnar transposition. The plaintext is written into a grid with the same number of columns as letter in the keyword. The columns are labeled by numbers representing the alphabetical order of the letters of the keyword, where identical letters have identical numbers. To read off the ciphertext, we start with the first row and read all letters in a column labeled 0. Then all letters in the second row in a column labeled 0. When we have finished all rows, we start again at the top with all columns labeled 1. Then 2, etc.

An example couldn't hurt. Let's encipher this short message with the keyword `TATTOO`.

`THIS MESSAGE WAS ENCRYPTED WITH A TRANSPOSITION CIPHER`

The keyword `TATTOO` is converted to numbers 2, 0, 2, 2, 1, 1, because 'A' is alphabetically first (zeroeth), 'O' is next, and 'T' is last. We write the plaintext into a grid with those column headings. There is no padding.

```
2  0  2  2  1  1
─────────────────
T  H  I  S  M  E
S  S  A  G  E  W
A  S  E  N  C  R
Y  P  T  E  D  W
I  T  H  A  T  R
A  N  S  P  O  S
I  T  I  O  N  C
I  P  H  E  R
```

We read off all of the letters in 0 columns: `HSSPTNTP`. Then all 1 columns: `ME EW CR DW TR OS NC R`. Pay close attention to the order of the letters. We take all 1 columns by rows. Now, all 2 columns: `TIS SAG AEN YTE IHA ASP IIO IHE`. The cipher text is

`HSSPTNTPMEEWCRDWTROSNCRTISSAGAENYTEIHAASPIIOIHE`

Reading and references

Émile Victor Théodore Myszkowski, *Cryptographie Indéchiffrable basée sur de nouvelles combinaisons rationnelles*, Paris: Société Française d'Imprimerie et de Librairie, 1902, gallica.bnf.fr/ark:/12148/bpt6k1265620p

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 50-51.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Myszkowski.pdf

Wikipedia, en.wikipedia.org/wiki/Transposition_cipher#Myszkowski_transposition

Merle E. Ohaver, "Solving Cipher Secrets," *Flynn's*, September 17 and 24, 1927, toebes.com/Flynns/Flynns-19270917.htm, toebes.com/Flynns/pdf/Flynns-19270917.pdf, toebes.com/Flynns/Flynns-19270924.htm, toebes.com/Flynns/pdf/Flynns-19270924.pdf

Programming tasks

1. Implement an encryptor for this cipher.
2. Implement a decryptor for this cipher. Be careful about incompletely filled columns.
3. Implement a dictionary attack for a ciphertext that was encrypted with the Myszkowsky cipher.

Exercises

1. Encipher this plaintext with the keyword ABRACADABRA.

   > Here began the experiences that quickly ripened Houdini into the World's Handcuff King and Prison Breaker, which he is, has been and always will be. In exploring his wits for exploits to amuse and entertain the audiences, Houdini hit upon the feat of escaping from ropes tied round him in every conceivable way.
   >
   > (from *The Adventurous Life of a Versatile Artist: Houdini* by Harry Houdini)

2. Decipher this ciphertext with the keyword DOLITTLE.

```
OAYOGHLIEOHALLMHAOKLNNGRTEHHDDWTODTSDDOEMAENSTRWONOOTE
HOOWOCOIAEAHUAARRTCDTEATNSEDTJDTMMSTARRTNEHLNTYWREILRC
ISEIDAPCNEUPEMRSNODFWELEENASRAAMLIHNLEANEWPERAAWT
```

3. Perform a dictionary attack on this ciphertext. The keyword is an English word between five and ten letters in length.

```
WHUHWHTIBEPSIIIYOAHNDNTLHMTBRESOHATENESLYLAHNTNLRSAFBA
TOTATEIGIUESLOTEROLWTATWLDAYSEGTOGMWHGNMICDWBIVIONEOFA
WCCGIERAHKGTKBAYMESTRLROTOMIOHTAOBENOEEEUWBYFTBIOFRTTO
AEERDLVRAEEKNTSKTEOAOEDNCUALHEITUNBEHOFYFERERHTINEHLYB
DTHSSMSTOELIEHEENGFEIEHAIRNSBENMEAIEEIDYLVOATIWAREDHCL
CNTSANITORLTNY
```

Unit 67
Cadenus cipher

The *Cadenus cipher* involves both a columnar transposition and a rotation of the columns. The key is expressed as a keyword in which repeated letters are dropped. The plaintext must be a multiple of 25 times the length of the keyword after repeated letters are removed. These are the steps in encipherment:

1. Divide the plaintext into blocks that are 25 times the length of the keyword (after dropping repeated letters in the key).
2. For each block:
    a. Write the block into a matrix as though for a columnar transposition. Each column has a corresponding letter from the key. Each column has 25 letters.
    b. For each column, roll it downwards by an amount determined by its letter in the key. Use ‘A’ = 0, ‘B’ = 1, ..., ‘V’ = ‘W’ = 21, ..., ‘Z’ = 24 (‘V’ and ‘W’ take the same value so that there are only 25 values).
    c. Apply a columnar transposition using the keyword, without repeated letters.
    d. Read off this block’s part of the ciphertext by rows.

An example seems necessary at this point. We begin with a plaintext that has 375 letters and the keyword `ORATIO` (because Shakespeare did not pronounce his ‘H’s).

```
TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER
IN THE MIND TO SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS
FORTUNE OR TO TAKE ARMS AGAINST A SEA OF TROUBLES AND BY
OPPOSING END THEM TO DIE TO SLEEP NO MORE AND BY A SLEEP TO
SAY WE END THE HEARTACHE AND THE THOUSAND NATURAL SHOCKS
THAT FLESH IS HEIR TO TIS A CONSUMMATION DEVOUTLY TO BE
WISHD TO DIE TO SLEEP TO SLEEP PERCHANCE TO DREAM AY THERES
THE RUB FOR IN THAT SLEEP OF DEATH WHAT DREAMS MAY COME
WUTEVUH
```

After removing repeated letters, our key is `ORATI`, which has five letters. So we break the plaintext into three blocks of 25 × 5 = 125 letters. We lay the first block into a matrix of five columns under the key (shown on the left below). The numbers of steps for rolling each column are ‘O’ = 14, ‘R’ = 17, ‘A’

= 0, ‘T’ = 19, ‘I’ = 8. The columns are rolled downwards (shown in the middle below). Then a columnar transposition is performed (shown on the right).

O	R	A	T	I
T	O	B	E	O
R	N	O	T	T
O	B	E	T	H
A	T	I	S	T
H	E	Q	U	E
S	T	I	O	N
W	H	E	T	H
E	R	T	I	S
N	O	B	L	E
R	I	N	T	H
E	M	I	N	D
T	O	S	U	F
F	E	R	T	H
E	S	L	I	N
G	S	A	N	D
A	R	R	O	W
S	O	F	O	U
T	R	A	G	E
O	U	S	F	O
R	T	U	N	E
O	R	T	O	T
A	K	E	A	R
M	S	A	G	A
I	N	S	T	A
S	E	A	O	F

O	R	A	T	I
T	O	B	T	E
F	I	O	I	O
E	M	E	L	E
G	O	I	T	T
A	E	Q	N	R
S	S	I	U	A
T	S	E	T	A
O	R	T	I	F
R	O	B	N	O
O	R	N	O	T
A	U	I	O	H
M	T	S	G	T
I	R	R	F	E
S	K	L	N	N
T	S	A	O	H
R	N	R	A	S
O	E	F	G	E
A	O	A	T	H
H	N	S	O	D
S	B	U	E	F
W	T	T	T	H
E	E	E	T	N
N	T	A	S	D
R	H	S	U	W
E	R	A	O	U

A	I	O	R	T
B	E	T	O	T
O	O	F	I	I
E	E	E	M	L
I	T	G	O	T
Q	R	A	E	N
I	A	S	S	U
E	A	T	S	T
T	F	O	R	I
B	O	R	O	N
N	T	O	R	O
I	H	A	U	O
S	T	M	T	G
R	E	I	R	F
L	N	S	K	N
A	H	T	S	O
R	S	R	N	A
F	E	O	E	G
A	H	A	O	T
S	D	H	N	O
U	F	S	B	E
T	H	W	T	T
E	N	E	E	T
A	D	N	T	S
S	W	R	H	U
A	U	E	R	O

We read off the ciphertext by rows. The remaining blocks are processed in the same way, and in the end we get this ciphertext:

```
BETOTOOFIIEEEMLITGOTQRAENIASSUEATSTTFORIBORONNTOROIHAUOSTMT
GREIRFLNSKNAHTSORSRNAFEOEGAHAOTSDHNOUFSBETHWTTENEETADNTSSWR
HUAUEROOSEOTSAAAEYLNAOSKHPDNTAYLMHNDOEITEEEIACHMBTDRNNSHESP
SNHTNFUUWTIHNTORTAAOTLCHPLSATRDTSOBPREDEGETRSHBUOEDOAHESEOE
TNEIHAEIDOEYLTCSAEDMOOLSOTMETOTENETMHDCICFAEOHTREOVLETPNFOS
EAHHURDDEICTSDTRBOWPYHARAEISMARLSMAROTTYAPEAWEHLUVETEIOHADV
MEHEYNATEWUEBSTOENPTS
```

Reading and references

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Cadenus.pdf

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.
3. Implement a dictionary attack.

Exercises

1. Encipher this text with the keyword `INTERNET`.

   > While my internet is down, I have to write my own plaintexts. It is very frustrating, but this is the cost we bear when we steal internet from the store down the street. I'm sure you don't want to hear about that. Perhaps you would like some historical bits of information about the Cadenus cipher? Unfortunately, I don't know any. Let me look some up online...

2. Decipher this text with the keyword `HEART`.

   ```
   DARHNAIIRTNRAEIWEDTLOAWANJSERHHAHTNIACUIRPUHWIYVTBITSE
   ESWAIEVOBODMGACNIRSHRPEFIESRSXHFOFESHHDNXDSEENCIHADNAG
   ESTSAOSWELAMSCDFN
   ```

3. Break this ciphertext with a dictionary attack.

   ```
   EWOIETEHTDOEINPEHAEAAERRRDNWIOTDHYHLEDONRPWTEPWSRFMNAE
   HHHEHNLQAORRGECAXEUHLHREHRSATNRASEESRWSLITAMITAPIIEPSI
   SAHWPWEWJOPEIDWUEDDSNATSSTTANTOARATEAAGLTISAIRNGENBUUI
   NFSEEFOXIODWATGNOEAFESEHDEIITIRDYSGSOGRENCBNIRUDLCSNTO
   AGBATTOITALNHNIGIIPLVDESHEHTTVIIHEUDTNTNPTAINSEPTPWEOS
   HETLWESLEEWAMHWBSLIRNCRRUUTTCSGRHEEYTLLMROMERENCOTHATE
   GNFHWAAIDGINUFYNRSNSAMORAUGSWBNTIFEREDILLTILEOEAERSHEY
   NTDDNEVIRRNSBXSTXWHERDGVITISNAMEPSOSHBHWWBWDTERTEEATAE
   DTSSLSTTRIATESUOEOSELOEENSNWEANEOAATASHAIRPPANWEOSIEHO
   NLCHNNTUDTISEE
   ```

Unit 68
Hill-climbing attack on the Cadenus cipher

In this attack we unlink the rolling and columnar transposition and vary the keys of the two operations separately; at the end, we put them back together. Therefore, we need a decryptor that applies them separately. Before we apply the attack, we have to determine the key length (or guess it). In the exercises in Unit 9 we found a cutoff above which we are confident that we have English text; we will use that cutoff in the algorithm. We will be using the usual technique of working with parent and child keys. To avoid local maxima, we also need to use a margin of error. For this cipher, we will vary this margin so that it vanishes when we reach the fitness cutoff.

0. set the alphabet to be `ABCDERGHIJKLMNOPQRSTUVXYZ` (no ‘W’)
1. initialize the parent shift key to an array of 0s
2. initialize the parent permutation key to (0 1 2 ...)
3. calculate the best fitness as the fitness of the unmodified ciphertext
4. set a counter to 0
5. while counter is less than 1,000
    a. copy the parent shift key into a child shift key
    b. copy the parent permutation to a child permutation
    c. randomly choose one of these ways to modify the child keys:
        i. change one member of the child shift key to a random number in 0, ..., 25
        ii. swap two randomly selected members of the child permutation
        iii.
            - randomly choose a number *n* from 1 to the key length
            - roll both keys leftward *n* of steps (with rollover)
            - subtract 1 from each of the last *n* members of the child shift key
        iv.
            - randomly choose a number *n* from 1 to 25
            - add *n* to each member of the child shift key, modulo 25
    d. decipher the ciphertext with the child keys
    e. calculate the new fitness of the new plaintext
    f. set the margin to be
        i. 0 if the new fitness exceeds the cutoff
        ii. the square root of the cutoff minus the new fitness, all divided by 10,

if the new fitness is less than the cutoff

g. if (the new fitness exceeds the best fitness) or
((the new fitness exceeds the best fitness minus the margin) and
(we roll a 1 on a 20-sided die))
    i. copy the child shift key into the parent shift key
    ii. copy the child permutation into the parent permutation
    iii. set the best fitness equal to the new fitness
    iv. set the counter to 0
h. increment the counter

6. convert the parent shift key to a keyword; for each member of the shift key:
    a. take *n* to be 25 minus the shift
    b. the letter of the keyword is the n^{th} letter of the alphabet (without 'W')
7. output the keyword

Programming tasks

1. Write a function that deciphers a ciphertext with separate keys for the shifts and for the permutation.

2. Implement the attack. Use your function from Exercise 1, and use tetragram fitness.

Exercises

1. Use your attack to break the example from the previous unit.

2. Break this ciphertext from the 2014 British National Cipher Challenge. It may take several attempts.

```
AFCAEUOTTACTHRIOLETCSERTHSHTRAHKYORPFRGEOADPPJNGLTERNE
FEOFIORTSDDOEEUMSCRUERNFETLAAFSTWIENTRVOONERHUAHRAVERE
ETSVSIELHLOSTDOALOYAESMNNDIGNNRHOHHTSNAOILNCNSSICREANN
EEIIIERWTANESRVOGIEIYWSSDGPVOIAISAOAEOAEDRNITRNXEIGRPS
SHADHDTOIPAATEXENNESAGROBTLESRNROIRYPBGEDCLLIWALALEENI
GRRNWYRLIMLPSTOLEFTRDMUARIEEEIIAOLNEWSAOHRTLSTOBETNSLV
FIVDOVTPOAEEISCIOHIPSEVEEDTEWFARNHEBLEAOTOHTTTEPNCKAON
HWETMVYPRREONNASGDEDOEEEOAAMTCICTTIFNADRESRTSEROSETRHC
ICTPSAAEHLDHSFXSOAOTCTBBSOEIRNSADLYTRRUNRCEPTTHREUHNKT
ACECEELRWNIREEEAESEEEIDISOGCEOMNRTEJHAGABSENITLWTRNBMI
ELSARETESRNGSNHEBIOSDIENAFLEISAHOCIFEVMFATANATRNIAGNHA
TNMIBNIUFENRTOTTRNYPAIDYIEGDNMERHHIOTRETCESSEILDRBCEPR
IGAESOADLTAHIEVEBRCENLEVASADNNTHNEITEIISAHUHHUAMONEFYH
LONWHAEEEEOSNEEYANEISETOGYITERLIHTCMIOIRARFDOETNIHTNEH
IIKAMRDMNADANAODSESEIYCLSIANTAOLTCIYMIDENTTHLTNDXTTTMA
SBLEAEETLISIRTWTURPFAILTEAOEFEISIIIYISIKVTWISPRBSINELP
```

HRMOHIAGNLSLVITODAISDPNYDDCAAOTAHCEHTUEIRREDAECTOSNRHV
NAODOIKOETCINENEURRISDCOURAGLVIMMUPPDITEANDITMAAIAIELE
ONNREEDAODBOIUMELROTNTTTGITNRLRIENNIKLYSOGSTCIFYPIPVID
VSSMNCEIASIITSNNEATITOMRHBNHNIDPRLREPOYNALSNVSDOSANESI
TFAENLTGODATTEEAISICROOTMSMFHAUENIRSGHYNWEINTEGODIILEE
DTARNOSRCAAENDTCUTTFDRBEHTMFITOORDRUIAOYAANOEELDOINHUS
GITEAORIECEVEMNTRATMTFPEUCUTAHAMTNEWONICDEEMRPAOLITOAF
ESOOSSPFNLNEEOOTACHLLIRSSXSOFPDFTFRNPRAEEAYLONAHAUTNTC
NTCBAWLONEFTOATECVOWDLWVNNEEDTIIOIGTEGMTAHEEATEFAAEPRR
CROSHEERRPALEDIENGIDRREOUHVESUROYTNSOSINUIUIOFPRDA

Part V
Grid-based ciphers

Unit 69
Polybius cipher

If we cram the alphabet into a 5×5 matrix, we have a *Polybius square*. Unfortunately, the English alphabet has more than 5^2 letters, so we may have to jettison one of them. Usually, we will put ‘I’ and ‘J’ in the same spot.

Now, if we add labels to the rows and columns, we can use the Polybius square to make a cipher, which we will call the *Polybius cipher* or *Polybius-square cipher*.

	0	1	2	3	4
0	A	B	C	D	E
1	F	G	H	I J	K
2	L	M	N	O	P
3	Q	R	S	T	U
4	V	W	X	Y	Z

The application of the cipher is obvious: we replace a letter of the plaintext with the row and column labels of its location in the grid. For example, with the matrix above, we can encipher this short message:

```
THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER
33121332 21043232001104 410032 042202314324330403 41133312 00 11311303 021324120431
```

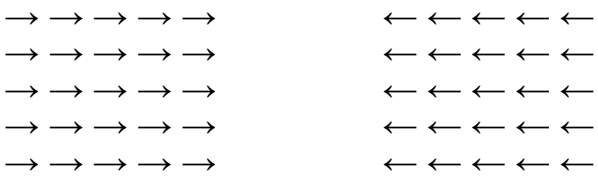
The ciphertext is

```
331213322104323200110441003204220231432433040341133312001131130302132412043l
```

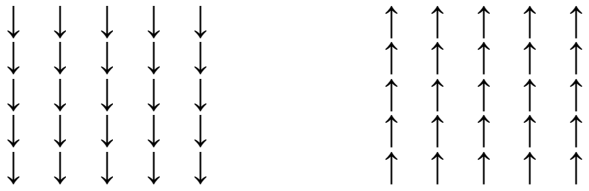
We are not forced to use the digits 0, ..., 4 as our labels, and we are not constrained to use the same labels for rows as for columns. The only constraint is that all row labels must be different, and all column labels must be different.

The cipher can be keyed by using a mixed alphabet. In Unit 26 we saw several ways to construct mixed alphabets from keywords. With this new cipher, we add a new dimension. There are many ways to lay the mixed alphabet into the matrix. Here a just a few:

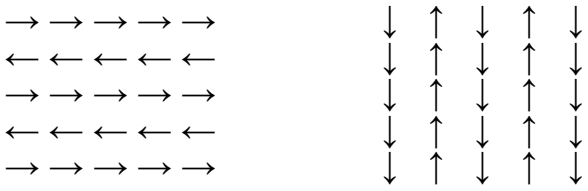
- by rows



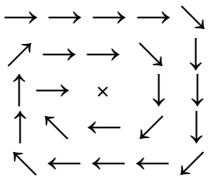
- by columns



- boustrophedon (by rows or columns in alternating directions)



- spiral, outside-in or inside-out, clockwise or counterclockwise



At this point, you should realize that no matter how we mix the alphabet and no matter how we place the mixed alphabet into the grid, all we have done is replace each letter with a two-character string (a code word). By listing all two-character strings in the ciphertext and assigning a letter to each, we convert the Polybius cipher to a monoalphabetic substitution cipher. In Unit 28 we saw a method for solving monoalphabetic substitutions automatically.

Reading and references

Practical Cryptography, practicalcryptography.com/ciphers/polybius-square-cipher

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 190-191.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, page 83.

Programming tasks

1. Write a function to fill a Polybius square. Allow for many options on how to generate the mixed alphabet and on how to fill the grid.

2. Write a function or script to encipher a plaintext with the Polybius cipher with a keyword, an alphabet-mixing method, and a grid-filling method.

3. Write a function or script to decipher a ciphertext with the Polybius cipher with a keyword, an alphabet-mixing method, and a grid-filling method.

4. Implement the attack mentioned above.

Exercises

1. Decipher this ciphertext with the keyword `POLYBIUS`. The mixed alphabet is constructed by adding letters that come alphabetically after the last letter of the keyword (in this case, start with 'T'). The grid is filled by columns. Labels are as in the example above.

```
24100231433231023221243121104332440302322101310010203040
40011121420100431344213244131332213031104244324204322020
30301011241313030121324201004313443113123131433231012120
20102433132410113343231044301011311040441332043143131400
10241032200043324013310142211140213101311131011024420100
431344
```

2. Break this ciphertext from the 2019 British National Cipher Challenge:

```
FBGAI AGCFE KEFEK CIAGC FCGAF CIBHD HEFCF AFBFA GDFCH
DFEKC IAKCI BGBGC IAHAF EKCFA KAIAG CFBFA GBFBI AFBHE
IAGCK CIAFC IBHDF EGAGA FCHDI AHEIA FCKDF CFAIA KCFBF
AIAGC FEHEF CICFB FEIAH EFDKA HBHDF CIEKA IDKCH DHEFB
HEKEF CFCHA FEKEK CHEHA KAGEF CKCIA GCFBF AGBFC GAIAG
CFEHE FCICF BFEIA FEHAH BFBHD FEIDK CHEHE IAFCG DFEKE
FDKAG CFBHE KEFEK CIAGC IAGCF EIBHE HEHDG AFEFE KEHEF
CFAIA GCFEK CFAGB FEHDI AGCKC IAIDF EHEIA FBHDI BHBID
FBIAG CFEKC KDGCI DKCHD IDFEG AFBGB GCIAF BFAFB IAHEH
BIBHB HBFEI AHEIA KCIAF EHEID FEGCK CICFE IAHDF BFEKE
IAFCK DFCFA IAKCF BFAFB IAFDI BIAFB FAFCI DHDFE KCGEF
BHEFE IAGCK CIAID FEKDK CFAFC FAGEK AKEFE GAFEK CIAFB
IAIAG CFEID KCKAI DFEGA FCIBG BGCIA GAKCH EKDFB HEHAF
DKAIA KCGDF BFAGB FCFAF BIAHE HAFBG BGCIA FBFAF CHBFE
FAIDK CHDFC IBHDH BFCGE FBIAF BKDFB KCFAH EGCKC ICFEH
EGCHD IBFAG DGAHD FCHAI AGCFE KDFCF AGAGE FBKDI AKCFA
KEIAG CFEKA FAFEF EKEKC HDFEK CHEFC FAIAF CKDFC FAGAH
DFCFA IAIAG CFEFE ICFBG E
```

Challenge

Unit 70
Playfair cipher

The *Playfair cipher* was not invented by Lord Playfair, but rather by Charles Wheatstone. At any rate, it is a *digram substitution cipher*, which means that it makes substitutions two letters at a time. It has little tolerance for double letters, so before we can apply the cipher, we have to prepare the plaintext by putting an ‘X’ between all adjacent pairs of identical letters. Since it works on pairs, we also need to add an ‘X’ to the end of the plaintext if it has an odd number of characters. (We do not need to put an ‘X’ between letters if they are in different digrams.)

The main engine of the Playfair cipher is a Polybius square. We fill it with a mixed alphabet that was generated in whatever way we like. Typically, ‘J’ is merged with ‘I,’ but some prefer to merge ‘Z’ in to ‘Y.’ The plaintext is processed two letters at a time, according to these rules:

- If the two letters appear in the same column of the Polybius square, then each is enciphered to the letter below it. If the letter is on the bottom row, then we use the letter at the top of the column.
- If the two letters appear in the same row of the square, then each is enciphered to the letter to its right. If the letter is in the last column, then we use the first letter in the row.
- If neither of the previous two cases hold, then a rectangle is observed in the grid such that the two letters are at two of its corners. They are enciphered to the letters in the other two corners. Each is enciphered to the letter in the other corner in the same row.

(Mathematicians in the audience recognize a torus when they see one.)

Now for an example. Here is our plaintext:

```
THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER
```

First, we prepare it with nulls, one between the two ‘S’s and one at the end.

```
TH IS ME SX SA GE WA SE NC RY PT ED WI TH AG RI DC IP HE RX
```

Let’s use the keyword `POLYBIUS` to mix our alphabet, and let’s not do anything fancy about how we lay it into the square:

```
P  O  L  Y  B
I  U  S  A  C
D  E  F  G  H
K  M  N  Q  R
T  V  W  X  Z
```

The first two letters of the ciphertext, `TH`, are enciphered to `ZD`:

```
P  O  L  Y  B
I  U  S  A  C
D  E  F  G  H
K  M  N  Q  R
T  V  W  X  Z
```

The next two, `IS`, are enciphered to `UA`:

```
P  O  L  Y  B
I  U  S  A  C
D  E  F  G  H
K  M  N  Q  R
T  V  W  X  Z
```

This continues, and the ciphertext is

```
ZDUAVMAWACHFXSUFRSQBIPFETSZDGQKCHIDIDFQZ
```

To detect whether we have a ciphertext that has been encrypted with a grid-based digram substitution cipher, we look to see if there are at most 25 different letters and whether the length of the ciphertext is a multiple of two (to disguise the cipher, however, one might change some ‘I’s to ‘J’s, so be careful). If there are any long repeated sequences of characters, then the starting character of each needs to be an even number of letters apart. Furthermore, if we plot the index of coincidence as a function of period, as we did in Unit 31, we often see a slight increase in the even periods over the odd periods (do not take this to mean that the cipher is periodic, just that this is a differential tool).

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XXI.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Playfair.pdf

Wikipedia, en.wikipedia.org/wiki/Playfair_cipher

Practical Cryptography, practicalcryptography.com/ciphers/playfair-cipher

United States Army, Field Manual 34-40-2, chapter 7, “Solution to Polygraphic Substitution Systems,” Basic Cryptanalysis, U.S. Department of Army, www.umich.edu/~umich/fm-34-40-2/ch7.pdf

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 217-219.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter XII, section I.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 198-202.

Joseph O. Mauborgne, *An Advanced Problem in Cryptography and Its Solution*, Fort Leavenworth (Kansas): Press of the Army Service Schools, 1914, www.marshallfoundation.org/library/digital-archive/advanced-problem-cryptography-solution

W. W. Smith, "Solution of the Playfair Cipher," in part IV of André Langie, *Cryptography*, translated by James C. H. Macbeth, London: Constable & Company, 1922, HDL: 2027/uc1.32106002774104 and 2027/uc2.ark:/13960/t0tq62t29

Programming tasks

1. Write a function that returns a boolean value representing whether it is likely that a given ciphertext has been encrypted with a grid-based digram substitution cipher.
2. Implement an encryptor for the Playfair cipher.
3. Implement a decryptor for the Playfair cipher.
4. Implement a dictionary attack. Remember to allow for many possibilities for the method of mixing the alphabet from a keyword and for the method of laying the mixed alphabet into the square.

Exercises

1. Encipher this text with the keyword `TRENDING`. Fill the square with whatever method you like.

   > The point I advance, if it need confirmation,
   > I'll prove by a witness that few will dispute,
   > A pink of perfection and truth in the nation
   > Where fashion and folly are all of a suit.
   >
   > (from *Nothing to Eat* by Horatio Alger and Thomas Chandler Haliburton

2. Decipher this ciphertext with the keyword `GROCERIES`. The alphabet was mixed by starting at the beginning of the standard alphabet after the keyword. The mixed alphabet was laid in by rows from left to right.

```
MDSOASOGTGKCDRBZEQVSKYMHFVIBDSKYMHCOROCEGODGABUICQMROR
AOEAIHPEVFHPDMQCXCNDPUMRKBBPASZKGQPLABKENPNBVIQCASYQWB
GZGUAEKYKBSHIQBUFSCPVLEQOEGUPBBNEQRFQYQCKSZGDCGUQSSIDC
KGOGKRXZEQDKFVSAUCOCLNMRRCHWCMOBVFPDNBLVXCPEDRMHFVPDFV
OVRCEAHRFSRLXCZMGQUQBXKGGSOBPUNPMDSHQBUIFNSGDUDUDCOWGS
RFYTCYMRDSLTRDBXARZRQGKDQITVPLFVOIASDPQWQRDRXCPEGECRVF
EDPLCDSDMCBAIQDQPLCOBNVBOZURBYXCNURQBXNQWSEKQUTCIQAELT
FICZEQSHOGHWGENLTMTCPLEKBAUNAEOW
```

3. Apply a dictionary attack to break this ciphertext.

```
CQAHUBVNTIZNTODRBAFRCEAWRKKRNODRVTMCNZITVTUBKHQCXEPRBN
TZFRZHMBABNOXEQGBTWRTZPODRCUQZOPKVUFDWONTZDRNKFWFKDEDR
BANOPGVBIWMTAWRKXEPRVTLQPUDKOMKATZMXZIKDVTLUDARPTBISBA
QCTPMKCKDBZNBTYCGNRLXVBNRABHFRLZKRQFKHQFPDOPZNDKMOQRTZ
PVBLLHBFOTVRADTBZNVPYCDVTOMBADTBISCLOTEVILTODRBNRAVNMC
NZFRRAEARKNKTOELRKCQBVKRIQMHRQDKQKBLITETKBDBQCPQRKQFKH
ZNVPZBLFDBQCZNVPITZDABDEVQHASIAWQPOYBAOPUKKHRLGPZIVQFR
EHVQQERAKVTZ
```

Unit 71
Hill-climbing attack on the Playfair cipher

We are going to begin with Cowan's attack ("simulated annealing") and make some improvements. In his method, fitness is measured as the sum of tetragram frequencies. A parent key (in the Polybius square) is set, and from it a child key is generated by swapping individual letters, swapping two rows, swapping two columns, or flipping the entire square. If the fitness of the decrypted plaintext from the child key is greater than the parent, then the child becomes the parent. This is a step upwards. However, it is easy with the Playfair cipher to become trapped in a local maximum fitness. To allow the algorithm to escape such a fate, a "temperature" is added by allowing a step to go downwards in fitness if the distance down is smaller than a random variable that drops off exponentially and which depends on temperature. As the temperature is reduced, the jitter in the motion in key space gets smaller and the distance downward that is allowed gets smaller. The temperature is allowed to slowly decrease to zero. If the algorithm has not found the global maximum fitness at that point, then it starts over with a high temperature. The algorithm actually does not know if it has found the global maximum, and requires human interference to stop it.

The first improvement that we make on this algorithm is to use tetragram fitness as described in Unit 9. Our definition of it is an average over all tetragrams in the decrypted text. As an average, it is not dependent on the length of the text. Therefore, it can give the algorithm a clear indication that it has found the maximum and can stop. A threshold above which we are confident that we have English text was found in the exercises of Unit 9. For the Playfair cipher, we remove all 'X's before evaluating the fitness, since they would have been added between any double letters.

Our second improvement is to use a constant margin of error for downward steps. In our algorithm, a downward step is allowed if the distance downward is less than a fixed amount and if a random variable is within a predetermined interval. That margin is 0.5, and downward steps are allowed only if the random variable lands in 5% of its range. The global maximum will be steep enough that the algorithm cannot walk downward out of it. So we can terminate if we have reached a point from which we cannot step upwards within a large number of tries (around 10,000).

Here is the full algorithm:

1. set the parent key to a Polybius square with an unmixed alphabet
2. set the best fitness to the fitness of the unmodified ciphertext
3. set the counter to 0
4. while the counter is less than 10,000

a. copy the parent key into a child key
b. randomly choose one of these modifications to the child key:
    i. swap two randomly selected elements
    ii. swap two randomly selected rows
    iii. swap two randomly selected columns
    iv. flip the square around the diagonal that runs from upper left to lower right
    v. flip the square vertically
    vi. flip the square horizontally
c. decipher the ciphertext with the child key to find a new plaintext
d. calculate the new fitness of the new plaintext
e. if (the new fitness exceeds the best fitness) or
   ((the new fitness exceed the best fitness minus the margin) and
   (we roll a 1 on a 20-sided die))
    i. copy the new fitness to the best fitness
    ii. copy the child key into the parent key
    iii. set the counter to 0
f. increment the counter

5. output the parent key

Reading and references

Michael J. Cowan, "Breaking Short Playfair Ciphers with the Simulated Annealing Algorithm," *Cryptologia*, 32:1 (2008) 71-83, DOI: 10.1080/01611190701743658

Programming tasks

1. Implement the attack. If you used a different logarithm base, you will want to experiment with different values for the margin.

Exercises

1. Break this ciphertext. What is the keyword?

```
UDSDAEEPVFHPKNNMPILPBMNGDOOGHPGDVFHIVQRSURBETIREAFHPAV
KFREHRRMFANFPUDMRAAUPIAGPEXFTGRUODWRBNFNDOTGPWQGDMNLQV
WEUWHGLDFSAUNOQPUALZSDZDGUFABEZDRBDFDVVQRSGMBEIZTDFNOP
PLPUUVRBAUGTVEHRARFKDRBEODUDVEUCAWRBPRDSNEBXRSLTPWQGRA
FAKGLPUWHGLZBFREIEREZLRETZWGYNLPDUFPECPZZDMUGUUTICGUIA
RAODOQGDUGIZGHUALZMUBVZDDMZDRBGUFAEFBRDMARDFOPBRPRNLOM
SDSRXNOREBRADMGKANMHKIDZUMOAPLOAAFUNRZARSIGHUSMGZDRDEP
UWBEIFREOPLSFNBWAUMPTLMGNLRZARSIPIAUXGZDPR
```

2. This ciphertext is from the British National Cipher Challenge before it was national (2001). Figure out how it was disguised. Break it. Make sure your decryption is a clean one; otherwise you may have made a mistake with its disguise. Can you find the keyword?

```
NZTFM YKDID MYLCY NSGZK VXKMX ALZDP MYLCY NSGYK VXKMX
AGKOG LCYUR EGPNY TFMZK DMNGL HWCLL DKYAB IYYKV XKMXA
DYOSJ PCDHU GDKIK UVXKM XADYO SJPCD KUJAS DLBEU VTNZT
FMZKD ZIKMG JLCEU DAYNV XKMXA IYGKP EJGHL HMAII YYKVX
KMXAI ZQSNJ OLHKJ VIKVX KMXAL YNJYG PDHLI YMCBS YLKCE
GYOSX XANJD NLKDP LIQZD KHGJG XANJD NLKDP LIQZL YDSGK
DWDOE JANEB SDAVV GKDCT GIZLY LSDFG DOEJA NEBSD AVXAG
LXADS TFEKI MCDMX XAOSD SJELX KMPEH CQDKS GZKDQ SIZGI
VMDSJ EGXKC ZMGPG LKUYM JGKYK IMZFN XADPN YKIZK IMKZY
GELJN YMIDN JDMOK IKYOL HDPOL LEDPC LSDYJ EUAGK DQISO
QDMGJ ZGEGO SDGLH FGPMC SBPEJ GUGXE JKSII MXAKY KZCMG
FWKKN PHJSK DSMBS FKHMD GQYKD MGMZG JIKJT GIZGE BYNAG
KDSNB DEDGL AWKDJ NIGYG GIXAK OAGBQ XKGIM TGEKB ZKJTG
IYNGJ CHYGX RKYOL LEHQK IBGEW GJQOK THCBS XALKB SMZNH
PDJMG FWKKN SMGMB GSDYQ NZTFQ IZIBX NEAXJ QBJZN GWKDK
USMVX DPDET FXD
```

3. Break this ciphertext. What is the keyword (hint: it is a name)?

```
BIPAFKWLIFETHACPEGKWCKKPIUDQCWPMLKBIAMSFFSAFQDANHOBQTB
BOKOSCQBTBQCFWUHHACPATIRFYKWUBQOIRHOOKUPZPQVONLCPKWOMK
GRCRKQVIPUBICMHACPATNLHWHFPMHOIQQTQNKNWOIXSFRCCRKQPRHI
OQQTQNEHQOOBQBUQGICPKPAPBTBTQOWOBIKUEFKMRCCRKRQYQYUQKT
CBCPKPAFGYSLPKCMQEKPUHAUDAIFKQBIAQQMINQCBICUKUOKMROWGY
FSAFQDHORFPUPKKFAIXBCPGARUZCUHXBKIMKWOFHQTRABNWOARTAOB
OUINTGUCQYOKMRQBCPKPFEKLPKCMQEKPUHAUMIHACPTQAQNEQKPAKP
UKQHHOFXQALNQAPZPUHDKRUKOKVNBIKUAZBFFKFYKWYLAMQNVNHOFS
QDUCQYOKMRQY
```

Challenge

Not square.

```
G0IJVEFCDLZMDECBNAV4VIWGBRCEACKTVETFCKEWDBIZPMRFNETSZOVCJPL
KIEIPYCICKTDIZUHTNBUEFTGTCAFCNTIJTSGFCGATKLVFVE2FUY2FGAJTNZ
GZKTCIFVTFEXACKTJCUYUVNUTCRFRFKT2FJBENIVGLJEGDDLCTPOOBCACBV
GJCCKVIFCEAILFCUFNELFIEFCPL2FTQCENGIRFCUFNELFSFVNCEGNBGNAQC
TFGAFCCIKTEYSA2FCFBJFBGAFCFBETBFIVKLPHNGPLFATAG0NGPL2FTQCEF
RFBRKZFCKFVJZEXVIWCIVKLNGHRDTH3FYANGTF2HPECWNODBCFJVENGTCVJ
NZGONJTFANK0FBNEYFNBBCENRKFJKYFGAQGFNJCNNZOZ5JGTFSDLCIFVNBO
PVEXCLHEXTHHZZEVMTKGACEDBCAKTAYJCAVF2NSCRVCVILK5JK0FBNEVEJC
NVCICNNZOZ5JGTGFRJFSDLCIGFILJPLKIEFCPL2FTQCEJCUFVIFCEAPLNSC
GLIHLCQFREJNTLISTEBKZPMRFRCTNVUTFBKENTAHTCNNZOZE0CESTEQUFNE
CGCNEBVGFCWEFGOPIVKLCINAPHCINSCNEAIDMUVFTNEZJBTCNGIRJOCNECH
```

PXCIRPLCTOZ2FGZLPXCHQJ0VIICIJYFILJPLKIEFCPL2FTQCENGFPGBWGGN
OPBDIPVULHPGKTNAJPLKIEFCAVPJZEFSIVCKCECGIVGAFCNSACEGNBZJEBJ
QFCEXJBTCNGIRJOCNECECBCLNACIRILFCDTDBPDCEZFQCTCCILFE0CESTEA
FCZJEBFPKENAQCASIJHUECJGKTWEK0FBNEUYVEENGLCBZVEKGJOTJTERVCV
IFVKTECNBVIZUJOOBJHFTJRRKCICSRKCEGZKTCYTKGVPHCIBFTDGTJDNBDI
CGPOCGTJXEJZTBCNEBQGTSCNBCKQ

Unit 72
Vertical two-square cipher

The *vertical two-square cipher* is a digram substitution cipher that uses two Polybius squares, one above the other. The plaintext is padded to an even length and divided into digrams. The first letter in each pair is found in the upper square, while the second is found in the lower square. The ciphertext letters are taken according to the rules:

- If the plaintext letters are in the same column, then the ciphertext letters are the same as the plaintext letters.
- If the plaintext letters are in different columns, then we find the rectangle that has them at two of the corners. The ciphertext letters are at the other two corners. The first ciphertext letter of the digram is taken from the upper square, the second from the lower.

  An example couldn't hurt. Take this plaintext:

```
THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER
```

It has an even number of letters, so we do not need to pad it. Let's use the keywords `POLYBIUS` and `KEYWORD`, and fill the first square by rows and the second by columns.

```
P O L Y B
I U S A C
D E F G H
K M N Q R
T V W X Z

K R F M T
E D G N U
Y A H P V
W B I Q X
O C L S Z
```

The first plaintext digram is `TH`. It defines a rectangle and we find the ciphertext digram `WY`:

```
P O L Y B
```

```
I U S A C
D E F G H
K M N Q R
T V W X Z

K R F M T
E D G N U
Y A H P V
W B I Q X
O C L S Z
```

The last plaintext digram is ER. They appear in the same column, so the ciphertext digram is also ER.

```
P O L Y B
I U S A C
D E F G H
K M N Q R
T V W X Z

K R F M T
E D G N U
Y A H P V
W B I Q X
O C L S Z
```

The full ciphertext is

```
WYAOKDALSNDBASGDUTYPTEDWCKEVEMUESXLYER
```

Decipherment with the vertical two-square cipher is the same process as encipherment.

Reading and references

Félix Delastelle, *Traité élémentaire de cryptographie*, 1901.

Wikipedia, en.wikipedia.org/wiki/Two-square_cipher

Crypto Corner, crypto.interactive-maths.com/two-square-cipher.html

Warren Thomas McCready (“Machiavelli”), “The Twosquare Cipher,” *The Cryptogram*, Nov-Dec 1972, 152-153.

Programming tasks

1. Implement an encryptor. Remember that there are many ways to mix an alphabet with a keyword and many ways to lay a mixed alphabet into a Polybius square.

2. Implement a decryptor. Remember that there are many ways to mix an alphabet with a keyword and many ways to lay a mixed alphabet into a Polybius square.

3. Implement a dictionary attack. Remember that there are many ways to mix an alphabet with a keyword and many ways to lay a mixed alphabet into a Polybius square.

Exercises

1. Encipher this text with keywords `HUDSON` and `EXPLORE`. Use the same methods for mixing the alphabets and for laying them into the squares as in the example above.

   > I take for granted that you are tolerably well acquainted with the different modes of life and traveling peculiar to European nations. I also presume that you know something of the inhabitants of the East; and, it may be, a good deal of the Americans in general.
   >
   > (from *Hudson Bay* by R.M. Ballantyne)

2. Decipher this text with keywords MAPLE and LEAVES. Use the same methods for mixing the alphabets and for laying them into the squares as in the example above.

```
XCSOKGSOMYHBMQBWSOLYEWLYMXPRHZSTNZQCMZLGMBMPWILWQQBPVG
ICHHVPPRQKIQMAMHMZXBHZAYHUHDBWVDTIMBAYNYVDNTIYUHHTKEHP
PGCNSCEVXCSOSMKXPWIRACTTHEQCVDSNNZELDBIYPGYTKEITKGSOLZ
OWHTMFLZHELOITDIITWRACTOSCMZSOLGFLRDLYAXSOLYHFLYSOAZTB
TWIYDBMXMKIQBXRXVTNLMNKZSOPYPCHTTRMWQQIQTBVGTWKZTBVGIY
YBIXPZNLQZMZMBPPBINTICMBKGBISIDZVDMOKGDWNTHEKEDUVDZQ
```

3. Perform a dictionary attack on this ciphertext. The keywords are short.

```
DUDAMXTENUICFGMPRLUBGGAMFQQIMXOLLWQQOMESITRSSGLQLIHMES
DAXQOIRXTMBDNHKDOUMODUWCRLFGGSBPILBOSXECTERWECNLVPNSMK
QDOFESQQLWHMESBALMQQBOSZDALOMNDOXGSPEGREFGECGXCBAFVUMI
DNNCGGQQSPKDATDBXDSPGSOUHCNXASHTFQSINITSFEKIASLISPGGBH
FFBPDZGCDGTLQXHTFQMNQNREMTDHHDRXOMLSAZQBFGIXGGSPEGREAT
LPECQDHMFQMNQNMISPHFLQSPMTSQMEDGSIPKBWSXFRCPEGLNWEDAVG
YTESELVETGMBQIELBPSPSPATESMOSPHEQQMLGGLWAEHEBZQEFEWTCB
FFGGTYHEDUECLAOXEDDQWEWCUDESLNRLBWMIBOSXECQBNSMCEOPQPI
BPFOLNLGNNDBLVDEAEFEYSGFGEBPQQAFTHREECIXTLLPGUHKSYGCED
GGQQSPKDATDBXDSPBPTXHTDUQPHEOUMIEDBSQDTLGUDABVOSRMQDOF
ESDIRETMNUFDSPDBBALM
```

Unit 73
Horizontal two-square cipher

The *horizontal two-square cipher* is a digram substitution cipher that uses two Polybius squares, one on the left and one on the right. The plaintext is padded to an even length and divided into digrams. The first letter in each pair is found in the left square, while the second is found in the right square. The ciphertext letters are taken according to the rules:

- If the plaintext letters are in the same row, then the ciphertext letters are the same as the plaintext letters but in reverse order
- If the plaintext letters are in different columns, then we find the rectangle that has them at two of the corners. The ciphertext letters are at the other two corners. The first ciphertext letter of the digram is taken from the right square, the second from the left.

  An example couldn't hurt. Take this plaintext:

  THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER

It has an even number of letters, so we do not need to pad it. Let's use the keywords POLYBIUS and KEYWORD, and fill the first square by rows and the second by columns.

```
P O L Y B   K R F M T
I U S A C   E D G N U
D E F G H   Y A H P V
K M N Q R   W B I Q X
T V W X Z   O C L S Z
```

The first plaintext digram is TH. It defines a rectangle and we find the ciphertext digram LD:

```
P O L Y B   K R F M T
I U S A C   E D G N U
D E F G H   Y A H P V
K M N Q R   W B I Q X
T V W X Z   O C L S Z
```

Then IS → NT, ME → WU, and SS → NW. But then AG is on a single row, so it is enciphered to GA.

```
P O L Y B   K R F M T
I U S A C   E D G N U
D E F G H   Y A H P V
K M N Q R   W B I Q X
T V W X Z   O C L S Z
```

The full ciphertext is

```
LDNTWUNWGAYMNXPUDBMGOIYKUPAHAYDIGRFDAO
```

Decipherment is the same process as encipherment, with the two squares swapped.

Reading and references

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/TwoSquare.pdf

Wikipedia, en.wikipedia.org/wiki/Two-square_cipher

Crypto Corner, crypto.interactive-maths.com/two-square-cipher.html
(Note: They reverse the order of each digram, compared to our method of encipherment.)

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 219-220.

Programming tasks

1. Implement an encryptor. Remember that there are many ways to mix an alphabet with a keyword and many ways to lay a mixed alphabet into a Polybius square.

2. Implement a decryptor. Remember that there are many ways to mix an alphabet with a keyword and many ways to lay a mixed alphabet into a Polybius square.

3. Implement a dictionary attack. Remember that there are many ways to mix an alphabet with a keyword and many ways to lay a mixed alphabet into a Polybius square.

Exercises

1. Encipher this text with keywords `BACON` and `CIPHER`. Use the same methods for mixing the alphabets and for laying them into the squares as in the example above.

   > For months and months the eye has been assailed by paragraphs and pages in the literature of two worlds, contending for or against the existence in the Shakespeare plays of a cipher that would assign the honor of their authorship to Lord Bacon.

(from *The Little Cryptogram* by J. Gilfin Pyle)

2. Decipher this text with keywords POLYBIUS and SQUARE. Use the same methods for mixing the alphabets and for laying them into the squares as in the example above.

```
FLHSHXESUORPOHLSVUAMYBIVMFQONAXPQYTDQVQFGHCELDBSEGMOSK
KHKLLIOOSOFTNYSEBLPMQTUOLSBDORDBBBCPUKDNLSBDICWYCUCOPW
CBUNTEQVNTOVCBDONFISQPTDKHNAQNDNDOQYQVNIOXCBYUNTOZCBQF
EPKHSVUXUMTIUONFUWMOOVCBDBQXCUNVLIDBOVURFKUOUVFOXOOUTD
EBNFNHPMQXCMNAOPUMQFDBUKDNLIXMQXOVCBBLLIOMUVHFNABWSVBG
UVRKNIELKLLIEVLDDOFLDBMDQLUOBLODUNUXUMFPPMQTUMDNODKEKF
MULIIVBGCMBYPEKFNAQNOWUVNHPWOUIDKLOFABNMDBAMWRNIUXCOBB
OFPBNAGYOUUVNXNDALLSOULIXMIDIPOVNTCBENNTLFLISEFKQPSDFP
OFDCQXIVMXFLVLXMUMABLIOOBGCMMUQKALEURKNICMODKEKFMUNIUE
PHEPUWMOLIIVNAUPMHDRUVNIOGPWCBBQKHNXBDTDLICMNABUUMDNVR
LICMUORTBELDMUEVLIOMUVHFNAOOGDDBLIXMBXFPYCQXUPBQCPOTMV
URTTCBODKEKFMUEVQYQKLIUVMONANHASBONWMULIOELIUVURDNRNNS
MVSKPZQKVF
```

3. Perform a dictionary attack on this ciphertext.

```
NIUGOTANORETOTQLWAQNLATSKAASARQNTHSARPQVLTCIEMXUNELTLA
NTPMQDLGKTDEBBDEESSHINKGADXSSEINKGMMLPTSKAASKCSFKBNDAO
DRLTHRRFZUSLXSRGBKUOUDNELSFHGOIBWFRGWAKZNPNOKARFLEONDN
STSASUUPPDBKTZNPEBHRLATSEMAENEUOSEGDGOLZNPIHNTKAWBLNUG
AETTDNQNTHSAUGONFORFUDNEUGBGPDSSOTZUSLROOELROTQFTHECNT
ARRGODNTKAVHQLSEHDKGFORGDKSURGNTGOLZMMRFFHHVNICNLZLEUD
NETSKUMLROAEAMVNNGHUBREOMPOOHHRBEMDESEUGANUHLUEOPOOMLP
```

Unit 74
Hill-climbing attack on the two-square ciphers

A hill-climbing attack uses two mixed alphabets and the parent key/child key paradigm that we saw for the Playfair cipher. When it comes time to modify the child key, we randomly choose to swap two characters in the first or in the second alphabet. It is not necessary to flip squares or swap rows or columns. To avoid being trapped in a local maximum, a margin of 0.2 works well.

Here is the algorithm:

1. set the two parent squares however you like
2. set the best fitness as the fitness of the unmodified ciphertext
3. set the counter to 0
4. while the counter is less than 10,000
    a. copy the parent squares into two child squares
    b. randomly choose which child square to modify
    c. modify that child square by swapping two randomly chosen elements in it
    d. decipher the ciphertext with the child squares to get a new plaintext
    e. calculate the new fitness of the new plaintext
    f. if (the new fitness exceeds the best fitness) or
    ((the new fitness exceeds the best fitness minus 0.2) and
    (we roll a 1 on a 20-sided die))
        i. set the best fitness to be equal to the new fitness
        ii. copy the child squares into the parent squares
        iii. set the counter to 0
    g. increment the counter
5. output the parent squares

Once a key is found with a hill-climbing attack, we can recover the keywords by swapping rows and columns. For example, if we find this (half) key:

```
T P R Q O
E M S U A
N I L K H
G B F C D
```

```
Z W Y X V
```

then we can rearrange columns to make the last row more orderly.

```
O P Q R T
A M U S E
H I K L N
D B C F G
V W X Y Z
```

Now we can reorder the rows:

```
A M U S E
D B C F G
H I K L N
O P Q R T
V W X Y Z
```

It appears that one of the keywords is `AMUSED`.

Programming tasks

1. Increment the attack for the vertical two-square cipher. Remember that there are many ways to mix an alphabet with a keyword and to lay the alphabet into a square.

2. Increment the attack for the horizontal two-square cipher. Remember that there are many ways to mix an alphabet with a keyword and to lay the alphabet into a square.

Exercises

1. Break this ciphertext which was encrypted with a vertical two-square cipher. What are the keywords?

```
MEXESAEPQHNPLAOKMGBSUNUTXTDCALPTOUERPTGHNVGHNBOQISPQFT
REFOIHREKLFAIHREGRDHQFOQISQXMEIBHUMQLHNAUNNACEIHMTIHRE
WEREALPQRDRESADCRDREPQNFMNHSDKKLEQNWHIBTPTFOUNRHCAREHS
CPDIFOSLYTHISLKLNXCQBSWHRENXMEHHDHPTGHEPTYFHOKDMIZGHKL
NXMEHBAERMGIVEICIVCQALREADWSTRPMGCONMESATRTNGCHQCWMGWT
LLIZKLGADBNEMEIHCEIHKHRDRRGKORPTEHDHPTGHHMDCDBPTEISNVT
DHPFTRPTEBTRQTCCHLFOPTOUERPMSTUTMCIAUNBSMFORSTAIDHDCLT
NAESCEIHAMTNTOMGUTNHSADCHQINDBHQGCOKDNXTDCREGRFHGSLLXT
QYLHPTADRHGIKLHNTXAHBBHLGCHLTTIAKHHUBGTRXEULIZADCWKLFQ
PSFWBTKLNELDFOGHQTNAWHECYTIHGAKDONDMIESX
```

2. Break this ciphertext which was encrypted with a horizontal two-square cipher. Can you untangle the keywords?

QMHCPVUNGIPBYNTNDKCIXGCITHMBKNVLQMEBGITRBPKNTSSFGPAPIS
HPTSMFQMFNMVFNHCMDSNICHCOPFHMOESFCPETIRRHCQINGBATIESWH
SXNITQCNLPNAIMPBGDBPBOEDQPDGEWTIPPCRADLQPPWRKPPIDDCTOC
WHMBKNQMDHRPUAYPHPGDPSILWHBWESSNDHEDMEPBLIEHRIPSBHMDQA
QAEBICDFILGHCTOYDISOWNECESQMDHWYILWHADIHHPDGEYYSGNMDQA
DFGNSHCOOCPSBDEQPBGAXCBIMBESASUCTMDQTSQSPIBDEYZSQSMHID
QPSXIDCSMVFPEIEXEAESRBPEZEIHTICQVPAPCRKNFUFPTYRAPEHBPV
MCUPEHAPTSQMYNWYMFPIQEHPMBSTIADIAHPTCVRBBNADAPUSCIZEIH
AHEBLFDHKUHPSRXCOIPKQAHPWHMBQIDKFPFUUFZPEDUPAHQIDGTIER
MBSPETOYAKIVKCGPMNDAHCGUQMDHDFDYEHLCRMCIKNOCSNMIYPPIBD
ELVPHHQMFPAKIVKCGPPCWHUSRHPVQPUCQMDHQAQMFEEIDLQMYNDEHP
PSNMBOERURPEWNQMFNSOQIDRUCNAQMIHWHTSQMDHAHESWHPSMPCIEH
DIEHODHIADMFCSEISPHU

Unit 75
Four-square cipher

The *four-square cipher* uses four Polybius squares, arranged in a two-by-two layout. The upper left and lower right are the plaintext squares; their alphabets are not mixed. The other two are the ciphertext squares and contain mixed alphabets. The plaintext is padded to an even length and divided into digrams. The first letter of a plaintext digram is located in the upper left square, the second in the lower right. They form the corners of a rectangle. The other two corners hold the ciphertext letters, the first in the upper right square and second in the lower left.

Let’s encipher this message:

```
this message is encrypted with a grid cipher
```

Its length is even, so no padding is needed. We will use the keywords `POLYBIUS` and `KEYWORD`, and fill the ciphertext squares in an unimaginative manner:

```
a b c d e   P O L Y B
f g h i k   I U S A C
l m n o p   D E F G H
q r s t u   K M N Q R
v w x y z   T V W X Z

K E Y W O   a b c d e
R D A B C   f g h i k
F G H I L   l m n o p
M N P Q S   q r s t u
T U V X Z   v w x y z
```

The first digram `th` is enciphered to `NB`.

```
a b c d e   P O L Y B
f g h i k   I U S A C
l m n o p   D E F G H
q r s t u   K M N Q R
v w x y z   T V W X Z
```

```
K E Y W O   a b c d e
R D A B C   f g h i k
F G H I L   l m n o p
M N P Q S   q r s t u
T U V X Z   v w x y z
```

This continues, and the final ciphertext is

```
NBSQHENPOROZLMLLOPZIRWOXAQIYUNAWYAFCOS
```

Reading and references

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Foursquare.pdf

William Maxwell Bowers, *Digraphic substitution: the Playfair cipher, the four square cipher*, American Cryptogram Association, 1959, page 25.

Wikipedia, en.wikipedia.org/wiki/Four-square_cipher

Practical Cryptography, practicalcryptography.com/ciphers/four-square-cipher

Crypto Corner, crypto.interactive-maths.com/four-square-cipher.html

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 221-222.

Programming tasks

1. Implement an encryptor. Remember that there are many ways to mix an alphabet with a keyword and to lay an alphabet into a square.
2. Implement a decryptor. Remember that there are many ways to mix an alphabet with a keyword and to lay an alphabet into a square.
3. Implement a dictionary attack. Remember that there are many ways to mix an alphabet with a keyword and to lay an alphabet into a square.
4. Implement a hill-climbing attack. Except for the decryption routine and margin, it can be the same as for the two-square ciphers. The margin should be about 0.2 when the fitness is lower than a threshold, but near zero above the threshold. Experiment with it until you find a threshold that works well for you.

Exercises

1. Encipher this text with keywords `WINDY` and `WEATHER`. Use the same methods for mixing the alphabets and for laying them into the squares as in the example above.

   > We all held the string as fast as we could, and tried to pull down the Kite; but it was impossible, for instead of bringing her down, we were all three dragged along down the meadow slope.
   >
   > (from *Adventure of a Kite* by Harriet Myrtle)

2. Decipher this text with keywords `NUMERAL` and `CIPHER`. Use the same methods for mixing the alphabets and for laying them into the squares as in the example above.

   ```
   CTBTLONUTEISIVRHSBOINKEUHTMNMMPNUUNGSBISDPEKEYFDMCDMDP
   PKMUOPDITPZCOIHCDMDPQQXTTHHQGUMUUSLHEUAPOHFGKSKIOIHOBM
   QDUEIHYMACKIQSURUNQUPHMUKEFGEEKTOHIPXAAMEUDPTIBTILEMPE
   PHSUQDEOQDBTBMTSTIPBTIEOUUMCRSOIEXCTNGSLLQNEOMYDHPTIMO
   MUQDNNMNCQCLBOQELLMUOPDIQX
   ```

3. Perform a dictionary attack on this ciphertext.

   ```
   EQVOMWHHQSPEHMLWCGEINSBEPEILCQFDCDXCSBAIROTWTBPPPLQCRW
   XCGLFHQPFTYTCDIPLARQROMWROQPOKBRMITABAFWSKGGUKHMGLHMNS
   IZLAPNFHROPEGGHMFQAWQNUKGSPEGSILLHGGTWBEROFLRWPEGSLAVO
   MIAELAPDTAQKAMFWMLHMSKUCBRPPIRROTZGFHRTOGLGZKAHPHATNQE
   FWLFOBDWHLARZIGAMBQEDKEQOWGKFQHIKOQIPEIABBNKGLZOROCBFT
   AOHQKQRWPEHFQHIMQPAWZQRWHQAWCVLLDTCYLZISSKQIKNGDPEHMPE
   LLGGTBROFGCDTGKAIMHAFTAAIRKMLLSKHLFPAWEIQCYLFPAWCWGAIG
   HSROKNFGPSGGHEFWSQQEDWMBFWETBWBGVYGGSQAWHEFPAWFCQPQIUG
   CQFDCD
   ```

4. Perform a hill-climbing attack on this ciphertext. What are the keywords?

   ```
   DSVNOOSGDPFOHMLETLEOLWSFFLMHSTCIAKNSMRSVACAILZRGGKUNAM
   THTNGKCIENSFRMGSEOSAMHNOISACMSFOTFHKUORALTLZAPLCACLUCS
   LMACUEMHVFGLFNLKNKKILZDHTOYFSNACLWTNGPBXAPAMNSCDSIFOPC
   PZLTYINOXDKAGKUNAMCIFKGRGLYDRARAUMATSANKKOPLMOAPTRAGNO
   XQYFOMLAMSBRNSCDRAPMACPWMOPZLTYINOYFOMGSPCELGFSTDSHNMX
   YFOMLWPMFWZKHNMXYFOMGWNSGQUQQIACWYNSMKFNSIMRFOBSVNVWRR
   MIHOTHKSSGPFGTKSULDTFOYDTOHNMRFOFONCLTFSSRMEFLRACSRACS
   MVRAONMRBRNSCDRAPMACGSEOOAONBQACQIYSDHBEMSLCGPFOUQGWEO
   RALZCSLMACHARAPBLCEIYICPUEVWFOMMRRATTOASMSFOUNMEPZLTRR
   GEGSFODGCDATTWHHBADRAMNNMRFOTNNNINLTGLDGTHTNYENSOHEYAT
   FOEOLSPLAGGLRAAMBSVKHOACAIPZLTTNTSRFGLCISIRAALXWNKVNRH
   PZLTLEGEKRQLNNQDGLCIRALMQTDZSDZKSFFCDXSGPOHEISKQHNMXYF
   MOLTFPHKMORMUNMEPZLKNKRACICQZNTNTNMMSQGLHADGKDABGKNLOD
   ISNNLIBSAKNKRGGKHYLTGHNNRAGLPZEIEOASMMFLMRMSFODFASUQRA
   CSAMLMFWOOMOGAQOPMFWZKRQOODWRACSDFOWHNVNOOVKCITOUSUORD
   LZNSMKTHAMRACSPWXSCINPZRSVROISNOGSFOHAFKSDIGQXAPYDRANS
   OIPLRFCIENMOCMLZNODSFOTNZORNCSATMHRQPMCKAPSAENSAKI
   ```

Unit 76
Phillips cipher

In the *Phillips cipher*, a Polybius square is filled, and from it, seven more squares are generated. Each square is used to encipher five letters at a time. The second, third, fourth, and fifth squares are generated from the first by shifting the top row downward one, two, three, or four rows. The sixth, seventh, and eighth squares are generated by shifting the top row of the fifth square downward one, two, or three rows. A square enciphers a letter by replacing it with the letter that is in the next column and next row, with wrap-around.

An example is essential when the explanation is as poor as the one above. Let's begin by filling a Polybius square with the keyword `POLYBIUS` and generating the remaining seven squares.

0	1	2	3
P O L Y B	I U S A C	I U S A C	I U S A C
I U S A C	P O L Y B	D E F G H	D E F G H
D E F G H	D E F G H	P O L Y B	K M N Q R
K M N Q R	K M N Q R	K M N Q R	P O L Y B
T V W X Z	T V W X Z	T V W X Z	T V W X Z

4	5	6	7
I U S A C	D E F G H	D E F G H	D E F G H
D E F G H	I U S A C	K M N Q R	K M N Q R
K M N Q R	K M N Q R	I U S A C	T V W X Z
T V W X Z	T V W X Z	T V W X Z	I U S A C
P O L Y B	P O L Y B	P O L Y B	P O L Y B

Notice that squares 0 and 4 have the same effect, as do squares 1 and 7. Now let's encipher a short message. The first letter enciphered with each square is highlighted with pink in the square and below. Notice the wrap-around used in enciphering 'T' to 'O' in square 0.

plaintext:	THIS MESSAGE WAS ENCRYPTED WITH A PHILLIPS CIPHER
square:	0000 0111112 222 233333444 4455 5 55666667 777700
ciphertext:	OKEG WNYYBRL AHG LYDPZVONM YEOI R EIVGGVEY POEKNT

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XIX.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Phillips.pdf

Programming tasks

1. Implement an encryptor for the Phillips cipher. Allow for several choices on how to mix the alphabet and how to lay the mixed alphabet into the square.
2. Implement a decryptor for the Phillips cipher. Allow for several choices on how to mix the alphabet and how to lay the mixed alphabet into the square.
3. Implement a dictionary attack.

Exercises

1. Encipher this text with the keyword `EDGAR`. Use the same method for mixing the alphabet and for laying it into the square as in the example above.

   > Now the irony is this. In this walk, so many times repeated, the world's greatest master of the terrible and the bizarre was obliged to pass a particular house on the eastern side of the street; a dingy, antiquated structure perched on the abruptly rising side hill, with a great unkempt yard dating from a time when the region was partly open country.
   >
   > (from *The Shunned House* by Howard Phillips Lovecraft)

2. Decipher this text with the keyword `RHYMES`. Use the same method for mixing the alphabet and for laying it into the square as in the example above.

   ```
   UBSSOOWGUHWHTWFUVHVYDIVLSPGFOWPGFWSNKHAXSAQWAUWUIFKNGU
   HQLRQYVGWOIAWGLFGWPFSIYUWNFMQYSHUYNFGIONVYRGYVSIFQNKUW
   SFGWSFUYHUIFCMSASQVUQLNGXIGRCUEASGUQWVMIFUIFGWAKWVFQVU
   BSGFRYKHGIBNIBUVXPAIYSIVUCZHWGKZBANRGFIFUMWEIYXUNOGWHR
   WUZYAUSIAUIHGFMQSIBAUYKYMV
   ```

3. Use your dictionary attack to break this ciphertext.

   ```
   RAPSZUHLHMRIXQGAPPHIZURAQKOHXLGHLYGOLKAUPSFALURHMSZUXN
   KAESXZZBRAYHUBHIKCLYDCDOZUDKALSBSMHMNTUSZVBVZIXAIZARIB
   UKXHFCAVPSHMIFLZAXBSIHLKHSTVUIZAGIBUCXHXUBSFGHWLUGKLDV
   DPHKEIKAXZQKAEIZVWAZZIXQKLHYSTHYRHUHPHVRDFAZHXBSRQNUDI
   MSZERAVCKYDIZSHFYSLUHXHZUZ
   ```

Unit 77
Hill-climbing attack on the Phillips cipher

Our hill-climbing attack on the Phillips cipher is similar to the one for the monoalphabetic substitution in Unit 28. The parent and child keys will be mixed alphabets, not sets of filled squares. To avoid getting stuck in a local maximum, we will use a margin, as we have done before. A good value for the margin is 0.5. Here is the algorithm:

1. calculate the best fitness as the fitness of the unmodified ciphertext
2. set the parent key to the alphabet without 'J'
3. set counter to 0
4. while counter is less than 10,000
    a. copy the parent key to the child key
    b. randomly swap two letters of the child key
    c. generate the eight squares from the child key
    d. decrypt a plaintext using the eight squares
    e. calculate the new fitness of the plaintext
    f. if (the new fitness exceeds the best fitness) or
       ((the new fitness exceeds the best fitness minus the margin) and
       (we roll a 1 on a 20-sided die))
        i. copy the new fitness to the best fitness
        ii. copy the child key into the parent key
        iii. set the counter to 0
    g. increment the counter
5. output the parent key

Once we have a key, we may not have *the* key. The reason for this is that if we roll the entire square to the left or right, then the cipher has the same effect. For the example in the previous unit, these versions of the first square (square 0) are equivalent. No matter which of these you choose, when you generate the remaining seven squares and encipher a text, the ciphertext will be the same.

```
P O L Y B    O L Y B P    L Y B P O    Y B P O L    B P O L Y
I U S A C    U S A C I    S A C I U    A C I U S    C I U S A
D E F G H    E F G H D    F G H D E    G H D E F    H D E F G
K M N Q R    M N Q R K    N Q R K M    Q R K M N    R K M N Q
```

```
T V W X Z    V W X Z T    W X Z T V    X Z T V W    Z T V W X
```

If we find, for example, that the key is `LYBPOSACIUFGHDENQRKMWXZTV`, then we can lay it into a square and roll it until it appears to be in order, and thereby recover the keyword.

Programming tasks

1. Implement the attack. Use tetragram fitness.

Exercises

1. Break this ciphertext. What is the keyword?

```
DWCFPHQAZAXMZZLZPLYARDZDMEHHDUUGLFGZPSKMFDWLHOIKIHRFOM
TLCQXCDZPCSLBREHQQZKDLNMAQLFEABIGVZHSMTNMWXBSASBZCWUKU
DGHSOZFHQAFQDHFLOAKRATQSLZLLZTZKMLQLFKAAIVBDFHULICCGZF
CBOAKFCATTILKVTALQLQDHIUULRIKUSLPZIQGKHCFQSZUGLZMZGMSO
WQVWQSZQLGDHCCGWPSOALCGFDFHMLTHQSWCLMAMZHZLYOCVZZICUDU
VHGDLHAWGHQCWLBICGHNRMGPLYWQLQSELHWULGLPHIHGUANGKACGBQ
LTGKWCRNTLRLWYVBLMLHGGKGRGAHIGDZHKGMSDBCOSZSHQNZKQVASZ
AVFHQDOUGBIIWKZFH
```

2. Break this ciphertext. What is the keyword?

```
BVGHUUWZKGDQLDDFZUFYVMUIVLNZKUXALMRWGHKWGHVABNNSNGHERZ
FYFOBFRBCBILBPNVMHPMUZIKSZFVMULFZZWOUWETFPXIFZAMZECGHM
MVFMRXEIFVHKFABURURHVQRQIPNOZYZUVMURHZPFVHKUGHMRURHVQC
XIKNHODARQVFRYVXNNZFGHBVIVLFAUHPBADNHOKICBVWKKZWGHQVRA
IZUIFMZEFZQVLCXEVFAEGHNNVMUPNFAZHKDQINZZFOHIERHZGFVOLF
AFMPNFAZHKRNHAHVLHAKUQLFPXNNIUUHUUHHVUCNHEUALKHPNSAHVE
BAKUHOFTNVNWHWWHBVURFHZNVLKMRUBWFXNWABNNKWAVSDWFUNGUBF
RHVEMZPFFUUCUGHKUISVURFZFPMZWPZSGFRDQYZFPXVBUBMVKUFZUQ
INXUWRHKCXFFLRSVLRQRRXFFLRWUZFQINQUNCHKXZMHFVKRWBCVFSW
ZZWOOWFRRXIRXYLRMBLNFAGFELK
```

Unit 78 (optional)
Phillips-RC cipher

The *Phillips-RC cipher* is a modification of the Phillips cipher in which rows and columns are shifted when generating the eight Polybius squares. The encipherment proceeds as in the Phillips.

Here is an example starting with the same first square as our example for the Phillips cipher:

```
    0           1           2           3
P O L Y B   U I S A C   U S I A C   U S A I C
I U S A C   O P L Y B   E F D G H   E F G D H
D E F G H   E D F G H   O L P Y B   M N Q K R
K M N Q R   M K N Q R   M N K Q R   O L Y P B
T V W X Z   V T W X Z   V W T X Z   V W X T Z

    4           5           6           7
U S A C I   F E G H D   F G E H D   F G H E D
E F G H D   S U A C I   N Q M R K   N Q R M K
M N Q R K   N M Q R K   S A U C I   W X Z V T
V W X Z T   W V X Z T   W X V Z T   S A C U I
O L Y B P   L O Y B P   L Y O B P   L Y B O P
```

If we can break a Phillips-RC ciphertext, then the keyword may not be obvious. We can roll the square until it becomes apparent, as explained in the last unit, but for the Phillips-RC we must roll horizontally and vertically.

Reading and references

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/PhillipsRC.pdf

Programming tasks

1. Implement an encryptor for the Phillips-RC cipher. Allow for several choices on how to mix the alphabet and how to lay the mixed alphabet into the square. Feel free to copy and modify your implementation of the Phillips cipher.

2. Implement a decryptor for the Phillips cipher. Allow for several choices on how to mix the alphabet and how to lay the mixed alphabet into the square. Feel free to copy and modify your implementation of the Phillips cipher.

3. Implement a dictionary attack.

Exercises

1. Encipher this text with the keyword `STORY`. Use the same method for mixing the alphabet and for laying it into the square as in the example above.

   > Once upon a time there lived a cat of marvellous beauty, with a skin as soft and shining as silk, and wise green eyes, that could see even in the dark. His name was Gon, and he belonged to a music teacher, who was so fond and proud of him that he would not have parted with him for anything in the world.
   >
   > (from *Japanische Marchen und Sagen* by David Brauns)

2. Decipher this text with the keyword `FRIENDS`. Use the same method for mixing the alphabet and for laying it into the square as in the example above.

   ```
   ZQCUBXHDEFKYZOHTOVMLZZQCBQMPXKSUHOAWFMNMBDPLMUMZXOUNXI
   OPFQLFQCUBSZMQVYULZCYDEHIUCZQBOQXFXPUZORULZFVXZQCUBXGE
   PTVYDQUZLHEZALXLDIZOMOWPZUQKAQNMCKBACQZXOPMMRULPEULACZ
   WZQMTLHQZOULIYPFNLZACCIBHCMZUHRNHHZLHFCLAGQXKEDEEEYPZN
   HLXXNQBOWGHVTZSUHGSOMULZZUCOWWFMOEXXUPUMFNILKKMNLGQTMM
   LGQSUMFEIVUBPQZLHDOLHIZIUOYNWALKICDZVYZXULOYGIFXXHPDLI
   CIEDECZIUQYIVKFQLZMWVGQZBMTLGQAKZOMNCLKZK
   ```

3. Use your dictionary attack to break this ciphertext.

   ```
   NGQFUQBVQLVAVWKHKTXAEPGFZETVMHMRUOVYQZZVSPZQOQMQZZRPOA
   MAQDKWUSBUQAXVBZGNSRYRZVUBBCNUIQPRLWPRMFETEXDWQUSQIMVB
   MUNSRGOTRVFTQZPRUELAZVUAMHRWPNTNQZGQVUUGBFOXUGVEXRVSQC
   BFEZIEXRSRZRZUVOFYIBZBFINCEUGVPNNEWRZKAURPYXQHRYQVSRZY
   VUTXFQBYUGYLMORNSRIFMMEWQZNFUSRFTTKMZKBXYYTNGQYQZMCBNR
   RVPLMAZRRPPYOQFVBYFRZDKVVSTYVRUNFHPCUGBHSPMRMELAHEZCND
   QVSRMFLVQMHQRQOTQFPYTNLZHMIRNPHEZSVSQTBCCEHED
   ```

Unit 79
Double Playfair cipher

For the *double Playfair cipher*, two Polybius squares are used. Each is filled with a mixed alphabet from its own keyword. The two squares are set next to each other. In addition, the key for the cipher includes a period, which is simply an integer. The plaintext is first padded with an ‘X’ if its length is odd, then divided into blocks that are twice the period in length. The last block is not padded to twice the period. Each block is written in two equal-length rows; this is called *seriation*. Each pair of letters formed by taking one from the top row and the one below it from the bottom row is enciphered together. The method of encipherment is to identify the top letter in the first square and the bottom letter in the second square. Then,

- if the two letters are in the same row, encipher them to the letter just left of the second plaintext letter in the second square (with wrap-around) and the letter just to the left of the first plaintext letter in the first square (with wrap-around)
- if the two letters are in different rows, form a rectangle with them at two corners; the ciphertext letters are the letters at the other two corners, with the one from the second square first

The resulting pair is enciphered *again* with the same rules.

Time for an example. Take this plaintext:

```
THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER
```

It has an even number of letters, so we do not need to pad it. Let’s use the keywords `POLYBIUS` and `KEYWORD`, and fill the first square by rows and the second by columns.

```
P O L Y B   K R F M T
I U S A C   E D G N U
D E F G H   Y A H P V
K M N Q R   W B I Q X
T V W X Z   O C L S Z
```

Suppose our period is seven. Divide the plaintext into blocks and write them in rows:

```
THISMES  ENCRYPT  GRIDC
```

```
SAGEWAS  EDWITHA  IPHER
```

The first pair is TS. They are on the same row, so we take the letters to their left, with wrap-around, and get LZ:

```
P O L Y B   K R F M T
I U S A C   E D G N U
D E F G H   Y A H P V
K M N Q R   W B I Q X
T V W X Z   O C L S Z
```

Repeat with LZ to get TW:

```
P O L Y B   K R F M T
I U S A C   E D G N U
D E F G H   Y A H P V
K M N Q R   W B I Q X
T V W X Z   O C L S Z
```

This continues until we have the completed ciphertext:

```
TWFAATNIOYRAXMTAMZAOMRIVASEAPRIGAAFQAK
```

Reading and references

NOVA Online, “Decoding Nazi Secrets: The Double Playfair Cipher,” www.pbs.org/wgbh/nova/decoding/doubplayfair.html

Noel Currer-Briggs, “Some of ultra's poor relations in Algeria, Tunisia, Sicily and Italy,” *Intelligence and National Security* 2:2 (1987) 274-290, DOI: 10.1080/02684528708431890

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.
3. Implement a dictionary attack. Remember that there are many ways to mix an alphabet with a keyword and many ways to fill a Polybius square. You will have to input the period or try several periods in your attack.

Exercises

1. Encipher this text with keywords DOMESTIC and FOREIGN and period 5. Mix the alphabets by adding letters after the keyword from the beginning of the standard alphabet. Lay the mixed alphabets into the squares by rows.

> Not being, at this moment, in the pay of any press, whether foreign or domestic, I will not, at this my third landing in English country, be in haste to accomplish the correspondent's office of extroversion, and to expose all the inner processes of thought and of nature to the gaze of an imaginary public, often, alas! a delusory one, and difficult to be met with.
>
> (from *From the Oak to the Olive* by Julia Ward Howe)

2. Decipher this ciphertext with keywords `GRIDIRON` and `FOOTBALL` and period 6. Mix the alphabets by adding letters after the keyword from the beginning of the standard alphabet. Lay the mixed alphabets into the squares by rows.

```
DTQFQAKMGIMEAEQHRVZDATAANIFOHUTMFIBXTTRFQMFIFIHDGURWTI
PSSBIKTEDFLAKVANUSOHIMQBVASSACONLDVEALAEHNGIGUELPEESGP
CBFNFIRFSIKVITNGQMDXQBXISIAHIASHAIGKLBECCQAFMMGUTTFDNK
EIDBSIHUCDNCOMKKGIMMGXTTTBRCBCRBDPBDSLZZLDQEDPBMSMRAFF
ELMESSBDOCEEREPELBIIPGQBTFTTKBTBXSKKGUQHFNUDQRDYLU
```

3. Perform a dictionary attack on this ciphertext from the 2004 British National Cipher Challenge. The keywords are taken from this list:

```
ANSCHLUSS
BLITZKRIEG
DEUTSCHLAND
DIRSCHAU
FATHERLAND
FEUERZAUBER
LEBENSRAUM
NORDWEST
RHEINTOCHTER
SONDERAKTION
WASSERFALL
```

```
ZONOP UXRFO VMNUS VERUZ XPPLS VOHMZ XGZBK TTQWL LFWAC
FTKTA HULIP LYBUP DUURL FXHXW TOSTZ IBODK WYLFQ FWYNF
EDZVQ RBOME SFHGT AHUUV QBIZR GFZNE WXWMV FCXMF WBLST
DISQA NGTPM CHISA CLVWX IKLFM OZCKW XHRNW MELKB GSNSA
MECOL KWEYP TPZDI DWKCW VFWOI ZSCID GLMTT PNUIS TVMII
SEKMI WLZBT CXXLF ZADTT BFQAE UGWMM XRWME VLVVF ZTDNP
ICPIZ LLICO GIHDN UBIOI OHNGZ WLGWU QFMBT PEWBO CDZPU
TZBKS PXFFT GVYUG ZUVEV LCAAQ FMPSO OMBVE TLZEW ISAQL
CPKIH ZVDSU TLVEL FCQUV VIMFS WWYOZ ICTSZ MSVZN HBNOX
SQFTD LFMCQ AMMLI MXLLF ZCICO YGEFU TCABO WRAQQ IYXLI
PUHIS ACLWM UVDGH PZISR QIWQT AUFQF SLOWV XWTWQ VMNOA
HCFME ZKFRC WFAMF QWFQM ZUFUU TPMHA QHFFF CNGGS UKDWL
EIIIQ ITKQI KDIMB OVUXP FMSLC PYXZM UMLIS W
```

madness's book on classical cryptography
unit 80: nihilist substitution cipher
last modified 2020-10-20

Unit 80
Nihilist substitution cipher

The *Nihilist substitution cipher* begins with an alphabet mixed by a keyword and laid into a Polybius square. The row and column labels are 1, 2, 3, 4, 5. The letters of the plaintext are converted to two-digit numbers by taking the row label followed by the column label. A second keyword is used in a manner similar to the Vigenère cipher. Its letters are also converted to numbers with the same Polybius square. Those new numbers are added to the plaintext numbers. Optionally, any sum that exceeds 100 is written without the leading 1; this does not lead to any ambiguities.

Your are probably expecting an example at this point. Let's begin with the keywords `POLYBIUS` and `KEYWORD`. If we fill the square in the least imaginative way, we have:

	1	2	3	4	5
1	P	O	L	Y	B
2	I	U	S	A	C
3	D	E	F	G	H
4	K	M	N	Q	R
5	T	V	W	X	Z

Our usual plaintext for this part of the book:

```
THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER
```

And here are the gory details (at least some of them):

plaintext:	T H I S M E S S A G E W A S ...
plaintext numbers:	51 35 21 23 42 32 23 23 24 34 32 53 24 23 ...
keyword:	K E Y W O R D K E Y W O R D ...
keyword numbers:	41 32 14 53 12 45 31 41 32 14 53 12 45 31 ...
ciphertext:	92 67 35 76 54 77 54 64 56 48 85 65 69 54 ...

The full ciphertext:

```
92 67 35 76 54 77 54 64 56 48 85 65 69 54 73 75 39 98 26
56 82 73 63 67 74 63 80 55 75 77 35 84 37 66 42 76 64 59
```

Reading and references

Wikipedia, en.wikipedia.org/wiki/Nihilist_cipher

American Cryptogram Association,
www.cryptogram.org/downloads/aca.info/ciphers/NihilistSubstitution.pdf

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 619-621.

Programming tasks

1. Implement an encryptor. Remember that there are many ways to mix an alphabet and to lay it into a square.

2. Implement a decryptor. Remember that there are many ways to mix an alphabet and to lay it into a square.

3. Implement a dictionary attack.

4. Modify the two-stage attack on the quagmire 1 cipher to make an attack on the Nihilist substitution cipher.

Exercises

1. Encipher this text with keywords `RUSSIAN` (in the square) and `FREEDOM`. Use the least imaginative way of setting up the Polybius square.

   > O God, how easy it is for a king to kill his people by thousands, but we cannot rid ourselves of one crowned man in Europe! What is there of awful majesty in these men which makes the hand unsteady, the dagger treacherous, the pistol-hot harmless? Are they not men of like passions with ourselves, vulnerable to the same diseases, of flesh and blood not different from our own?
   >
   > (from *Vera, or The Nihilists* by Oscar Wilde)

2. Decipher this text with keywords `ANARCHY` (in the square) and `NIHILISM`. Use the least imaginative way of setting up the Polybius square.

```
44 77 59 47 45 66 78 57 36 53 56 83 47 76 89 76 44 83
38 63 58 67 65 79 53 44 26 76 66 47 55 87 36 76 60 43
79 56 67 80 53 53 56 83 45 77 67 67 37 57 39 45 58 44
89 80 44 67 39 76 66 85 55 79 34 56 60 45 45 53 68 58
34 53 50 43 78 77 85 88 44 86 68 64 79 47 97 50 53 67
47 85 45 76 68 47 43 43 46 56 57 85 76 80 27 73 60 47
```

```
58 45 88 67 24 43 57 66 75 77 55 66 23 64 27 76 79 44
76 49 27 73 49 43 78 46 99 46 25 73 40 45 85 76 88 67
23 64 68 43 78 85 85 48 57 47 26 67 66 66 78 67 53 44
56 57 47 83 66 69 36 76 26 44 57 77 59 59 65 47 56 66
58 73 69 67 57 64 57 66 58 55 75 59 35 77 56 77 49 56
58 46 63 76 39 73 59 47 95 70 23 44 49 64 56 56 57 80
33 64 27 45 85 76 88 67 23 67 26 76 79 73 97 67 66 65
27 56 87 77 59 67 56 43 27 55 49 56 55 69 56 73 48 44
58 85 89 50 23 77 56 77 49 56 57 70 36 67 37 56 47 76
85 60 57 47 39 76 75 46 76 59 57 53 30 43 57 66 55 48
43 56 59 83 69 76 89 50 63 76 57 66 58 55 75 59 35 43
40 77 58 45 55 88 27 64 49 56 66 47 55 69 37 76 66 76
76 56 58 80 36 55 30 64 69 43 56 58 56 73 59 56 48 45
68 80 36 55 50 53 65 73 78 58 44 44 26 74 68 43 58 59
45 44 56 85 46 73 56 69 33 77 56 67 55 76 68 69 37
```

3. Break this ciphertext with a dictionary attack. Both keywords end in `-IST`.

```
46 86 52 67 74 45 74 42 36 65 45 66 36 45 57 35 103 54
56 55 68 73 52 64 48 38 106 52 64 35 74 85 55 74 44 46
86 52 64 56 38 73 43 56 64 54 94 42 64 47 74 74 42 64
54 45 74 42 64 46 57 83 34 37 74 47 66 63 47 45 35 64
63 47 35 65 64 44 36 45 37 97 72 37 54 44 94 32 43 46
54 75 55 53 64 46 64 44 56 54 44 97 55 34 74 45 83 43
36 65 48 75 55 53 68 54 84 33 67 54 46 86 52 53 65 56
67 42 44 57 54 74 64 53 36 44 66 36 37 77 46 86 52 35
38 37 74 43 43 37 64 74 64 53 37 58 73 63 55 35 55 66
66 53 37 38 93 33 44 46 55 64 66 35 54 48 75 33 36 45
45 64 34 43 37 46 83 52 64 46 44 97 52 37 77 75 73 44
56 54 37 65 55 34 46 57 83 66 36 65 48 84 33 67 64 37
74 33 67 46 35 84 34 34 38 35 94 75 66 74 44 75 52 36
66 37 97 44 54 68 35 93 63 36 46 44 63 52 44 35 36 73
52 45 77 44 04 35 44 55 35 97 44 73 65 37 75 52 53 65
35 03 54 56 46 35 93 35 57 54 45 64 62 53 37 36 96 72
36 44 65 75 35 64 36 54 74 35 63 35 65 85 44 56 54 64
74 33 34 65 37 84 44 53 68 44 04 52 64 46 35 03 44 36
65 48 06 33 73 68 64 64 44 56 54 68 66 63 47 44 75 83
66 53 64 74 65 55 63 35 68 83 42 64 68 74 74 43 43 37
65 74 33 35 44 54 75 75 45 57 37 94 42 44 74 45 03 35
37 75 44 75 55 34 74 68 65 33 73 65 46 97 75 63 54 65
75 55 53 68 54 73 53 34 74 65 77 54 67 54 37 75 35 47
34 37 94 44 36 56 54 84 66 34 64 44 75 35 64 48 45 86
35 37 38 47 83 54 37 37 48 84 33 67 77 35 03 44 34 48
35 75 55 53 45 37 93 52 76 35 74 04 42 37 38 57 66 32
53 35 65 83 32 53 68 77 85 66 53 37 46 66 46 33 37 65
75 35 55 54 46 86 35 45 77 35 03 73 43 38 38 76 52 36
47 38 83 44 34 45 66 83 35 57 68 74 74 43 43 37 65 84
36 73 54 65 75 36 76 44 65 66 43 56 35 68 75 44 43 64
54
```

4. Break this ciphertext with the two-stage attack.

```
34 80 57 87 47 63 47 25 88 56 78 76 44 58 24 60 65 57
45 34 86 44 58 95 75 63 44 86 25 67 57 57 45 36 57 43
77 86 87 47 34 89 27 56 65 77 66 33 50 24 66 86 58 43
65 50 36 77 65 77 64 65 56 36 60 64 64 77 57 67 55 66
78 75 63 54 69 44 88 64 65 67 36 57 47 67 55 67 63 76
67 47 89 74 75 75 66 66 27 90 68 74 67 35 59 25 88 86
65 44 54 69 66 68 55 75 45 33 67 56 76 65 86 55 35 48
47 89 74 56 73 67 47 53 60 86 56 76 37 60 44 96 56 65
66 56 79 43 58 75 64 73 37 47 56 67 78 87 43 44 59 56
88 65 78 46 66 50 55 58 87 54 47 34 79 43 89 74 56 76
53 48 27 57 75 56 75 37 46 34 79 77 87 63 37 78 25 97
74 58 84 53 48 56 76 56 55 53 37 49 25 57 65 87 45 37
47 24 67 57 75 56 44 69 64 76 56 87 63 35 47 55 77 78
67 45 34 48 46 99 77 65 55 37 47 44 80 68 75 67 66 66
25 77 78 87 45 34 48 55 89 86 58 43 53 80 33 67 78 75
76 76 50 24 68 58 75 75 66 48 24 60 88 86 66 76 78 56
57 75 94 64 57 60 23 60 55 78 47 66 50 24 77 56 87 86
53 57 63 58 56 78 46 35 57 63 60 55 56 46 37 47 53 57
56 87 45 57 49 25 59 87 58 64 43 76 24 60 94 56 77 63
50 47 89 74 56 45 75 67 55 89 75 78 57 37 47 26 58 55
58 43 65 50 36 77 56 87 86 54 79 44 88 65 84 66 44 67
47 80 65 55 44 44 79 44 96 56 95 63 37 78 25 77 78 87
45 34 48 55 89 77 75 45 65 67 47 89 74 56 53 37 56 25
80 87 58 77 65 59 43 67 55 65 56 66 48 24 60 54 87 63
35 46 34 69 87 86 84 53 67 36 76 75 87 44 35 69 34 89
56 86 53 67 59 43 60 54 75 76 54 78 47 60 95 54 47 34
79 43 58 54 75 44 65 79 56 77 64 56 57 54 86 25 80 87
58 76 53 48 53 90 66 77 64 46 67 43 67 94 56 46 34 57
64 80 88 84 47 57 79 43 58 55 56 56 37 47 26 88 58 54
76 53 48 36 67 86 56 53 44 49 25 77 78 67 47 67 47 56
68 88 87 53 37 47 25 58 86 84 45 46 67 34 79 77 97 77
63 50 47 89 74 56 44 35 76 27 57 87 86 53 44 49 25 89
58 64 45 36 80 24 77 78 68 76 53 48 53 57 58 68 44 35
78 55 60 54 87 63 35 67 47 96 56 86 76 54 60 34 89 75
58 67 45 89 56 76 56 64 54 57 89 26 58 87 56 56 66 67
63 58 86 95 63 37 87 25 57 56 95 47 34 68 44 80 68 88
67 36 48 24 66 97 57 64 34 48 36 89 75 58 67
```

Unit 81
Bifid cipher

The *bifid cipher* is one of Félix Delastelle's inventions. It uses a keyword to fill a Polybius square and a period to determine how the plaintext is divided into units that are enciphered together. The only way to adequately explain is through an example. Here is our short message:

```
THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER
```

We will use the keyword `POLYBIUS` and a period of seven. In the simplest way, we can fill the square thusly:

```
  0 1 2 3 4
0 P O L Y B
1 I U S A C
2 D E F G H
3 K M N Q R
4 T V W X Z
```

Next, we divide the plaintext into blocks of length equal to the period seven. The last block is short, but that's OK. We can encipher it in the same way as a full block.

```
THISMES SAGEWAS ENCRYPT EDWITHA GRIDCIP HER
```

Now let's encipher the first block. We write below each letter the row and column labels that address that letter in the square:

```
T H I S M E S
4 2 1 1 3 2 1
0 4 0 2 1 1 2
```

Next, read off the coordinates from the upper row and follow it with the lower row. We divide it into pairs, and remap those pairs back into letters by using the same square.

```
42 11 32 10 40 21 12
 W  U  N  I  T  E  S
```

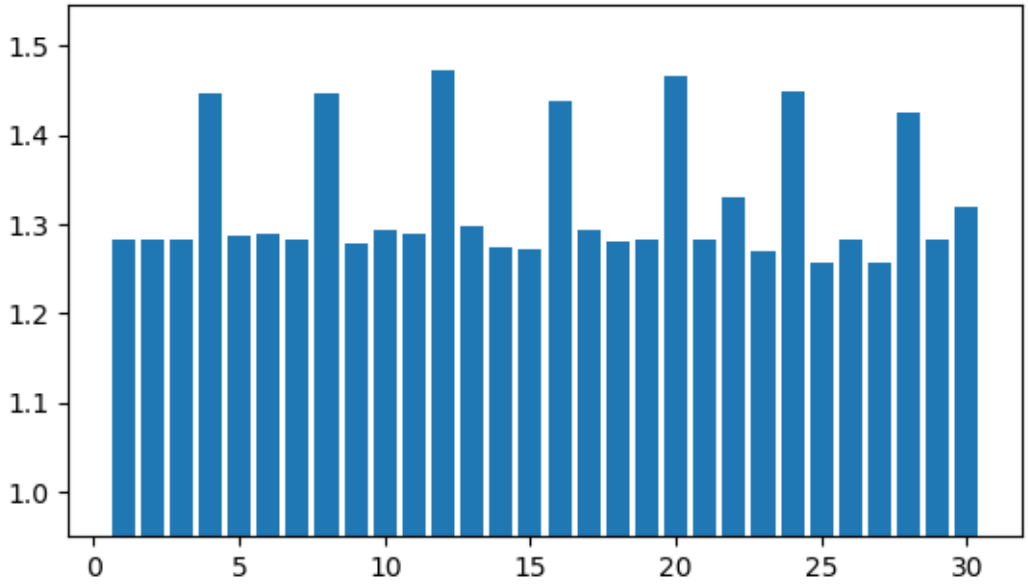
The full ciphertext is

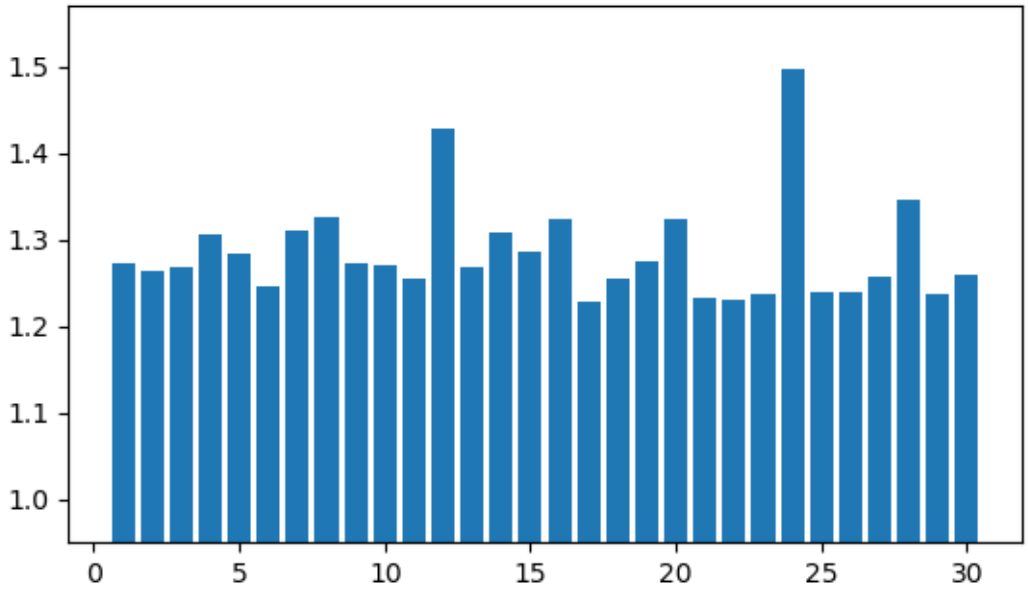
```
WUNITESUFVSQSNGAPVHXPFVWULPXGSUYTBPFRC
```

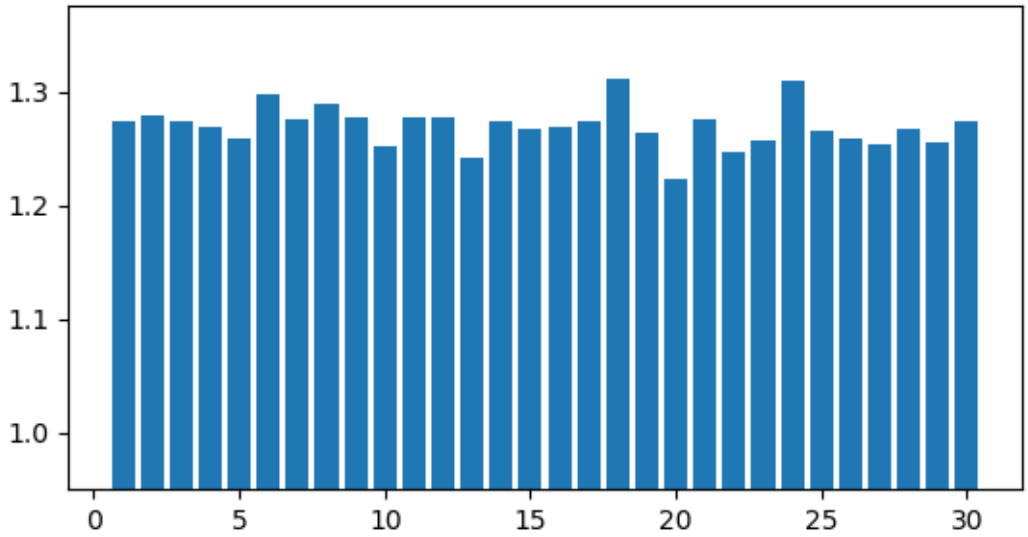
If the period is specified as zero, then the convention is that the entire plaintext in enciphered as one block.

The breaking of letters into smaller parts (in this case two base-5 digits) and separating the parts of each letter from each other is called *fractionation*. We will see this again.

If we are given a ciphertext and want to break it, the first thing we need to do is find its period. One approach to this question is to graph the index of coincidence as a function of the period in the same way as we did when examining the polyalphabetic substitution cipher in Unit 31. Here are three examples from real ciphertexts that have periods four, twelve, and zero. As you can see, the peaks are not as large as they were in our analysis of the polyalphabetic substitution, and it is not always easy to find the correct period in the graph.







Once the period is known (or guessed), we can apply a hill-climbing attack that strongly resembles the one we built for the Playfair cipher in Unit 71. We need to change the decryptor function, of course, but the rest of the algorithm remains unchanged.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 210-211.

Félix-Marie Delastelle, *Traité Élémentaire de Cryptographie*. Paris: Gauthier-Villars, 1902, archive.org/details/8VSUP3207b

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, page 243.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Bifid.pdf

Wikipedia, en.wikipedia.org/wiki/Bifid_cipher

Practical Cryptography,
practicalcryptography.com/ciphers/bifid-cipher

For other approaches to cryptanalysis, see:

> Practical Cryptography,
> practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-bifid-cipher
>
> António Machiavelo and Rogério Reis, "Automated ciphertext-only cryptanalysis of the bifid cipher," Universidade do Porto technical report DCC-2006-1,
> www.dcc.fc.up.pt/~nam/publica/dcc-2006-01.pdf

Programming tasks

1. Implement an encryptor for the bifid cipher. Remember that there are many ways to mix an alphabet from a keyword and to lay it into a Polybius square.
2. Implement a decryptor. Remember that there are many ways to mix an alphabet from a keyword and to lay it into a Polybius square.
3. Write a function to take a ciphertext and try to find the period. It will be similar to your function for the polyalphabetic substitution, but you will need a new way to detect a peak.
4. Implement a dictionary attack.
5. Implement the hill-climbing attack by modifying a copy of your attack on the Playfair cipher.

Exercises

1. Encipher this text with keyword `SOCIETY` and period five. Fill the grid in the simplest way, as we did in the example above.

   > How self-contradictory, in the first place, is the nature of man! How sociable he is! also how unsociable! We have among animals the gregarious and the solitary. But man is of all animals at once the most gregarious and the most solitary.
   >
   > (from *Modern Society* by Julia Ward Howe)

2. Encipher the text in Exercise 1 with the same keyword but with period zero.

3. Decipher this ciphertext with keyword ROBERT and period 6.

```
RRHIKFERHMKSUFBPEVBETUFWEAOALEQERYMKHBTSLCRBANVTTRKNNP
VFBEHALCBFNVRSDNNTTFNPETERHMKSPPENBMRHBKSLTLGCAWFNBETV
ODKSTFVOAURLIVRVQUHQHEUGLAREWZHHPEARHRCYLASRHYUEHAOAVN
ORAZETAOOVNZOPCRBNBRPVNUOPCRBNBRWVNAPBCNLOLKBLWVFDGRME
RBBLVPBSEAEAPRBUGVBEDPRTROFQPTNEASAFCECTSTOISAVRDOHTQC
LICRRHNNNPBZRRHIKFERHMKSPINQDTABLYVFPHNRYSAPICBRPABBNG
FVORQDXORNUGHKLGOYBAOSQPFNHTSEWPCANNHSNWYFEEEA
```

4. Decipher this ciphertext with keyword SAMUEL and period zero.

```
FQAVKKIOQSEUWUKSURZICUSQINORBSRLYSILSPYUSSLRUYUSSLRGRS
UAVKUEOGCHOHATFDMQAWFBAERKQUSVRBILMZCVOTSQZUDTIYMVYUTN
HYBYUTNHYWFKZPOZYTOPDNYFN
```

5. Break this ciphertext with a dictionary attack.

```
REKTAXKSIIZVNCOGEOOMEKGUFQUKXMSPYBTBBRVYKGSOMSVHTOLDXP
AYODCNEHVDTCVYAKHPSBWXVKBBOVYICWOLVITUHTWEFTVPUUBDUPPL
RMFYUYCKUHVWWCQNETERWOZANOOPPTLDQTPOFCHSRCCTAHRCIKZVNG
WCWNEHUCQXQOWQKQNFYFDGVSVDBLSEEZHBONAIQWFBYUQYMDRLRZNY
HEEIEQXVMHTCEFTVPDWBFNDHPWVFEWSWLGNQEPAATVHROHCKOQSQHI
TFRIELFOUAPUTFABREUCOSBQLELFZEEKBSRTITEMWZVIACVEVKNSKM
PIPXBDCQGONFSHNOVFTVYPMGYUEPMQBMXLDCQGONBSHHZVFCGSZAK
```

6. Find the period of this ciphertext. Then break it with your hill-climbing attack. What is the keyword?

```
GRRAVYGIOLRGTSCYNRTYWYHVYGUMLRRENOUVRVEERIYITRLVOACOTD
EYYGNMTVBECVEERSYWIRFRONCUCPARIYIVDNVCBKXGADXETDRYGOEG
TPGYFRRMOQCUSQBYDFVTGFMRROMCSDWEAIBGWTHFYAVBYCHTBAEXCU
MROOACIDIVIYIVVOGVRHRQADGGQRYOHAQXMEYNDDYSPQYYEYPVYGUM
DGRKYLDGIYIVIYIYIAIUVRMEOGZQPOEZDDUBYCAROAKOEGIMNGTMSS
UIDLEGTYTVEDVUIDIYXVKRFGNMRAAHTEFBNBEUCHAWCFMWAAQYPAXZ
OOGCOWVGVUCTGAMGDQGYNDUYYPPRANMWAFSHHYVUXOIVRMLYERNGHN
VBPCROUTRIPDRYASNYMNUUCGOZUIHIGQZVSCFKPBEUQYAPOARGGOAC
PCXFFTBRSOYCDURROTKSLKRBBAZYQRRIABVLYPUYCAFOOITICFFYOY
EYLRRALPORWIHLYGUMNGTMRAAHTKFYAAHXFOHZAOKVOGCAIYIVADXE
CRREPFYRDHZGRDEUICIFLFERPXETATSYGEMFPDWZDYYNPBDOVUCTGI
GIEDWERAEDBNTAIYIWSCYYAIUIAVUMNQTYSTNFBDWBNGHNLCREPAEX
CRMRNACVOYCYGNOUIAIURGOZCYRAKYGVUIHICTASQRZOTYTVAGXMRD
EYYGNMRYEDCQAZIDIZTARVM
```

Unit 82
Trifid cipher

The *trifid cipher* is a generalization of the bifid cipher to three dimensions. Yes, Félix Delastelle invented it, too. Instead of the Polybius square, a mixed alphabet is placed in a 3×3×3 cube. Breaking the plaintext into blocks of equal length is the same, and period zero means the whole text is one block, but the fractionation is done with three base-3 coordinates. Since we have 27 spaces in the cube, we do not need to drop any letters, and need to add one. The new character can be space or some item of punctuation.

Here is an example. Suppose we want to encipher this message with keyword KEYWORD and period eleven.

```
THIS MESSAGE WAS ENCRYPTED WITH A GRID CIPHER
```

The mixed alphabet can be KEYWORDABCFGHIJLMNPQSTUVXZ_, and we can put it in a cube so:

```
    0       1       2
  0 1 2   0 1 2   0 1 2
0 K E Y   C F G   P Q S
1 W O R   H I J   T U V
2 D A B   L M N   X Z _
```

The coordinates of a letter are the layer number, the row number, and the column number. So ‘G’ has coordinates 1, 0, 2. We divide the plaintext into eleven-letter blocks, and write the coordinates under each letter.

```
THISMESSAGE WASENCRYPTE DWITHAGRIDC IPHER
21121022010 00201100220 00121010101 12100
11102000200 12002010010 21111201120 10101
00121122121 01212022001 00100122100 10012
```

The coordinates under the first block are read out by rows and broken into sets of three. Each triplet is remapped to a letter in the cube.

```
211 210 220 101 110 200 020 000 121 122 121
 U   T   X   F   H   P   D   K   M   N   M
```

The full ciphertext is

```
UTXFHPDKMNMYOYQPQEEVBEETFRILJKCNCMEWHR
```

Reading and references

Félix-Marie Delastelle, *Traité Élémentaire de Cryptographie*. Paris: Gauthier-Villars, 1902, archive.org/details/8VSUP3207b

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; pages 210-211.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, page 243.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Trifid.pdf

Wikipedia, en.wikipedia.org/wiki/Trifid_cipher

Practical Cryptography, practicalcryptography.com/ciphers/trifid-cipher

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.
3. Write a function to take a ciphertext and try to find the period. It can be the same as the function you wrote for the bifid cipher.
4. Implement a dictionary attack. There are many ways to place a mixed alphabet into the cube; just use the simplest one.
5. Implement a hill-climbing attack. The modifications to the cube that you need to consider are swapping two element, swapping two planes in any of three directions, and flipping in any of three directions (up-down, right-left, front-back). Feel free to make a modified copy of your attack on the bifid cipher. For the fitness function, you may want to strip out any of the 27th character in plaintext. The margin of error for stepping downward should be variable, and about 5% of the fitness.

Exercises

1. Encipher this text with keyword `FAIRY` and period 8.

The children who read fairy books, or have fairy books read to them, do not read prefaces, and the parents, aunts, uncles, and cousins, who give fairy books to their daughters, nieces, and cousines, leave prefaces unread. For whom, then, are prefaces written?

(from *The Orange Fairy Book* by Andrew Lang)

2. Encipher the text from Exercise 1 with period zero.

3. Decipher this text with keyword `VIRTUE` and period 5.

```
TPHVXTTZXYSK_FAAGJVRIIDN_VRDVEVZJFPDFBIPIN_AYII_RIKEVW
BYKKVPDPFGDIONZITBSCOFRMQOHXIAFVDHCXSV_IQTBHLSVMEQDQIU
WAHXIAFVDHCXWTPPAAWDSGSBHDSVMEQDVRAKRUCOGTDTUEZBDALKRR
```

4. Decipher this text with keyword `CIRCLE` and period zero.

```
VEDGTIACIGQLVCCCYOAJGCRPCNVRLTDLYOAJPCFGBJVJMGCORGBJLL
SDCTBEWDOAHMZLQCKMGIGMLTSGIJICCIGKNIGMLTUINCEIGEC_QHKS
EIHQTJCKEEGVEOIFPHURNTSAIMPYZMAIHNHMWMJWNMPYZHKPNMMEPK
MJIZNMMFOXWPMK
```

5. Find the period (not so easy) of this ciphertext and break it with a dictionary attack.

```
NMINDWATDAOAEAPYHH_IMALEWTJUPNP_QUKFNUPAPGNERKEIK_KPEX
GWFLV_SGTHHN_CRGYGGZPMDPYKGDTBIUIMAGHNNJOFHVEEDIMKPNHN
N_CQTABYWQUJWMIWHJPTNVTCQGFPGDJCAQADEKLQIYFXLWFHPNLCGB
LPJJCENIRJPLVP_HPADLPGEFHMNBHANZLQUSIPCGIEAHCBKMPTNVTC
QGFSGPJCHAUKAGTGPNIEOVNIZBPIEQDDDVLCGADOOWLCDRSKDZNACE
YHQWGNYTDBTZISEOLJIRR_LXGBQRIZUPPDWASMZXPPEBEBZJAXIHUP
XPSHSBABNZFERLIJHCENBFJPLYGTYXYVOIZBYFPLNDCBTIMTUSKSLG
WONVAPTMGEFFUKBICSIHIFVPPRCHUWXBKHIEOOYQNISHS_JUKJPRDP
WVPYHPEDOIWLUUKEXASDUETZJPXINKJIBPBHIFVGPXCKAFIIPUXDHE
YABHNSKADTMPCQAYEJSPAPJ
```

6. Find the period of this ciphertext and break it with a hill-climbing attack. What is the keyword?

```
RUPHNRDTHBYKTTRTQUXCAKYINADHLHT_BXOQRLWZNBZHZOMSVAFYWR
SDFYEIEIRNHSSUGTXNMNAFHMQTETDBTYPMHDIJYKZHBKEQOFMYJ_TH
YUGQEXUAG_WYYUPYCANTFTFDF_DSKATHOPDNRKMPRICTOOFHIPAIRU
IAYIOYPFOQZ_S_RGFPEVUCSYAHAXWXPRNGMHQBKVACI_DBDCBJQATT
CUTODHIKMGQHQPQXAHA_TMEUJOIHNKT_HBYKTPRQRYYLJPEKEOUSZH
SWDOTIDMMDLAFTUXI_RFFPLCUTLPMDNNWCARHVMMAERURHMPDVVGHA
SDZRRVUKHDJDJOJQOVTSBMPFNUYHAZFKH_Z_EIASHXSSSAUGSXPSFI
ERENHZWYYYYXUDSAQXX_X_FEEDQDIQQOBMDENPODWJHHNJUTCOANXT
AZIVPIT_YDJTCNTPGYAXHZUSQRHVMMAERGBOIPILCDQPDPNXEQOQAH
K_CKNHRBTZYWPYPKNMCFYHNJM_DEPFSAKSCTCQAPXTHHRRVP_INRXP
RH_SSVFYHDNMDBGHQ_RSKKJHINI_AMQMAPDKLGOAYYWZJYKPFK_FQA
```

DUGIETADUGFBAXHQLXQY_TUEFIH_RLGEBIFQFLBHWNFLOXYSTLHUXI
CFVISOJKITNYYZTTYBHHSUQDRREBH__NQQFVKRPDPYBPQFKTCIJIPH
MWUTCYHDETFJQNOUBIPARJVOOTGACPKPUEBPHSHSRTOQBKXZZXSYKL
KDAPEYWYGRTEISIYMRSIILBAOQSYLUDYIYWMTBCJVDOAHTSXOKUDCC
YHY_WDQEOXSRXTVYIIEH__TDBQHMTDSHDPVKWAAOWVQORJKKOHDIIX
POCQCXTRDYQPFNIKKPARJTUOFOCRLUXESZXHADINCGYAZQZSBRHEPL
WISCAYYKRQNNPHPNQXEOFSIHFJVUOETL

Unit 83
ADFGX cipher

The *ADFGX cipher* is a Polybius cipher with row and column labels A, D, F, G, X followed by a columnar transposition which may be keyed with a keyword. Decipherment must be done in reverse order. The choice of the labels A, D, F, G, X is due to their low likelihood of being mistaken when transmitted in Morse code.

Some use a variation in which the transposition stage is a permutation cipher rather than a columnar transposition.

One way to attack the ADFGX cipher is to modify the hill-climbing attack on the columnar transposition cipher. However, since we do not know the contents of the Polybius square in advance, we cannot use tetragram fitness. Therefore, we use the index of coincidence as the function that we wish to maximize. For each permutation that we try, we decipher the ciphertext with a grid containing `ABCDEFGHIKLMNOPQRSTUVWXYZ` and evaluate the IoC. If we can maximize the IoC at a value resembling the IoC of English, then what remains is a monoalphabetic substitution, and we can use the attack from Unit 28. For permutations of length greater than five, there is a complication: There may be more than one permutation that gives the same maximum IoC. In that case, we will have to try several candidate permutations until we find the right one.

Reading and references

Practical Cryptography, practicalcryptography.com/ciphers/adfgx-cipher

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 339-344.

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.
3. Implement the attack described above.

Exercises

1. Encipher this text with the keywords `PINK` for the square and `FLOYD` for the transposition.

   > It's been three months of lockdown. Please, someone send vegan brownies to Old Pink, in care of the Funny Farm, England.

2. Decipher this text with the keywords `FRANCE` for the square and `PRUSSIA` for the transposition.

   ```
   AFAAGAGGFGADFXFFGFAGFDAAFFFGAGAAXFXAFDXFAAAGFGAFGDXGFF
   FXFAAGAGXXGAAXAAAFDXGXFFFGXXDDFGGFFXFDAAGFGDGXXFDADADA
   ADGAGDGADADADAGGDDFAGGAAGDGDXDDADDDDFDDDDFXDGFGDAXAFFG
   GADFXADDDAGFFAAGAFGDGDAFDGADAAAGXAAAGAGGGFGDDAFAAFDGDF
   DADFFDAFADAGDFFDFDFDDXAGFFAAGGXAADFAADDFFAGDFFAFGGXAGG
   XFGFXDGXGGFGGAADXDDGAFXADGAAGXFFFDGGAAXDXXFFAFGXAGDGGF
   AGDDFFGXAAFFAGXAAAXXDFFGAFXDGXXXAAAAXFGGDDAGAXFGAXDXAG
   FGAAFGFGXFFAGADFADDGAAFDFGXXGFAFDFGGDXGDDFFDAFDADAAAGF
   DFDFDDFAGADDFGFGAADFXAFGGXXDGGDGGFDD
   ```

3. Break this ciphertext. What are the keywords?

   ```
   DXGAGFDAAAXAAAADAFADAAGFDADGAAAGFAGADAGFDGAGFGAGAAADAA
   GDAGADADFFADDDDAXDAFFGDDDXADAAGAGFXDADXFDAADAGDAFAAFAD
   AGFFFDADADAGXAADDFAAAAGXGFDGDGGDAFDFXXDDGDFFDDDAFFDAGA
   AGDXADAAXDADFDXDADGGFDFDAGGXGDGXGDDGXGDDAGDFFXFFDFXGDG
   DFDXGDFFDAFGAFGFFXGDGDFFXGXFXFFFFFGXFDGXFFXGDDFFAXAGGX
   DXDDGGFGGAGGADAXDGDXFDFFFDDGDAXFDXGXGDFXGGDFDGDXFFXGXX
   DGGDDXGADFFDAGFXDGFDAXXDXDAGFFGXADGFGXXDFDGXDGFFDXGDFF
   FXFFGAFDXXDGGGFADFFXFFDGFGXDGXXXFGDGGDDFGAGDFFFDDAXDFF
   AGDAFXADGAGDXADGXFGGXDAXFFFXGAXGGDGGFGXXXGADGGFAAGAGFD
   GGGDAXXXDDGXAGDDGDXXAGDFGGFGFDGXXXFDXAFGFFDDXFFFDGFGGG
   XADFFXFXXGGGDXXFDGDDGFGFGXXFAAADDADFAXXGDADDAAGDDDXAAG
   AFDAAADXGAGDADFDXAAADDFFGGAXDDDAGFDXDXXAGXFFAAGAGDADDA
   GADADDDDDAAGFFADGFAAFAFDDDGFDDDDAAGADAGDDAGFDFADDXADDA
   AAFDAAGAADDAAGAGDFGAAFDAADXGDGDAGAFFDAAADDAFXFGAAAAAGF
   DADDXDXFXXFGDXDXFXDFAGXDXFFDDXXFGFGXGXGFXADGXGXXXDXDAX
   FDAGXFFXFXGDFDDAXAFGADFGDGDFGDXDXDGDFFGDAGDFFGFGGDXFDX
   FXDADGDFXXFFXDAXFFGGGFGXDDGDFFFDDFXDDADGDDGXGDXAFXXGFD
   FGGGDFFXXDGFDDGAFFFDDXGXFFXADAGADAFXDFDDAAADFDAGADDADX
   XGAAFXDDADDADFDDDAGDFGFAAADGAADDFAADADDDADGFDAAAADADAA
   GXDGDAADAAFAXDXAADDDGAAAGDDDDGFGDDFAADXAAAADAADDXDAGAG
   ADDAGDFAFGAAFFGXAAAAGXAAXGGXXGAAGGXDAAFFGADDDDDAAXADXA
   AA
   ```

4. Break this ciphertext. What are the keywords?

AAAFXAFFGAAXDDGGAAAXDGGFGDAAGADAGGDFDDFGAFGDGADDDGAFXG
DAFXDAXDFADADAFFGGDFGAGFGADFAFFAXAFAXFAXFAGAXDAFFXXAGX
GFDFGDDDXAXXAXAXFAFXFXXGDXXAFFADFFFFAAFAXADXAXGGGAFXDF
FDFDDGGGDADDGGADAAADFFGGXAAAFGADFDXGDAGDADGFFFGAAGDGDF
DDFDXFGAGDADDGDGFDDAAAAGFFAGAFGDFGAXFGDDAGGFDAFDGDDAFA
DDGGAXFGDGGDXAADDGFFGDAGAGGAAGDAFFDGFGADDAGGAFGDDDFGAA
GGAGDFXGDFFDAXFDXAGFAXAXAXDAFXXADXDAXFADGXADDADGAGFXDX
XXXAXFXAXFFXFAGDXADGAAGFGAXXXGDGADGFAAGFADXGAXAGFDFAAF
AFXAADFGXAXGFXFDADGAAXDXFGXAADGGAGGDXFXAXFAXFADXFXXXDA
DGDFXDAGGGDDDXGAGAFGGDDGAXAXAAFGFAGDXADGGDXDFDXGXDFAGA
XFDGFXDADXXFDDAFAXXXDXAXFAGGGGFAFDDXDFGFAGDAFADDGXXDFA
AAADAGDAFGAFXDXGGFGGGFFDDADAAGFGGGAGDGAGFGFDAGFDADDADA

Unit 84
ADFGVX cipher

The *ADFGVX cipher* is the extension of the ADFGX cipher by using a 6×6 Polybius square. Because the square has 36 places, it holds the full English alphabet and all ten digits. The choice of the labels A, D, F, G, V, X is due to their low likelihood of being mistaken when transmitted in Morse code.

Some use a variation in which the transposition stage is a permutation cipher.

An approach to breaking the ADFGVX is to look for a columnar permutation that results in the fewest distinct digrams. We are looking for a decryption with mostly letters and few or none of the digits. After the transposition, the Polybius cipher is broken as usual, if possible.

Another approach to breaking the cipher is to extend the attack on the ADFGX to use 36-character alphabets and a 6×6 Polybius square.

Reading and references

Practical Cryptography, practicalcryptography.com/ciphers/adfgvx-cipher

General Solution for the ADFGVX Cipher System, Washington D.C.: U.S. Government Printing Office, 1934, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/FOLDER_269/41784769082379.pdf, archive.org/details/41784769082379

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 339-346.

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.
3. Implement the first attack described above. The length of the permutation is an input.

4. Implement the second attack described above. The length of the permutation is an input. As a subproject, implement the attack from Unit 28 for an alphabet of 36 characters. In calculating the fitness, it is acceptable to delete digits from the text.

Exercises

1. Encipher this text with the keywords `WEATHER` for the square and `PICNIC` for the transposition (remove repeated letters in the keywords).

   > To give a picnic party a fair chance of success, it must be almost impromptu: projected at twelve o'clock at night at the earliest, executed at twelve o'clock on the following day at the latest; and even then the odds are fearfully against it. The climate of England is not remarkable for knowing its own mind; nor is the weather "so fixed in its resolve" but that a bright August moon, suspended in a clear sky, may be lady-usher to a morn of fog, sleet, and drizzle.
   >
   > from "The Picnic Party" by Horace Smith

2. Decipher this text with the keywords `MUNCHKINS` for the square and `CYCLONE` for the transposition (remove repeated letters in the keywords).

   ```
   DGDADVAADDDDDGGDFVDGDVFGDGDDFADDADGDDGDDDAGDADAGDDDDDD
   GADFFGADGDDDGADAFFGDAAFAGGDAGAADAGDDFGDFDDADDGDAGFDGFG
   GADGFFDDADGFAAFDAFDAFGFGFFGAFADAAFDDGGGDGDDFDDFDAFDADG
   ADXDFFGDGVVFVGXFADVFVXGFFXAXVAFGXVFAFVFFVGGXDFFVXVVVXX
   FVXVGVFXDFVAXVAVDGVVAFFFAVFVVXDFGDDDVXDXAFFAGXFVAVGVDX
   FXVVXFDVGFVVFFDXDFVFFVDXFXDFFAFDXFGXFDXXADVDDDVDDFVVAD
   VDDADGDDDFDADADGADFGAADADFAFDDAADDAGDGGFADGGADDDGDGGDF
   ADGFGDFDFFDFGAAADDDVGADFDGDAGADDFFAFDGGFAGAAADAAGGDFFA
   ADFFDDDFAGDDGFDDDDADDDDFADFDFDGFDAGDFDFAADDFADDADDGDAG
   FFDDGDDDADDAFDADGDDFGADFDFAADFDADGGDAGDDADDAAADGGAFDAA
   AFDAFFAADGDDAAFDDDDDFDFVGDADDDAFDDGGFGDDAGDDGADADDADFF
   DFDGDFAFAGADDGAGGGDGDFAGDGDGDGDDVDVDDFADADDVFGFFGDFGGG
   DFDADDVFVXVFVDDXFXDVAVFVFGFAVVVGDAVFGFDDVFFAXDFVGFFVVV
   FVXFFFVFDGAGVAVDGFVXAFADVFXVFXVFVDFVVFVFFDVVDVDVXAFFVV
   AVFVAGVDDFAADVVFVFXXDFVXVXVVAGADXDAFAFVVVVVVXDVFAXFAXF
   XFDGDDXVDVFXXAVXDVFDXFFVVAVFXADFDFVXDVXXDFFDXDVXFVDXVV
   FDFFFXVVFDAVGFFXFXFFVFVAGVFDDFVGFFXDVFAAFVAFAFGDDDFFAF
   XFVVAFGFDAXDDVXXFAFVVVGFFXXVVFGFDGGGVAXFFVXGFDDXDAGDFG
   DFVVXDAFVF
   ```

3. Break this ciphertext. The transposition has length nine. What are the keywords?

   ```
   GFAADFVAGDXFVAAFADGDDFAADAXAVDAADAGAAADAXDAVGAVFDAGFVA
   XDDDGDXGDAGAFAVDXAFAAGXDFDADFAVFFADAAGVDADXAXFAAVADAVD
   XDDAFDDFXFFADFVADAAGAAAGAFGVFGVDAAAAGDAAXDFAAADAAGVGAG
   AAADVFXDDFADGDAAAAGGDDFAAAAFVDXDAGXGDAAAVGAAVADDXDXAAF
   ```

VDGADDFGAGVFVDXAGDVDAAGGXAAAAFVDGFDFGFADGADDAFFDGGXAAG
FGGFDAAADFDAADGAAFDGGGXAFDGAAAFFFDGADAGGDFDGXAFAFADAAA
GAADXAVFXDGFVAGDDAVAXFAAXDXGDDVGGGVFXDADDAFDVFDDVDGDDA
GAXDVAGAAGVFAAVDVDDAVDAFGDFDXAAAAAAADAAAFADAGAFGADADDD
XGDDXAAGXADAXAAFAAXGDAVDGGAAFAFFDAAADAXFGFADXFDDAAGFXG
DFDGFFAGGAAGXAAGXAVADDGDXFGGAADDGAGDAGDAXAVDAGAADXDAAV
GVAADDFVAGDADXGXDAAXDAFADAAAAAAXGFFDDXAVAVFADADDAAADAA
AGAFDVGXFAFXGXFAGXAXADVADXFXAXAVDVGVFGAXGGGFFGAADXAVAA
AFDGGXDGVFDGDXDAAFAAFXFGAAAFAFDAAAAAFAAXAADGGXAXAAGVDA
AADADADVADAXDFGXAGGFDVFGGDAXAADXAXAAAGAVAGDGFXAXAADFFA
AVDADAGADAAVAXFFDADAAAAXFAFVAFDXFADVDGDGAADXAA

Part VI
Ciphers based on matrices

Unit 85
Matrices and vectors

A *vector* is an ordered set (a list) of numbers. The length of that list is its *dimension*. The numbers in the list are the *components* of the vector. In this section of the book, vectors will be written as *column vectors*, i.e., as a column of numbers. The name of the vector is written either in bold face or with a vector arrow over it; for example:

$$\mathbf{A} = \vec{\mathrm{A}} = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$$

The components are written with a subscript. For the above example, $A_0 = 2$, $A_1 = 3$, and $A_2 = 5$. Notice that we strart counting with zero; this is not the usual convention, but we are programmers and we use languages that index lists beginning from zero.

The set of all vectors with the same dimension is called a *vector space*; its dimension is the same as the dimension of its vectors. The *origin* of the vector space is the vector all of whose components are zero; This vector is called the *zero vector*, **0**. In a vector space, we can add vectors by adding their components. If **C** = **A** + **B**, then

$$C_i = A_i + B_i$$

for all $i = 0, ..., d-1$, where d is the dimension of the vectors. Since it does matter in which order we add two numbers, it is also true that we can add vectors in either order; we say that addition is *commutative*. Of course, if we add the zero vector to any other vector, then that vector remains unchanged; for any vector **A**,

$$\mathbf{A} + \mathbf{0} = \mathbf{0} + \mathbf{A} = \mathbf{A}$$

A *scalar* is a number that is not part of a vector. We can multiply a vector **A** by a scalar by multiplying each component by that scalar. If **B** = c**A**, then

$$B_i = (cA)_i = c\, A_i$$

Recall from Unit 7 that we saw how to multiply vectors (it’s not the only way, but it is the only way that we will need) by defining the inner product (scalar product, dot product) as

$$\mathbf{U}\cdot\mathbf{V} = \sum_{i=0}^{d-1} U_i V_i$$

An *m*×*n matrix* is an two-dimensional array of numbers with *m* rows and *n* columns. For example:

$$\mathbf{M} = \begin{pmatrix} 3 & 4 & 2 \\ 7 & 9 & 5 \end{pmatrix}$$

The numbers in a matrix **M** are its *elements* (or *entries*) M_{ij}, where *i* labels the row, and *j* the column, in which a component exists. For the example above, $M_{01} = 4$. The *dimension* of a matrix is the number of rows by the number of columns. For our example, the dimension is 2×3. The set of all matrices with the same dimension has algebraic properties similar to a vector space. We can add two matrices in the set by adding their elements:

$$(\mathbf{M} + \mathbf{N})_{ij} = M_{ij} + N_{ij}$$

The matrix all of whose entries are zero acts as the additive identity. We can also perform scalar multiplication:

$$(c\mathbf{M})_{ij} = c\, M_{ij}$$

Multiplying matrices is a little more complicated. We can only multiply two matrices **A** and **B** if the number of columns in **A** matches the number of rows in **B**. The rule is that if **C** = **AB**, and **A** has dimension *m*×*n* and **B** has dimension *n*×*p*, then **C** has dimension *m*×*p*, and

$$C_{ik} = (\mathbf{AB})_{ik} = \sum_{j=0}^{n-1} A_{ij} B_{jk}$$

for all $i = 0, ..., m-1$ and $j = 0, ..., p-1$. Notice that if *m* and *p* are different, then we can multiply **AB**, but not **BA**.

The set of all square matrices with the same dimension is special. A matrix is *square* if it has the same number of rows as columns. In this set, it is possible to multiply in either order. In general, however, the result will depend on the order, so matrix multiplication is noncommutative. In this set of square matrices, the additive identity is the matrix all of whose entries are zero. Adding it to any other matrix leaves the matrix unchanged. The multiplicative identity is the matrix **I** such that

$$I_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

All of the entries on its diagonal are one; all other entries are zero. The multiplicative *inverse* of a square matrix **A** is another matrix called $\mathbf{A}^{-1}$ such that

$$\mathbf{A}\,\mathbf{A}^{-1} = \mathbf{A}^{-1}\,\mathbf{A} = \mathbf{I}$$

To find the inverse of a matrix we must go on a short detour.

Given a row number and a column number, the *minor matrix* of a matrix is formed by taking the matrix and removing that row and that column. For example, the (1, 2) minor matrix of this matrix

$$\mathbf{A} = \begin{pmatrix} 4 & 3 & 9 & 4 & 3 & 2 \\ 7 & 6 & 8 & 5 & 7 & 0 \\ 8 & 2 & 1 & 3 & 2 & 5 \end{pmatrix}$$

is found by deleting the second row (because the first is row 0) and the third column (because the first is column 0):

$$\mathbf{M}_{1,2} = \begin{pmatrix} 4 & 3 & 4 & 3 & 2 \\ 8 & 2 & 3 & 2 & 5 \end{pmatrix}$$

The *determinant* of a square matrix is a number derived from the matrix. We will define it in a recursive manner. The determinant of a 1×1 matrix is the value of the only entry. The determinant of a 2×2 matrix **A** is

$$\det \mathbf{A} = A_{00}\, A_{11} - A_{01}\, A_{10}$$

The determinant of a square matrix with arbitrary dimension is found from the determinants of its minor matrices:

$$\det \mathbf{A} = \sum_{i=0}^{n-1} (-1)^i\, A_{0\,i} \det \mathbf{M}_{0,i}$$

where $\mathbf{M}_{0,i}$ is the (0, *i*) minor matrix of **A**. With this definition, we can find the determinant of any square matrix by working our way down until we are working with determinants of 2×2 matrices. An important thing to know is that a matrix is invertible in the set of matrices if and only if its determinant is invertible in the set of numbers. In the realm of real numbers, all numbers are invertible except zero. However, in modular arithmetic we know that not all numbers have inverses.

The combination of a power of −1 and the determinant of a minor matrix that we see in the formula above is called a cofactor. In general, the *cofactor* of an element A_{ij} is

$$C_{i,j} = (-1)^{i+j} \det \mathbf{M}_{i,j}$$

The matrix **C** whose *i*, *j* entry is the cofactor of A_{ij} is called the *cofactor matrix* of **A**. The *transpose* of a matrix **A** is another matrix $\mathbf{A}^{\mathrm{T}}$ that is obtained by exchanging the rows with the columns of **A**:

$$(A^{\mathrm{T}})_{ij} = A_{ji}$$

If we take the transpose of the cofactor matrix of **A**, what get is called the *adjugate matrix* of **A**. Finally, we are able to say that the inverse of a square matrix **A** is the multiplicative inverse of the determinant of **A** multiplied by the adjugate matrix of **A**.

Another way to invert a matrix is by using elementary row operations. There are three *elementary row operations*:

- swap two rows
- multiply every element in a row by a scalar
- add one row to another (add elements that are in the same column)

To invert a square matrix, we first extend it with a copy of the identity matrix with the same dimensions. Then we apply row operations until the identity matrix appears on the other side. The matrix that is now where the identity used to be is the inverse matrix. As an example, consider this matrix:

$$\mathbf{M} = \begin{pmatrix} 2 & 1 & 2 \\ 1 & 3 & 3 \\ 5 & 6 & 7 \end{pmatrix}$$

Extend it with the 3×3 identity matrix:

$$\left(\begin{array}{ccc|ccc} 2 & 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 0 & 1 & 0 \\ 5 & 6 & 7 & 0 & 0 & 1 \end{array}\right)$$

Swap the top and middle rows (R_0 and R_1, since we count from zero):

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 3 & 0 & 1 & 0 \\ 2 & 1 & 2 & 1 & 0 & 0 \\ 5 & 6 & 7 & 0 & 0 & 1 \end{array}\right)$$

Replace the middle row with $R_1 - 2R_0$, and the bottom row with $R_2 - 5R_0$:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 3 & 0 & 1 & 0 \\ 0 & -5 & -4 & 1 & -2 & 0 \\ 0 & -9 & -8 & 0 & -5 & 1 \end{array}\right)$$

Replace the bottom row with $R_2 - 2R_1$:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 3 & 0 & 1 & 0 \\ 0 & -5 & -4 & 1 & -2 & 0 \\ 0 & 1 & 0 & -2 & -1 & 1 \end{array}\right)$$

Swap the middle and bottom rows:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 3 & 0 & 1 & 0 \\ 0 & 1 & 0 & -2 & -1 & 1 \\ 0 & -5 & -4 & 1 & -2 & 0 \end{array}\right)$$

Replace the bottom row with $-(R_2 + 5R_1)/4$:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 3 & 0 & 1 & 0 \\ 0 & 1 & 0 & -2 & -1 & 1 \\ 0 & 0 & 1 & 9/4 & 7/4 & -5/4 \end{array}\right)$$

Now that the leading entries of each row are all 1, we need to put zeroes in the off-diagonal entries on the left side. Replace the top row with $R_0 - 3R_1 - 3R_2$:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -3/4 & -5/4 & 3/4 \\ 0 & 1 & 0 & -2 & -1 & 1 \\ 0 & 0 & 1 & 9/4 & 7/4 & -5/4 \end{array}\right)$$

Now $\mathbf{M}^{-1}$ is the square matrix on the right-hand side.

Next we need to discuss the product of a matrix and a vector. We will only be concerned with square matrices here. A n-dimensional vector is the same thing as a $n \times 1$ matrix, so we can handle multiplication in the same way as we multiply matrices. The product of an $n \times n$ matrix $\mathbf{A}$ and an n-dimensional vector $\mathbf{V}$ is another n-dimensional vector $\mathbf{U}$ whose components are

$$U_i = \sum_{j=0}^{n-1} A_{ij} V_j$$

Reading and references

Wikipedia:
en.wikipedia.org/wiki/Matrix_(mathematics)
en.wikipedia.org/wiki/Square_matrix
en.wikipedia.org/wiki/Minor_(linear_algebra)
en.wikipedia.org/wiki/Determinant
en.wikipedia.org/wiki/Adjugate_matrix
en.wikipedia.org/wiki/Elementary_matrix

Programming tasks

1. Go back and find your function that evaluated the dot product of two vectors.
2. Write a function that adds two vectors.
3. Write a function that multiplies a vector by a scalar.
4. Write a function that adds two matrices. It is OK if you only consider square matrices.
5. Write a function that multiplies a matrix by a scalar.
6. Write a function that multiplies two matrices. It is OK if you only consider square matrices.

7. Write a function to multiply a square matrix by a vector. It should return another vector with the same dimension.
8. Write a function that finds the minor matrix of a square matrix given a row number and a column number.
9. Write a function to find the determinant of a square matrix.
10. Write a function to find a cofactor of an element in a square matrix.
11. Write a function to find the cofactor matrix of a square matrix.
12. Write a function to find the transpose of a matrix. It is OK if you only consider square matrices.
13. Write a function to find the adjugate of a square matrix.
14. Write a function to invert a matrix by using elementary row operations.
15. Write a function to determine whether a square matrix is invertible.
16. Write a function to find the inverse of a square matrix. It should first check whether the matrix is invertible and have some way to alert the main program if it is not.

Exercises

1. Find these products:
   a. $\begin{pmatrix} 6 & 6 & 7 \\ 4 & 9 & 7 \\ 1 & 5 & 5 \end{pmatrix}\begin{pmatrix} 6 & 6 & 4 \\ 5 & 0 & 8 \\ 9 & 7 & 6 \end{pmatrix}$
   b. $\begin{pmatrix} 4 & 3 & 2 & 6 \\ 7 & 2 & 9 & 1 \end{pmatrix}\begin{pmatrix} 7 & 0 \\ 5 & 4 \\ 3 & 8 \\ 1 & 5 \end{pmatrix}$
   c. $\begin{pmatrix} 5 & 9 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} 4 \\ 7 \end{pmatrix}$
   d. $\begin{pmatrix} 5 & 7 & 0 \\ 8 & 4 & 6 \\ 9 & 7 & 4 \end{pmatrix}\begin{pmatrix} 7 \\ 1 \\ 3 \end{pmatrix}$
2. Find the inverses of these matrices:
   a. $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$

b. $\begin{pmatrix} 2 & 2 \\ 1 & 3 \end{pmatrix}$

c. $\begin{pmatrix} 17 & 19 & 15 \\ 10 & 12 & 13 \\ 16 & 18 & 17 \end{pmatrix}$

d. $\begin{pmatrix} 3 & 9 & 5 & 8 \\ 7 & 0 & 6 & 2 \\ 2 & 3 & 2 & 6 \\ 4 & 7 & 5 & 1 \end{pmatrix}$

Unit 86
Matrices over the set of residues

For use in ciphers, we need to work with numbers from the set of residues. Review Unit 14 if you have forgotten what we mean. For the rest of this part of the book, we will be working with matrices and vectors whose elements are taken from the set of residues, usually modulo 26. This means that every number, whether it appears in a vector, in a matrix, or alone, is a member of $\mathbb{Z}_{26}$. All multiplications and additions (and subtractions) must be done modulo 26. The inverse of any number must be found by the algorithm in Unit 21.

In the last unit, we saw how to find the inverse of a square matrix as the inverse of the determinant times the adjugate matrix. We now have to be careful that we handle our arithmetic correctly, and that when we find the inverse of the determinant we find its inverse modulo 26. If that inverse does not exist, then the matrix is not invertible.

Programming tasks

1. Find your function from Unit 21 to get the multiplicative inverse of a number in modular arithmetic.

2. Make revisions of your functions from the previous unit so that you now have a function that multiplies a square matrix by a vector. It should return a vector all of whose components are in $\mathbb{Z}_{26}$. Allow for the option to change the modulus.

3. Make revisions of your functions from the previous unit so that you now have a function that finds the determinant of a square matrix whose elements are in $\mathbb{Z}_{26}$. Allow for the option to change the modulus.

4. Write a function to determine whether a square matrix over $\mathbb{Z}_{26}$ is invertible. Allow for the option to change the modulus.

5. Make revisions of your functions from the previous unit so that you now have a function that finds the inverse of a square matrix over $\mathbb{Z}_{26}$. Allow for the option to change the modulus.

Exercises

1. Find these products modulo 26:

   a. $\begin{pmatrix} 9 & 6 & 10 \end{pmatrix} \begin{pmatrix} 10 \\ 18 \\ 13 \end{pmatrix}$

   b. $\begin{pmatrix} 15 & 9 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 24 & 25 \end{pmatrix}$

   c. $\begin{pmatrix} 11 & 9 & 18 \\ 5 & 19 & 7 \\ 6 & 10 & 10 \end{pmatrix} \begin{pmatrix} 4 \\ 11 \\ 20 \end{pmatrix}$

   d. $\begin{pmatrix} 25 & 18 & 19 & 18 \\ 14 & 19 & 24 & 15 \\ 8 & 4 & 1 & 7 \\ 23 & 8 & 24 & 6 \end{pmatrix} \begin{pmatrix} 23 \\ 13 \\ 5 \\ 21 \end{pmatrix}$

   e. $\begin{pmatrix} 16 & 22 & 1 & 8 & 1 \\ 24 & 9 & 16 & 1 & 25 \\ 2 & 4 & 2 & 19 & 11 \\ 11 & 17 & 0 & 1 & 12 \\ 18 & 9 & 25 & 1 & 5 \end{pmatrix} \begin{pmatrix} 13 \\ 4 \\ 16 \\ 7 \\ 6 \end{pmatrix}$

2. Find the inverses of these matrices over $\mathbb{Z}_{26}$:

   a. $\begin{pmatrix} 8 & 19 \\ 21 & 18 \end{pmatrix}$

   b. $\begin{pmatrix} 15 & 14 & 8 \\ 0 & 15 & 11 \\ 6 & 7 & 0 \end{pmatrix}$

   c. $\begin{pmatrix} 21 & 3 & 0 \\ 8 & 8 & 19 \\ 13 & 2 & 4 \end{pmatrix}$

   d. $\begin{pmatrix} 16 & 19 & 2 & 14 \\ 20 & 8 & 0 & 13 \\ 0 & 5 & 19 & 0 \\ 13 & 8 & 8 & 11 \end{pmatrix}$

e. $\begin{pmatrix} 4 & 17 & 4 & 14 & 7 \\ 11 & 14 & 15 & 6 & 8 \\ 4 & 4 & 7 & 17 & 2 \\ 2 & 13 & 0 & 0 & 0 \\ 19 & 2 & 11 & 15 & 11 \end{pmatrix}$

Unit 87
Hill cipher

The *n*×*n Hill cipher* is a block cipher that uses matrix multiplication to encipher each block of *n* letters. (A *block cipher* is a cipher that acts on blocks of texts, rather than individual letters.) Each block is written as a vector whose components are the numerical equivalents of its letters, where ‘A’ = 0, ‘B’ = 1, ..., ‘Z’ = 25. Encipherment is done by multiplying this vector by the key, which is a (square) *n*×*n* matrix. Decipherment is done with the inverse matrix. All operations are done modulo 26 (or the length of the alphabet, if we are using a different one).

Let’s work an example. Suppose we have this short message and that we want to encipher it with the matrix that follows:

```
THIS MESSAGE WAS ENCRYPTED WITH A HILL CIPHER
```

$$\mathbf{M} = \begin{pmatrix} 7 & 8 & 11 \\ 11 & 2 & 8 \\ 15 & 7 & 4 \end{pmatrix}$$

We first divide the plaintext into blocks of three letters, and pad with nulls if necessary:

```
THI SME SSA GEW ASE NCR YPT EDW ITH AHI LLC IPH ERX
```

The first block, `THI`, is expressed as this column vector:

$$\mathbf{V} = \begin{pmatrix} 19 \\ 7 \\ 8 \end{pmatrix}$$

This block in enciphered to a new vector:

$$\mathbf{U} = \mathbf{M}\mathbf{V} = \begin{pmatrix} 7 & 8 & 11 \\ 11 & 2 & 8 \\ 15 & 7 & 4 \end{pmatrix} \begin{pmatrix} 19 \\ 7 \\ 8 \end{pmatrix} = \begin{pmatrix} 17 \\ 1 \\ 2 \end{pmatrix}$$

This vector becomes the block `RBC` in the ciphertext. The full ciphertext is

RBC GUG KAG EQY GQM IXR DEV ISN ZAV OAD FDQ TST BCL

Decipherment uses the inverse matrix:

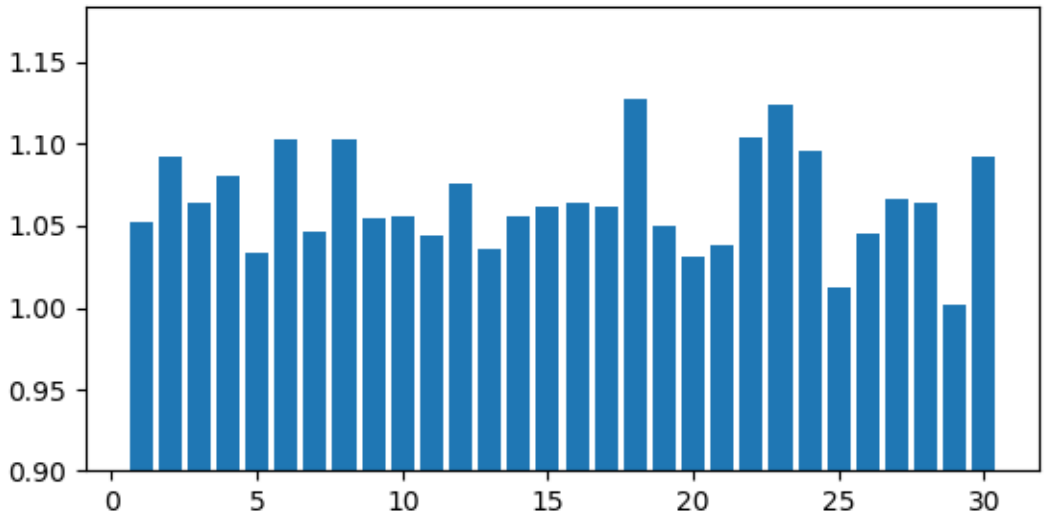
$$\mathbf{M}^{-1} = \begin{pmatrix} 12 & 5 & 22 \\ 20 & 5 & 13 \\ 11 & 5 & 12 \end{pmatrix}$$

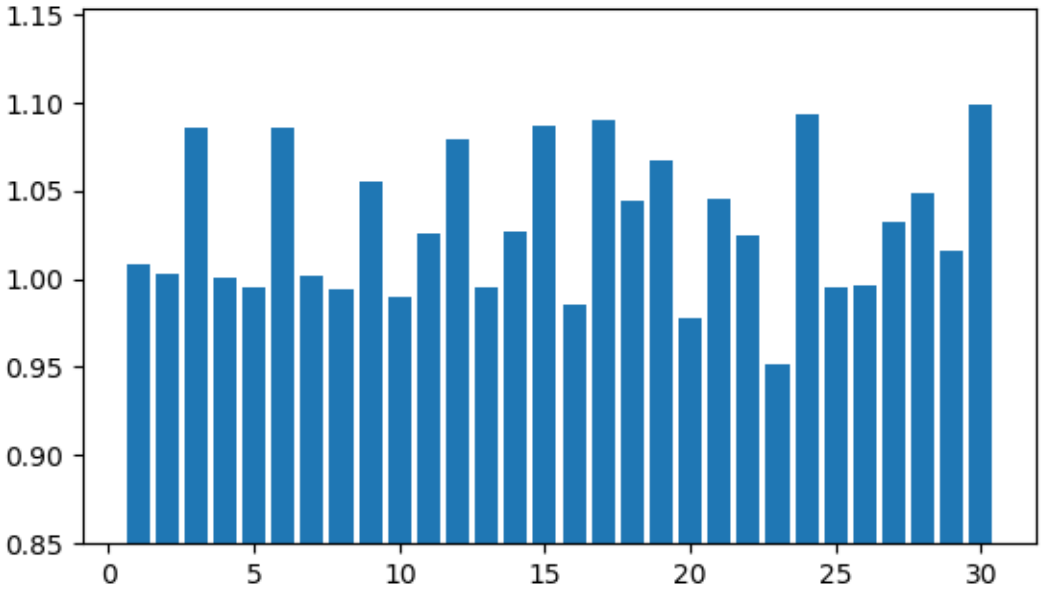
For the first block of the ciphertext:

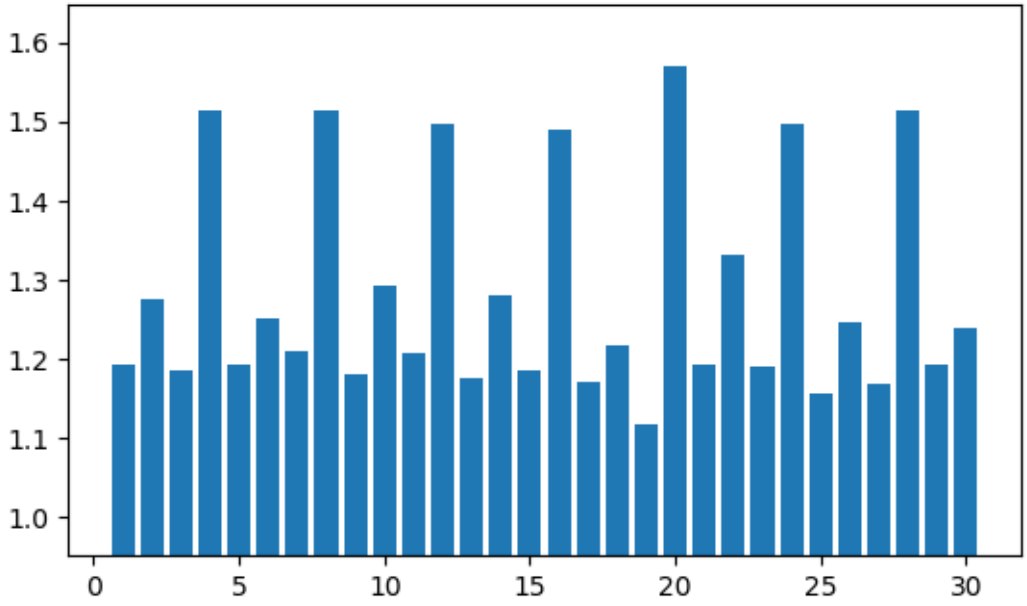
$$\mathbf{V} = \mathbf{M}^{-1}\mathbf{U} = \begin{pmatrix} 12 & 5 & 22 \\ 20 & 5 & 13 \\ 11 & 5 & 12 \end{pmatrix} \begin{pmatrix} 17 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 19 \\ 7 \\ 8 \end{pmatrix}$$

The matrix used in a Hill cipher can often be expressed as a keyword by using letters that are equivalent to the entries of the matrix. For the example above, the keyword is HILLCIPHE(r).

Given a ciphertext whose blocksize is unknown, it is often possible to find it using the index of coincidence. The same technique we used in Unit 31 to find the period of polyalphabetic substitution cipher can help here. However, the height of the peaks in the graph of IoC versus blocksize is not always above a predetermined threshold. Often it is even difficult to discern the peaks. These examples are for 2×2, 3×3, and 4×4 Hill ciphers. As you can see, it is sometimes difficult to determine the block size.







A more reliable way to find the block size is with the index of coincidence for digrams, trigrams, etc., provided that it is not larger than five. In the exercises from Unit 10, you found ranges of the values for IoC2, IoC3, ... for typical English text. A Hill cipher, since it enciphers each block always in the same manner, will have the same IoC*n* as English when the block size is *n*.

Hill had another cipher, in which the plaintext was inserted into a matrix that was multiplied by the key matrix. In this cipher, the key matrix was chosen so that it was its own inverse. An interested reader can see his second paper in the references below.

Reading and references

Lester S. Hill, “Cryptography in the Algebraic Alphabet,” *The American Mathematical Monthly* 36:6 (1929) 306-312, DOI: 10.2307/2298294, www.jstor.org/stable/2298294, web.archive.org/web/20110719235517/http://w08.middlebury.edu/INTD1065A/Lectures/Hill Cipher Folder/Hill1.pdf

Lester S. Hill, “Concerning Certain Linear Transformation Apparatus of Cryptography,” *The American Mathematical Monthly* 38:3 (1931) 135-154, DOI: 10.1080/00029890.1931.11987161, www.jstor.org/stable/2300969, www.cs.jhu.edu/~cgarman/files/Hill2.pdf

Abraham Sinkov, *Elementary Cryptanalysis: A Mathematical Approach*, 2nd edition, revised by Todd Feil, published by Mathematical Association of America, 2009; www.jstor.org/stable/10.4169/j.ctt19b9krf; chapter 4.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 404-409.

Wikipedia, en.wikipedia.org/wiki/Hill_cipher

Crypto Corner, crypto.interactive-maths.com/hill-cipher.html

Chris Christensen, “Lester Hill Revisited,” *Cryptologia* 38:4 (2014) 293-332, DOI: 10.1080/01611194.2014.915260

Programming tasks

1. Implement an encryptor. Allow for any block size. Verify that the matrix is invertible before enciphering.

2. Implement a decryptor. Allow for any block size. Remember that the key is the matrix used to encipher, so you must find its inverse.

3. Write a function to try to determine the block size. Perhaps it could return a value of zero if it is unable to clearly determine the size.

4. Implement a brute-force attack. For an $n \times n$ matrix, there are n^2 values to vary. You should discard matrices that are not invertible.

5. Compile a list of four-letter words that can fill a square matrix so that it is invertible. Do the same for nine- and sixteen-letter words.

6. Implement a dictionary attack. You should discard matrices that are not invertible.

Exercises

1. Encipher this text with keyword `HILL`.

   > This, and enough, premised, I go souse into my personal history. My maiden name was Frances Hill. I was born at a small village near Liverpool, in Lancashire, of parents extremely poor, and, I piously believe, extremely honest.
   >
   > (from *Memoirs of Fanny Hill* by John Cleland)

2. Decipher this text with keyword `PROFESSED`.

   ```
   SNSPTIRUZMCUZBWJZIJVGYIVSGJUUZYWDDWMDLPVWKTVYJPUGLPAGQ
   LLYBVHLIPLXXIKKGKAOHSDCBDSKSKXTTTMERLNKQFBNDBPULJUSHQR
   TWNSNSLWDPWGBAKKMZMOEPKGDGEABNTIARIAVQHMAZEKPGYKRXXXTT
   EKNSNSYMOISTGYXSSRJAKUAYVKKKPCXGYRHKCARPVPOIAQNYGGLVRM
   XJBUYLXVHSSRKIIRPAZVKVYJBKJFGSDYIGSCBDPBVRHCMTLIJXHJMK
   RPOCNWRZMELXHWBRWVZWXTJFHTKDWMHVYPHHXLDEYYDJZREVWQDAZE
   JOGHUXREVVGEDIZOTGELXOSJFUMVXCCPCZVJWMMKITKYZHIIXJMXRD
   XSBDISSHTZOGYNOZOGBLZUSFSNSWPCQAFPGFRAIMKLHABVGGUZNTJS
   HHRXSUVUUBZTYWDDCJVBTBKJEQDCQIHZNPPGSNSZDODCJPFICIAWHE
   PFKHRWOTGXJMDIZFXPDQQISQZHERGXTSSZEWNGEQKHTYIXJMTOGDKN
   LUN
   ```

3. Can you determine the block size for this ciphertext?

   ```
   IPKGRGNLJUQQGFTQMMYKABOCFYRBGEWEOMYICYODZYLOMJGFXXZDYN
   TOBIBZLTITEMDSQITHEDUUWUQJOJVAWIFKVXFUDCGFUTECOTWBSQLO
   CVCSEBSSNKMBJFRGECMIERCGEVAXSBLVBZEAQNIHGHJZGBJMIJJQHN
   MAKHJCBIOZRDTAZMJDSWORRBHXOOCFDBMWNYAPXZASTVSXIOFITERX
   WVQMLAFSWXSTWKUVWSEFOIBKUQIHZAELVRPVUQQSGBPCZRKTVAKBLK
   ZXZMAWMNUHBMOLUWNIMUMICASOLWHJVLEKONNGOFCDKKTQSGEEZELI
   BCGDCBCIJLEOHEJODPHRISLZVAWZVJJZXJFHESLRYWDITGQUMERAFO
   XEJUWGCSQYGGJYVAZPJVHMPGYBMKERBHCFJUUUUJGPGICASFJHWWCN
   TEMAYXBUQZVDYDGHQQRMGBWBZKMRAIXDUYMTQXDRMQZVMTDZUYTRVK
   WIMPVAREZPRLJIURMIDPAMSDRQZOZRIARCPJPLHWWICHDRQCWVILQK
   AIMXMYWVEJRQZOZDUXOOGYXNNOMZXRMYJYHGRLFWJWPJHNADQCTXFR
   EHIJDNXKZONJTWWCRYESVCGWGVUWALSSPQACCGEVAXSBLVIERGK
   ```

4. Use a brute-force attack to break this ciphertext. Is there a keyword?

   ```
   TOPGRHYADTVKXXSQYQXZUBWLGFALFCMGEZUBEBHGQKWODTGFYUBIMO
   WJFVKXEVFOVTRYCNXWOJYMSVZWPLANLUCCTOKEUIPVWVSOJWURXAFV
   CUMPCUQJDEWJFVSNEVJQEFIWJHQTVHYASVFVSQZWFVSAMROUTOHBIG
   WSFVCUQJHECURMNQEZFVIGPCMYURTOLQLBXWOYOFZSSAGBITRYAJSA
   CRFIHEANLIANGMGPZWAHQWFCDIVGQWDTZWPYDTXWQAYWRMSSZWXWIK
   KRZGGDJK
   ```

5. Use a dictionary attack to break this ciphertext.

RMYKUKRLHLXETRSMCYXFOUYMHHNOJAMEERFEDXQRXOWOSIEAPXFDOR
ZBCZZHLNBRMYRRGVTCNOZNYSOHYDORWPKRHPCEJEKOPGAJTDRPERMY
VZUCHCDORSWGAYJMJOVBULXRVXTGKOZNHAQDYZHKYRMNXKMCMIREMZ
ITKDDLYRYWGDJZUZNXCEJKALYRYCEEDMTBMAQYUHHNWJWCEEDMTHMK
REIZXYOCQEUHSMWJLYTTQDKSEUHZNHPXFWGDOVAVJSBROHYBYUCFOC
IKHXIKPSLUJYJWWAWUHQKSAGXXMHHNMJOOINZNHROZUMKMCYRBEMLK
VTCAJUJZUBRODKSFEBJZWPBUMJOYGNGGCNTZZBSJZUHYYVEBOWNZUL
TPRRLHJWWAWURYPKRKQSEAJUPUKITIYFYHYBQBEQBQQYUUUYUFCRLH
WFGVXCYKCBTRHRSAUEZNXKKYBTRRMYXISLBNSQJVNZPWGJLYKXMGVU
DPQRHPQAFXISRBXWOSIEAIHUHYBDGAZMZSMYBHXLGUSTSIKHCVOKKO
PYDDORZCVOLXYWBWGDRLUXLRYWNLWBZNHOSTZNXGWKZREYMTNDKOMA
FSIHRADORGHSOAFZWQMRMLSSWXWOFQZGCECLPAPROXSEIMLL

madness's book on classical cryptography
unit 88: attacking the hill cipher with cribs
last modified 2020-08-14

Unit 88
Attacking the Hill cipher with cribs

An $n \times n$ Hill cipher has n^2 parameters, and therefore we need n^2 constraints to completely determine the key matrix. These constraints take the form of equations that we get by matching a crib to a piece of ciphertext. If the crib is too short, or some parts of it repeat, then there may be too few constraints; in that case, we have to brute-force the remaining parameters.

When we are matching a crib to a piece of ciphertext, we must remember that we can only use parts of the crib that cover complete blocks. Let's return to the example from the previous unit. If our crib is `MESSAGE`, then we move the crib over the plaintext and try each position until we can find a key that gives a good plaintext.

```
MES SAG E
RBC|GUG|KAG|EQY|GQM|IXR|DEV|ISN|ZAV|OAD|FDQ|TST|BCL
```

In the above position, the constraints from the first block are (remember that the key is a matrix **M**, and 'A'=0, 'B'=1, ...)

$$\begin{aligned} 17 &= 12\,M_{00} + 4\,M_{01} + 18\,M_{02} \\ 1 &= 12\,M_{10} + 4\,M_{11} + 18\,M_{12} \\ 2 &= 12\,M_{20} + 4\,M_{21} + 18\,M_{22} \end{aligned}$$

From the second block we get

$$\begin{aligned} 6 &= 18\,M_{00} + 6\,M_{02} \\ 20 &= 18\,M_{10} + 6\,M_{12} \\ 6 &= 18\,M_{20} + 6\,M_{22} \end{aligned}$$

From the third block we do not get any constraints, unless we brute-force the next two plaintext characters. The equations we get, where x and y are the missing characters from the third block, are:

$$\begin{aligned} 10 &= 4\,M_{00} + x\,M_{01} + y\,M_{02} \\ 0 &= 4\,M_{10} + x\,M_{11} + y\,M_{12} \\ 6 &= 4\,M_{20} + x\,M_{21} + y\,M_{22} \end{aligned}$$

Collect the equations for the top row:

$$\begin{aligned} 17 &= 12\,M_{00} + 4\,M_{01} + 18\,M_{02} \\ 6 &= 18\,M_{00} + 6\,M_{02} \\ 10 &= 4\,M_{00} + x\,M_{01} + y\,M_{02} \end{aligned}$$

From these three equations, we can solve for M_{00}, M_{01}, and M_{02}, for each choice of x and y. We then collect the three equations for the middle row of the matrix, and the bottom row. For each of the $26^2 = 676$ choices for x and y, we get a complete key matrix (unless some of the equations are not independent and so cannot be solved). We invert each matrix and decipher the ciphertext. Unfortunately, for this position of the crib, none of the plaintexts that we get are acceptable.

Next, we move the crib over by one position:

```
 ME SSA GE
RBC|GUG|KAG|EQY|GQM|IXR|DEV|ISN|ZAV|OAD|FDQ|TST|BCL
```

Now we must brute-force the first character of the first block and the last character of the third block. The process is similar to the above, and again we do not find an acceptable plaintext. We can shift the crib one space at a time. When we do find a good plaintext, it is in this position:

```
    ME SSA GE
RBC|GUG|KAG|EQY|GQM|IXR|DEV|ISN|ZAV|OAD|FDQ|TST|BCL
```

We need to check all possibilities for the first plaintext character in the second block and the last plaintext character in the fourth block. For each of the $26^2 = 676$ choices, we set up nine equations. The equations form three sets of three. Each set allows us to solve for one row of the key matrix.

Programming tasks

1. Implement the attack. Build an attack for 2×2 and 3×3 Hill ciphers, at least. If you are very clever, perhaps you can build an attack for any block size. Be careful in handling cribs of various lengths. Use tetragram fitness to determine when you have found an acceptable plaintext.

Exercises

1. Finish the example above by hand to find the key. Check that it matches the one we used in the previous unit to encipher the message.

2. Break this ciphertext with the crib `FOLLY`. What might the keyword be?

```
GHTSYUESFCYMNNBGCGESCNGHAXDULTNVXOYRWPWTKPGUNKAWNPDVES
BBRFNTRNBGOWXLWTYNDPNSDLEYGUBBEHYQLAGOGJCNGIYUWTEULBBB
REYVZVBGCGESAEDNCQLHDLZZGRXOYKGUNKAWNNNSCNYBYCOLYRYWLI
AWNLXOGULXGBDNLGYTAEXWGJDLEXYQFBWDDLJHBFFXNNNSDLERDDLH
QGXREYWTKBWQDGGPGULNGJDNBBREEXWKMTOIZBTRMUDPYSWXYQNSUW
```

```
WLYDMOUUGSYJAMMLAYMOUUGHTCWTKVDNPDOLYUXRCILNGJESYVGVKE
FWDNEACILWLZNXTCGYWMEQNQWKZBCILULJUVDAOCWGEOGUNKAWNSGY
YNGIXWYOXLNSXRCXANFWGGYBGHTCWTKMYJLJUSGIDVUU
```

3. Break this ciphertext with the crib `PITILESS`.

```
GQTIRDBVCMCVUCJCPZFCSPEOFOPRBDTOQGQAHWBXNBWLEWJCNUBRMR
ETIEVCJSKZLCUDWHWBGESEPEVBRGXWCHDQSFAHWNRUJMHTRRPAAZGY
GEZTDGDLELCSEMNLOCGESEPMNRUAFKGQTUHGHSFCPEIYGHCRDCSIHW
RTIOJVKAXSRIGSFZMKNAZWMVIRDVUHYDBPEUPEVRBDOFSJVCHSFLCS
LWFRBDREQCRUKPMFDHAHOEMNZRJQLYYGHGFZSQHLAQBPQTQHGFSLQB
WKHOFNMACHRFGXEKUYYDBFYZVJGATEDYGGXERWCPPKGISLAALBOCST
CIYIVFMKPUSKCEGFYWUWSQMAYALQCEKMWWFSNSWMCVUCJBQDUWCKLS
YDBTUNDKPXMKEIZAFKIBRDXAGKE
```

Unit 89
Affine Hill cipher

The *affine Hill cipher* is an extension of the Hill and affine ciphers. We take the Hill cipher and add a constant term to the equation. Of course, since we are dealing with actions on vectors, the constant term is also a vector. When the key is a matrix **A** and vector **B**, a block of plaintext in a vector **P** is enciphered to a block of ciphertext in vector **C** with this equation:

$$\mathbf{C} = \mathbf{AP} + \mathbf{B}$$

Decipherment is done by solving for the plaintext vector:

$$\mathbf{P} = \mathbf{A}^{-1} (\mathbf{C} - \mathbf{B})$$

This cipher can be factored into a Hill cipher followed by a Vigenère cipher. This should be obvious from the first equation above. Furthermore, if in that equation **A** = **I** (the identity matrix), then the affine Hill degenerates to a simple Vigenère cipher.

Programming tasks

1. Implement an encryptor. Remember to check that the matrix is invertible.
2. Implement a decryptor. Remember that the key includes the matrix for encipherment and that you must invert it.
3. Modify your brute-force attack on the Hill cipher to accommodate the affine Hill cipher.
4. Modify your dictionary attack on the Hill cipher to accommodate the affine Hill. Use one keyword for the matrix and one for the added vector.
5. Modify your attack on the Hill cipher with cribs to accommodate the affine Hill cipher. Each equation will now have a constant term on the right-hand side. There are n^2+n parameters for which to solve.

Exercises

1. Encipher this text with the given matrix and vector.

$$\mathbf{A}=\begin{pmatrix} 21 & 3 & 0 \\ 8 & 8 & 19 \\ 13 & 2 & 4 \end{pmatrix}, \mathbf{B}=\begin{pmatrix} 5 \\ 7 \\ 9 \end{pmatrix}$$

   After marriage arrives a reaction, sometimes a big, sometimes a little one; but it comes sooner or later, and must be tided over by both parties if they desire the rest of their lives to go with the current.

   (from “Three and—an Extra” in *Plain Tales from the Hills* by Rudyard Kipling)

2. Decipher this ciphertext that was enciphered with the given matrix and vector.

$$\mathbf{A}=\begin{pmatrix} 8 & 19 \\ 21 & 18 \end{pmatrix}, \mathbf{B}=\begin{pmatrix} 23 \\ 11 \end{pmatrix}$$

   ```
   VSWEZMIFVTXDROVWQYJVVKQRZMVSWFWQXZJFFESTHETMQRQYXQHCID
   BCTUNNOPNPKLNBJIIFRUQDVPVAGBEYDFUPWQWJNVWXTMHCWFXQXJIF
   BLZSLZNUISVTXDROVWQYJVVKQRPINCFJHDXQGPBLDWONQNTUUCKLOO
   FBWXHDMPRVDRCAGWBXDRCABRTMNBHICAUBQRMFOLZWBCLZZMNZHUMC
   SLOLQNBVCYOLCLJWRJKLZMJGXUPONJGNPBKLVVMJDQHEGANJNAIDNA
   HEKLOOMFOLKRWVBEWXUPQRTQAINTDWWDKGBXXQST
   ```

3. Break this ciphertext with a brute-force attack. What are the keywords?

   ```
   HFPQQDGRZVKFKUEMZTGEUZSIUVNGRFBCWNHNGBGWUOZAGEGIWZGCPK
   NZBHZRBKEJTZZFXXWNSDZCUKSQUKGUTXZRGOGWWIIHGEGIPQPXWZKM
   OEJRPQWNXPGKZZHBUUBDHRZJSYMGPNTZKMZIGEAZCQBFZVTZKBZIGE
   GMSVHCHNWTHPDAFPBRNPGTZGJRLZAZZYSYYAHNSVEKHPPQPXWZKMSC
   HPCVZVZVTJJGKGZDGESIGTPZHBUZJOTHZVCNAFGVGSSMAMCBZGHNPD
   WZGOJOHNGBWAHPSMDEFBDPHNUHUCNSGQHNKMIAUFNSKGSUUZGOOMQD
   UXNGTJWG
   ```

4. Break this ciphertext with a dictionary attack.

   ```
   QJCSKYUYIUASSRXELHDERHZYKTHNQMJDIHYIAKLZBCKYQXYMVNKILM
   CFIRYOZZILKMCTRNDAONWVAUPAPJKDBYLWKQOLYHNSKYOUGDALEKJT
   JNMQJJTFGJKDOWNXOAWLOGTXTEDSLLUNTQDZNNSGWLDQGQQTENQTTU
   QCOVQULXROLPWCOUGDEJWZASZTOQYQOESIIWYXPPOAIWHQLZQHSLQR
   CVYJDGMTMOGPIBRDYJFAVGWDTOAWJRSDATSAIEIHSEQDIXBHTDMTDA
   IYKTLSVKAQGFYYWRNAYWYKTJQBSCQFJYGRATUGWLKLAUQ
   ```

5. Break this ciphertext with the crib `AFTERNOON`.

   ```
   KGMJEIVFJJGJHCKOGBRDZRHBUETXPIWNHUVFEWPTHHDXODTKUCEWUO
   CITNOLEYCEUOUNHFRFKIXNPXIGEYKOCLHUVFPLWMETRIVFXZDGUHCV
   ```

```
HYWKTGCOURTTPHEDHYDZPBVHWNNPELPUWFHYKOGJTLDXTQDLMHHYSX
XNNNSXHHTGCUWFHYMEKGCBDPICETHUVFEYWFDICUETRIVFVRDGEMPT
XZHHTGCTPXVTNDMLUVPLEFTGCKKNDIEDHYNDMWTGCNPVEFQYWXNSVT
TQUTCXTGCIUZCLRFTKCHTZVFHNTEUREPEIVSWFGGHFPAVCWWSTPM
```

6. Break this ciphertext from the 2013 British National Cipher Challenge with the crib `TELEGRAM`.

```
HCHKK MEYXE YMEZW RLFMS RPJSU BPWHR PMDEY XWHRM IQKEC
JNBCM CJFHY HSELD JXTSH IGEXH BMBIB GJRCN VAQJX FJYHL
LQPAF DJHAO NUCJF LRXTS XKTNQ JATUU VBGSH OHKOC VXHRZ
WWHFY EEZIU DSJFJ IBGQK XSUIN IFDPI HRXVZ BGGYT SXTPV
NETDB QMZAT SXRDW HLXKT EIWAZ WRTSX JWHSH VSIGE YRKBZ
WJQYI UVPTU VFMPR SOTWI DTEXL NMKNB YYMTV JIJWR JQFLZ
WJDHM LCVXH RZWWH FYEQH XMIQK ECUZK PJSBF KXVAY NZSUY
AJRRJ QRDWG DHENG CMRNR RBOVU FDUAE PRGAU QTBWN PLOJV
RLSVP TNBPI KHOQL PFXOT FMCWM KIRFG YNVBZ AXRS
```

Part VII
Stream ciphers

Unit 90
Stream ciphers

A *stream cipher* enciphers characters of the plaintext one at a time. It generates a *key stream*, which is a pseudorandom stream of characters. Each character of the key stream is combined with one character of the plaintext to give one character of the ciphertext. To decipher, the same key stream is used to recover each character of the plaintext from a character of the ciphertext. If the key stream is generated independently from the text, then the cipher is called *synchronous*. If the key stream depends on characters from the text, then it is called *asynchronous*. If that dependence is on the last few ciphertext characters, then the cipher is *self-synchronizing* (or *ciphertext-autokey*), because the receiver of the ciphertext can resynchronize after reading a few ciphertext characters.

There are three variations on how key-stream characters are combined with plaintext characters. Each corresponds to one of the polyalphabetic ciphers that use unmixed key alphabets.

- Key-stream characters are added to plaintext characters, as in the Vigenère cipher. This is the standard variation.
- Key-stream characters are subtracted from plaintext characters, as in the variant Beaufort.
- Plaintext characters are subtracted from key-stream characters. This is similar to the Beaufort cipher.

Whatever information the encryptor of a stream cipher stores as is goes from one plaintext letter to the next is its *internal state*. Typically, the key is used to generate the initial contents of the internal state. The state changes as the encryptor advances through the plaintext. The key stream is generated from the state.

In contrast to stream ciphers, ciphers which act on a block of text at a time are called *block ciphers*. For example, periodic polyalphabetic substitution ciphers and permutation ciphers take blocks of a fixed number of plaintext characters and encipher each block in the same way.

Reading and references

Wikipedia, en.wikipedia.org/wiki/Stream_cipher

Exercises

1. Decrypt this ciphertext which was encrypted with a synchronous stream cipher, and the key stream is based on the Fibonacci sequence. (The Fibonacci sequence in modular arithmetic is periodic, by the way.)

```
ZPWFFVUVDHXCQRRTYGDYHFWECMWAHGIUWVZBCDLXHTGZNLIVNSVQWM
UVGCTSFPSZOXPHMYKAAFVLJSYBTSIAUMGDUANILUFBRBHDCDCJPJGL
ADGFFSBVLJQWOKGOFSSNWLPSYQLETFJZSXWYQZTCZMLUQEHZLMWSQB
ZPVRUAZTUWNTYVVGWSSFKNFNJYHNSQHSCVUOIKGPFIIGEBLERBYKWW
CQSVTPXVHIBEADNBRQEFDNSQGBZDTSRBKSJSBTHUJAUVALQBNBZDSK
YULFDOSVWGIWIEWABZZ
```

Unit 91
Trithemius cipher

The simplest stream cipher is the *Trithemius cipher*. It is synchronous. We can look at this cipher in three ways. First, we can say that it starts with a shift of zero and adds one to the shift with each character. Shifts are applied to plaintext characters modulo 26. Second, we can say that the cipher generates the key stream `ABC...XYZABC...XYZABC...` which is added to the plaintext to get the ciphertext. Third, we can view it as a Vigenère cipher with key `ABCDEFGHIJKLMNOPQRSTUVWXYZ`.

As explained above for general stream ciphers, the Trithemius cipher has three variations on how key-stream characters are combined with plaintext characters. In addition, there is another variation in which the initial shift is taken to be nonzero.

Like the atbash cipher, the Trithemius cipher has almost no security. Once the adversary knows the cipher, s/he can decrypt any ciphertext.

The internal state S of the encryptor is a simple counter. As we have done before, we can think of the plaintext $P = \{p_i\}$ and ciphertext $C = \{c_i\}$ as each a series of integers. In these terms, the action of the encryptor for the standard version of the Trithemius cipher is

1. Set $S = 0$
2. For each p_i in P
   a. $c_i = p_i + S$ modulo 26
   b. $S = S + 1$

Reading and references

Wikipedia, en.wikipedia.org/wiki/Tabula_recta#Trithemius_cipher

Johannes Trithemius, *Polygraphiae libri sex*, Reichenau: Joannis Haselberg de Aia, 1518, www.loc.gov/item/32017914, book 5.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 135-136.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter XI, section III.

Programming tasks

1. Write a function or script to encipher a plaintext with the Trithemius cipher. Give your function some optional arguments to allow for the three variations on the method of combining key-stream characters with plaintext characters, and for the possibility of a nonzero initial shift.

2. Write a function or script to decipher a ciphertext with the Trithemius cipher. Give your function some optional arguments to allow for the three variations on the method of combining key-stream characters with plaintext characters, and for the possibility of a nonzero initial shift.

3. Write a function or script to break a ciphertext that was encrypted with some variation of the Trithemius cipher. Use tetragram fitness of the plaintext to determine when you have found the right variation.

Exercises

1. Break this ciphertext:

```
PVPJXYQFHGYKZXGDQPUZSDCXKMZPVJCYAJGIEIEHHCEPUZYIQPIFTK
BXYVDRTJWVAGUWAKCCIDVLRJGIJNISQFHPOVVMBZNBYMDUCXKMZAPN
WZZXGWENLHJVNQWCOXPDRIMLBFSETGVTDLFLCICOYSIUWGIKUYPRCS
QYDLYKZFIMWICSHFAHPJIAMMRTYNPQEKLHIHQZFOASVBMSMSMJNEAU
WHTVSMBVJAFDWBUEYWOLLJDETROQRVQ
```

2. Break this ciphertext:

```
HSUFOWQFNYJLEJOTVJGDYNRPJITVKKLXSFFZIMKWMSCVVLZBRBLXAG
QKQVCPQXZLKAKUMUPZSSMFKIWNTJFRXTGCFLVGIZWWUHVQXDBXHRWF
CUCAKYWWGZZJCUBDPYELCGWIAMWEOPINSXZNKHJPPGCDEJKXMUFQSE
QNYGQYGUOXFFVBVXMWJXDGQEODQTATXBNWILAIEPNQNDAISXUPECGG
OWMYUVILISKSNRYMCMTRWFYDBPGXDWGCESDJQGCBRZFBORLTTVHQZN
YWKKLNRCEFSTSJPTHBKSDRHLZYQFKVOHKSQOSVRPGTVZWPHOQCKZYL
MTVHKWFDLBRKVXNGRTFRMBAYIEDUFHVVQPYEWSUICGRXDVNLPOMCSJ
MBXFQPYSWKGTPOXTTNCREWMYKKEYRCECMMKMYBNWHTJSQCKYICWJKR
TF
```

Challenge

```
TAEVZRVZLPSLZAERFZRVDQMMUNVIOPNGNRCKDWHLOCQHZNFMQMQDWXXZJCQ
JQUNNOZDKKVAHVKSFLDKOAIOVOTBORRESLSWNXBZNAIBPGGZZMKBBUYYZNB
LZEVPAEWBPGYFZSLTGMBFTITXRGRZJNGKVZRTHWASLSASLTFJBIMRFZMPPA
IBUFJKYRKVZNAIBXEMOVXBSLLTGRPLLDQEXEIZDEINBSWWJMDRLYBUBFSWY
IWBUBFGUNGRVRRJJYYNRITTUYJFNGNGNRNBSVNQHPCMYCDHHBUCHBMXVKBJ
WPTWKXFYGTEEVBFQUNGKBT
```

madness's book on classical cryptography
unit 92: autokey cipher
last modified 2022-01-13

Unit 92
Autokey cipher

The *autokey cipher* (or *autoclave cipher*) uses a key stream that begins with a keyword which is followed by the plaintext itself. Hence the name of the cipher. It is asynchronous. In the standard version of this cipher, key-stream characters are added to plaintext characters, modulo 26.

An example will help to elucidate. Let us encipher this short text with the key `AUTOKEY`.

`THIS TEXT WAS ENCRYPTED WITH AN AUTOKEY CIPHER`

The key stream is written under the text, and the ciphertext under that is the sum of the two, modulo 26:

plaintext:	`THISTEXTWASENCRYPTEDWITHANAUTOKEYCIPHER`
key stream:	`AUTOKEYTHISTEXTWASENCRYPTEDWITHANAUTOKE...`
ciphertext:	`TBBGDIVMDIKXRZKUPLIQYZRWTRDQBHRELCCIVOV`

If the length of the key is L, then the internal state $S = (s_0, s_1, ..., s_{L-1})$ of the autokey cipher is the last L characters encountered by the encryption routine. The state is initialized by filling it with the letters of the key. For each letter in the plaintext, the character of the key stream is found by shifting the state vector to the left. The leftmost character is popped off the left end of the state vector to become the key-stream character. The current plaintext character is pushed into the vacancy at the right end of the state vector.

Let's rework our example in the language of state vectors, just to belabor the point. The state is initialized by filling it with the key:

S = (‘A,’ ‘U,’ ‘T,’ ‘O,’ ‘K,’ ‘E,’ ‘Y’)

The first plaintext character is ‘T.’ We push it onto the right end of S and pop off the first key-stream character, ‘A.’

S = (‘U,’ ‘T,’ ‘O,’ ‘K,’ ‘E,’ ‘Y,’ ‘T’)

k_0 = ‘A’

The next plaintext character is ‘H,’ which we push onto the right, and pop off a ‘U.’

$$S = (\text{‘T,’ ‘O,’ ‘K,’ ‘E,’ ‘Y,’ ‘T,’ ‘H’})$$

$$k_1 = \text{‘U’}$$

You get the idea. This continues until the full keystream is generated.

There are three variations of the cipher that correspond to the three variations of combining keystream characters with plaintext characters: Vigenère (standard), Beaufort, and variant Beaufort.

Reading and references

Blaise de Vigenère, *Traicté des chiffres ou secrètes manières d’escrire*, Paris: Abel l’Angelier, 1586, HDL: 2027/ien.35552000251008, gallica.bnf.fr/ark:/12148/bpt6k1040608n, gallica.bnf.fr/ark:/12148/bpt6k94009991

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; chapter XVI.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter XI, section II.

Wikipedia, en.wikipedia.org/wiki/Autokey_cipher

Practical Cryptography, practicalcryptography.com/ciphers/autokey-cipher

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Autokey.pdf

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 147-148.

Programming tasks

1. Write a function or script to encipher a plaintext with an autokey cipher. Allow for the possibility of choosing any of the three variations of the cipher.

2. Write a function or script to decipher a ciphertext with an autokey cipher. Allow for the possibility of choosing any of the three variations of the cipher.

3. Implement a dictionary attack on the autokey cipher.

Exercises

1. Encipher the following text with the keyword STREAM,

   a. with the standard autokey cipher (the Vigenère-like version),
   b. with the Beaufort-like variation,
   c. with the variant-Beaufort-like variation.

   > The deep hollows which separate the hills are thickly covered with fern and heather, over which blocks of granite are scattered in all directions; and, as in all similar districts, each valley has its own clear mountain stream, which receives the innumerable waterfalls descending from the hill-sides.
   >
   > (from *Devon: Its Moorlands, Streams and Coasts* by Rosalind Northcote)

2. Decipher this ciphertext with the keyword RIVERBED and the standard autokey cipher.

   ```
   KPZWYFIUKVGCZEGIFGUDZEUKFWSFJYLHWJFDQGMVJQNXVFPLVPOXLR
   WRUKXZQPIFLTXCBZVTIAQOEWACAAUNDKNSDFDASIOFUKTISHSBFUOM
   CPVGCWQSSXKASGPVWRYARQFGEOIXGLWEXOYEUSIFSSDRJYLHZTQKIO
   SSAGZXCPYGQCRFNSFVMCNOPWKYSIRFLSVTKVHLOEDPKQLGHXIPBANQ
   ELWZNHNRRSBOGTDSGHRYGLGIRHWOOYJPGFWESFQXMQEYGZGXC
   ```

3. Decipher this ciphertext with the keyword FUTURE and the Beaufort-like variation of the cipher.

   ```
   MNPWKEQCQEUXKYKMWTPJLUYDAMVWADNOKYAQMFGIPNLAPFADSFZWBL
   OVBRMAGATHOUBPJJHTPRDUSCZZVDASTVPLXYYMXWEGYHPWAJQWVWLA
   INXQDAYLTZWMERQPBDRW
   ```

4. Decipher this ciphertext with the keyword ELECTRIC and the variant-Beaufort-like variation of the cipher.

   ```
   EPIGUCJSLHHZRZUGCXVYPPTMFOFZYDAMAVKWVVKTCRATCJNNWPELOA
   LICUWOMROBYBKMICOVUIVOITZVJVSDAEALTPHTNWFUTSHGAOEFPCEK
   OAVGSNIPZMWKLBOSDAEATQZDNGQQQPBGIBBA
   ```

5. Break this ciphertext with a dictionary attack.

   ```
   IVIDVWYYSWAFNTVFDLJQVDAELSGSMSXEEMPWGRGWPQTPOFEACINUPH
   QYEPXMZSVVHPYXBROOHKVWOXNYIXWWLASKMEYEWXLCUPBKAWSHTALF
   WBMJGJXMDBZWYWDEMZMPAJDVTBQJTHNXQJTHVORWGUIYGKLHSPXKPI
   MUFPNWNLPBYWUXXKXQPRGLGVWATPKGSOKEEROOXAAQQNRWSUZTRWSE
   NRMVPASQHBDJSAFRJYHIEJCHDEEXTBOXFJQRKTGJQQHLWSAPPTWIWV
   ```

6. Break this ciphertext with a dictionary attack.

   ```
   PVALSAXRIUTGBRNFNMPKKRATZLWEMOACVIFCJDVZNABCNIMLXYIAHS
   OGPTEAGKLSKMWGSVBKYKTCSDPIRSSWKPJTFHCEXGSVQTBCHQJVZGEW
   ```

CVCICDXYBQLKRWLXJMKDWRKZVSKXOPVXQQCWKFAWDKHJTDNOULXXNM
GPVRQUZFBZRLMVGDNPBHTYWQNNAABKFLATGRIGQHTCHGDHDETMWSEP
SSAJZERDTHXWTMRUQ

Unit 93
Hill-climbing attack on the autokey cipher

This is a modification of the hill-climbing attack that we used against the Vigenère cipher. Given a key length, we vary each character of a key in turn until we cannot increase the fitness of the decrypted plaintext any further.

The algorithm follows. The purpose of the flag is to exit the loop if it has run through all characters of the key but not improved the fitness.

1. set the best fitness equal to the fitness of the unaltered ciphertext
2. choose a random parent key with the given key length
3. set flag to FALSE
4. while the flag is not TRUE
    b. set flag to TRUE
    a. for *i* running from 0 to key length minus 1
        i. copy the parent key into a child key
        ii. for *x* running over A, B, ..., Z
            - set the i^{th} character of the child key to *x*
            - decrypt the ciphertext with the child key to get a plaintext
            - calculate a new fitness of the plaintext
            - if the new fitness is greater than the best fitness
                · set the best fitness equal to the new fitness
                · replace the parent key with the child key
                · set the flag to FALSE
5. output the parent key

Reading and references

Practical Cryptography, practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-autokey-cipher

Programming tasks

1. Implement the attack. Allow for the possibility that the cipher is one of the three variations. Feel free to copy and modify your code from the analogous attack on the Vigenère cipher.

Exercises

1. Break this ciphertext:

```
KPDWZTEEHVSSJGUUHVOTIWHZLALTAIEKBHSLRAEHFSEDRRRDIWTFWG
SNPKAAGSWGFZEFDQLTNSXZEGKIYRLALTAAJOETUDPJEOSLRLJKPKST
AWAWLVUNUCHOMBMYFSGLAWYVOEETZQSCYEDBRIBBPXSMSQSBLWALMQ
AFLRNRRSHFJZULBIPTSPINDUGWALMQAFQRRVYRZYWVMGYGKHLTODPC
KZDKCSTURCPLYOWWLMIVJHZTBOPMTRPGAHFWKZRYNFYWRONRJVREAJ
KERAGHRFYHBPKFEPRUYEKFEGKIYRLPRBAIYSGAGILWPXZSLTZACVYO
OXHGFYGVAVTJLWFWEEWALMQAFYROLIZPWJHCMRBXYYMPAJKMAHXGHW
HWELMLIHHVEVMNFZEFWSYIXVDNPTUVOPDKQOAGHDFKPHSTRUMACMGU
EVHNUTOCCTONWKAVOECAEDTUHRHHPCMOMBZZXYTAMIKJRZEGWDFCCH
IEKWOHECXJLCQKAAWLASBZOIHTSPRSSAMSSJILSUTWDFDPWKPMJAVP
EVXMETUHPHXFVAFWLALBXCJTOEPNMJVMQOIAWALQVLRQVMOEUPFSXK
VMLACCJKCELHBRMGHZXGGVMNF
```

2. Break this ciphertext:

```
OTDBPBEBOCNBUEMOEGMSNQNPNYUMXWKPPMDSDEHZMDMTRCNAYAMGUJ
UHBTNLMKPQIWJVEYNRMGWMUETBJPVLEQBHPZARJJYUXWZQQYXSAOTP
RZPVEYOAUAXXYDIBZNMDXEPPTJGFLGOFZSCCMETIPZZTTDHHICGDNQ
FSOCYNIWKEHWSTDVMKZMYWGNMLNYLCFSQLCCKDCWRGYKHOHZEJBTQJ
RWUKBODTYIYDTSMNUSHSNLPKFNQCCUVBQORDZSTEVJXZAWOALSILAI
UEDF
```

Unit 94
Attacking the autokey cipher with monogram frequencies

This attack is a modification of the attack on the Vigenère as collection of Caesar shift ciphers. For the autokey cipher, we are unable to find the key length from the index of coincidence, so we will have to try various lengths until we find an acceptable plaintext. We will partition the ciphertext into slices, each of which (if we have the correct key length) has been encrypted as an autokey cipher with a key comprised of a single character. For each slice, we try the 26 letters of the alphabet as the key and choose the one that results in the closest fit between the monogram frequencies of the decrypted slice and the monogram frequencies of English.

Programming tasks

1. Implement the attack. Allow for the possibility that the cipher is one of the three variations. Use the cosine of the angle between the vectors of monogram frequencies. Feel free to copy and modify your code from the analogous attack on the Vigenère cipher.

Exercises

1. Use this new attack to break the ciphertexts in the exercises of the previous unit.

2. Break this ciphertext:

```
PLEKLARLUSGSOSAOWAWLQWGBLSBLUFPAVSGNAPDLIPLHTGBCARDAQE
IVVJGVQHVMQOVGKIPHISELWDWHWFFTWJFRHIPYIGNGOSGFZABUKNIK
FQAIFXBFVRTXKMPBPUPPRVSCZEVQAEUXZBXNHWXZHBOAVHWNKALQDC
GEJWOFRMSRMNFRPVNMGJOIMAKMYAMNHXMOEKTOKWVSISRXUQWJITEJ
PWOSQTGSHDSKNSXMPSDCASLOZNLKFAZTNHJJZEQXIYPFYOGEOVGRYL
ENSSPDNSPKHTWJMNHAMCYMOSLHIRORSMOKSMHBUDMVQGKMAGNLQAQQ
AZOSDEEGPPPZGDMSQEPTPILMWMVVVTVVFVQZSVK
```

3. Break this ciphertext:

```
MZHPWWXIFYIIOPPVGLADUZNZRQWWVJDBJRRNIWAYAKRXRULWRCZLUY
CKLLOWLMLAWNUOECNAKGZMONJAQZEJAQEOOGFZEHEDZPIRGLLACCYZ
```

```
KVZLUYXUAIKLPFXEKKOHZVBDRJUVIPVWEZDCLGOULRRYQIUAWARYME
HWKLWOTAYFOQWJPGYTLQIEFRTOSSWGUVNNGODRTGGJMKPMASCXAWYI
BAQEJXJCBGORUWRDUBPWCGWNKBNPXNEKKRRVRJRDWXRUPLLOWLYQEU
QXVRWFSABAMQFKGBACPMDIVTCHGRXWHMPXASWDLWIZXAXOQZMZARMS
AIXFXQUPHJOTKUFKGVKEXKWMQUWYBDDYXPW
```

4. Break this ciphertext:

```
MEKXVYDNJQTQFAFWUXIUNNLJTADQMAGWBDTNVGONJNOAHNMRDZCEUS
SCSELQTZAETCPALNSBNPLZMYMKXDHZYNBYCOXFBOLFIUZPVWMWNAAP
PKHLOCSFPNHRHIEWAVPJUYEDNIHXMJUQKDTK
```

Unit 95
Running-key cipher

In the *running-key cipher*, the key stream is another piece of text, usually taken from literature. Sometimes, it is a plaintext that you have seen before.

Reading and references

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 236-238.

Auguste Kerckhoffs, “La cryptographie militaire,” *Journal des sciences militaires* IX (1883) 5-39 and 161-191, www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf, www.petitcolas.net/kerckhoffs/crypto_militaire_2.pdf, part III.

William F. Friedman, Methods for the Solution of Running-Key Ciphers, Riverbank Laboratories Department of Ciphers Publication 16, Geneva, Illinois, 1918, www.marshallfoundation.org/library/methods-solution-ciphers

Programming tasks

1. We would ask you to implement the cipher, but you can just use your Vigenère functions with a really long keyword.

Exercises

1. This ciphertext was enciphered with a key that is also meaningful English text. Recover the plaintext and the key.

```
SULTFZHWTFUQRFTVMXCZOAAUARHGPACVPRPOXZEVBNUHANEHMWGKVU
DMCSTUHYVVEVFMNWZMJVJKLMLPALXPRLOFLNWGGYLHGLRXGGNAVTCU
NEMDAEPZRBZAAIBXIPSBUZHRJKYHSXBFWPQPXKAVTCTTBDHTQYUKKM
KOIWXXUHAZPGXYRMJKARNNWPDFJDIIPXBEVMDOWVEWGZBUXNPJZPUN
CMACRDNILMCANTHIAWALGFRBEVRRUZLGTJGOXBCSMLXYEPQRRCEUCA
```

INXSEWXSWTRCJQEZAHXZRWHPOKXLXMEIVVZMBGZJPLWLAWSIIPHKZA
GIMGGLRXKUTGKICNIEBXYSIMML

Unit 96
Progressive Vigenère cipher

The *progressive Vigenère cipher* (also known as *progressive-key cipher*) is a synchronous stream cipher. It is a modification of the Vigenère cipher in which the keyword is shifted by an amount each time it is used, as with a Caesar cipher. The shift is called the *progression index*.

An example will clarify things. Start with our usual plaintext and encipher it with keyword `KEYWORD` and progression index 3. Shifted by 3, the keyword becomes `NHBZRUG`. Shifted by another 3 it becomes `QKECUXJ`. You get the idea.

```
plaintext:     THISMES SAGEWAS ENCRYPT EDWITHA STREAMC IPHER
keyword:       KEYWORD KEYWORD KEYWORD KEYWORD KEYWORD KEYWO
shifted key:   KEYWORD NHBZRUG QKECUXJ TNHFXAM WQKIADP ZTNLD
ciphertext:    DLGOAVV FHHDNUY UXGTSMC XQDNQHM OJBMAPR HIUPU
```

We can attack the progressive Vigenère if we know the length of the keyword and the progression index (or we can try values until we find them). With these two numbers, we can remove the progression. What remains is a Vigenère cipher, which we can break using the techniques of Units 34-38. For the example above, once we know the key length is seven and the progression index is three, we can subtract the progression as follows:

```
ciphertext:     DLGOAVV FHHDNUY UXGTSMC XQDNQHM OJBMAPR HIUPU
progression:    AAAAAAA DDDDDDD GGGGGGG JJJJJJJ MMMMMMM PPPPP
new ciphertext: DLGOAVV CEEAKRV ORANMGW OHUEHYD CXPAODF STFAF
```

Some special cases:

- progression index = 0: Vigenère cipher
- key length = 1, progression index = 1: Trithemius cipher

Reading and references

American Cryptogram Association,
www.cryptogram.org/downloads/aca.info/ciphers/ProgressiveKey.pdf

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.
3. Implement the attack described above. You will need to try all possible key lengths and progression indices, until you find a good plaintext.

Exercises

1. Encipher this text with keyword `TARGET` and progression index 25.

   > We now take it so much for granted, we are so conscious of constantly progressing in knowledge, arts, organising capacity, utilities of all sorts, that it is easy to look upon Progress as an aim, like liberty or a world-federation, which it only depends on our own efforts and good-will to achieve.
   >
   > (from *The Idea of Progress* by J. B. Bury)

2. Decipher this text with keyword `REASON` and progression index 2.

   ```
   ZRHMANGVTIWGZAWPZVOOLMLXQQISKIJGXQRXFQEYAKNFGZTVTYDBLD
   EEFQOYDQIXMCNLQSSWXVQIQCMMCWHBLKPMUHPKLWTVQDKPGXMFQNOZ
   OHSGLZHMAYLCSGHHGQYIWZIVBQFVKXDGNIMYRALFYPGGQIUYIIAKCA
   TTFNXTDVGNWRPSTEJEDMAUNZSSFGYIHTESMDTTXKEUABQBGRDHVZRJ
   LQYRWYJIPVYDICNYOELVEGTGAKUCJJVBMKFWOYSUBBBTIHADHVDGLB
   GLUXAKNLMSKHONVTSZCDWSSUCMPQULATYCEUSDCDQTKLEJFRFKOVUG
   ZLRKJDBKYSLXCZWRZZTWBGB
   ```

3. Break this ciphertext.

   ```
   HKOTUMFYIJMSACEODTFEMRQHXNHOCXUARPBHNCZFIHLFDGCFHFAXBJ
   WYIAEJNQSFGDGUWKZNMXPMOQPPAITGTPJKJSGAYZZIQREBSTEFKIMW
   SWNSCNTRRKXKYORQWGXRVGJUNYYOIMAGFZTFIWSUUOWXGTQFHZMJYQ
   WCQEWACGGASQGHOXMBXCQGVGFVOHRISHXQYYJCDMBQASRAIRQZZZWM
   MAYFHJDBMDGUTCHOETWCKIOPUZAOAFUOOBZNDGZJXQWBVKYQQTLBEQ
   UPUJIUWVGRTCAZQBDZSVDSLVXQKJIXWLZYFMCPXELOYEOWYENKVASN
   AKMOEGPVCGTKFEKBBUHVQRDARSDEJHLVRVMRBMSQQJWCSRXKAEHSUF
   IOUJGMPOSTXCZCEPTEKFXHNALWZVMMXEXNQSUVCXAUIILZUJTXAABI
   XOMHNONIVUZONHYUJECRLJPQJUYTFPVEMDLRVAXACNXVITBOWNENNV
   GREKCDYCSFZPZMCIUDMCKHUUYMBRKIQOESQWUNTVSIVDYGWEDCYORQ
   WUJSVLEKHSJFAVDAGERCJYEJVYXCYWUOWAXDYQSPVTGHXGJNZYLHAO
   OUQPSJIPHKULHDQALOVVMVORRDQMGVCZHYIIKMFPLEDJQQGFKYORKU
   ```

WZUVOGLKJYLEXZMDFQFXUGZEBOKOANKNRTLTXCBOGYQUKMRRIZWQOZ
TLGXARDHYJ

Unit 97
Solitaire cipher

The *solitaire cipher* is a synchronous stream cipher in which the key stream is generated with a deck of playing cards. It was invented by Bruce Schneier for the spy novel *Cryptonomicon* by Neal Stephenson.

Solitaire uses a standard deck of poker cards with two jokers. Usually, one of them is black-and-white and the other is colored. We will call one of them “joker A” and the other “joker B.” We also assign a number to each card in the deck. The clubs (♣) are numbered 1 (ace) to 13 (king), the diamonds (♦) 14 to 26, the hearts (♥) 27 to 39, and the spades (♠) 40 to 52. Both jokers are numbered 53.

We begin with the deck in numberical order, clubs followed by diamonds followed by hearts followed by spades followed by joker A and joker B. To key the deck from a keyword or phrase, we perform these steps for each letter in the keyword:

1. If joker A is on the bottom (the last card) of the deck, put it just after the top card. Otherwise, swap joker A with the card below it.
2. If joker B is on the bottom of the deck, put it just after the second card. If joker B is the second-to-last card, put it just after the top card. If neither of these is the case, move joker B down by two cards.
3. Do a triple cut by swapping the stack of cards above the first joker with the stack of cards below the second joker. The first joker is whichever is closer to the top of the deck. The second is closer to the bottom.
4. Look at the bottom card. If it is a joker, do nothing for this step. Otherwise, take the number corresponding to that card and do a count cut by taking a stack of that many card off the top of the deck and putting that stack just above the bottom card.
5. Convert the letter of the keyword to a number, where ‘A’ = 1, ‘B’ = 2, ‘C’ = 3, ..., ‘Z’ = 26, and do another count cut by taking a stack of that many card off the top of the deck and putting that stack just above the bottom card.

The keyed deck is the initial internal state of the cipher, and from it we generate the key stream. We repeat the following process until we have enough key-stream letters to encipher our plaintext:

1. Same as step 1 above.
2. Same as step 2 above.
3. Same as step 3 above.
4. Same as step 4 above.
5. Look at the top card and take its corresponding number. Take a stack of that many cards off the top of the deck. Look at the new top card and record its number. Put the stack back onto the top of the deck.
6. If the number you recorded in step 5 is 53, throw it away and go back to step 1. Otherwise, if the number is greater than 26, then subtract 26 from it. Convert the number to a letter, where 'A' = 1, 'B' = 2, 'C' = 3, ..., 'Z' = 26.

The key stream is combined with the plaintext in a Vigenère-like manner, i.e., with addition modulo 26. However, it is not exactly the same; there is an offset of one. The cipher uses 'A' = 1, ..., 'Z' = 26 for this process as well. To combine a plaintext letter with a keystream letter, convert both to numbers in this way. Add them. If the sum exceeds 26, then subtract 26. Convert back to a letter.

The solitaire cipher is difficult to break unless we know more than half of the keyed deck.

The official instructions say that the plaintext should be padded to a multiple of five letters, but we are not strict about that.

Reading and references

Bruce Schneier, "The Solitaire Encryption Algorithm," Schneier on Security, www.schneier.com/academic/solitaire

Wikipedia, en.wikipedia.org/wiki/Solitaire_(cipher)

Programming tasks

1. Write a function to generate a keyed deck of cards from a keyword or key phrase. Do not use the optional step in Schneier's description of the keying method on his web page.
2. Write a function to generate a key stream given a keyed deck of cards.
3. Write a function or script to encipher a plaintext with the solitaire cipher and a given key phrase.
4. Write a function or script to decipher a ciphertext with the solitaire cipher and a given key phrase.
5. Study what happens to the deck as it is keyed with a keyword or key phrase. Can you devise a method of recovering the keyword from the state of the deck? If so, implement your idea.

Exercises

1. Encipher this text with the key phrase `LUCKY YOU DO NOT HAVE TO DO IT BY HAND`.

   The best poker hand is the royal flush, followed by a nonroyal straight flush. Then comes four of a kind, followed by a full house. Then comes a flush, followed by a straight, then three of a kind, then two pairs, then one pair, then nothing. You don't want to have nothing.

2. Decipher this ciphertext from the 2010 British National Cipher Challenge. It was encrypted with the key phrase `DIE ALCHEMISTEN RISE AGAIN`.

   ```
   AGXJE SSFKM MJMHX ZGJWB CPCVX EBNDK UQOCE DUTIC NPARQ
   PEDIX ZAVYM WZSVT BBVMT HJIGW XZAPJ HJMYN MXRGO RXOWE
   ULMJS AAENC WVUYI FQUTR XEDEJ BLWAA DFPBW ZAXJD DZOTM
   GSEZG NQWJY MFNWL SLTQD URZVQ RKOTS VDNHY EIITY RRWGC
   CSLKS UKHDR LDBZE DXSGV UGMTB NZQJT CBZTT IBWKQ PXUTQ
   MZDIH HKWZK SJEEH FBFYP GYSIH OKKOB JFJSN XSIKS NBMTN
   IADVT CXYZZ AOKQY WXNIZ JWOFZ VSPQQ GASYU MJDEL MDDHV
   ZTNFH MOOLN XAFPE VBHGS TJFMC IFNHZ YCYGG WAQYB UNNHD
   WHSLP IBFAP PDNQN DOCNU RXEAI RZNLR XAKVX XAMPU ZOPOK
   TJVQK IZDSC NC
   ```

3. Decipher this text with the deck 5♦, 6♦, 7♦, 8♦, 9♦, 10♦, J♦, Q♦, K♦, 9♣, joker A, 7♣, 5♣, J♣, Q♣, J♥, Q♥, joker B, A♥, 2♥, 3♥, 4♥, 5♥, 6♥, 7♥, 8♥, 9♥, 10♥, 10♣, K♥, A♠, 2♠, 3♠, 4♠, 5♠, 6♠, 7♠, 8♠, 9♠, 10♠, J♠, Q♠, 3♣, 4♣, A♣, 8♣, 2♣, K♠, K♣, A♦, 2♦, 3♦, 4♦, 6♣. Can you also reconstruct the keyword?

   ```
   SCYWLWXCACDWWIFZSXIMHVMBRIWHCLNLMZIHHWWCHVOZNJCKPPALVN
   MGFNJRLCQFHDKNHZHKRAGIFXGKQQSLNEDGKOTOFRFNZAJOWWVZAPGG
   MLURKZGQDMHEHKYEBLRUPMRPKFHFFMCIDKGYLOFLQQLSOMYAEGCPDY
   RWWJLTXRKQLOLXGCHCSCQMDUKZWMLMKNTHMJQNVORTTIDRHQZBEIJC
   JNTMRTHYNVGAID
   ```

Unit 98
Hill-climbing attack on the solitaire cipher with partially known key

The approach of this attack is to fix the known keys and shuffle the remainder until the fitness of the first few (between fifty and one hundred) deciphered characters exceeds a threshold; then we begin again with the key we have at that point and try to maximize the fitness of a longer part of the deciphered text. For the maximization, we use a hill-climbing technique with the parent/child key paradigm. To generate a child key from the parent, we randomly swap two of the unfixed cards, or do a three-way swap of three unfixed cards. This is a difficult attack, and it may take a long time or need to be restarted many times. It is

Programming tasks

1. Implement the attack. Use tetragram fitness to rate plaintexts. Experiment with the fitness threshold and the length of text to use in the first stage.

Exercises

1. Break the ciphertext from Exercise 2 of the previous unit. The deck is 8♦, 9♦, 3♥, K♦, A♥, J♣, 5♥, 6♥, 7♥, 8♥, 9♥, 10♥, 6♣, K♥, 4♦, Q♦, 2♠, 10♣, A♦, 5♠, 10♠, joker A, K♣, 4♠, 7♠, K♠, Q♣, 9♠, J♥, Q♥, 2♥, 7♣, 3♠, 6♠, 8♣, ?, ?, ?, ?, ?, ?, ?, ?, joker B, ?, ?, ?, ?, ?, ?, J♦, ?, 7♦, ?.

2. Can you break this ciphertext? The deck is 9♦, 3♥, 4♥, 5♥, 6♥, 7♥, 8♥, 9♥, 10♥, 2♣, K♥, A♠, 2♠, 3♠, 4♠, 8♦, 7♠, 8♠, 9♠, Q♥, Q♠, 6♠, 2♥, 2♦, 4♣, joker A, 3♣, 6♣, 3♦, 9♣, 10♣, 7♦, A♣, 10♠, J♠, K♣, A♦, A♥, ?, joker B, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? Can you also recover the keyword?

```
BHSJKOZKSQZBKVJLAICTFVGPJQJKMJCWBKTGMKUXWWIAPSCHXWUIPG
HSPRDSVSADSBHIAMWWRQCGWPSNIMKZXOXYKGCRHKWCSMGZICJYCJCW
XTBKLTAMPCFAQIIJTIWDLCPXFZKIRSVJSCKQATFLPZYCWJECXZYKID
BINPIFQJHWRECYODKTTYKHDYYULKEQFCGPGVGPQZHQAWFRMYUESBID
PSCIIGUJXXOKRSCEOLCHIDXEQCGKBVNRLNNGIXIYKYGFYUITKKHFVC
```

Part VIII
Codes

madness's book on classical cryptography
unit 99: codes
last modified 2020-07-16

Unit 99
Codes

Historically, a code is a way of hiding the meaning of a message by replacing whole words and phrases with code words. But a newer meaning of *code* is a way of converting information from one format to another. Codes can be used as ciphers. The symbols of the plaintext are *encoded* in the ciphertext symbols. *Decoding* is the inverse operation. For example, the ciphertext symbols of Morse code are the dot, dash, and space. The encoding of a single plaintext symbol is a *code word*.

Two main categories of codes are fixed-width codes and variable-length codes. In a *fixed-width code*, all code words have the same length. In a *variable-length code*, this is not true. For example, Morse code is a variable-length code, and the Polybius cipher is a fixed-width code (in base 5).

Breaking a ciphertext that is encoded with a fixed-width code is easy for us. Once we know the length of each code word, we can build a table of them and assign a plaintext symbol to each. Then what we have is a monoalphabetic substitution cipher.

Variable-length codes have two subcategories: prefix-free codes and non-prefix-free codes. In a *prefix-free code* (also called simply a *prefix code*), no code word is a prefix of another. That means that no code word looks like the beginning of another code word. Therefore, if one knows all of the possible code words, decoding is unambiguous. In a *non-prefix-free code* (or *non-prefix code*), this is not true, and decoding is difficult, even with knowledge of all code words.

Reading and references

Wikipedia, en.wikipedia.org/wiki/Code and en.wikipedia.org/wiki/Prefix_code

Programming tasks

1. Implement the attack described above for fixed-width codes. The length of code words may have to be an input.

Exercises

1. Break this ciphertext that was encoded with a fixed-width code.

```
ADGJBDGJAEGJBEGJADHJBDHLADGJBDGLBDHJADHJADGKAEGJADHKAD
HKAEGJADHKADGKAEGLBDHKADGKADHJAEGLADHLADHJADGLBEHLAEGL
AEGJADGLADHJAEHJBDHKBEHJBDGLAEGJAEGLAEGLAEGJADHKADGKBD
HJBEHLBDGKADHJADGLBDGLAEGJBDGLAEGLADHJADGLBDHKADHKAEGL
BDGKADHJBDHJADGJADHKAEGKADGJADHKAEHJBDHKBEHJBDGKADGJAD
HLAEGJADHKADGKADHKBDHKAEGLBDGKAEGJADHKADGKAEGLBDHKAEHJ
BDHKBDHKADHKBEGJADHJBDHKADGLAEGLBDHLAEGJBEGJADHJBDGLBD
GKADHJBDGKADGJAEHJAEHKADHJADHJAEHKADHJAEHJAEGJADHKAEGL
BDHKAEGLBDGKADHJBDHJBDHKBDHKAEGKBDGKADHJADGLBDGLAEGJBD
GLAEGLADHJADGLBDHLADGJBDGLADGLADHJADGJAEHJAEGJADHKADGK
BDHJBEGLAEGLAEGJAEGLBDGKADGJAEHJADHKBDHKAEHKAEGJBEGJAE
GLBEGLADGLADHJBDGLBDHKADGLBEGJBDHKADHKADHLADHJADGLBDGL
ADGJAEGLAEGJBDHKADHKBDGLAEGJADHKAEGJAEGLADGJADHKAEHJBD
HLBDGKADGJAEGLAEGJBDGLAEGLBDGKADHJBEGLBDGLADHJBDHKBEHJ
ADGJBDHJBDHKBDHKAEGKAEGLBDGKBDHKBEGLADGKBDGKAEGLADGJBD
GJAEGJBEGJADHJBDHLAEGJAEGLBDGKBDHKBEGLAEGLAEHKAEGJBEGJ
AEGLBEGLADGLADHJBDGLBDHKADGLBEGJBDHKADHKADHLADHJADGLBD
GLADGJAEGLAEGJBDHKADHKBDGLBDGLBDHKBDGLBDGKADHJBDHLADGJ
BDGLBEGJBDHKADHKBDGLAEGJAEHJADHJADGLAEGJADHKADGKAEGJAD
HKBDGKADHJADGLBDHKBDHLADHKBEGKAEGJADHKAEHJADGJBDGLBDHL
ADHJBDGJBDGJADGJBDGLBDGLBDGKADHJBEGJBDHKBEGLBDGJAEHJBE
HJBDHKADGLAEGLBDGKADHJBDGKBDHKAEGLAEHJADGJBEHLBEGKADGJ
AEHJADHJBDGKADHJADGLBEHJADHJADHJBDGJADHLADHJADGLBEHLBD
GLBDGJADHJADHJAEHKBEHLADGJADHKAEHJBDGLAEGLBEGLAEHKAEGJ
AEHJBDHLBDGKADHJAEGLBDGKADHJADGLAEGLBDGKADHJAEHKBDGJAD
HJADGJBDGLBEGLADGLADHJBDHKBEHJBEGKADGJAEGKAEGJADHKADGK
ADGJAEHJADGJAEGJBDGLBEHLBEGJBDGKADGJAEGJADHKBDHLBDHKBE
GLBDGJAEHJBDHJADHJBDHLBDHKADGLAEGLBDGKAEGLBDGKADHJAEGL
ADGLBDHKBEGLBDHJBDGJADHJBDHKBEHJADGKADHJAEGLAEGLAEGJAD
HKADGKBEGLAEHKADGJADHKAEHJAEHKAEGJBEGJAEGKAEGJADHKADGK
AEGLBDGKADHJAEHJADGJAEGJBDGLAEGJADHJBDGLBDHLBDGKADHJAD
HKBDGLBEGLAEHJAEHJADHJADHKBDGJBEHLADGJBDHLBDGKAEGJAEGL
ADHJADGLADGJBDHJBDHJAEGJAEGLBDHLAEGJAEGLBDGKAEHKAEGJAD
HKAEGKADHJBEHLADHJBDGLADGLADGJADHKBEGJBDGJBDHKBDGLADHJ
BDHJBEHLBDGKADHJADGLAEGLBDGKADHJADGLADHJBDHLADGJBDGLAD
HKBDHKAEGLBDGKAEGJADHKADGKBDGLBDHKADHLADHJADGLBEHLADGL
ADHJBEGKADGJADGLAEGKADGJBDHJBDGJADHJAEGJADHKAEGLBDGKAD
GJAEGLADHKBDHKADGLAEHJAEGJAEHJADGJBDGJAEGJBEGJADHJAEGL
BDGKAEGJADHKAEGKAEGJAEGLBDGLBDHKADHLADHJADGLBEHLBEGKBE
GLBEGJBDGKBDHKBEGLAEGLBDHKBEHJAEGLBDGKADHJBDHLADGJBEHL
AEGLBDHKBDGKADHJADGJADGLAEGLBDGKADHJADGLADGJBDHJBDHJAE
GJAEGLBDGLADGJBEHLAEGLBDHKAEGJAEGLBDGLADHJBDGJBEHJBDHK
BDGKAEHJADHJADGJADGLBDHKBDGKAEHJADHJADGJADGLAEGJBDGLBD
GKADGJBDGJBDGJBDHJADHJAEGLBDHKBDHKBDGJADGJAEGLADHJBDHL
BDGKADHJADHKBDGLBDGKADHJAEGLBDGKBDHKBEGLADGKBDGKAEGLAE
GJAEGLBDHKADHLADHJADGLADGJBEHJAEGLADHJADGLBDHLADGJADGL
AEHJBDGLAEGJAEGLBDHKBEGJBEGJBEGLADGLADGLADHJAEHJAEGLBD
```

```
HKBDGKADHJADGLAEGLBDGKADGJAEGLBDGLBDGKADHJBDHKBEGLADGK
BDGKAEGLAEGLBDHKBDGKADGJADHLADHJBDHLBDHKADHKAEHJADHJAD
GLADHJAEHJADGJAEGLAEGLBDGKAEGJBDGLBDHJBEGLAEGLADGJAEGL
AEGLBDGKADHJAEGLAEGJBEGKADHJAEGJAEGLADGJBDGJBDGJBDGLAD
HJADHJBEGKADHJAEHJBEHKBEGLAEGJAEGLADHJADHKADGJAEGLBEGL
ADGLADGJBDGJBDHJBEGLAEGLBDHLBDGKADHJADHKAEGLBDGKADHJAD
GLADGJBDHJBDHJAEGJAEGLADGJBEGJAEGLBEGLADGJBDGJBDGJBEHL
AEGLBDHKBDHKAEGKADGJBDHLADGJAEGLBEGJBDGKBDHKBEGLAEGLBD
HKBEHJAEGJAEGLBDGLBDHLADGJAEGJBDGLAEGLBEGJBDHKADGJAEGL
AEHKBDHKBEGJAEGKADHJAEGLADGJADHKAEHJBDGJBDHKBDHKAEGKAD
HJAEHJADGJAEGLAEGJAEGLADGJADHKAEHJAEGLBDGKADHJADHKBDGK
BEGLADGLADGLAEGJADHJAEHJBDHKADHKADGJBDGJAEGJBEGJADHJBD
GLAEGLADGJADGLAEGLADHJAEHJAEGLBDHKBDGKADHJADGLBEHJADHJ
ADHJAEGLBEHJBDHKADGLAEGJAEGLBEHJBDGJADGJBDGLBDGKADHJAE
HJADGJBEGJADGLBDHKBDGLBDGLBDGKADHJADGLBEGKAEGJADHKAEHJ
AEGLBDGKADGJAEGLBDGLBDGKADHJBDGKADGJAEHJADHKADHJADHLAD
HJADGLBDHJADHJBEHJBDHKADGLADHJBDGLADHJADHJADHKADGJADGL
ADGJBDHJBDHJAEGJAEGLBDHLAEGJAEGLBDGKADHJAEGJAEGLBDGKAD
HJADGLADGJBDHLADGJAEGJBDGLAEGLBEGJBDHKADGJAEGLAEHKBDHK
BEGJAEGKADHJAEGLBDHKADGLADGJBDHLADGJAEGLBEGJBDGKAEGLBD
HKAEGLADGJAEGKADHJBDHKBEGLAEGLBDHKBEHJAEGJAEGLADGJADHK
AEHJBDHJBEGLADGLADHKAEGJADHKADGKBDHLAEGJAEGLBDGKBEGJBE
GLADGLAEGJBDHKBDGLAEGJAEGLBEHLBDGLBDGKADHJADGLADGJADHK
ADGJBEGJADGLBDHKBDGLBDGLAEGLBDGKADHJBEHJAEGJADHJBDGJAE
HJADGJBEHJAEGLADHJADGLAEGJAEGLADGJADHKAEHJBDHLADGJBDGL
AEGJBEGLBDGLAEGLAEGJADHKAEGLAEGJBEGKADHJAEGLBDHKBDGLAD
HJADHJAEGJAEGLAEHKBDHKAEHKAEHJBDHKBDHLADHKADGJBDGJADGJ
ADGLADGKADHJADGLADGJBDHJBDHJAEGJAEGLBDGKBDHKBDGJADHJBE
GLADHKAEHJADHJADGLAEGLBDGKADHJBDGKADHJAEHJADGKADHJAEGJ
ADHKADGJADHKBDHKAEGLBDGKADHJADGLBEGKBDHKBEGKADHJADHKAE
GLAEHJBDHKBDHLADHKBDHLADHJADHKAEGLADGJBDGJAEGJBEGJADHJ
ADGJBEHJAEGLADHJADGLAEGJAEGLADHKADHJADHLADHJADGLBDHKAD
HKBEGJADHJBEGJBDHKADHKBDGLAEGJAEHJADHJADGLAEGJADHKADGK
BDGKBDHKBDHLAEGJADHKAEGLBDGKADHJBDHLBDHKADGLBDGJAEHJBD
GLBDGKADHJBDHLADGJBDGLAEGLBDHKADGKADHJAEGLBDHKBEGLAEGL
ADGJADGKADGJAEGJADHK
```

2. Break this ciphertext that was encoded with a variable-length (prefix-free) code. You will have to do at least part of the process by hand.

```
85639848098198385169256961876929811858199917638058563l
691816781608769285639608563981836997651692135796569980
806357638018380635763569167606796985097187839606996048
563967631228767967081605298192606963583760331811692856
396085639812995838762626083985639848999176361608585809
923981608769218585639017656048563976033181836398315285
606398183934856398483856060283608385533856318583639648
758594608161608585639848098191358991692836398018358783
856160569618160876928560839954856398060812858099239801
```

```
838081585859691858563901765604917637603318180639698363
980183838518185392084189605797606756961481606785639606
996718165922876754846087856356965809819801828060816583
639831528460876087616385856062184846087656960808018280
608165838098196985671298560093606065921854608169608563
569616960636080760698581181580583912292856396069967181
659229954846087856356965809819135899846087608761638585
608362916556783878195678998184836081818480183133135797
608732831844608185639806081283604856396032836069616596
285815696156961856381608761636398163912356598563985576
556961604173607651692836397608732631812384639362831845
696185639676087853608728580992392876716928580992392991
618199285606318991018585394608185809923928767831528580
992392996312836260539263583695796998081185853958783858
563969439802608069167606983858160878378160801830317651
831851810181819380635763481561638596992060856385639639
816098383608563984648758594608161608585639581648718181
935656960808063185846087819856356965569611060878583152
858099239287670878558558369858360696063608076069858118
158058397606985569879285809923929954585801838360585675
616385091692545858098198360585806087320908785183585583
698558515698585631858336061575801838563569655696113579
831528998184626035859384806357635838563909838580184608
78560485635838060602
```

3. Break the following ciphertext. Although it is a variable-length code and not prefix-free, you should be able to find a weakness and decrypt it.

```
624435638696115072679850656809026349262963492606798506
332249696349265867914624435644163322496342167914680902
626349265862108644167914658680902634926791463869611507
267985065680902633224963421679146441644163322496067914
680902626332249680902634926567914611507263322496809026
791464416332249680902680902626791462967914610679146296
349267634216791463322496386244356386068090262633224968
090262443564416857219634926115072658644163492611507265
626115072633224964416791464416332249611507267914696349
263861067914611507268090267914658624435638680902634926
441634926342167914680902626244356386067914629644167914
633224969634926586791463421624435606268090268090261150
726332249638644167634926115072634216962633224963860679
146244356386809026349269610606586419763492611507269634
926964197680902633224962443562962963492621086386067914
680902626791462108644163863322496809026244356349263863
322496296441679146962108611507262443568090267985063322
496067914638696798506586791467624435638679146586332249
696349265867914633224964416332249644162108641644168090
262443568090262108680902624435634926386961150726798506
568090263492644167985064416809026791463421624435638685
```

7219626244356962680902626791465629633224962443563868O9
026791465580456809026791462967914634216791463868090264
416332249611507267914656115072624435634216332249611507
262443562967985068572196349261150726586441656261150726
332249644167914644163492611507264416791463868090267914
638696791464416332249638658680902626791469634926586791
467914636210862443561063322496296791463868090264416963
322496296296791465869634926586791460611507263492621086
564416809026798506562443569633224962962967985069634926
386441624435644168090263492676296791468090268090267914
611507264416349261150726586244356062443568090264416349
261150726416349268090262624435638634926809026267914611
507268572196244356441679146342167914633224963862443563
860629679146441644169634926342164162443563863322496809
026244356349263864416349267624435658679146386809026244
356963322496296296791463860680902626332249696349265867
914641634926349264197624435644163867914679146586791465
868090263492679146386961150726798506568090263322496386
586586791469611507267985065680902680902626791465626115
072633224964416791464416349261150726857219634926115072
658644164167985069634926386809026115072633224964416809
026962443565626791461150726441679146386961150726798506
568090263421679146441644163322496067914644163322496809
026809026267914629679146106791462963492676244356386586
244356106244356586210863322496296296791468090268090267
914611507264416349261150726441634216332249629629606115
072634926210865644163492676296791468090268090267914611
507264416349261150726791461067914638624435638634216349
265867914611507263869624435656267914611507264416244356
386586244356106244356586210863322496296416244356809026
441634216791464416441633224960679146441696332249638641
679146809026115072633224963864416763492611507263421679
146586762443561150726441680902641679850633224969634926
586791463322496386586809026267914638641679850633224969
624435656267914611507264416210869626342162108629680902
624435656296791467914638696115072679850656809026244356
349263863492611507264416210865679146115072679146386961
150726798506568090262443563492638633224962443563421644
168090263492634216332249641976791469611507267985065680
902633224963863322496296798506441624435644163421634926
115072679146586244356767624435696210862968090262443564
416809026349262967914680902626244356441696244356562679
146115072676115072634926342164416349263421679146562108
638641976262443560626441696263492634926296791461150726
441676115072634926342165633224962963492633224962968090
263492696332249629624435676349261150726386244356332249

Unit 100
Baconian cipher

The *Baconian cipher*, also known as *Bacon's cipher* or *biliteral cipher* or *biliterarie cipher*, was invented by Francis Bacon a long time ago. It is a fixed-width code in which the ciphertext symbols are 'A' and 'B.' The code words are in the following table:

A	AAAAA	J	ABAAB	S	BAABA
B	AAAAB	K	ABABA	T	BAABB
C	AAABA	L	ABABB	U	BABAA
D	AAABB	M	ABBAA	V	BABAB
E	AABAA	N	ABBAB	W	BABBA
F	AABAB	O	ABBBA	X	BABBB
G	AABBA	P	ABBBB	Y	BBAAA
H	AABBB	Q	BAAAA	Z	BBAAB
I	ABAAA	R	BAAAB		

If we replace the ciphertext symbols 'A' and 'B' with '0' and '1,' we obtain a five-bit binary encoding.

Reading and references

Francis Bacon, *Of the proficience and advancement of Learning, divine and humane*, London: Henrie Tomes, 1605.

Wikipedia, en.wikipedia.org/wiki/Bacon's_cipher

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Baconian.pdf

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter V, section I.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 882-884.

Blaise de Vigenère, *Traicté des chiffres ou secrètes manières d'escrire*, Paris: Abel l'Angelier, 1586,

HDL: 2027/ien.35552000251008, gallica.bnf.fr/ark:/12148/bpt6k1040608n, gallica.bnf.fr/ark:/12148/bpt6k94009991, pages 200b and 241-243b.

Programming tasks

1. Write a function that takes a letter of the alphabet and returns a five-bit binary representation. You may store the five-bit number as a string of 0s and 1s or as an array.

2. Write a function that takes a five-bit binary number and returns the corresponding letter.

3. Write a function or script that encodes a plaintext in the Baconian cipher. Allow a switch so that it can use the symbols ‘A’ and ‘B’ or ‘0’ and ‘1.’

4. Write a function or script to take a ciphertext that contains two symbols (*any* two symbols) and decodes it as a Baconian cipher. Be careful that if you interpret the ciphertext symbols incorrectly, then you will find code words that are not in the table above. In that case, you should automatically correct the mistake.

Exercises

1. Break this ciphertext:

```
B88B88BBB88BB8888B888888B88B888B8BB8B88888B88B8B8B88B8
8B88BB88BBB88888B88BB88B8BB888B888888BB8B888B88B888B88
B88888B88888888B88BBB88BB8B8B888B88B8B88BB88BBB88B88B8
8BBB888BB8B8888B8888888B8B88B88BB88BBB8BBB8B888B8BBB88
8B8BB88B888BBB888888B8B888B88B88B88BBBB88B8888888B888B
88B88B88B8B8BB88BBB8B888B8B8B8B88B888B8BB8B88B888BB88B
B88BBB88B88B888B8BB888BBB8B888B88B88B88B88BBB88BB8888B
88888B88B8BB888888B8888BB88B88BB88BBB88888B88BB88BBB88
B8888B888BB8B888B88BBB8888BB88B88888BBB88B888B88888B8B
888B88B88B88BB8BB8888B88B88B8B88B88888888BB888B88B88B8
8B8888BB8BB88BB8BBB8B88BB88BBB88B8888B8B8B888B888BB88B
8B88BB88B8B8BBB88B8BB8B8888BBB88BBB888B8BB88B888BBB888
888B8B888B88B88B88BBBB88B8888888B888B88B88B88B8B8BB8B8
88B8B888B88BB8B8888BB8B88BB8B88B8
```

Unit 101
Triliteral cipher

The *triliteral cipher* (or *triliterarie cipher*) is a fixed-width code that uses the ciphertext symbols ‘A,’ ‘B,’ and ‘C,’ and these code words:

A	AAA	J	BAA	S	CAA
B	AAB	K	BAB	T	CAB
C	AAC	L	BAC	U	CAC
D	ABA	M	BBA	V	CBA
E	ABB	N	BBB	W	CBB
F	ABC	O	BBC	X	CBC
G	ACA	P	BCA	Y	CCA
H	ACB	Q	BCB	Z	CCB
I	ACC	R	BCC		

Some add `CCC` to represent a space, or put the space at the beginning and shift all the letters down one.

If we replace ‘A,’ ‘B,’ and ‘C’ with ‘0,’ ‘1,’ and ‘2,’ then we have a three-digit base-3 (ternary) encoding.

Programming tasks

1. Write a function to take a letter and return a three-digit ternary number. Allow for the possibility that `000` might represent a space. You may want to represent the ternary number as a string or an array.

2. Write a function to take a three-digit ternary number and return a character. Allow for the possibilities that the alphabet starts with ‘A’ or with a space.

3. Write a function or script that takes a ciphertext that only has three symbols and finds the best decoding as a triliteral cipher. There are six ways to assign the three symbols, and two ways to decide if `000` is a space or not. Use tetragram fitness to choose the best decoding.

Exercises

1. Break this ciphertext:

```
V4V4VVV444AV4AV444A44V4A44444A444AAA44A4VAA444A4444AAV
VA4AAA44V4AVV44444VVA4AVV4A444A44V4V444A44V4A444A44V4V
V4A44444A444AAA44A4VAA444A4444AAVVA4AAA44V4AVV44444V4A
AV4AAA4AVVAV4VV4AVV444AV444A44AAV444V4V4VV44AV4V44444A
AAV4A4A444AVAAVV4V444AAAVVAV4AV4VA444A4444V4VV44A44AAV
A44V4VVVA444VVAAV4VA44444VA4AVV4V444V4V4VVV444AV4AV444
44AV4A44444A444AAA44A4VAA444A4444AAVVA4AAA44V4AVV44444
V4V4VV4AV444AVA44AV4AV4V44444AAAV4AA444V4V4VV4AV444AVA
V444AVV4A4AVAAVV4V44444AAAV4AA444V4V4VV4AV4444V4VA4V4V
VA4V444AV4444V444AA44V4V4VV44444AAAV4AA4444VVAV4AVA4AV
44444AAAV4AA4444A44VV44AV44A44V4VVVA444V4V4VV4AV4444VV
4AV44AV44V4V44444AAAV4AA444V4V4VV4AV44444VV4444AA44AAV
44444AAAV4AA444V4V4VV4AV44444VAV44AAVVA4444VAA44VAA4AV
444VVAAV4VA4444V4V4VVV444AV4AV44444AV4A44444A444AAA44A
4VAA444A4444AAVVA4AAA44V4AVV44
```

Challenge

```
IECIEAEAHEAHDGBGCDCIDGADIAEAEHEAHGCDIADBHFEGBGDCAIDFIBEBGDA
GGDCFBIEAIICEHFCCDGHDAFBGCIEAIDAGDBEHCHFGDCGCEFCHCDGEGCBGFB
IFGDCICEFIAEGCGBDBFGFCHBHEHFAADHHADIADCDIADGEHBEIAFHCAHFIBF
BGFIEAEAIDICBFHAHFFIBFHABGEEAIDHAGBEAFHEHCAIEBFHEAIHFAEBIGC
DAFIFHACIDGBDCGDAHFDBGHBEIBDGDCHADFAHDCIAEHDGCIBFBEGFHAIDCF
CHDGCGECDAHAEICDGFBIDGAEBHAIEAGDIAEHAFHFBHEAGDCCGDFBGAFHIAF
IFACDGDAGGADIDAIBDFHCDGCEGCHADEIAGDAEHBIEAGDBDGCFHABGDBEHHF
CGCDECGHFBCEGDCIGADGBFBIFDGCICEHFAHEAHAEADGEHBEIAEAGHAEHFAH
CFAEIHADICDDAIBGDEIAAHDDGAHFAADIBIFHEACHFHDBCDGCEIAEIAHEHEA
CHFDGCGCEFBGFBIDCGECIDAICHEAEIIBFIAECHEAIEDAHHBFIEADAGAGDHE
BEAIEGBGCEHFCCDIICDGDCFCHCDGCGEAHEHEAEHBAFHCHEIAEDAGGCDAGDI
EAAHEHEABFHIAEDAGEHBAIEAIDDHAFBIAHFFHBAEIDCIHFAIFBBDHAGDEBH
EIABIFADIHEAAHEFBGBIFDGCECICEIBEHDCGDCIEGAEHAFAHHFCDIAIFBEG
BGDCFBIGDAEHBIAEDAGAEIHAFBHFCDGEHBIADHAEEAHAGDAEIHEAHEAFCHD
GCEGCADGHEBIAEIDAHADIFBAFHHFBIEADCIIFAEGCGADHCFCGDCGEBGFIFB
DCGCEIIDADAGIDCAEIEAIFBHCDIAGDDGCHBFAEIAGDBHEIEACHFEGBAIDCH
EEAIDGABEHEAICDIEAIIFAFAHEHAAHEEAGEHAAHFFCHAEIHDACIDFBIGDCC
IEFAHDHBAHFFCHDCIHECEAIHADHCFGEAEIAGBDCGEEHADAIAFHDAHIFBHFA
BFHIEACIDIECBEHGCDIDCDGCFIBDIBIADDAHIDCADGAEIGFCHFABDGAGDHA
EFHCEICHBEHFAAGDFBIGDCEICEHBAHFGADCDICGDIFBDICEIABGDCDGIBFB
DHADIBGEAEIAGDAIFIEAEHBIADIBFHDBAGDHBEAEIEAGHEAHAFAGDIEAAGD
CGDBDHCGDDICGDCBHFAIEBDIAFHIFBBGDCHFBGDFHAGADDBGHEBDIABFIBE
GAGDCGDBHFCGDDHAADHGCDIECICDEAGAIDGDAGBDHEBIDAIFBEGBCGDFBIF
HBFCHADGEIAHAFHFBHFABFIBHDAFHEHBAIEAFHCHECHFHBEIDAADGGDAEAI
DAHGEBAEIDGAIDCCEGEAGIBFGDCEICADGHEBEIAHBEAEIAFHCHEHFCEBHAD
IAGDADGEAIHDAFIAECGFIBGDACIDAGDHEBAIEFIAHAFHEAAEHIECBHEIAEI
BFBEHAIEIAFGCEIBFAGDICDGDAEBHAIEIFAHFAHAEAHEHBFAEIIAFAEIDAI
FBIEBGAHFGEBCGDCDGHBDBDGHFAGDAGDBHBEAEIHADADIBHFBEGGCDFBIFB
```

IAHFDAGBHEHDADCGCIEADGHBEEAIEGBCGEHFCGCDEGCDGAFAHDAGBDIADIH
DAIDCDGAIFAFAHIDCIEACDIDCGDIAGAEDIADBGFGBGCEGAEGDABHEIEAAFI
AHFAEHHAEFAHBIFHDBGADHEBDAHGCDIECDIAGDAAGDDCGCIEHBEDGCIFBDC
GICEICEHBECGDCDIGEBDGCADGDIAAGDFBIFAHDGACGEHDAFHAAEHAEHFHCA
EHDGCDGCFBGDIADBIIADAGDEBHAHDDCGICEGDABHEEIAIAFFHAHAEHEAGAD
DCGBDIADIHADDICDAGIAFAFHCDIIEACDIGCDHBFEIAAIFDGCBDHCFHDICEB
GGDCADGADGFHAEGBAIEAGDDAIADGBFIGDCICEEICBHECDGBEHHFADCIAIDG
DAFIBHFAGADEGCAHDFAHAHEHEAHCFIDCDCGADIAEGIADGDBFBGGECEAGADG
HEBIAEAIFAHFHEAAHEFHABIFHBDDAIADGHEBADHDCGIECIDAADGDAGDCGCE
IEBHGCDIECBEHDGCHBDAHDGCDAEGICDGDAEHBAIEIAFHAFEAHHAEHAFBFID
BHDAGBHEEIAGBEGECCFHDAHGCEIBFICDDGACDGCIDEIABDGCGDIBFBDHECI
EBHCGDEAGDAIDBGBFGDICCGEEAGGADHEBIEAFAIAFHHEAEAHFAHBFIHBDDA
GEHBAHDCDGCIECDIDAIADGDAGGDCCIEHEBHAFDAGCEIBHEHAFGDAGDABEHA
HDDCGIECCIDDIAGADGADGDCDAIBHDDGCIFBADGGFBBIFGDCIECDAIBDHDGC
BIFAGDBGFIFBGCDEICADGEBHDHADCGCIEDCIAIDDGAAIFHFAGBDBFGDGAGD
CADGCDGFBHDCGHDADAHDGCIECDAGAHDDAIGAEHAEIAEGEAAEHAFHCFH

Unit 102
Morse code

Morse code is a variable-length code. The ciphertext symbols are the dot · the dash – and space. Sometimes the space is written with a slash (‘/’); it is placed between code words. In addition to the English alphabet, there are code words for other European characters, digits, and some punctuation. In the table below, you might notice that some code words are not unique.

A	·–	J	·–––	S	···
B	–···	K	–·–	T	–
C	–·–·	L	·–··	U	··–
D	–··	M	––	V	···–
E	·	N	–·	W	·––
F	··–·	O	–––	X	–··–
G	––·	P	·––·	Y	–·––
H	····	Q	––·–	Z	––··
I	··	R	·–·		

0	–––––	5	·····
1	·––––	6	–····
2	··–––	7	––···
3	···––	8	–––··
4	····–	9	––––·

Ä	·–·–	Á	·––·–	Å	·––·–	CH	––––
É	··–··	Ñ	––·––	Ö	–––	Ü	··––
&	·–···	'	·––––·	@	·––·–·	)	–·––·–
(	–·––·	:	–––···	,	––··––	=	–···–
!	–·–·––	.	·–·–·–	-	–····–	+	·–·–·
"	·–··–·	?	··––··	/	–··–·	#	···–·–
$	···–··–	%	·––––	;	–·–·–·	_	··––·–
~	·–···						

Reading and references

Wikipedia, en.wikipedia.org/wiki/Morse_code

Morse Code World, morsecode.world/international/morse2.html

Programming tasks

1. Write a function or script to encode a text in Morse code.
2. Write a function or script to decode a text from Morse code.
3. Write a function or script that takes a ciphertext containing three symbols and finds the best interpretation of those symbols as dot, dash, and space, and decodes appropriately.

Exercises

1. Encode this text in Morse code:

```
IN MORSE CODE THE LENGTH OF A DASH IS THREE TIMES THE
LENGTH OF A DOT. THE SPACE BETWEEN DOTS AND DASHES IN
THE SAME CODEWORD IS THE LENGTH OF ONE DOT. THE SPACE
BETWEEN LETTERS IS THE LENGTH OF THREE DOTS. THE SPACE
BETWEEN WORDS IS THE LENGTH OF SEVEN DOTS.
```

2. Decode this text:

```
-   ....   ..   ...       -   .   -..-   -       .--   .-   ...
.   -.   -.-.   ---   -..   .   -..       ..   -.       --   ---
.-.   ...   .       -.-.   ---   -..   .   .-.-.-       -
....   ..   ...       -   .   -..-   -       ..   ...       .-
.-..   ...   ---       -.   ---   -       ...-   .   .-.   -.--
..   -.   -   .   .-.   .   ...   -   ..   -.   --.   .-.-.-
```

3. Decode this text:

```
::'';':;:'::':';::'';';;;'':;;':;':;:';:'';:;;';;;'::;
'':;::':;':::';'';'::';;':'';'::::':;';'';'::::'::':::
'':;;';;;'::;':;::';::'';:';;;';'';:::':'':::;':':;:';
:;;''::';:';':':;:':':::';'::';:';;:':;:;:;''::';':;;;
;:':::'';:;;';;;'::;':;:''::;:':;'::;':;::';''::;:';;;
':;:'';:';;;';'':;::'::':::';':';:'::';:';;:':;:;:;'';
:';;;':;;'';:;;';;;'::;'':::::':;':::;':'';';;;'':::'::
;'::;:'::;:':':;:'';'::::':;:';;;'::;';;:'::::'':;';:'
;;;';'::::':':;:'';:::';;;':;:'::';:';;:'':;;:':;::':;
'::';:';':';::;';':;:;:;
```

Challenge

```
01110001010101000101000101010000000111010111010001010001011
10111010001010101000100010111010001110001000111010101110001
11000000010101000101000111011100010101110001011101010001011
10001110001000101010000000111000101010100010000000101010111
00011101110111000101110101000111000101110001110111010001000
00001110111011100011101000000010111000000011100010001011101
01000100011101110100010111010001011100010111011110100010101
10000000101110101000101000111010001000000010111000101010000
00010111000000010101110100010101110001110100011101011101000
11100010100011101110111000111010000000111011101110001010111
01000000011100010100011101110001000101110101110101110000000
10100011100000001010111010001110111011100010111010100010111
01010001110111011100010111011100010101000000011100010101010
00100000000111010100010001010100011101011101000101110100010 1
00010111011110100011100010100011101111011100011101000000001110
11101110001010111010000000111000101010100010000000111011100
01110111011100010111010001010100010000000101010001110001011
10001110100011101010001011100010111010001110101000000011101
11010001010001010101110001000111010000000101000111010000000
10001110101011100010001011101000111010111010001010001010100
01000000010111011101110111000101110101110101110000000111010
00111011101110001011101110000000111000101110100011101011101
11000000011100011101110111000000010100011101110001011100011
10111010001010001110100010000000101010100011101111011100010 1
11011100000001010101000101110001011101000111010100000001110
00101010100010100010101000000010111011100011101110111000101
01110001011101010001110101000000011101010100010000000101000
10101110100000001110101110111000111011101110001010111000000
01011101110001000101110100010000000101110101000101000101010
00111000100011101000101000111010001110111010000000111000111
01110111000000010111000000011101110001110111011100011101010
00100011101110001011101011101101110
```

Unit 103
Monome-dinome cipher

The *monome-dinome cipher* is a prefix-free variable-length code. The key is a keyword and a permutation of the ten digits. The cipher uses a 24-letter alphabet, so we have to boot out two letters. Usually, we merge ‘J’ with ‘I,’ and sometimes ‘Z’ with ‘Y’ or ‘S.’ The keyword is used to generate a mixed alphabet from the remaining 24 letters. This alphabet is put into a 3×8 grid. The first two digits label the second and third lines, while the remaining eight digits label the columns. The code word for a letter is the row label, if any, plus the column label. Code words that contain one digit are called *monomes*; those with two, *dinomes*.

Let’s work through an example, using the keyword `KEYWORD` and the list of digits 5, 9; 3, 4, 1, 7, 6, 8, 2, 0. One way to generate the mixed alphabet (remember that there are many ways) gives us this grid:

	3	4	1	7	6	8	2	0
	K	E	Y	W	O	R	D	A
5	B	C	F	G	H	I	L	M
9	N	P	Q	S	T	U	V	X

If we encipher the message

```
THIS MESSAGE WAS ENCRYPTED WITH A CODE
```

we get

```
96 56 58 97 50 4 97 97 0 57 4 7 0 97 4 93
54 8 1 94 96 4 2 7 58 96 56 0 54 6 2 4
```

Of course, we don’t want the spaces to betray our encryption method, so they are removed. The ciphertext is

```
9656589750497970574709749354819496427589656054624
```

With a ciphertext that is long enough to reliably use statistics, the monome-dinome cipher is easy to break. The two most common digits will be the row labels; the rest will be column labels. Once

we know all the code words, it becomes a monoalphabetic substitution cipher, which we can break with the method in Unit 28.

Reading and references

American Cryptogram Association,
www.cryptogram.org/downloads/aca.info/ciphers/MonomeDinome.pdf

Programming tasks

1. Implement an encryptor for the monome-dinome cipher. Allow for the choice of letters that are merged in the mixed alphabet.
2. Implement a decryptor for the monome-dinome cipher. Allow for the choice of letters that are merged in the mixed alphabet.
3. Write a function to tabulate the frequencies of digits in a ciphertext.
4. Implement the attack mentioned above.

Exercises

1. Encipher this text with the keyword `AUTOMOBILE` and digit list 3, 1; 8, 7, 5, 4, 2, 6, 0, 9. Use standard choices for mixing the alphabet and merging letters.

   > When the windshield was closed it became so filmed with rain that Claire fancied she was piloting a drowned car in dim spaces under the sea. When it was open, drops jabbed into her eyes and chilled her cheeks. She was excited and thoroughly miserable. She realized that these Minnesota country roads had no respect for her polite experience on Long Island parkways.
   >
   > (from *Free Air* by Sinclair Lewis)

2. Decipher this ciphertext with the keyword `HIGHWAY` and digit list 1, 3; 6, 4, 0, 8, 9, 2, 5, 7. Use standard choices for mixing the alphabet and merging letters.

```
814693514915148717157143639417151710173230030149397173
215393026143035938391514383861430359304141638714151430
261430363017383614304392614306936364151438386143051732
151619143838301438173230714386143041212141538143617383
845419439414386143018415161514383891516617361410321915
143838814930145173215163917614305293961732389151615148
394143817109734329415399157141710938381774939417159151
617103630416143961419415717191564068924389193014916286
939439438415391415161416391751490171916141530179161710
```

```
36191493832301491516323814103219151438381043919216141 6
47939141691516161438394151416391741538364301490301493 9
36939304173943812915163917617151732309030149393693930 4
1739
```

3. Break this ciphertext:

```
48080528768180845127654528754815752534525751815380268 6
15287284821250645025757757748005325757818780815386805 1
25952806257821480504825752848752152568081878086806457 6
54525728781530123462648180872845148450813521525165448 4
57287872849525448051257576545287811568153508182521804 5
08684565282575212805102574575281280518754266545275425 1
51818052805252515251480652182152626481804687284838646 5
28026861257654266545259818468193481868428051595250542 8
04502578254528081595280284548186575153418469521525186 5
05251280518481654528214805048242872845081805052180525 1
87465459525054280450257821480504825752842828257452516 8
12846181808159450257821819575259 84
```

Challenge

```
BEAHAICIHAICGAHBFAFBEBFDGBEAGGCIBFCFHAFDHCGBEBFBGAGAIBEAHGD
ECFAIFAIBEBFAFGFEAIFAIBEBFAFGFHAICGAHBFAFAGAHAIHAHAICIGDFGA
FAIGEFBFFBFCFBIBEAHCGAFBFBHAIDGBHAFBFBFHAEBGBFBFGHAHFBFBEBE
BFAFAEBHBEAHBFCGFGAICFBEBFDGBEAICIBFCFHAICGAHBFAFBFBGGCIBFA
IBEAHBFAFAECFBFAEAFBEAGAEFBFBEBEBFAFCIAICFBEAHBFHAICGAHBFAF
BEBFDGBEBEAHBFGCICIAIBICFCEBFCFBEAEBHHAEBGBFAGAEAFBGCIBHAEA
FBFGHAHCGFGAICFBEBFDGBEFBFBEBEBFAFAICIBGAECFBFAICFGCIDHCIBE
BFCEGBEAIHAGGDHHGCFDHAEDEBHAICFBGBEAHBFIBFDHAGAEAFBGBEAHGBE
AGGCIDECIBFBG
```

Unit 104
Straddling checkerboard cipher

The *straddling checkerboard cipher* is another prefix-free variable-length code and is very similar to the monome-dinome cipher. The key is a keyword and a pair of digits. The keyword is used to generate a mixed alphabet, which is placed into a 3×10 grid. The two digits label the second and third lines, while the digits 0, ..., 9 label the columns. When the alphabet is laid in, the spots in the top row under the digits that match the two row labels must be left empty. There will also be two empty spaces in the third row, or two additional characters can be added, such as space and period. The code word for a letter is the row label, if any, followed by the column label. Code words are monomes or dinomes.

Let’s work through an example, using the keyword `KEYWORD` and the pair of digits 5 and 8. One way to generate the mixed alphabet (remember that there are many ways) gives us this grid:

	0	1	2	3	4	5	6	7	8	9
	K	E	Y	W	O		R	D		A
5	B	C	F	G	H	I	J	L	M	N
8	P	Q	S	T	U	V	X	Z		

If we encipher the message

```
THIS MESSAGE WAS ENCRYPTED WITH A CODE
```

we get

```
83 54 55 82 58 1 82 82 9 53 1 3 9 82 1 59
51 6 2 80 83 1 7 3 55 83 54 9 51 4 7 1
```

Of course, we don’t want the spaces to betray our encryption method, so they are removed. The ciphertext is

```
83545582581828295313982159516280831735583549514 71
```

With a ciphertext that is long enough to reliably use statistics, this cipher is easy to break. The two most common digits will be the row labels. Working from the beginning of the ciphertext, knowing

the row labels allows us to unambiguously break it into code words. Once we know all the code words, it becomes a monoalphabetic substitution cipher, which we can break with the method in Unit 28.

Reading and references

Practical Cryptography,
practicalcryptography.com/ciphers/straddle-checkerboard-cipher
practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-straddle-checkerboard

Wikipedia, en.wikipedia.org/wiki/Straddling_checkerboard

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 200-201.

Paolo Bonavoglia, "The straddling checkerboard cipher," La crittografia da Atbash a RSA, 2020, www.crittologia.eu/en/critto/straddle.html

Friedrich L. Bauer, *Decrypted Secrets: Methods and Maxims of Cryptology*, 4th edition, Berlin: Springer-Verlag, 2007, pages 56-57.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 635-636.

Programming tasks

1. Implement an encryptor for the straddling checkerboard cipher.
2. Implement a decryptor for the straddling checkerboard cipher.
3. Implement the attack described above.

Exercises

1. Encipher this text with the keyword `AUTOMOBILE` and digits 3, 1. Use the easiest choice for how to mix the alphabet.

   > What I was going to say is this: wouldn't it be much better to turn your car into the means of making an honest living, and at the same time having some rattling good fun, rather than sell the thing for less than half cost, and not only get no fun at all, but not know how to get out of the scrape in which you've landed yourself?
   >
   > (from *My Friend the Chauffeur* by C. N. Williamson and A. M. Williamson)

2. Decipher this ciphertext with the keyword `HIGHWAY` and digits 1, 6. Use the easiest choice for how to mix the alphabet.

```
151818142173563162626215116526363181711951705611015721
653217116305636201191864151010612651159561563626490631
161612122962191111106301119648152921726362162171062117
111718101864863122364611162630262161863186190516192181
756258236263615191921735165671817101861186307151165263
632621166596315718615151618626362186301110261119631819
191862263111812626490519296364611162011262630111618626
332611526201812418165171574181611176215230632176263563
646111620751710620612171421735151618626363216210217011
611165116171057152121126326262111116217351656165111563
056362011951761115151578116301141816517401805621018171
151515630563630116111918611062620184
```

3. Break this ciphertext:

```
714240734049764717070383037176141456714240456713560705
340465407040404570646346701072804051467149693571314634
171423707053404654037014607694097257137340346647474064
964654041491457442404654037094049373409714267147404941
405714932172497442354237054260040462409876461464034114
671424051467149649766007142404749147170371314670744235
423714046724653671407045676840940493734093461499404987
671424049720407014141149456001235421743703717142671456
714240456713570370461714940972540971162326467135716727
110127671424070760012370455646714065427270461714234624
070704046713600764640746469341407340497671423462457270
717047493462414914571424047493465347040141394046713717
671424046407340497671423462704217209840564768040141840
346249409725409711714267147493465347040649407440714240
467116945371714267171424040467246536713146701416007142
407142401494045707437142744235427014564676731072454070
649404130040964940146076346934940571746767014170676346
2714267163706
```

Part IX
Miscellaneous ciphers

Unit 105
Symbolic substitution

A *symbolic substitution cipher* is like a monoalphabetic substitution except that the ciphertext symbols are not letters. We can attack such a cipher by *transliterating* it, i.e., by assigning a letter to each symbol, then applying the hill-climbing attack from Unit 28.

Reading and references

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998, pages 173-180 and 183-184.

Simon Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, New York: Random House, 1999; see the section "The Babington Plot" in chapter 1.

Edgar Allan Poe, "The Gold-Bug," 1843, en.wikisource.org/wiki/Tales_(Poe)/The_Gold-Bug
Edgar Allan Poe, "The Gold-Bug," 1843, en.wikisource.org/wiki/Tales_(Poe)/The_Gold-Bug, www.eapoe.org/works/tales/goldbga2.htm

Arthur Conan Doyle, "The Adventure of the Dancing Men," first published in 1905, now in *The Complete Works of Sherlock Holmes*, London: Simon & Schuster, 2012.

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter VII, section II.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 90-91, 93-94, and 174-176.

S. Tomokiyo, "First Codebreaking in the American Revolution — Benjamin Church's Cipher," cryptiana.web.fc2.com/code/church.htm, 2009-2014.

Benjamin Church, Jr., George Washington Papers, Series 4, General Correspondence: Benjamin Church Jr. to Maurice Cane, July 1775, www.loc.gov/item/mgw443691

Friedrich L. Bauer, *Decrypted Secrets: Methods and Maxims of Cryptology*, 4th edition, Berlin: Springer-Verlag, 2007, pages 44-45.

Johannes Trithemius, *Polygraphiae libri sex*, Reichenau: Joannis Haselberg de Aia, 1518, www.loc.gov/item/32017914, book 6.

Blaise de Vigenère, *Traicté des chiffres ou secrètes manières d'escrire*, Paris: Abel l'Angelier, 1586, HDL: 2027/ien.35552000251008, gallica.bnf.fr/ark:/12148/bpt6k1040608n, gallica.bnf.fr/ark:/12148/bpt6k94009991, pages 286-343.

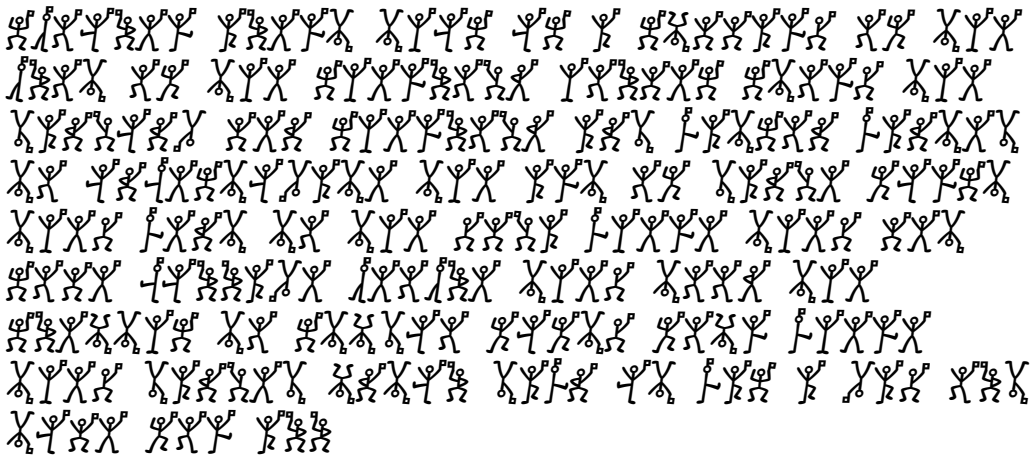
Martin Gardner, *Codes, Ciphers and Secret Writing*, New York: Simon & Schuster, 1972, sections 4, 6, and 7.

Exercises

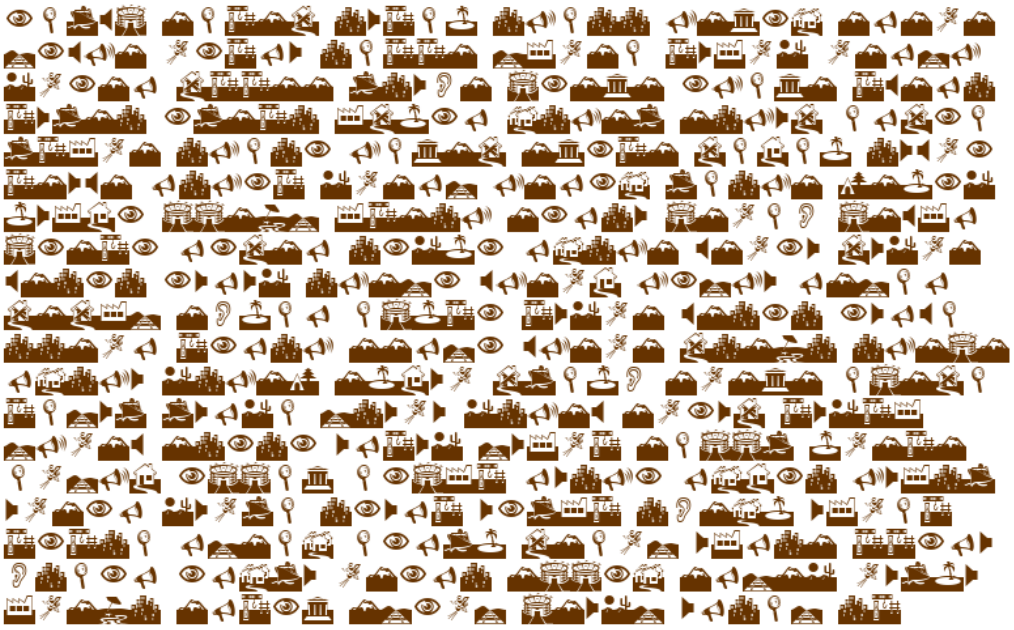
1. This is the ciphertext from Edgar Allan Poe's "The Gold-Bug" (the original version). Break it.

```
53‡‡†305))6*;4826)4‡.)4‡);806*;48†8
¶60))85;1‡(;:‡*8†83(88)5*†;46(;88*96
*?;8)*‡(;485);5*†2:*‡(;4956*2(5*—4)8
¶8*;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡
1;48†85;4)485†528806*81(‡9;48;(88;4
(‡?34;48)4‡;161;:188;‡?;
```

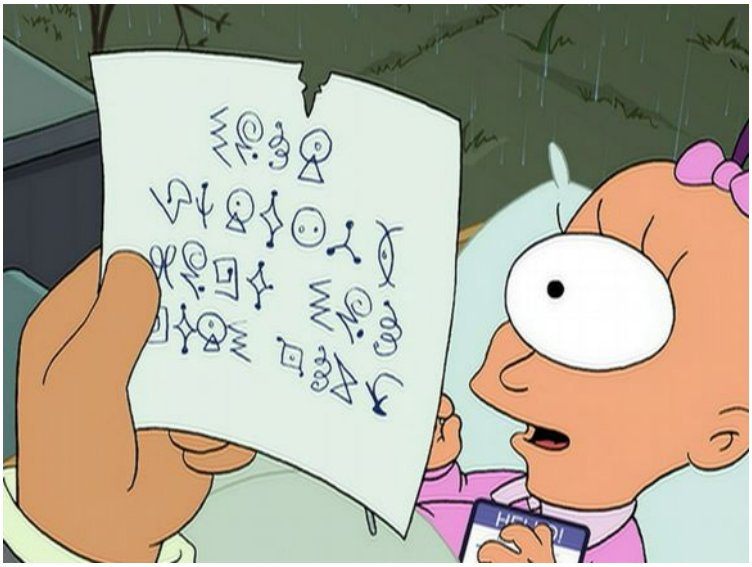
2. This ciphertext uses the symbols from the Sherlock Holmes story "The Adventure of the Dancing Men." Some symbols did not appear in the story and were designed by the font's creator. Break the ciphertext.



3. This ciphertext is from the 2002 British National Cipher Challenge. Break it.



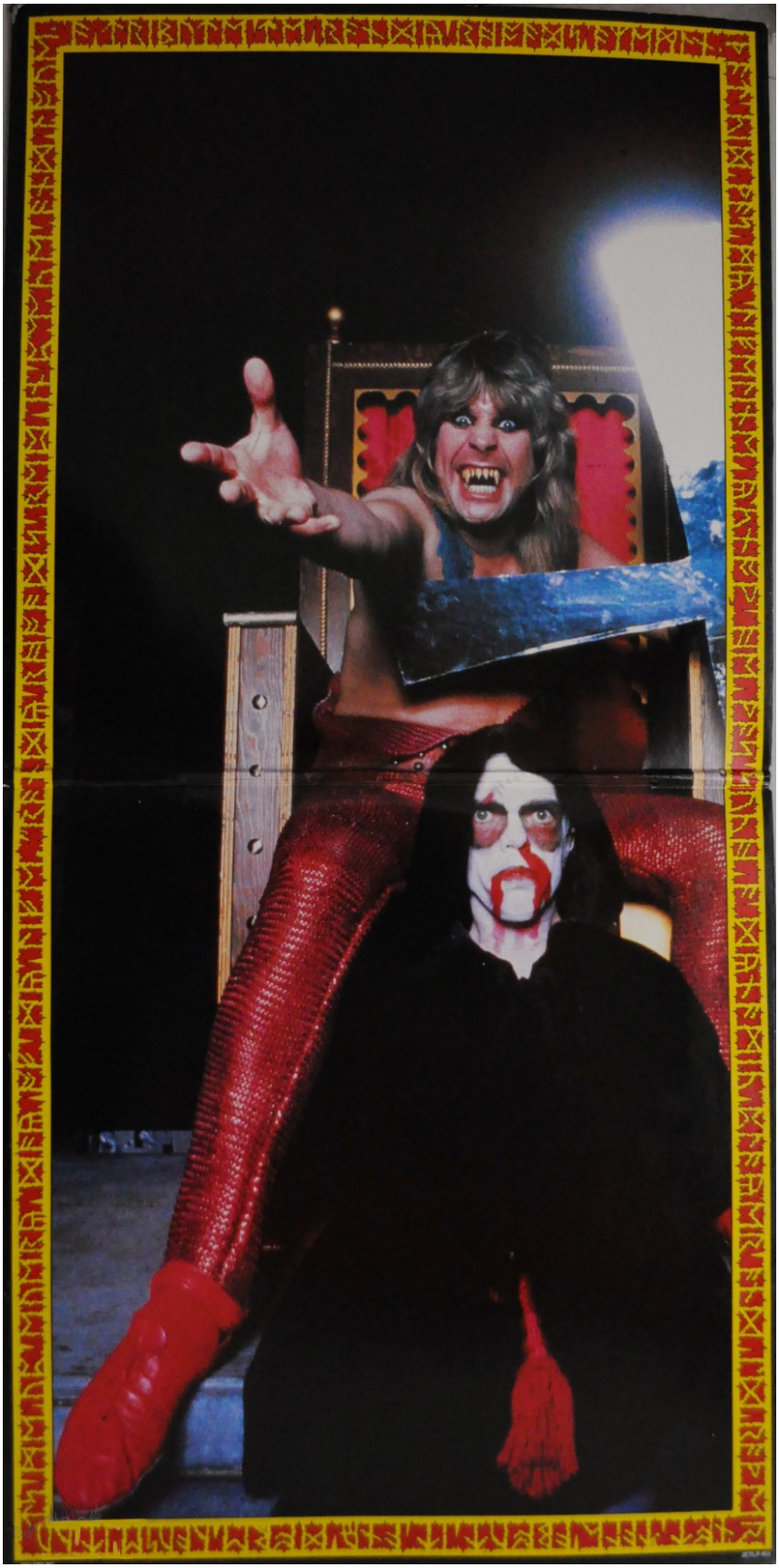
4. What does the note say?



5. ᐳᑎᒥᐯ ᒥᐯ ᐳᑎᗅ ᖁᒥᒣᖁᗅᗅ ᒪᒥᖁᑎᗅᒥ ᒍᐊᐺᕮ ᑌᗅᕮᐱᗅ ᒍᐯ ᐳᑎᗅ ᕮᒥᗅᗅᕃᒍᕮᗅᐯ ᒪᒥᖁᑎᗅᒥ ᕮᒥ ᐳᑎᗅ ᒥᕮᐯᒥᒪᒥᐸᒪᒥᒍᗅᐯ ᒪᒥᖁᑎᗅᒥ

ᐸᐧᐃᐸ ᖁᒥᐧᐃᑌᒍᒍᑌᒪᐧᐸ ᒍᑌᒪᒥᐧᑎᒍᒐᐸ ᑌᐧᐧᐃᐧᐯ ᒍᑌᐧᐃᐸᐳ ᒥᐳ ᐯᒥᐧᑎᒪᑎ ᒪᑎᒥᑌᒐᒍᒥᐧᑎᐧᐧ ᐸᐯᑎ ᒥᐳ ᐧᐃᑕᐳᑎᐧᐧ ᒥᐳᐯ ᐯᑎᒍᖁᑎᐯ ᒍᒪᐧᑎ ᑕᒥᐧᐃᐣ ᒍ ᐳᒥᒪᐳᒍᒪᐳᐧᐃᑎ ᑌᐧᐃᒍᒪᐧᑐ ᒍᐧᑐᑐ ᑕᒥᐧᐃᐣ ᒍᐧᑎ ᐳ ᑌᐧᐃᐳᑎ ᐯᒥᐳᑎ ᒍᐧᑐᑐ ᐯᒥᐳᑎᐧᐃᐸᐳ ᒍ ᑐᐧᐃᐳ

6. What is Ozzy trying to tell us?



Unit 106
One-time pad

The *one-time pad* (*Vernam's cipher*) employs a key that has the same length as the plaintext. The key is combined with the plaintext to produce the ciphertext, usually in a Vigenère-like manner. Because for any ciphertext there exists a key that can decipher it to *any* plaintext, the one-time pad is the only provably secure cipher. The problem with it is in key distribution: The sender and receiver must have access to the same key. Sending the key introduces insecurities, and carrying a large codebook of pregenerated key material is inconvenient.

We can view the one-time pad as a limiting case of other ciphers. It can be seen as a Vigenère cipher in the limit of a long key or infinite period. It can be viewed as a stream cipher in the limit of perfectly random key generation.

The key must *never* be reused. If two messages encrypted with the same key are intercepted, then the security of the cipher is destroyed. Also, the key must be random, lest the adversary find a pattern in your key.

Reading and references

Claude E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal* 27:3 (1948) 379-423, DOI: 10.1002/j.1538-7305.1948.tb01338.x, HDL: 11858/00-001M-0000-002C-4314-2

Wikipedia, en.wikipedia.org/wiki/One-time_pad

users.telenet.be/d.rijmenants/en/onetimepad.htm

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 394-403.

Programming tasks

You can just use your routines for the Vigenère cipher for this one.

Exercises

1. These five ciphertexts were encrypted with the same key. Recover the texts and the key. This may require some trial and error.

```
DNBYDYWSXBNXHPRWYBHMSGHEXVRINUBSQIXEBHBQKUUHAFHXWJSNSN
JGALTYUVCGLXZIWSD

DNBTZHDGGAYBRSWVFZABDLHTNXKLQLINQKAXBCPPPWIQBIXERVIZTL
JNIWVLJQ

SZTCNAOOOWNIZXBXVEHMUDQANIFYAZMFROQYTLTGUWTGJIOKFJPEYC
KBELERWRBWNYVSJU

DNBHMZQHJXISVEBEGMUUMCSJRWKHUZTKAKUXXTSBPVBGYPYUDNMSFL
HBHSZQMSJFCNNG

KYPVZAKOTUYGSXQUJYBUQDSJRMDMNUFYCZKYCZSCOLYCWFWKXTFNDA
JRHSUNWVWLSMXD
```

2. These two ciphertexts were encrypted with the same key. See if you can recover the texts and key.

```
PPHKLXETAGOTLJGODXRODJBSJQGYGXJLGTKSGPHBTFEJSDXKEESUWL
DDPVCVPKPZQMGFAEFTQLUKSBEJLHHHSKZFXNOYIDGMQOICSMZQQIUF
YQBGCEZXVDJTBOGTXDRPWRIDTORXABUGDGIRGGBYHYRIRIJEKTLOMK
IMKERZHFGZVCMOORRAMIHIRERZCPLCYFJKWPQRSHKTQIHBAYZXLBZO
UJZRXNYBGIEVWJUXBSTOGWABQHGHPUCRISHTSTHDCCRAWABIZPEOWQ
VOBJYYSCNUUPYUFOVBZAOSCSODBKWGWRRVUQQRHDVHTJYSLGZRHWAK
WYWTNPITYMPCQSRUKAHSJLKCZYEIDOEDVAHFIQGHNYODMLOTWANYAG
XIHUVSNMGVLKYHBJTNUSVFFAFNKNMBMZPCN

HRETWPKMQUJEBMKEIMIKXNEEMHOZIETHUNQFKTKWPERUSDTRDTWJIA
RUGMOSLDCRFTHPMOJTRHWDEIYUKUCQNRFGRJDGFOABXJCMPNRPBUUP
RHMKXOGFCFJEJFWPKGRPMGPGCKHBTFPVUPIIKBQSBFEJQQCENZWDUW
IXUOVXSMMVACYCYRWCAQOGICTCKWXJGIVPVQQUNYXJBMHDIIXDLYJQ
RTGECARZHPPCHMFDRCUPJHIXDRXNPRUDQGVRPDQTVEEFBZVNNZEPCU
RLANGYFOPVUBLRPVAEUMJZHAUDAKQUBHLKBSFLEGYJCFDSXDCXHUUL
MIAWDJXESREIEWQCGECWCFLGMMEDTXFCMSFSDWTSZWAXCIZEJIJANP
AWLWTFMCAEBKMZFUUYVPVAPPDCPTLVXOMROVANDAUH
```

Unit 107
Slidefair cipher

The *slidefair cipher* is a periodic digram substitution cipher. The mechanics of the cipher involves sliding one copy of the alphabet against another. The plaintext is divided into digrams, with padding if necessary. For each letter of the keyword, the lower alphabet is slid until that letter is under the ‘A’ of the upper alphabet. The two plaintext letters in a digram form the corners of a rectangle, with the first in the upper alphabet and the second in the lower. The ciphertext letters are taken as the other two corners of the rectangle, again with the first in the upper alphabet and the second in the lower. If the two plaintext letters are in the same column, then the ciphertext letters are taken as the two letters in the column immediately to their right.

As an example, let’s encipher this message with the keyword `FAIR`.

```
THIS MESSAGE WAS ENCRYPTED WITH SLIDEFAIR
```

First, divide the plaintext into digrams:

```
TH IS ME SS AG EW AS EN CR YP TE DW IT HS LI DE FA IR
```

To encrypt the first digram, `TH`, we line up the alphabets so that the first letter of the keyword is under the ‘A’ in the upper alphabet.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
. . . A B C D E F G H I J K L M N O P Q R S T U V W X Y Z A B C D E F G H . . .
```

The ciphertext letters are read from the same columns as the plaintext letters, starting with the one in the upper alphabet: `CY`. For the next digram, we use the second of the keyword, ‘A.’ This continues as we cycle through the keyword. The final ciphertext is

```
CYSIWUBJBFWEKIWVMHPYWBFUONSHATNUVKRI
```

There are three variations of the slidefair cipher, corresponding to the three ways of arranging plaintext and ciphertext alphabets in the Vigenère, Beaufort, and variant Beaufort ciphers. The example above used the Vigenère-like variation. For the variant-Beaufort variation, we use the same prescription, with the exception that the keyword letter is taken in the upper alphabet and aligned with the ‘A’ in the lower alphabet. For the Beaufort variation, the lower alphabet is written in reverse order.

If we have a ciphertext that is sufficiently long, we can find the length of its keyword with the method of Unit 31. The period that we find with that method is twice the length of the keyword, since we encipher twice the number of letters in one period.

A hill-climbing attack on the slidefair can be done in the same way as we did for the Vigenère cipher in Unit 37, by varying each letter of the keyword to improve the fitness of the plaintext. When we can no longer increase the fitness by changing any letter in the keyword, we have likely found the correct one.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; page 199.

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Slidefair.pdf

Programming tasks

1. Implement an encryptor. Allow for the option of choosing one of the three variations of the cipher.
2. Implement a decryptor. Allow for the option of choosing one of the three variations of the cipher.
3. Implement a dictionary attack. Check all three variations of the cipher.
4. Implement the hill-climbing attack described above. Allow for the choice of the three variations of the cipher.

Exercises

1. Encipher this text with keyword `RULER`. Do it three times, once for each variation of the cipher.

   > The slide rule is a device for easily and quickly multiplying, dividing and extracting square root and cube root. It will also perform any combination of these processes. On this account, it is found extremely useful by students and teachers in schools and colleges, by engineers, architects, draftsmen, surveyors, chemists, and many others.
   >
   > (from *Instruction for Using a Slide Rule* by W. Stanley)

2. Decipher this ciphertext with keyword `CIRCLE` and the Vigenère-like variation of the cipher.

```
FVVMCRNCTCAWRPVMWZFVHTNEGVDKBVMJIHWLUVAPNCFVTPNEGNKMBK
GNTOQVCNOAAVMGIQALGEUZVMPCNEATFVDMCRPGADZMPGDCBVDQWEOI
TGFMNKLGWEAGRPJCFPPGDPPJCJWZDUJKCTWISUHGGV
```

3. Decipher this ciphertext with keyword ARGUE and the Beaufort-like variation of the cipher.

```
NMNPTSBUSWHWNKCPQYANEHLDSSMRMATEBDGMDDTAWPROKOIGRAVMNY
KTJMETXNKVTCNBGAATEONSMNYRPWNZSNAENEWJRGSFBAXLCWNMVVGB
WZTUJYATTUKQHHNKPYANDMNAZOOGGBXIYSVKOGNBMAPSJGOCGBLZWT
UF
```

4. Perform a dictionary attack on this ciphertext. The keyword is a common five-letter English word.

```
GMUIVVNAVLFQRCQKPEGPCRYFWRECXDGFQPXWERWEQQDIJKNICZVFNS
JFDRPDRZAZTFISIYHAQVJWRIULSQUZPEFOGATAGCMEEMPWQZAVHCOC
TXCEQJQIMICTWTEGALEVDJHRQVDSTSXWUZXDLRYEHPVFEWBZAVXYMF
MTAZONPECYMFYCALTNSOFFRRLNCJFAYOAVYRCZHQAVPLVHGPBKUCHJ
HTRPFGMHWZIWAWSNBIAGSECPZFMMMEEHXHZXQPDKNRCTCSQJQIAEPO
RZQVUVASSYFOUMPEOHPHSDAMGL
```

5. Find the length of the keyword used to encrypt this ciphertext. Break it with the hill-climbing attack.

```
UGSYGABUJSNMQFARCNMMNUKPJTMLUGJYQQLJHEXGVAVGZMAAJBWIND
BFYNAGHXOJEPJTQLABYOBUNHHRCPMZVSYLRFFJPMRJBFRUOASHEOYE
XKAJUGTYZBBBPINYMVVPGGDXHPPRWZPLFROFSPTSAPNICKGYUMVFSB
PMEJTSRHYCSZQBOICKIZBGZZSTDNPRURCEGYVCWXZDEXWF
```

Unit 108
Nicodemus cipher

This is a real dog of a cipher. To encipher a text with the *Nicodemus cipher*, first encrypt with a Vigenère cipher, then break the text into blocks of five times the key length and apply a columnar transposition to each block, using the same key as the Vigenère. Here is a short example:

```
KEY       KEY       EKY
———       ———       ———
THI       DLG       LDG
SME       CQC       QCC
SSA       CWY       WCY
GEI       QIG       IQG
SEN       CIL       ICL

CRY  →    MVW  →    VMW
PTE       ZXC       XZC
DWI       NAG       ANG
THN       DLL       LDL
ICO       SGM       GSM

DEM       NIK       INK
US        EW        WE
```

The ciphertext is read off one block at a time, including the final, incomplete block:

```
LQWIIDCCQCGCYGLVXALGMZNDSWCGLMIWNEK
```

Variations on this cipher can use a different block size, or replace the Vigenère cipher with Beaufort, variant Beaufort, or any other polyalphabetic cipher.

Here is how to break a ciphertext encrypted with Nicodemus:

1. Guess the length of the keyword. You probably guessed wrong, so try a range of values.
2. Reconstruct the columns by taking five characters from each block. Ignore the final, incomplete block for the time being.

3. For each column, decrypt it as a Caesar shift cipher. Use the frequency-matching technique from Unit 19. You will, after this step, have a text that is only encrypted with a columnar transposition and a keyword that has been scrambled with the same permutation.

4. Break the text you found in step 3 as a columnar transposition cipher. Use the hill-climbing technique from Unit 60.

5. Use the key from step 4 to unscramble the keyword.

6. Use the keyword to decrypt the full ciphertext. Now the final block is included, and you get the full plaintext.

Reading and references

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain; page 216.

American Cryptogram Association, "The ACA and You," www.cryptogram.org/cdb/aca.info/aca.and.you/aca.and.you.pdf, 2005 edition: web.archive.org/web/*/http://www.cryptogram.org/cdb/aca.info/aca.and.you/aca.and.you.pdf, 2016 edition: web.archive.org/web/20160418014322/http://cryptogram.org/docs/acayou16.pdf; the relevant page is also at www.cryptogram.org/downloads/aca.info/ciphers/Nicodemus.pdf

Programming tasks

1. Implement an encryptor and a decryptor. Feel free to import your routines for Vigenère and columnar transposition ciphers.

2. Implement the attack. The key length can be an input, or your program can try a range of key lengths. Again, feel free to import some of your other routines.

Exercises

1. Encipher with the keyword `GEOMETRY`:

   > I will not, from henceforward, talk to any squarer of the circle, trisector of the angle, duplicator of the cube, constructor of perpetual motion, subverter of gravitation, stagnator of the earth, builder of the universe, etc. I will receive any writings or books which require no answer, and read them when I please.
   >
   > (from *A Budget of Paradoxes* by Augustus De Morgan)

2. Decipher with keyword `MURDER`:

```
KWRQDIMJXQAQFSMKPGRJMGRSNHLICFKHHBHLRRXVTPDRQTVBSZRSFY
TCVIZZQORNWXIHMLQDZYSZRTJWVJVEVQACOIZBDUXGLVCMOXPRMFEV
VSFRRVCLISNLVRKIORYIXWFEQAQZPKFRVJKYRBNYJVLNITERISYVYN
AYFVFWURXEIZFVLZYYOPKVUENITEMREQFGIKPFKLMFTXINLZVUOHKR
SSMIDEUFTNWJYKVDWXNINNSNHXFBUWRSPITDUZKCFXIGLULRRILHNM
WWDPLFLKEHPZUQNLRVKJVCIGVIYEUYHDDWDLPWSJESMGQYSNVJZJJS
VNBGLUWHDKFJXWEPEDQQPFYNPFFIMIZJNYLNWFRWJWLAWVYDEZOZEI
RRSVZLVVUAABLHKQWJIHMMADUTKLKIXNHGI
```

3. Break this ciphertext:

```
ZIMSHCUJAROQDMOVOFSNZJVXRDEKFLWRQLEANXAFKXOQISWRWOLZLK
SJUKSMSSMEICJBJCDTXOCHMCGCGBKJVDWOQLLRMKRQJWKNDSOIKOGQ
ZBVTMIUOGSWUMISHSXWPBNPQTKVBUSVUUFEKIAJWZQIELXRXNWRTZB
DCYTHUFSIKVAYHAXGAKQVJEVNRCNOSKKBVVUVQRRNZTNUAJCRAXVGN
JTQMSNNMYFBSSHZZFFFLFLOEIIIIRNRUAURQNCKHBCSMKOUEUZHHLK
XLSHIFJDRDZDUYCSMZQGRVWVPOTFEWACIKWJYORFRGJVJSD
```

Unit 109
Fractionated Morse

The *fractionated Morse cipher* first encodes a text with Morse code using dots, dashes, and × between letters, and ×× between words. The stream of symbols is taken three at a time and replaced with letters according to an alphabet key that may be mixed by use of a keyword. Since ××× never occurs, there are $3^3-1 = 26$ combinations, which fits the number of letters that we can use. If necessary, we pad with one or two × symbols. The fractionation occurs when symbols from the same plaintext letter are separated and become part of different ciphertext letters.

As an example, let's encipher a short message with the keyword `FRACTION`. If our message is

```
HIDE THE CANDIES UNDER THE PILLOW
```

we first encode it in Morse code, with × as the separator:

```
H    I  D   E  T H    E  C    A  N  D   I  E S
····×··×–··×·××–×···×·××–·–·×·–×–·×–··×··×·×···××

U   N  D   E R    T H    E  P    I  L    L    O   W
··–×–·×–··×·×·–·××–×···×·××·––·×··×·–··×·–··×–––×·––
```

The resulting symbol stream is divided into groups of three. We need to pad the last group with two × symbols.

```
···  ·×·  ·×–  ··×  ·××  –×·  ···  ×·×  ×–·  –·×  ·–×  –·×  –··  ×··  ×·×
···  ××·  ·–×  –·×  –··  ×·×  ·–·  ××–  ×··  ··×  ·××  ·––  ·×·  ·×·  –··
×·–  ··×  –––  ×·–  –××
```

Now let's set up our key. Each letter in the mixed alphabet is associated with one triplet of symbols.

```
 F R A C T I O N B D E G H J K L M P Q S U V W X Y Z
 ──────────────────────────────────────────────────
 · · · · · · · · · – – – – – – – – – × × × × × × × ×
 · · · – – – × × × · · · – – – × × × · · · – – – × ×
 · – × · – × · – × · – × · – × · – × · – × · – × · –
```

By replacing each group of three symbols with the letter that appears above it (where the groups are written vertically), we get the ciphertext:

FONABLFUVGIGDQUFYIGDUCZQABTOODSAJSP

We can attack the fractionated Morse cipher with the same hill-climbing attack in Unit 28. However, we will not be seeking the best plaintext, but rather the best match to English text that was enciphered with the key ABCDEFGHIJKLMNOPQRSTUVWXYZ. So, we must encipher a corpus of text with this key and from it compile a table of tetragram frequencies for the attack to use. We must also augment the attack with a margin of error and allow downward steps about 5% of the time, as we did for the attacks on grid-based ciphers, in order to avoid becoming trapped in a local maximum. A good margin to use is 0.5. Once we have found a key that gives the best match to English text that was enciphered with the key ABCDEFGHIJKLMNOPQRSTUVWXYZ, we can use that key to simply decipher the ciphertext. To say this another way: we are factoring the cipher into a fractionated Morse cipher that has the standard alphabet as its key, followed by a monoalphabetic substitution; first we break the substitution, then we decipher the fractionated Morse.

If instead of three Morse symbols per ciphertext symbol we use two, then the cipher is called the *morbit cipher*. This cipher is keyed with a nine-letter keyword, which is then converted to digits in the same way as with a permutation cipher. So, for example, if the keyword is FRACTIONS, the substitution uses this table (note that most people use the digits 1-9 rather than 0-8):

```
F R A C T I O N S
3 7 1 2 9 4 6 5 8
-----------------
. . . – – – × × ×
. – × . – × . – ×
```

A plaintext such as

THIS IS MORBIT

is thus enciphered with that key:

```
T H     I   S     I   S     M   O    R    B     I   T
–×.···×··×···××··×···××––×–––×·–·×–···×··×·×
4 3 3 6 1 3 1 6 1 3 1 5 4 9 4 7 1 2 3 6 1 4
```

Reading and references

Practical Cryptography, practicalcryptography.com/ciphers/fractionated-morse-cipher

American Cryptogram Association,

www.cryptogram.org/downloads/aca.info/ciphers/FractionatedMorse.pdf and
www.cryptogram.org/downloads/aca.info/ciphers/Morbit.pdf

Programming tasks

1. Implement an encryptor for fractionated Morse.
2. Implement a decryptor for fractionated Morse.
3. Implement a dictionary attack on fractionated Morse. Remember that there are many ways to construct a mixed alphabet from a keyword.
4. Implement the hill-climbing attack described above.
    a. Encrypt your textual corpus that contains spaces but no punctuation with the key `ABCDEFGHIJKLMNOPQRSTUVWXYZ`.
    b. Compile a table of tetragram frequencies from the encrypted corpus.
    c. Put the pieces together.
5. Implement an encryptor for morbit.
6. Implement a decryptor for morbit.
7. Modify the hill-climbing attack on the fractionated Morse cipher to attack the morbit cipher. You will need to compile a new frequency table.

Exercises

1. Encipher this text with the keyword `LINGER`. Use the same method for constructing the mixed alphabet as in the example above.

    > Two hours. One hundred and twenty minutes. Anything might be done in that time. Anything. Nothing. Oh, he had had hundreds of hours, and what had he done with them? Wasted them, spilt the precious minutes as though his reservoir were inexhaustible.
    >
    > (from *Crome Yellow* by Aldous Huxley)

2. Decipher this ciphertext with the keyword `ODIN`. Use the same method for constructing the mixed alphabet as in the example above.

```
LVVJESICJCMBOTVWQSICJOSQCQEIBJLXOCTEQMLVTVCCNSJCJEESCB
JBJGTVCCNSJCJEEWWQMJDCJLWXOCWQCSCMCICVWPMOWPGTPPIPOUJB
MEMDESIBPIDVTVVEPQCCMBOXOCVTWULSJPOTXMLWQEOSQTTIATVMOV
ICOSVVCILTIBJGWTVSTQMNCDUOTPVXBAWPGSJGCSIIJGUBHCOXOCTQ
```

```
MNCDUOVJBDWCOXLVXOCWTXNCPIPQEIPOUHQICDWPBNUGSEMNCQEIMB
JIBJGXOCVXOCTQMNTIGTPPIIJGTTEBJCIOTVVENWPGVF
```

3. Break this ciphertext with a dictionary attack.

```
FHRNJLMKCRAKPSYMEWYMRNJYUKRMAAFKPSGRMSFLEJYSPWHUWPMVMF
MUVSICVKJPRUUMEEQAMAJAYMEEFTEJHRAFHUSDMNEYVNKGRRMRKEQA
TMNACUJKPSCMRQIMFFHWMLSFPMMADEJLQMEMHSYRETVQKADRMNQAIT
MGUJTKNYUKAGKIAYRMRAIFCQERMMGSCUFPMMGVQNEIPWHRMQKGQFHV
NALEJYWPMRAHUVEYVKCUENJFKPQMJPMUREJLVMPUAYVNQAMFFHREQA
TQUUMNEATKNYUKYREJPWYQKCSIFCRJMFYUJJLQEIYRWCMIAYEAPRQF
AFMPVKIPSTQAMFMEEVNIMRFTRREKNEADEJJLEJYWICMCSCMIKKCRAK
PQQMMAIPUVNQAAFGWHWPMRNJYUNJLMGRUEADNQAEPSIPKRB
```

4. Break this ciphertext with a hill-climbing attack. What is the keyword?

```
NTQQMKATXTMQHLGQBSMQMIWOMLLDMDURBKTQXLJVQQLADWOGVHESHL
BLLHNTUNSLBHGSSHTMLMHTNBGUAHDQSNNLTWNOBEQWNATXJVLMOQLA
HDQVVHATULTQLJQNDVBTUHMVMVBQLAOKHBMBHTBQAMHFNDSXMMLANB
BQLBKEMGQVHBLESMBBHATXJSBOBKNKLTQLLHIXJVHEXJQHEBQLBGWO
BMVHNBKHLTNVXESMVBWOKHMNHNTUFUFNOBLEBKGWVKDBKGBLHMLTSN
NLFVWOGVNVBKEMGWVKTXJWNDSQHGMNLTBKEMDQWMTXHLHIQAMHFOMS
SMDVKBBLBLKFOGSSMUGSQHNSBQLANMKATVWOUDQVXTMWNGQHLFMQHJ
SBGSQOQWOBLESMBBHGMBHTBVQVWMLOKGWQC
```

5. Encipher this text with the morbit cipher with the keyword `BACKWARDS`.

> A snip of the scissors, and six inches of white beard fell to the floor. For the first time in thirty years Mr. Simpson felt a razor on his face. Then his hair was cut and shampooed; and an hour later he sat gazing at a dark-haired, clean-shaven man in the glass who gazed back at him with wondering eyes.
>
> (from *Stepping Backwards* by W. W. Jacobs)

6. Decipher this text with the morbit cipher and keyword `EMPLOYING`.

```
91135312162113954378843612911399373188429113251391896 7
35863313131218791533433878431211394861888421854368931 3
18837391835685434673191613612591631867631885818667295 6
53291133858378513944325911385132794256584467356896735 8
63188313391879421811321467356848389384658858853138464 34
84724813581438443534896672788936954348794769846
```

7. Break this ciphertext which was encrypted with morbit.

```
45753474349861687845498437267567764731376449843726737 6
37475477223646477586164687347464724362549434944371343 3
27429423236462746843576362549274343471623345964436631 3
98236432947163133278496161776443 32
```

Unit 110
Hutton cipher

The *Hutton cipher* was invented by Eric Hutton to challenge strangers on the internet. Its key is a pair of keywords; one generates a mixed alphabet (where the remaining letters are added starting with the beginning of the standard alphabet), while the other serves as a set of shifts that are applied periodically (for this cipher, ‘A’ = 1, ‘B’ = 2, ..., ‘Z’ = 26). This may sound like a quagmire 3 cipher, but there is an added complication: after each letter is enciphered, the plaintext letter and ciphertext letter swap position in the alphabet key.

As usual, we will work out an example. Suppose we want to encipher this message with keywords `HUTTON` and `CIPHER`:

`THIS MESSAGE WAS ENCRYPTED WITH HUTTON CIPHER`

We begin by mixing the alphabet with the first keyword. Remember that for this cipher, we add the unused letters starting from the beginning of the standard alphabet.

`HUTONABCDEFGIJKLMPQRSVWXYZ`

The first plaintext letter is ‘T,’ and the first shift is ‘C’ = 3, so the first ciphertext letter is ‘A’:

`HUTONABCDEFGIJKLMPQRSVWXYZ`
`  3→`

Then we swap the two letters in the alphabet key:

`HUAONTBCDEFGIJKLMPQRSVWXYZ`

The next plaintext letter is ‘H,’ and the next shift is ‘I’ = 9, so the second ciphertext letter is ‘E’:

`HUAONTBCDEFGIJKLMPQRSVWXYZ`

So we swap those letters:

`EUAONTBCDHFGIJKLMPQRSVWXYZ`

Next comes ‘I’ which we shift by ‘P’ = 16. We wrap around and get ‘A.’

```
EUAONTBCDHFGIJKLMPQRSVWXYZ
```

This continues until we have the full ciphertext:

```
AEAIVQTKANRFZVABZIJZITDURFVLZVSZDPUC
```

There is a variation, known as *Hutton cipher 2*, in which the shift is increased by the value of the first letter of the alphabet key at any given time. For reference, when enciphered with Hutton 2 and the same keywords, the message above gives

```
JPQIGHMFJOOZXBEJVUMUKJKNYDWEFYSABVKR
```

Reading and references

hutton-cipher.netlify.app/howto.html

Programming tasks

1. Implement an encryptor. Allow for the choice of cipher variation.
2. Implement a decryptor. Allow for the choice of cipher variation.
3. Implement a dictionary attack. Allow for ... you get the idea.

Exercises

1. Write an essay about why the inventor should have used ‘A’ = 0, ‘B’ = 1, ..., ‘Z’ = 25.
2. Encipher this text with Hutton 1 and keywords `GEOLOGY` (mixing) and `EARTH` (shifts).

   > We know little of the earth’s internal parts, or of the materials which compose it at any considerable depth below the surface. But upon the surface of this globe, the more inert matter is replenished with plants, and with animal and intellectual beings.
   >
   > (from *Theory of the Earth*, Volume 1 by James Hutton)

3. Encipher this text with Hutton 2 and keywords `OCEAN` (mixing) and `LAND` (shifts).

   > By the present theory, the earth on which we dwell is represented as having been formed originally in horizontal strata at the bottom of the ocean; hence it should appear, that the land, in having been raised from the sea, and thus placed upon a

higher level, had been of a different shape and condition from that in which we find it at the present time.

(from *Theory of the Earth*, Volume 2 by James Hutton)

4. Decipher this ciphertext with Hutton 1 and keywords `VALLEY` (mixing) and `HILL` (shifts).

```
BSDULXXUPXKHLZXVJRKOQZVXUVOGQHZAGHSSELJXQSKHVLHMBEDGCC
AXNAYIAZGFNMVFFEEUFPGIYEIDZTPJYNNBLGSNXUYVTWYTIFLUOJTN
RTTPLBDEUPJMVMNMWAQRMTTQPLBBOGDLUIUJTXZIZXUOBDPDABMNNU
JJAMUYDDDCXWYFHDHWPNUALTPYYMKVTGJIXFBMXMGIWNZEHDUAFCWZ
KIIHQIQVEWKIEFILDPEBSGNUOCFDQMEWXXTLJVS
```

5. Decipher this ciphertext with Hutton 2 and keywords `RIVER` (mixing) and `SEA` (shifts).

```
KDXLRZUWOVTDBCZBXMFVIGBWTXMKIDVEOFYBAJSJBUOBQRALSLQTGN
JUGFJZXCIOZUEVOKOKBAMMORVLQREGSGVMGVBZPDCDNRHOSPGEZXOC
BCETWNDJLYMRRIVUQGTLRZFBIYBLHBMGIVPDHASDMJLTPLUJCUBSBB
UPQJEBRBDBTVGNWELCHCMIWBFUPSSNKBPPXUH
```

6. Perform a dictionary attack on this ciphertext. It was encrypted with Hutton 1. Both keywords are five-letter English words.

```
ZVFPHUFWADZGGCAXLVLCWBKQGYMFKFWWWHHNOLADQIRTEZHPBHKFSU
UQZFNMQRLMGCVZFWEJYIGYNCMGAVLXPAUQHKCRCXGENGTKKFUILCKK
MOQACODGRSJYMJHYXKMVBMQFZQULXZMUQXMHWAFITBDLSIMIHCGLVA
IGLICLRVBNNZFPGJYJZPLQIGNUZASDGJVGPNGGHEXZCYVCNCLLWECG
REILNCNOKYKLSCGJJWPUWUOON
```

7. Perform a dictionary attack on this ciphertext. It was encrypted with Hutton 2. Both keywords are five-letter English words.

```
SABOOKONJEMXGEUWGFUDDZOOFCTMJUVATUYEQYNCNFLBHZSGZHJUTK
WGCCYQLYETQCCSUHMRYHQKJVNPXILEHYNHJEKQGKHEXTZSSGINBQTA
JKOGEPHOMKYATHZHPCCIBXQLOSVNVHLDMYBRZQFLMAWIQLZJGYHLJK
TSMZVMTZXYOOZEPMNJBZTRZMRSSMICIDAGNDSNYNFIRFYXXRHYVFMI
NDSVVROMLNVLTMEIFOXLUSMZJEUIXHMURCMOHTVOHYNNSDLYLUUPAL
DFRGDQUUYPVVQRXHRUEWWFSCLLTWMLBBPXLNJEVUGOQFZVHAHVAPKU
HIVGXJZVKLYTBJTGIIQEDRXUGKUUVSPFXTJCYVPCKOTTZSBKAKGWCE
QGHGIJPOYELOMJSOPOSNBDFDRSJWKTEDUCFMIE
```

Unit 111
Scrabble cipher

The *Scrabble cipher* is an invention of this author in response to the Hutton cipher. It is an attempt to create a cipher with a similar level of security but with simpler rules. Scrabble™ is a trademark of some large corporation. They seem like nice people, and we are sure they would be happy to let us use the name, but we haven't had time to ask them yet.

The basis of the Scrabble cipher is the monoalphabetic substitution. When each letter of the plaintext is enciphered, it will be done with an alphabet key. That key, however, changes with each letter that we encipher.

The first thing to do is choose the initial key. There are four choices for how this is done.

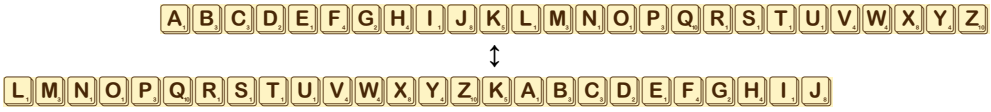
1. Use a keyword to generate a mixed alphabet. For example, if our keyword is KEYWORD, then we have

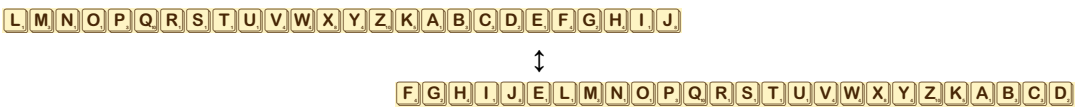
2. Shuffle the alphabet with a keyword. Here is our example with the keyword KEYWORD. We begin with the standard alphabet:

   For each letter in the keyword, we do a shuffling step that swaps the letters before and after our key letter. If there are no letters in one of those groups, we just do the swap anyway, with one set being empty. So, we start with the standard alphabet and swap around 'K':

   Then we take the result and apply a swap around the next key letter, 'E':

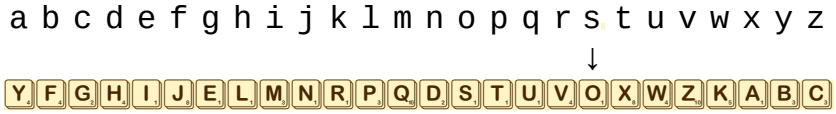
We continue and use the remaining letters of KEYWORD to get this keyed alphabet:

3. Generate an alphabet from a keyword as in option 1 above, then shuffle it with another keyword as in option 2.

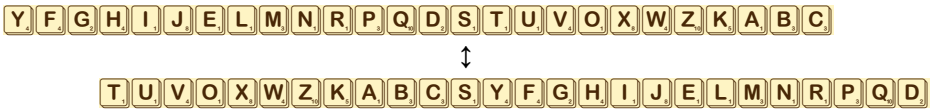
4. Use a randomly permuted alphabet.

The encryption process employs the same shuffling step as we used above in creating type 2 keys. For each character of the plaintext, a substitution is made, and then a shuffling. Each shuffling is swapping the block of letters before the plaintext character with the block of letters after it.

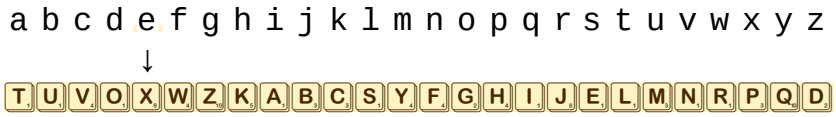
As an example, suppose we want to use the alphabet key from keying option 2 above and encrypt the message “send help”. First, find the substitution for ‘s’:



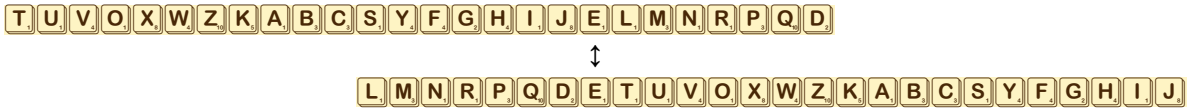
We get an ‘O’. Next, shuffle the alphabet around the letter ‘S’:



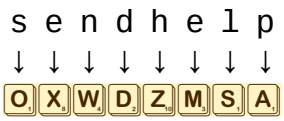
This new alphabet will be used to encrypt the next letter of the message.



We get ‘X.’ Another shuffling, this time around the plaintext letter ‘e’:



This continues until the entire message is encrypted.



Reading and references

Challenges on RingZer0:
ringzer0ctf.com/challenges/300

ringzer0ctf.com/challenges/301

Programming tasks

1. Implement the four keying options.
2. Implement an encryptor.
3. Implement a decryptor.
4. Implement a dictionary attack for the cases in which the key is generated from one or two keywords.

Exercises

1. Encipher this text ...
   a. ... using keying option 1 with keyword `QUACK`.
   b. ... using keying option 2 with keyword `MAVEN`.
   c. ... using keying option 3 with keywords `PLAY` and `HARD`.

   Scrabble™ is a crossword game for humans, but did you know that there are computer programs that can play the game? One such program is Maven by Brian Sheppard. Maven was incorporated into the video-game version of Scrabble™. Quackle is an open-source program that also comes with a graphical interface.

2. Decipher this ciphertext with keying option 1 with keyword `CROSSWORD`.

   ```
   JNNQZVMEEAOLRZNQJXWULPCSSURGDXEEVPMKSYZNBKAQIVUXJVWANQ
   CRSBZEQJWMYCPBAPDOWGNXKQJPPJFZGGHRNUTAXUKHHBHPHIQHFE
   ```

3. Decipher this ciphertext with keying option 2 with keyword `POINTS`.

   ```
   LVMXWUTRDMQJQQNGEIBXHIRHUPOMOZDWDDCHKKJFWTNEMTCTOFGDTT
   GORQFLCGJMWXKRGFECSLKJPALVXJJKVIKEDTJTIPINNLEAWDFVBVMC
   LWKNDXDMAJLFCOBXLNWMVFJEWRTLFCBFWKDDQTUDERPVUJPSHMD
   ```

4. Decipher this ciphertext with keying option 3 with keywords `BOARD` and `GAME`.

   ```
   ZVXDIMUCIXANYNGBSYZIZZFWZTXXZNSKKSYVQCZVCAXECAYOHLPTZO
   CWACHINTHDUEGGHKPQTSKMEGYEEIEEXAXPIONLMACKHWPWQBYQNXTI
   VIZEPGLHAZBONYNEFYRRXVJNNWIBGPRHRHDEAHNBHJBYIEBGCPQBLF
   ZMGEYEUMPRDCMIBXAZWSXYXIYQFKOJMBQBORISUHMGOQZRMXBFSVUX
   ```

```
NRBMVCRNKVEJLOPWPPYTYEGSHDSOYSXIKWLGTMPRUHATHDOIJSZLCO
LSYFIGNOKBVSEUXTZA
```

5. Break this ciphertext with a dictionary attack. It was encrypted with keying option 2.

```
ERMQNRLMWFTKUDJHHUNFKHSIUIMXCXFWJRRXLEXRSBLCHRCIYXQPKZ
ZNGHGDQUMKYIXMLNZFSOHLDJJWOOPHZXTENDRKZHBDGZGJEKGFMIUJ
SZXOFMBGIHJRLBVNXTDHLQJDQCLJMEFJPEQXZJLRCPVLHVAXNOLLUC
XJDMLYMJWNZMUQRJFOESDRWMJZBNQJWEZMPALHOGLFKXVBTUROUYDI
```

6. Break this ciphertext with a dictionary attack. It was encrypted with keying option 3.

```
IMUFQFYMAECPAUAMUSSOYDCVJIOUMIGKVRUHEFTCPIDMAFNORGTREE
PPIIFTMZUNZMTJMNSILAEWEDIUHNIBEATJBDRTDNZNHJBGDQUJDXAG
CFGUWWCANYGQYNERIDVHTTETIQJTMVJTMBLYIAVGFXSEDRWPRVDZCM
ZPQNDQJWETZNFXEPXSDADLDDUMEHDLKHWQBJOVAFASTTPPIYQVKJXQ
REDDKWTKFQCJQTEEWSCJIMIQJOAXKBHZPTHEYZIRQNZMUYMMATDK
```

Challenge 1

Here are a plaintext and its ciphertext. Can you recover the key? The initial key alphabet was generated with keying option 2. Can you also recover the keyword?

```
WHENALANTURINGWASSEARCHINGFORNEWCRYPTANALYSTSTOJOINTHEWORKA
TBLETCHLEYPARKHEUSEDACROSSWORDPUZZLEFROMTHEDAILYTELEGRAPHTO
TESTTHEAPPLICANTSTHOSEWHOCOULDCOMPLETEASECONDCROSSWORDINTHE
ALLOTTEDTIMEWEREINTERVIEWEDANDACCEPTEDINTOTHEPROGRAM

PDMCOMMLKRTAENINDKQUKIMHQLRGOWAAJTMFUSVGEZCWNNCSPYOYNOZRYKO
MRFNPNHKHSQZEGBONPGRTPRZAPBYUXNVUCFDVTZFJTHIGTIEOTWXAWDSDCJ
IKTYIQWJMHZHEZSFMWCXKFMPEHTQGSSXQTNJMXYQUVPUNQKNWQTVHUCCYDQ
DGIFNYEFDAAIPLVZXULXDZVLEDWGMGFAQQUUMFNIHTRZFFSIKVBT
```

Challenge 2

Here are a ciphertext and the beginning of its plaintext. It was encrypted with a randomly scrambled key alphabet. Can you recover the key and decrypt the entire text?

```
SUCIEIEOANRWBUHKWYJTPDUTWRBEZJTOQLFHUDTTNBKBRMOZQJWVNDGPRAW
PUOUREAYNAAKNLPMDUSAJHGWHLCPMXUAVHEFFFENJIZNNEGWUFYRDXPXNDA
LMRPXMLPEKHBJQCVCRUDXGYXKVPUKZUCKQWSMYAICBYNMBFOTQBFCGQEPBM
PYINVFLVZRWZNVCHIXCPKDHCSDZLDJQBPHEXKJBCSILJUDZNXFDMSQZZUMK
FGERXBWNHZPWXQRPQKXRTYWDYLVUG

BLETCHLEYPARKISNOWAMUSEUMRUNBYTHEBLETCHLEYPARKTRUSTANDSUPPO
RTEDBYTHOSEWHOVISITTHESITE...
```

Unit 112
Homophonic substitution

In a *homophonic substitution cipher*, a plaintext symbol is enciphered to one of a set of ciphertext symbols. Each plaintext symbol has its own set of ciphertext symbols, and the sets do not overlap. The members of a set are called the *homophones* of the corresponding plaintext symbol.

In the monoalphabetic substitution cipher, encipherment is a one-to-one function from one set (the plaintext symbols) to another (the ciphertext symbols). In that cipher, it is clear that decipherment is also a one-to-one function and as such is unambiguous. This lack of ambiguity in decipherment is one feature of a good cipher. With the homophonic substitution, encipherment is a one-to-many mapping. If the choice of homophone is made randomly, then encipherment is not *deterministic*, but is *probabilistic*. Decipherment, however, is a many-to-one mapping, and is therefore deterministic. So, while one plaintext can have many different ciphertexts, all of those ciphertexts share the same plaintext. A probabilistic cipher has the advantage of being resistant to brute-force and dictionary attacks, since it is unlikely that such attacks will find a matching ciphertext.

There are several ways to build a homophonic substitution cipher. Here are just a few:

- Assign the same number of ciphertext symbols to each plaintext symbol.
- Assign a number of ciphertext symbols to a plaintext symbol in proportion to the frequency of that plaintext symbol. This hides the frequencies of plaintext letters and makes analysis based on monogram frequencies difficult or impossible.
- Assign ciphertext symbols based on a system, such as a grid.
- For each plaintext symbol, choose a homophone randomly from a set.
- Choose homophones from a set in cyclic order as the text is enciphered.

Let’s work through an example. Here is a short message:

```
this message was encrypted with a homophonic substitution
```

And here is a randomly generated key that assigns two homophones to each plaintext letter:

plaintext:	a b c d e f g h i j k l m n o p q r s t u v w x y z
ciphertext:	m O F r y q Y A d p U V X L w c E l x z a B e j t Z D h i s u H K P g N T v W I J n o R C S b Q f G M k

The first letter of the message, ‘t,’ can be enciphered to ‘z’ or ‘S.’ The second letter, ‘h,’ can be enciphered to ‘A’ or ‘P.’ One possible complete ciphertext is

zAgxWyxCmKueDCyLFRtnSuregSPDPwWJnAwIdiCaOxzdSbzdwI

A hill-climbing attack can break a sufficiently long ciphertext. Since we cannot a priori know how many homophones are assigned to each plaintext letter, and to avoid running off toward a solution that gives the plaintext THETHETHETHE..., we need to define a textual fitness that incorporates both monogram fitness and tetragram fitness. One possibility for this is to use the cosine of the angle between frequency vectors (see Units 7 and 8) as the monogram fitness, and define the full fitness function as

$$F = F_4 (2 - F_1)$$

where F_1 is monogram fitness and F_4 is tetragram fitness. Since F_1 is always between 0 and 1, the expression in parentheses is between 1 and 2. The algorithm for the attack uses the parent/child paradigm. Child keys are modified in one of two ways: randomly chosen homophones are swapped, or a randomly chosen homophone is reassigned to a randomly chosen plaintext letter. Like many times before, we will use a margin of error to allow occasional downward steps, in order to avoid becoming trapped in a local maximum fitness; 0.1 is a good margin to try. Here is the full algorithm:

1. randomly generate an initial parent key
2. decipher the ciphertext with the parent key to get a plaintext
3. set the parent’s fitness to be equal to the fitness of the plaintext
4. set counter to 0
5. while the counter is less than 1,000
    a. copy the parent key into the child key
    b. if we flip a coin and get heads
        i. randomly choose two symbols in the child key
        ii. swap those two elements
    c. if we got tails in the coin flip
        i. randomly choose one element in the child key
        ii. randomly choose a plaintext letter
        iii. reassign that element in the child key to that plaintext letter
    d. decipher the ciphertext with the child key to get a plaintext
    e. calculate the child’s fitness of the plaintext
    f. if (the child’s fitness exceeds the parent’s fitness) or
    ((the child’s fitness exceeds the parent’s fitness minus the margin) and
    (we roll a 1 on a 20-sided die))
        i. copy the child key into the parent key
        ii. set the parent’s fitness equal to the child’s fitness

iii. set the counter to 0

g. increment the counter

6. output the parent key

Unfortunately, the algorithm does not always settle into a maximum and needs to be restarted with a new random parent key. Of course, if there are constraints on the cipher, such as our example above in which each plaintext letter had exactly two homophones, then the algorithm can be similarly constrained; this helps.

It is possible to specify a key using a keyword. Here is an example to how it may be done. There are, of course, other ways.

plaintext: `abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz`
ciphertext: `KNIGHTSroundtableABCcDEFfghiJjkLMmOPpQqRsUVvWwXxYyZz`

Some homophonic ciphers use numbers as the ciphertext symbols. For example, the *Grandpré cipher* uses a square grid to assign numbers to letters. The rows and first column of the grid contain keywords, and all letters of the alphabet should be present. The rows and columns are labeled with digits. The homophones for each letter are the various two-digit numbers made from the row label and the column label. Another example is the Mexican Army cipher disk which had five concentric rotating disks to assign three or four two-digit numbers to each letter.

Reading and references

American Cryptogram Association,
www.cryptogram.org/downloads/aca.info/ciphers/Homophonic.pdf,
www.cryptogram.org/downloads/aca.info/ciphers/Grandpre.pdf

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter IV, sections I-III.

Merle E. Ohaver, “Solving Cipher Secrets,” *Flynn’s*, May 19, 1928,
toebes.com/Flynns/pdf/Flynns-19280519.pdf,
toebes.com/Flynns/Flynns-19280519.htm

A challenge on MysteryTwister C3:
www.mysterytwisterc3.org/images/challenges/mtc3-madness-01-polyhomophonic-substitution-01-en.pdf

Wikipedia, en.wikipedia.org/wiki/Probabilistic_encryption

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, page 322.

Python tips

Python has a dictionary data type. A dictionary is like a list except that items are indexed not by integers but by any object. For us, it is most useful if those objects are strings. To define a dictionary, we could use this statement:

```
x = {"A": [1, 3, 4], "B": [5, 7, 8]}
```

Our new dictionary x associates the letter ‘A’ with a list of numbers. To retrieve that list, we call on x with index ‘A.’ For example, we might want to do something with each number in that list:

```
for y in x["A"]:
    print(y)
```

To add to a dictionary, use the `update()` function:

```
x.update({"C": "See?"})
```

or simply use a new index:

```
x["C"] = "See?"
```

To remove an entry from the dictionary, use `pop()`:

```
x.pop("A")
```

To change a value in the dictionary, simply assign a new value to it:

```
x["A"] = "Aaayy!"
```

The `choice()` function from the `random` module chooses an element from a list or string randomly.

Programming tasks

For these tasks, you will need to decide on a way of storing the key.

1. Implement an encryptor.
2. Implement a decryptor.
3. Implement the hill-climbing attack described above. Allow for the possibility that ciphertext symbols are integers.

Exercises

1. Encipher this text with this key. Choose homophones randomly.

```
plaintext:   a b c d e f g h i j k l m n o p q r s t u v w x y z
-------------------------------------------------------------
ciphertext:  P H O N E h o m e A B C D F G I J K L M Q R S T U V
             W X Y Z a b c d f g i j k l n p q r s t u v w x y z
```

```
TOP SECRET XXX FOR YOUR EYES ONLY XXX THE EYES OF ET
THE EXTRATERRESTRIAL WERE MODELED AFTER THOSE OF
ALBERT EINSTEIN XXX DESTROY THIS MESSAGE AFTER
ENCRYPTING
```

2. Decipher this text with this key.

```
plaintext:   a b c d e f g h i j k l m n o p q r s t u v w x y z
-------------------------------------------------------------
ciphertext:  L y t k A u w b T 4 2 m q X P x 5 h e G o z s 3 v 6
             M   l B   c U   n r Y Q   i f H p
             N     C   d V     Z R   j g I
             O     D     W     a S     J
                   E                   K
                   F
```

```
LevQptOYeBCuhQrGcD2EvHRJcVfA3DitVfEtUxdBiHA3HevqyRngNi
CXQHjFfIiVtHFlJQoxxEjMXlmSsCitOeAnDKJCjgypItLZyDapryBh
fSiEzDXxpYtGoNITPXsQonlvPocMzCxiBuDhjFlJcLKGcDtUxbFhKA
3GtSZHOUaElKsQlTwUIYoqyBhfvSpqVwdHNngQbLzCYQIWtBkIdOGI
dEYpryBiRudRqPxcSaCfuSjDMtcmAGHEjWghRowcmvxjQxSiITRaNn
IQVIeujD5pAYtvGbWfrL2DgJdFtTxcDiJE3KdMjkBjHRyiBM2
```

3. Break this ciphertext.

```
rucdupmdEvrYJbRNGIsRuOElKOHdMMTCOnvPGKFEvblSbfAOvsTJfN
OIKGnmnDrlSdGnvsOWGCFsGNGUbRnOdJMCGNeCTTalnHNOluJJfBOJ
RmuHsRBRUbRnOJDfnslhOATKoHrMqTCTnSBTlnrupJlnFAOCoHrMMO
KOuSEIOcxrneJbOCTlCOJGBTOnpcdJbbRNGIbGnOEldKsTdCJGfKEG
CTZrSTzgvTMxyqcTWOlCnyKnBlrcBfcOCOTHCTfFCOHCTlFPRGHWRm
cHPbrnTPrnOYKTfaYKOTaDbGCHYRCHYOKTfxETCrfcbdNboNnUxfdn
IcfnTPfKnPGCuWOfvbTKWbTvsOCWbdcOWrcTDyKKfuvYmKCTuSzfCO
zOlCzlxxZlPcfSOTrpsvJTfJOTcrDQTCxrimGCxTfFxOHqxOLqcTDQ
JcGluxRnTxduQJcgnL
```

4. Break this ciphertext.

```
afPSKaMjEnjNiaYNzgvLAAPNiWrLBMEtRDBXBvTLRWLAGoXrDYBlEP
UNjBYCWoizlbtyVamJefiNumVqXNJhbtyVjESYeLSKfVGETUJIbGSR
ufJqlNVgdDvKZHimMnjUfZSMCgjEfYkjUmMWVTjAQhofYemQNwCofa
ZNzwDATUYyPWaGemjCrojSKPOuXEQYAdnHLRcjCgTtRBtALoXuGZBf
jkZUmYSJtwTKNLoebqhNVbXrtsQuiNkHcrLutwoZWsJBtJSuuZCwiD
```

```
KlHTYIcKJbqlBzrZoXbbXSqajEzMAumLTyDWzhmYkjUfZWPlSvaiXN
PwYhooCgKZnVzrjfYnYUmYSigtBPWRItgfrmhuumPqvMBMukCofjAA
JSzCTaVvjSMuLkhRLBQrfPGESjWPODgKPBoidWvnZwKUPjUbPdEJUE
MHdHIqCSlalEVYArmVGEGWojWsiBDQBoiGKntRfuTJJnKfleKZHJUQ
NUVuClIIqHCNvJSVzHqlSGWoZWsPSDJBoinCwrHiumPTcEQwKadqZA
NJlItxtSRDeHcKrfEMCRmmXHLuChrGZWZcEUEXoKLoVLwWMvMcHeAc
IIYACBoMBwoGjCTzistoPgAYNlsZGvGWRKfigJlnHLRcLrLPgnjNJY
sQELWoMSsJSGQBoJgrZamtomaQlNPHYEBdEPsJEqbGRfrIqrhyiSHC
uTLWoQGAhIIaYNzgaiNidItyJLSwYcWzemQNiaMcIvHiWYTUMyQSbl
WRchRiKfPvLAAPNPWrtluLYSjArfJgYcBvYAajEvwtTZAefJJTgiWo
iYAgUJioflBzDrAjIbZawKmhuefQnMNPfMkYUfMSPwKmJNQdNVLSlW
qIdSRChRVufikMEiAlcIrqtTrmleIXWRCdRJlThgoLVSuLAtoXBzoj
WsJStVBesJmDoIQZAgUPVomrfGwaGIbHPGbbcwKEluJzLSzCJoMCEg
PKfPdorclbojWvtrLZSYAJWRbGwmaLumEiTUVoruZfMnMUmYSPwkCT
eHJCSzQNgujZzlWzlUUNQotleQzHVAZNQemJSdKcNiMAufPDNREYaK
mlWzrfiUMwTDHbJkVhWgjAKfJGEkteGRhKLjBaLbIgQikUNhouLodI
FciwKDZST
```

5. Break this ciphertext.

```
86 42 25 81 84 63 70 97 87 74 25 27 42 97 84 89 41 59
58 77 84 88 81 21 17 31 18 23 35 84 31 63 75 27 47 84
60 24 23 34 42 27 24 77 74 87 45 93 93 57 45 14 60 24
23 95 70 94 47 31 21 86 14 18 75 88 27 34 25 14 23 70
69 81 81 11 31 18 96 79 84 97 95 57 70 87 63 14 23 66
88 73 21 17 14 18 13 77 22 13 81 69 41 94 69 96 35 11
97 79 95 57 42 94 78 27 87 58 78 88 14 23 86 46 41 79
58 13 74 79 60 42 47 75 47 36 69 27 46 70 78 25 27 87
74 47 97 94 45 41 89 66 57 33 41 95 33 66 11 66 93 70
42 57 86 18 96 22 93 78 86 42 89 96 34 66 24 88 60 32
14 23 84 25 73 21 48 88 34 74 78 93 25 14 60 48 88 34
66 78 93 95 17 74 95 70 95 11 74 66 94 78 95 33 17 73
60 11 14 18 87 89 13 88 87 87 25 27 77 87 46 63 73 73
24 27 66 59 79 13 24 35 13 73 78 48 69 84 77 27 74 94
69 96 22 86 97 78 81 69 97 87 88 81 73 25 66 35 31 75
69 17 23 11 97 75 35 36 78 27 74 59 77 27 77 97 96 57
66 59 86 14 18 88 27 22 33 14 60 63 66 63 73 73 95 45
66 88 97 13 93 24 32 23 11 31 18 75 32 27 13 69 74 57
48 18 93 32 70 18 27 69 69 96 46 57 97 87 70 69 25 27
34 60 24 79 13 88 31 60 32 97 89 81 78 96 75 78 66 94
63 66 48 78 78 81 97 88 34 57 73 23 25 34 74 11 58 77
89 66 97 77 73 48 42 59 77 93 69 97 31 47 41 59 86 14
18 41 57 48 27 86 18 59 41 69 14 46 24 95 33 66 17 32
46 48 93 89 18 94 74 79 31 77 23 63 31 60 73 22 31 77
34 47 46 24 14 22 42 87 35 58 33 75 94 48 47 93 13 77
22 13 73 69 95 33 41 48 35 93 32 97 22 33 31 23 63 31
70 87 35 45 14 23 17 42 45 27 96 66 35 93 33 31 95 88
75 21 66 22 84 63 93 23 74 57 84 31 95 45 66 42 59 69
93 79 87 31 57 66 94 89 14 18 95 69 59 17 96 23 58 69
```

96 57 84 95 45 42 74 59 24 87 88 58 79 18 27 88 70 58
47 87 87 63 31 60 18 27 88 70 87 21 70 88 31 60 60 77
32 23 97 17 81 79 96 75 77

Unit 113
Polyphonic substitution

In a *polyphonic substitution cipher*, more than one plaintext letters are enciphered to the same ciphertext symbol. Here is a very simple example of a key for such a cipher:

```
plaintext:    a b c d e f g h i j k l m n o p q r s t u v w x y z
ciphertext:   A B C D E F G H I J K L M A B C D E F G H I J K L M
```

As you can see, in this example, both ‘a’ and ‘n’ are enciphered as ‘A.’ Deciphering a ciphertext is not unabiguous; there are many ways to decipher a text; we hope that there is only one way that gives a sensible plaintext. In other words, while encipherment is deterministic, decipherment is not. This makes polyphonic substitution a bad cipher.

One can generate a key from a key phrase like this:

```
plaintext:    a b c d e f g h i j k l m n o p q r s t u v w x y z
ciphertext:   O N C E U P O N A T I M E I N A L A N D F A R A W A Y ...
```

When the key is created this way, we call the cipher a *keyphrase cipher*.

An attack on a ciphertext uses the parent/child paradigm in a hill-climbing algorithm, very much like that in Unit 28. However, to evaluate each key, we must find the best decipherment with that key before calculating the textual fitness. The method to do this resembles our hill-climbing attack on the Vigenère. We start at the first position in the plaintext and try each possible character that can be there, given the key. We keep the one that gives the best fitness for the entire text. Then we move to the second position and try all possibilities that are allowed there, and keep the one that results in the best overall fitness. This continues until we reach the end of the text. We start over at the beginning. The cycle repeats until an entire pass through the text results in no increase in fitness. At that point, we believe that we have found the best decipherment for the given key. The overall attack starts with a parent key, then modifies it in every possible way, and for each possibility (child key) finds the best decipherment. If it finds a higher fitness than the parent has, then the child key replaces the parent. The algorithm terminates if no child key results in a higher fitness. Unfortunately, this algorithm is not guaranteed to find the solution, and we may have to restart with a randomly generated parent key until we can find an acceptable plaintext. Even then, some human intervention is often needed to finish up.

Reading and references

A challenge on MysteryTwister C3:
www.mysterytwisterc3.org/images/challenges/mtc3-madness-02-polyhomophonic-substitution-02-en.pdf

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939, chapter IX, section I.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, page 787.

Programming tasks

1. Implement an encryptor.
2. Write a function that finds the best decipherment for a given ciphertext and a given key.
3. Implement the attack described above, for the case in which each ciphertext symbol can represent either of two and only two plaintext symbols.

Exercises

1. Encipher this text with the given key.

plaintext:	a b c d e f g h i j k l m n o p q r s t u v w x y z
ciphertext:	H M F A A K E B C L I D J G J E D K G C M H B L F I

> Polyphony is the art of imitating sounds of various kinds, usually, without attempting to deceive the hearer as to their direction. It may therefore be studied independently of ventriloquism. Already the art is much in vogue.
>
> (from *Three Hundred Things a Bright Boy Can Do* by anonymous)

2. Find the best decipherment of this ciphertext. Use the given key. You may have to finish it by hand, once your function gives you something recognizable.

plaintext:	a b c d e f g h i j k l m n o p q r s t u v w x y z
ciphertext:	D C I M J K A E M G C L A F L I G J H E D B K H F B

```
DFMAJILLLFHDELFEEJHEMLJDFMEDEJMEEJKELLJHIEJAJLKLMKJKEM
IEKDHDELFIJJHIJHHMBJDFMMFDMJGDDEJDHDHLLDEMLFEJEDEJMKLH
CLDJFJEJEDEJMKLHCLDJFJEMAEHEJJJEEJEDEJMEMHHELIDFMEMHKM
KJDFMEMHFJMAECLDJHJBJJFCLJHHJMFJMAECLDJDFMKMEEMFMJHIJM
CDCLJCMEEJJFJHHEJEDEJMEMAHJLK
```

3. Break this ciphertext. Each ciphertext symbol can represent two and only two plaintext letters. At the end, you may need to tweak the decipherment by hand.

```
GFLLKILLJEAFHCJIEEBCHIEAFABCFKCBEAFLCCJEAFBCDEFICLBCCC
HBCBCEBCLKCEHCBCEAILCCJBDGBHDABILLEFBAEFIHCBLBIEJDECFG
CECJBCBCKCEIEIEECLCJBCBDEAFIAJDCAGIBLFBLDCCBFFHABIAFGC
ECJJDBCLALKIBFHIDIABIKCEJCEBCEAIBCJIBFJEJIFBIGLDHGECFI
ABCGFHEEEBCCMGCLACJAFECCFJAEFABDEGLDCCFECHIEAFFCECCEAB
CJDGEDIDCJLIJKHBFHCAIEECIEJIICHFABCBHCLLJBCEECJHFHCEHB
FLFEKCBECJDELFHAFECEHDABCILBFABCBJDJEFALFFCLDCCGFLLKED
JCIFIEBFGGDBLEIEECELIJKLFEJJLACJABCHAFILDIAIEJABCKEBFA
JGAHFEAFBDCEIGIDEABCKLIHCFJADEAFILFKCLKLFJEGDEGBFFHFJA
EADLLEFEDGEFIJBCEECEABCLIJKGJEBCJIFJAAFEIEJIEFABCBHFHI
EBJBBDCJDEABCEGFLLKHIEELDCEADIDLILLKHCIEJBCJIEJDEIEBFB
AADHCIEJHFCBFIHFJCLEHCBCFBFJGBAIFBBCBDEEGCLADFEIEJIGGB
FKILABCECHCBCGLILCJJGFEIFBHEIEJCKCBKJCEDBIFLCJCAIDLFIA
BCGFHEEHIEGFDEACJFJAAFIEECIEJABCGDBLE
```

Unit 114
Polyhomophonic substitution

Polyhomophonic substitution incorporates both homophonic and polyphonic substitution. Here, a plaintext symbol can be enciphered to any of several ciphertext symbols, *and* a ciphertext symbol can represent more than one plaintext symbol. Encipherment is probabilistic, but since decipherment is not deterministic, this is a bad cipher.

Here is a quick example. Consider this (randomly generated) key:

```
plaintext:   A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
-----------------------------------------------------------------
ciphertext:  E A B H N D H B C F L J A G I T H D E J P N B F G J
             M C E Q Q G L O T K M R F V P X M R I S Q P D L S K
             X  N U V I O U   Y W Z X V  U W Z Y R S O Z Y W
```

With this key, the word `SECRET` could be enciphered as `ENBDQJ` or as `ZQNRVS` or as `INBWQY`, but `ZQNRVS` could also be deciphered as `MEELEY` (not a word).

The attack on this cipher is similar to the one in the previous unit. We must modify it to accommodate the fact that each plaintext letter has homophones.

Reading and references

A challenge on MysteryTwister C3:
www.mysterytwisterc3.org/images/challenges/mtc3-madness-03-polyhomophonic-substitution-03-en.pdf

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.

3. Write a function to find the best decipherment for a given ciphertext and given key. You can do this by modifying your function from the previous unit to lengthen the key.

4. Implement the attack described above, for the case in which each plaintext letter has two homophones, and each ciphertext symbol can represent either of two plaintext letters. You can do this by modifying your attack from the previous unit to double the length of the key.

Exercises

1. Encipher this text with the given key. Choose homophones randomly.

plaintext:	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ciphertext:	Y D D A W I S T V B R A E F X I V W O J Z Y P Z H U N H F Q C S J N B T M E K O Q L G K C X U L G P R M

There were fine lawns and beautiful flowers everywhere, but Polly and Rose loved the shore, and surely the salt air was delightful, and the beach a lovely place on which to romp.

(from *Princess Polly at Play* by Amy Brooks)

2. Find the best decipherment of this text with the given key.

plaintext:	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ciphertext:	Y C R U G I T N W O Q G K Q J P P X L R E F L D U D O X E F W M Z N A S I C S H B Z B V M J Y H K A T V

```
WOHWSPCCDXHSDWREKWELDKXIJXCCXCSWSCPOXCEDTWAHJYXMWHOSHJ
OKLXBOXOCJJXSWJFBIWHPOPEBCOXCFXCJXLDPSXIXFHPXHPOIPSKPH
XCPOHJOGFZOHAASPOAOYSWOFPEBCOIGDEIOCREZSXSUZHHJOSWYAOS
AWWCCXNQPWDNCYSEBONPHXXCBUDWSHPWIWQXOLNCCAOKAWOOXXMPXH
WAHJWIXFJGFVWACHJODSPWGPODNICXNSPWJXDDCYAWYIJJPHWCIHJW
JEBCWSPWOHPOAWXIGBOMDCKXXKPXSJOIOEJ
```

3. Break this ciphertext. Each ciphertext symbol can represent either of two plaintext letters, and each plaintext letter has two homophones.

```
QZAGUXPRKKVNFOTAPQBUWNQNVUWMAODHPAWMROHKFMBOCOBNJYUJWH
RNOSOKNPAVQWNVUJRNUSWQAHMWPYAVOFUPRWRQWOFMBUPSSOPNJRCH
RPBGUPYTYQPGMCTVPSQLAUSXUHWCNGUKNQSNJWCGQHRNMCUSQXQIOG
QAUJAQXGYHCHOUJWAGUJWAOSRPNHQWQDXKXQNGPKXGPAGEQWWUHPKM
WPRTUFVNVPYPNASQLNHMRXONIOUWJWWJMJUVQDPRJDYKMXOAJXUKMM
UBDUQBBPTHQDUJZZOBORTOKHRBMJWNMCMJURQRWEUABQTHPYUHZCOB
UWTUKJRBMJWAMZMJOIIHYYQDIQKPXUQNIQSRJNGFVNTQDQANBQTAJM
WXONRUORHWUHAJWVQDKQDDTMWNSQKNKPDDLQFGYUZJNJUPPSUZQTTP
```

KUWQRAVHKKJWDYUPAQDURGJDOMRUCUAKPPWMRGOSUOYPONGULMWZYJ
LNIGJTVOQLGRUINMSWCOWOSQNJQRLQWRQAXVARQDOPCQJRPNAGUWJT
NQAHQWQCNVOQCUKQRNVHKTBQRUOWPTOWUNBJWCJWAVQNQXUBJTPRUD
UXOWNNQIVHLGNVUUHTAQNJQWQZNGOPDUPFOQWPSODPAHMOYERQAVHW
CPWUAGOIUPDAGQZAVOUSQXQAJLZJOYWXUTMFUKMXAHQVPJNJPTTBJQ
VPNVUBUZMBUAVQAJNVQPNUOWPMYHNNDONMTTGUUHAVQKNOORUWNOBO
WXTARMNMOSECPSAGOCSUQNSTPKJQWPWUZSOWTVRQAUDJPNPRHAVAVU
JSLMRLUWABPAJMRQWNGOYHZOJXXUWHQNODESMTWWNGOXHRNGOFPHWJ
CWMBUJAAGOUWDDHPGGQAURMSHOWHAPYJANYUNTARQAMZAOWPRWWMAI
HAGFTTVJRPJDVAAVONSVNGKOUFPNMNUNGQNNVUUTSMGUQRWQNJQWKI
HNVAGOHSPASQWDYHROPQZLDOQAQCUVPMOWHZCJLTDNEJRVWUUSKNQW
WJWDUPTVMNVUBIVHYUAVUKTWUOSKAQWWPXUBJLPRMAPAPDDKNOUBUW
QRWWKUUHRNVUHBMRRWQNJQWQDGSOBMKPUKPJMRKNGOESOCQBUMNGUB
RQNJQWQYBSUBQKKUPPHMWPIJAGJRUJCZUSORTUPXQYOXOWNQBVMPAJ
DJAEAGOBUQBUUZLOGNHQWKNQNGHKKAPAUFOWNMCTMTSKOJPGUPSMWD
KMZCOWOSQDAOWWURTJOKNGOASORWQZOMOWNPPHWLOAVURQSUAUWXQS
UAGQRAVUIQSJAKOYZXSHWCPVQXUAMTPNGUZPLNAVPAAVUOTSMGUPWF
JWWHPABJXQD

Unit 115
Pollux cipher

The *Pollux cipher* is another cipher that uses Morse code. Like in many such ciphers, a plaintext is first encoded in Morse, with a separator between letters and two separators between words. The dot, dash, and separator are then enciphered with a homophonic substitution to digits. The key is the mapping between Morse symbols and digits.

As usual, let's run through a short example. Consider this plaintext:

```
THIS MESSAGE WAS ENCRYPTED WITH POLLUX
```

We can use this key, for example:

0	1	2	3	4	5	6	7	8	9
·	·	×	–	·	·	×	×	–	–

The first step is to encode with Morse.

```
–×···×·×···××––×·×···×···×·–×––·×·××·––×·–×···
××·×–·×–·–·×·–·×–·––×·––·×–×·×–··××·––×··×–×···
××·––·×–––×·–··×·–··×···–×–···
```

Finally, replace the Morse symbols with digits, according to the key. For this example, choices of ciphertext symbol from sets of homophones were made randomly. One possible ciphertext is

```
325050744651022386060556141218288424221987492004775281294812435294336583563612304620337156860444765894693320855703147158684 18
```

To break a ciphertext we can brute-force the key, or we can perform a hill-climbing attack. Because the cipher has two stages, Morse followed by homophonic substitution, we will attack the substitution. Once that is accomplished, it is easy to decode the resulting Morse code. Since the substitution is between an intermediary text of dots, dashes, and separators and the final ciphertext of digits (or some other set of symbols), we must build a frequency table for a corpus of English text that has been encoded with Morse, and use this table in defining a fitness function that we will work to

maximize; we recommend tabulating the frequencies of 10-symbol groups (there are only 3^{10} = 59,049 such groups to consider). The hill-climbing algorithm will be similar to ones we have used before. We will use the parent-child paradigm. For each step, the parent key is modified by randomly selecting one element and changing its value; i.e., pick one digit and change which Morse symbol it represents. The ciphertext is decrypted back to Morse code with the child key and its fitness is calculated. If the fitness exceeds the fitness of the parent key, or if it comes close on rare occasions, we replace the parent with the child and continue onwards. When we are no longer able to find a higher fitness for some number of tries (like a thousand), we terminate the algorithm and decrypt all the way (including decoding the Morse code).

Reading and references

American Cryptogram Association, www.cryptogram.org/downloads/aca.info/ciphers/Pollux.pdf

Programming tasks

1. Implement an encryptor. It should check that the key is valid, i.e., that it contains at least one homophone for dot, dash, and separator. Choices of homophone should be random.
2. Implement a decryptor.
3. Implement a brute-force attack. Remember to only try valid keys.
4. Implement the hill-climbing attack described in this section.
    a. Take your textual corpus containing upper-case letters and spaces and encode it with Morse code. Replace the space between letters with × and between words with ××. Compile a frequency table of dekagrams (10-character sequences) from the result.
    b. Modify your fitness function from Unit 9 to handle dekagrams of Morse code.
    c. Implement the hill-climbing algorithm. Experiment to find a good margin to use for allowing downward steps about 5% of the time. Terminate after 1,000 children if the fitness does not change.

Exercises

1. What is the size of the keyspace for the Pollux cipher? Only include valid keys.
2. Encipher this text with the same key as in the example above.

```
CASTOR AND POLLUX WERE THE TWIN SONS OF ZEUS AND LEDA.
ZEUS SEDUCED LEDA BY DISGUISING HIMSELF AS A SWAN.
UNFORTUNATELY, ONE OF THE TWINS WAS ACTUALLY FATHERED
BY LEDA'S HUSBAND, SO ONLY ONE OF THEM WAS IMMORTAL.
WHEN CASTOR DIED, POLLUX SHARED HIS IMMORTALITY WITH
```

HIM, AND THEY ALTERNATED DAYS ALIVE AND IN THE UNDERWORLD.

3. Decipher this text with the key (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) ↔ (−, ×, ·, ·, −, ×, ×, ·, ·, −).

```
0821286428550299199917291603959310406749550637781245916024817067885419096702512419854225624921000579376847312345488956382362010825145209587502158275885227191853075332172003366968337635084051809575207635508026848750396450686405976767236068035806534598643255467788586683426345405004623412721582371753072576981549063308564679760491974068489395189130286343265370354991770180351494633431695283213652376731077357932123148140268225184184858562876796771972160500415373313052229575598335253502653872130505924368778576928662892179269445091635002590753331093749568781871925984713119182326868268035183925705018333675397614082521830583751823768064876543886357193510786271283540752391726332575033652417226170112225290629107517469650588712677579211929310446985039862529086418350945421293089
```

4. Break this ciphertext with your brute-force attack.

```
7646364311625374521503865256856122945035748641639569896535094374104588058656681093555285972502777294525182172666834512924530187689182536190862242657219359811034569572207895986286421993691347584050269817253184093200256827713041898637043305856841123676377584589427023743404737219003475894192555244635645320743603526109886969377780403011898910226484034108378966134442115355920669252844521041229726963307021399935056360481365981668854421656335319163774266139231027038709291216634204920360130992670497384800903912190837584074852609925478171635772456279743806299923461181864912092086605393701327254528570347588469302936451392950844107782693562842616639727834480734624647236184134998774259699730360426462457264462029427032767936652941924369731513080098866998043601429835724957253106036613354520679824698692196721085021225558040230992027426927098198016430987026225305730187182901135748679260029844398403050564220635638520900302610802854841573586091319932786961803243957209871216138048076293819639863518864800638660804391678993755214030662374529056825209102477310936880037688174131864026126706226587624018894896623781111318241502557207871703901087430841327692679216296048163492756385299903182911985743979309926269133426911864343317502167214193976123429616293369031948708305840513159344580603160384300013122996241737177395156
```

5. Break this ciphertext with your hill-climbing attack. You will find that it is possible despite the brevity of the text.

```
194925739853721143540660461564856257706111381886975538
846790763873570253246089065227281124416275893915209311
569888272969902426589642812919609523052402344860214605
379208
```

6. Break this ciphertext which uses a larger set of ciphertext symbols. You may need to reduce the margin in your hill-climbing attack for this one. Did I use a keyword?

```
CEDAMPSBVEXTDHYVACSBUPVOPCGKHWNNHFRXBETJFTLKBNLQPMNSEA
MSADMHSJKNHUBHGRUEAYCEZAOOQHZYRHOFKBFSXRLTAAWSXLSFUHYS
SBGIKHTSTZLADBEUVVQNCVZKLIEHTGCALBCTMOIQOQFOFSGSONTQDE
PGODRKDEKNBMHDSYTTLVCVPPPBEMOXPXOQSBXVFUSMXD
```

madness's book on classical cryptography
unit 116: doubled-over substitution
last modified 2020-10-28

Unit 116
Doubled-over substitution

The *doubled-over substitution cipher* was invented in 2011 by Viktor Veselovsky, albeit by a different name. His name for it was “monoalphabetic substitution with camouflage,” for a reason that we will explain below. The cipher involves interweaving two halves of the plaintext and a substitution that involves both upper- and lower-case letters. It is a probabilistic cipher, since the point at which the two halves are divided and the interleaving are both done with some randomness. Decipherment is unambiguous; it is always possible, with the correct key, to recover the plaintext.

Here is an example to show how the cipher works: Start with a plaintext, such as

```
The quick brown fox jumps over the lazy dog.
```

Split the text at a randomly chosen place near the middle:

```
The quick brown fox jum
 ps over the lazy dog.
```

Throw away the punctuation, convert the first part to upper-case letters and the second to lower-case, and replace spaces with ‘_’ in the first half and ‘/’ in the second:

```
THE_QUICK_BROWN_FOX_JUM
 ps/over/the/lazy/dog
```

Next, interweave the two parts. This is also done randomly.

```
TpsH/E_QovUerICK/t_hBROe/lWN_aFzOyX/_JUdoMg
```

The key is a permutation of this plaintext alphabet:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz/
```

For this example, we use this key:

```
plaintext:   ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz/
ciphertext:  ROUNDTABLE_knights/abCcdeFfGHIJjKlMmoPpQqrSuVvWwXxYyZz
```

After making the substitutions, we arrive at the ciphertext:

```
aSvBzDftrXbjVLU_zWfMOsgjzpcifGTZgydzfEbJrnl
```

In decipherment, we first invert the substitutions. Separating the two halves of the plaintext is possible because of the use of upper- and lower-case alphabets.

Optionally, we can remove all spaces from our plaintext, and remove '_' and '/' from the key.

Veselovsky's original name, "monoalphabetic substitution with camouflage," came from an earlier version of the cipher in which the plaintext was not doubled over. In that version, the plaintext was converted to upper-case letters, and lower-case letters were added randomly as camouflage. The substitution proceeded as in the example above. Decipherment involved inverting the substitution and removing the camouflage.

We can attack this cipher with a hill-climbing algorithm that seeks to maximize the fitness of the plaintext using the parent/child paradigm that we employed in Unit 28 for the monoalphabetic substitution. As we have done before with other ciphers, to avoid becoming trapped at a local maximum textual fitness, we will allow downward steps by a margin of 0.2 about 3% of the time. In order to converge on the global maximum, we must shrink this margin down to zero after we have reached a fitness at the lower end of acceptable for English (see the exercise in Unit 9). Another complication is that there is about an even chance that the algorithm will have the two halves of the plaintext in reverse order.

Reading and references

Challenges on MysteryTwister C3:
www.mysterytwisterc3.org/images/challenges/mtc3-veselovsky-11-mono_split-en.pdf
www.mysterytwisterc3.org/images/challenges/mtc3-veselovsky-13-mono_nospaces-en.pdf
www.mysterytwisterc3.org/images/challenges/mtc3-kallick-27-cloakedsub-02-en.pdf

Bruce Kallick, "A Modified Simple Substitution Cipher With Unbounded Unicity Distance," Cryptology ePrint Archive, report 2019/621.

Programming tasks

1. Implement an encryptor. Allow for the choice to include or exclude spaces.
2. Implement a decryptor. Allow for the choice to include or exclude spaces.
3. Implement the attack described above. Remember that spaces may or may not be included in the ciphertext.

Exercises

1. Encipher this text with the following key:

   ```
   TWAIN_story/aBbCcDdEeFfGgHhiJjKkLlMmnOPpQqRSUuVvwXxYZz
   ```

   I have found out that in one way you are quite different from other people. You can see in the dark, you can smell what other people cannot, you have the talents of a bloodhound. They are good and valuable things to have, but you must keep the matter a secret.

   (from *A Double Barrelled Detective Story* by Mark Twain [Samuel Clemens])

2. Decipher this ciphertext with the following key:

   ```
   HOMEWRK_duetomrw/AaBbCcDFfGghIiJjkLlNnPpQqSsTUVvXxYyZz
   ```

   ```
   BWHzVALJzhaqGyUHzgQimEzGkabltTeFGPRUHMWzaGVHmqqzExJGdT
   JzokwrPgBWmiBGRzxLdJaQzBxaqTnlGQkzVlEprbJzOItgpWJEzGRd
   WTqAMvQWitFzgGckg_lWQzmgGQiz_dVaGLJOHMzpeGcgUVJHaGBbAm
   TWzxJEQGVczLWqApWJGzB_VqzWGALWJcHPSAzLlEaUGrRG_daGzMjr
   gVLmJTazMdWqQzmBdrbVaLJmzjgWaTapGHmE
   ```

3. Break this ciphertext.

   ```
   dsLIiVEzmLivklyWeIpzdiDqBKIGzmkuLVDzJrIWDABIbvzLcsDbip
   idIVzEacSIueNsLIqEjtcdeIekzt/XLiVzMRIqktzvmAiIsLcIyvzt
   mRuuNEKzLIDXRQiIyteeqzmiKzviyLPkqzmNQzIDdsiDX/LiBYzIvm
   esLIkzftalWyuYzLuRAVkzULIaiEMIGstiUsIdNsLIqzNqszmDkBuz
   IOmLLkAiI_FutyevHziqIsDKzuiMBIfkKzWDqAvNpztNvdzjsiQkzv
   RLBzmkuzIEcaQ/NIqKsLczImRdXbtzmcktuezIKkiuDzSAuBINqjAk
   azGIXRWyDHpKIzmtiAIsLcwkIz/LlpRX/acqHIzGteivszIemkuzsL
   IUvDymuRi/vzNIlz/amckAzmiKzVtkkqA_zImOkuzLsNqztVABWjIt
   emIGzistzSiyVVLItANIERqzvcaAeImeksLzIQCFkQtRLeIBuYFzji
   drpQkHIKAtz_smekuIGDzidIqKzAVmkzauRGIVUkyadztiA_qItKAz
   IXeskLqIyavFzmRQkBzvmkIdukzSsDkumiSVzV/mkzXRWLpIOLKzeG
   ltgeIDNIqKBzSauFNqOyLIjbLkzlDRUuLzVWukpIOLeGLLYAIzemsL
   IekGazjIsLRWcIbpDKddtaAIyaarLzqkwBIF_yHIkteIbuzmDitwkA
   LBIsLcIeaIcLz/JkkL/OqzVLcWjIdsmLIELzyeizIVtNppeYzIKRGM
   ziVzMRziXiYzipvRMkvmkuzXNvmzizVvuiqMkzXRQiq
   ```

4. Break this ciphertext.

   ```
   evEevFfOmBbGoSwFbVsvXMZBbvHOafdfAmiFwfRmBWkafblOjFwSfl
   APVbRTNTHSmTNTXfduikbGTaRRffTFSeBFTwfukOafPWMVDTubHFRP
   wqfIQYmAbmEEbaMVwWEvWBeBOjvjBufdafWmjBMpXVveBowbCfmEMY
   WPvBDOlVXfWkdOEFwWRjFeuxBmqTEXVCeMmdqfePkFEjkIZvTvOdef
   mlfheEFfBqLmfSWEnLwvOTfafNbjKlmmVvTSnEXmTWAugTWFNejKXv
   vmbBluNbqdeZVFqEECfLmefEneqmBFjuwSfYOZEFdVTRjSGZooANqT
   ```

YmfulVOjbxdEmVGvPdkgeQZgkWjqPAkFmwVWTFutuTeFwvmkWEIvfP
fBExAmOjbPHAiuCVYfdlfLOFZveOBmVoHQXSeQkFjKKjnwEbmafeBE
fLFEwftXuACtqFwmdvbuGoPqwqPXRjFwSMkjufuWoPTfEeSFeNEqTY
BbFWmuCWmEEeRgfPvFweBoFjbIfWNmvPMTMTDuaedfDIYOiwPGqaWm
TvBfOifmuEBbdNqDbfEeFTYHmWEFuVmfeFVEEtHfmkfFBPfEEjqqbZ
vOmBYOPFvwfDmTfuEfdtjqmFWedZemqFeEWqEfBTFeYfBFSIPnOvKq
TfeBToReaCmuVfkOjAuARYPvfOvOmPDFbWLuFMmeqBFPkPfATXAeof
VBKuFAimFwdjvbOGVommPNZTXwkFjwSSTvfMPVkTExmuYOWafEFZvO
mPdRRbPBoDnujfSvMEejPVZdjqufEmFwjWMFZYOPWvBmSeVTXauWNF
PfmGqMVYEPWBvOMDmVkfBmDbHSMfjDSHeFwvVLfjdFWeVBSXRmCbHf
duTXEVDellfdjVveOmVBojbSMVTBANvOieBDfmVmjVodOffTuImYOG
mumjVZFBTbXFReBsjZveuBDjRldtjYbaOFwbTEfqmMbjZYlPFwvfOW
BTeXavWALNdPSfWFeMbBDfPEeSnjdVfPfgfdSLndfWqmFfESjlvbPx
dmaqDbjibMGDPbmGVIVFeqPttmu

Unit 117
Duplicitous ciphers

A *duplicitous* cipher is one which can be decrypted to two different plaintexts, depending on how the decryption is performed. The structure of the cipher does not involve interweaving two ciphertexts; rather, all ciphertext characters are needed to decrypt each plaintext.

A very simple duplicious cipher was invented by Jack Levine. In his cipher, two sets of numbers are assigned to the letters of the alphabet:

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
set 1:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
set 2:	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52

The first set is used for the first plaintext, and the second set for the second plaintext. The ciphertext consists of pairs of numbers. In each pair the first number is the average of a character from the first plaintext and a character from the second plaintext. The second number is the difference from the average to either of the two characters. For example, if we want to encrypt “definitely yes” and “possibly not so,” then the first characters are d and p, whose numbers are 4 in the first set and 42 in the second set. The average is 23, and the distance from the average to each of the two is 19; so the first two numbers in the cipher text are 23 and 19. The full ciphertext is

23 19 23 18 25½ 19½ 27 18 24½ 10½ 18½ 9½ 29 9 28 23 26 14 33 8 35½ 10½ 25 20 30 11

Decryption is simple: For each pair of numbers, the character in the first plaintext is the difference between the numbers, and the character in the second plaintext is the sum of the numbers.

Let’s work an example of another duplicitous cipher and run through the encryption of two plaintexts. This cipher will encrypt one plaintext in a base-5/Polybius-cipher scheme, and another plaintext in a 5-bit binary scheme. The first plaintext must be five times as long as the second; if not, it is padded with nulls (usually ‘X’) until it is so (spaces are not used). The base-5 encryption will use different symbols for the horizontal and vertical labels of the Polybius square. The binary encryption will involve permuting those symbols. For our example, we begin with two plaintexts:

#1: DUPLICITY IS THE SAME AS
#2: LIES

We also need two keys. We can construct them from keywords CANARD and DECEPTION. The first keyword is used to fill the Polybius square. Remember that we have only 25 spaces, so we must remove one letter; typically we merge ‘J’ into ‘I.’

	U	V	X	Y	Z
A	C	A	N	R	D
B	B	E	F	G	H
C	I	K	L	M	O
D	P	Q	S	T	U
E	V	W	X	Y	Z

To encrypt the first plaintext, each letter in it is replaced by the labels on the rows and columns of the square, just as in the Polybius cipher:

D	U	P	L	I	C	I	T	Y	I	S	T	H	E	S	A	M	E	A	S
AZ	DZ	DU	CX	CU	AU	CU	DY	EY	CU	DX	DY	BZ	BV	DX	AV	CY	BV	AV	DX

From the second keyword we construct this keyed alphabet, and assign numerical values to the letters:

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

The second plaintext is converted to a 5-bit representation based on the keyed alphabet:

L	I	E	S
15	5	1	19
01111	00101	00001	10011

We take the ciphertext from the first plaintext and swap the two symbols if the corresponding bit from the second plaintext is 1.

AZ	DZ	DU	CX	CU	AU	CU	DY	EY	CU	DX	DY	BZ	BV	DX	AV	CY	BV	AV	DX
0	1	1	1	1	0	0	1	0	1	0	0	0	0	1	1	0	0	1	1
AZ	ZD	UD	XC	UC	AU	CU	YD	EY	UC	DX	DY	BZ	BV	XD	VA	CY	BV	VA	XD

The final ciphertext that hides both plaintexts is

AZZDUDXCUCAUCUYDEYUCDXDYBZBVXDVACYBVVAXD

The key to cryptanalyzing the cipher we have just described is to separate the ciphertext symbols into two sets, those for row labels and those for column labels. Once that is done, the rest is straightforward. Let’s work through an example. Suppose we have this ciphertext:

```
AJCDICHAFBHBAEHADFHGJAHAEIFBBCIEFDJADCICHAFBJICIBFAJHAEIBFI
EBCCAIHEAEBHADHHAAJEIHIHBHDAHAEDCCIFDBHBJHGHBJADCICCBHABFGH
BEHDDCGHBEBECIBFGCICHAFBJADCCIEDFDAHJAAHHBCDFBHGJAAHEIFBGHE
BHDHAEADCCIFDHDCDHGBEBECIBFGCICEIBCJAJBCIFBJAGEBFHABFCIJAIC
CIBFHAJAHIHBICCAIFAHGJCIACFDGHCAHAJGFBHIIFDECIFDHBAJEIICFBD
HEICACIICHGDHDCAEEBGHHABFAJCIJGAJBEICJAAJCIFDGHFBACHBCIFDAH
GHAJAHEIBFGHBFACBCFDGHHDJAAHEIFBGHJAHAEIFBJAIECDHAACICAJCDG
HAJCBHGDHAJJGJGJG
```

We begin by breaking it into digrams:

```
AJ CD IC HA FB HB AE HA DF HG JA HA EI FB BC IE FD JA DC IC
HA FB JI CI BF AJ HA EI BF IE BC CA IH EA EB HA DH HA AJ EI
HI HB HD AH AE DC CI FD BH BJ HG HB JA DC IC CB HA BF GH BE
HD DC GH BE BE CI BF GC IC HA FB JA DC CI ED FD AH JA AH HB
CD FB HG JA AH EI FB GH EB HD HA EA DC CI FD HD CD HG BE BE
CI BF GC IC EI BC JA JB CI FB JA GE BF HA BF CI JA IC CI BF
HA JA HI HB IC CA IF AH GJ CI AC FD GH CA HA JG FB HI IF DE
CI FD HB AJ EI IC FB DH EI CA CI IC HG DH DC AE EB GH HA BF
AJ CI JG AJ BE IC JA AJ CI FD GH FB AC HB CI FD AH GH AJ AH
EI BF GH BF AC BC FD GH HD JA AH EI FB GH JA HA EI FB JA IE
CD HA AC IC AJ CD GH AJ CB HG DH AJ JG JG JG
```

Now let's see which symbols appear together. In this chart we put an 'X' if two symbols appear in the same digram.

	A	B	C	D	E	F	G	H	I	J
A			X		X			X		X
B			X		X	X		X		X
C	X	X		X			X		X	
D			X		X	X				
E	X	X		X			X		X	
F		X		X			X		X	
G			X		X	X		X		X
H	X	X					X		X	
I			X		X	X		X		X
J	X	X					X		X	

By studying this chart, we can divide the ciphertext symbols into two sets. For example, 'A' and 'B' do not appear in the same digram, but both can appear with 'C.' So 'A' and 'B' are in the same set, and 'C' is in the other. The two sets are {A, B, D, G, I} and {C, E, F, H, J}. To uncover the first plaintext, we reorder each digram so that a symbol from the first set comes before a symbol from the second set:

```
AJ DC IC AH BF BH AE AH DF GH AJ AH IE BF BC IE DF AJ DC IC
AH BF IJ IC BF AJ AH IE BF IE BC AC IH AE BE AH DH AH AJ IE
```

```
IH BH DH AH AE DC IC DF BH BJ GH BH AJ DC IC BC AH BF GH BE
DH DC GH BE BE IC BF GC IC AH BF AJ DC IC DE DF AH AJ AH BH
DC BF GH AJ AH IE BF GH BE DH AH AE DC IC DF DH DC GH BE BE
IC BF GC IC IE BC AJ BJ IC BF AJ GE BF AH BF IC AJ IC IC BF
AH AJ IH BH IC AC IF AH GJ IC AC DF GH AC AH GJ BF IH IF DE
IC DF BH AJ IE IC BF DH IE AC IC IC GH DH DC AE BE GH AH BF
AJ IC GJ AJ BE IC AJ AJ IC DF GH BF AC BH IC DF AH GH AJ AH
IE BF GH BF AC BC DF GH DH AJ AH IE BF GH AJ AH IE BF AJ IE
DC AH AC IC AJ DC GH AJ BC GH DH AJ GJ GJ GJ
```

We can now break this ciphertext as a Polybius cipher. To recover the second plaintext, we compare this to the original ciphertext. If we had to reorder a digram, assign a 1, otherwise a 0.

```
original:   AJ CD IC HA FB HB AE HA DF HG JA HA EI FB BC IE ...
reordered:  AJ DC IC AH BF BH AE AH DF GH AJ AH IE BF BC IE ...
bits:        0  1  0  1  1  1  0  1  0  1  1  1  1  1  0  0 . . .
```

The full bitstream, broken into groups of five, is

```
01011 10101 11110 01100 11110 01100 01011 10101 11100 01100
11100 11000 10000 10001 11011 10101 11110 11011 11011 11100
10001 01111 10010 11010 11110 10001 01011 11100 11101 01011
10100 01010 01100 01011 01011 10000 10000 01011 01101 11110
11000 10011 00111
```

When we try to decipher this as a Baconian cipher, we have a problem: code words like `11110` are not valid. However, if we reverse the rolls of 0 and 1, we can decode the bitstream. This simply means that we should swap the row and column labels that we found earlier. To finish decrypting, we must break what we get as a monoalphabetic substitution cipher.

We have another example of a duplicitous cipher. Before we present it, we should briefly describe the cipher that inspired it. The final challenge for the 2019 British National Cipher Challenge involved a modification of the bifid and trifid cipher to a fourth dimension. A 2×2×2×2 “grid” has 16 cells into which we can put letters; this is not enough. A 3×3×3×3 “grid” has 81; this is too many. As a compromise, the four-dimensional box used in the cipher has size 2×2×2×3. This allows 24 letters to be used. Since we need to remove two letters, we can merge ‘J’ into ‘I’ and ‘Z’ into ‘S.’ This should make the British happy, since they never use ‘Z’s. The key for the cipher is a keyword for mixing the 24-letter alphabet in the box and a period for seriation and fractionation. Each plaintext letter is first encoded as the mixed-radix number describing its location in the box. A *mixed-radix number* is one whose digits have different bases. The base is often written as a subscript. For example, one such number used in this cipher is $1_2 0_2 1_2 2_3$ (or simply 1012). This number has three binary (base-2) digits and one ternary (base-3) digit.

Let’s encipher a short text to see in detail how the cipher works. Suppose we have the keyword `KEYWORD` and period 5, and this plaintext, which is padded to the fit the period:

THIS MESSAGE WAS ENCRYPTED WITH A NEW CIPHER XXX

The mixed alphabet is

KEYWORDABCFGHILMNPQSTUVX

The box is four-dimensional, so it is very difficult to draw it on this two-dimensional page, but we can instead list the contents and their coordinates:

K	0000	D	0100	H	1000	Q	1100
E	0001	A	0101	I	1001	S	1101
Y	0002	B	0102	L	1002	T	1102
W	0010	C	0110	M	1010	U	1110
O	0011	F	0111	N	1011	V	1111
R	0012	G	0112	P	1012	X	1112

We can now encode the plaintext as a fixed-width code using the four-digit mixed-radix numbers as code words.

```
1102 1000 1001 1101 1010 0001 1101 1101 0101 0112 0001 0010
0101 1101 0001 1011 0110 0012 0002 1012 1102 0001 0100 0010
1001 1102 1000 0101 1011 0001 0010 0110 1001 1012 1000 0001
0012 1112 1112 1112
```

Write each vertically, and group them in fives (the period):

```
11111 01100 00010 10001 10001 11010 00111 00111
10010 01111 00110 01000 10100 10100 01000 00111
00001 00001 01000 11101 00010 00010 11010 01111
20110 11112 10111 10222 21001 20111 00120 12222
```

Read off each group by rows:

```
11111 10010 00001 20110 01100 01111 00001 11112 00010 00110
01000 10111 10001 01000 11101 10222 10001 10100 00010 21001
11010 10100 00010 20111 00111 01000 11010 00120 00111 00111
01111 12222
```

Unlike the bifid and trifid ciphers, the box in this cipher does not have the same lengths in each dimension. Therefore we are unable to recombine the digits into letters. So the final ciphertext is

```
11111100100000120110011000111100001111120001000110010001011
11000101000111011022210001101000001021001110101010000010201
11001110100011010001200011100111011111222 2
```

To break this cipher, we first have to identify the period. This is easy if we break the ciphertext into blocks and adjust the block size until the ‘2’s only appear in every fourth block. Then we can

reverse the transposition to the digits. We then decode the result using an unmixed alphabet. What remains can be broken as a monoalphabetic substitution cipher using the hill-climbing technique in Unit 28. The keyword can be recovered from the key to the monoalphabetic substitution.

Before we move on, we should note that the ternary digit need not be the last coordinate. In the BNCC challenge, the ternary digit was the third. Which coordinate uses the ternary digit should be clear during cryptanalysis, since '2' can only appear there.

We are now ready to describe a second example of a duplicitous cipher. Like the first example, it encodes symbols from the first plaintext with coordinates, and then permutes them according to the contents of the second plaintext. Like the BNCC cipher described above, we the coordinates that we use are for a four-dimensional grid in which we place a mixed 24-letter alphabet. However, since we cannot rely on a positional number system (i.e., one that keeps digits in order), we have to expand the set of ciphertext symbols so that each coordinate has its own set; we did this for the first example cipher above, when we used ten row and column labels instead of five. Unlike the BNCC cipher, we will not use seriation and fractionation.

Let's work through a short example to see how the cipher works. Start with these two plaintexts. One is padded, since they must have the same length (spaces don't count):

```
#1:   DONT MAKE FUN OF ME
#2:   RIGHT BACK AT YOU X
```

The key consists of two keywords; suppose they are SNICKER and LAUGH. For labeling cells in the box, we can use these sets of symbols: {A, B}, {C, D}, {E, F}, {G, H, I}. Then encoding the first plaintext is done with this assignment, using the mixed alphabet obtained from the first keyword:

```
S  ACEG   R  ADEG   H  BCEG   T  BDEG
N  ACEH   A  ADEH   L  BCEH   U  BDEH
I  ACEI   B  ADEI   M  BCEI   V  BDEI
C  ACFG   D  ADFG   O  BCFG   W  BDFG
K  ACFH   F  ADFH   P  BCFH   X  BDFH
E  ACFI   G  ADFI   Q  BCFI   Y  BDFI
```

The first plaintext is provisionally encoded as

```
ADFG BCFG ACEH BDEG BCEI ADEH ACFH ACFI ADFH BDEH ACEH BCFG
ADFH BCEI ACFI
```

The second encryption scheme enumerates the permutations of four objects. Luckily, there are 24 of them. We use the second keyword to assign a mixed alphabet to them:

```
L  (0, 1, 2, 3)     C  (1, 0, 2, 3)     M  (2, 0, 1, 3)     S  (3, 0, 1, 2)
A  (0, 1, 3, 2)     D  (1, 0, 3, 2)     N  (2, 0, 3, 1)     T  (3, 0, 2, 1)
```

U	(0, 2, 1, 3)	E	(1, 2, 0, 3)	O	(2, 1, 0, 3)	V	(3, 1, 0, 2)
G	(0, 2, 3, 1)	F	(1, 2, 3, 0)	P	(2, 1, 3, 0)	W	(3, 1, 2, 0)
H	(0, 3, 1, 2)	I	(1, 3, 0, 2)	Q	(2, 3, 0, 1)	X	(3, 2, 0, 1)
B	(0, 3, 2, 1)	K	(1, 3, 2, 0)	R	(2, 3, 1, 0)	Y	(3, 2, 1, 0)

The second plaintext then becomes this list of permutations:

(2, 3, 1, 0), (1, 3, 0, 2), (0, 2, 3, 1), (0, 3, 1, 2), (3, 0, 2, 1), (0, 3, 2, 1), (0, 1, 3, 2), (1, 0, 2, 3), (1, 3, 2, 0), (0, 1, 3, 2), (3, 1, 0, 2), (3, 2, 1, 0), (2, 1, 0, 3), (0, 2, 1, 3), (3, 2, 0, 1)

We apply these permutations to the blocks of symbols from the provisional ciphertext of the first plaintext:

```
#1:   ADFG BCFG ACEH BDEG BCEI ADEH ACFH ACFI
#2:   (2, 3, 1, 0) (1, 3, 0, 2) (0, 2, 3, 1) (0, 3, 1, 2) (3, 0, 2, 1) (0, 3, 2, 1) (0, 1, 3, 2) (1, 0, 2, 3)
      ---------------------------------------------------------------
      GFAD FBGC AHCE BEGD CIEB AHED ACHF CAFI

#1:   ADFH BDEH ACEH BCFG ADFH BCEI ACFI
#2:   (1, 3, 2, 0) (0, 1, 3, 2) (3, 1, 0, 2) (3, 2, 1, 0) (2, 1, 0, 3) (0, 2, 1, 3) (3, 2, 0, 1)
      ---------------------------------------------------------------
      HAFD BDHE ECHA GFCB FDAH BECI FICA
```

The final ciphertext is

GFADFBGCAHCEBEGDCIEBAHEDACHFCAFIHAFDBDHECHEAGFCBFDAHBECIFICA

Decipherment requires that one knows the alphabet key(s), the choice of symbols for the digits of the mixed-radix numbers, and which digit of those numbers is the ternary digit. However, this is not a particularly difficult cipher to cryptanalyze, as we shall see. First, we break the ciphertext into blocks of four symbols. Then, by noting which symbols do not occur in the same block as others, we can assign them to groups, each of which represents the first, second, third, or fourth digit of the mixed-radix numbers. We can then assign a mapping from those numbers to the 24-letter alphabet. The resulting text can be analyzed as a monoalphabetic substitution cipher, which can be broken with the hill-climbing technique in Unit 28. The orderings of the digit symbols (i.e., the permutations) can also be mapped to the 24-letter alphabet. The resulting text can again be analyzed as a monoalphabetic cipher. Finally, it is possible to recover the keywords (if any) that were used to reorder the key alphabets.

Let’s crack this ciphertext as an example:

```
ZTXSXTYRVSYXXZUTRWYTTSYWTUZXTWZUTZSXWTSZWTYSVRZXVUWZSTXZYTX
RSTYWRTZXWVZUVSZXYUVXTUXYWZVRTRYXUTYWTZUWYUTWTUYWZVXUTUXZRT
XYYTXRRTZXTSWYSYVXSZTXVZSWTXZSVUWYRTZXRXYTZRTXYSTWVRZWRTWYU
VYXWYTSTRXYYRXTZTXRWTYSTRYXTXUYTXZSTUXZTZWUTRYWTYUWXVSZWTYS
VUYWTRXZSTWYYTXRRTZXWTYUVUZWUVYXZTXRTYRXUZTXVSZXYTWRXTZRTUZ
```

```
XZVXSTRWYUTXYYTWUUVZWVSXZTWYRXYTRZTXRTYSWTYUXTWSYZTWRXTYUTS
YWYUVWXRTZSYVXTSYWVSXZTXYRVUWYYWRVVRWYYRWV
```

Our first step is to break the ciphertext into blocks of four symbols:

```
ZTXS XTYR VSYX XZUT RWYT TSYW TUZX TWZU TZSX WTSZ WTYS VRZX
VUWZ STXZ YTXR STYW RTZX WVZU VSZX YUVX TUXY WZVR TRYX UTYW
TZUW YUTW TUYW ZVXU TUXZ RTXY YTXR RTZX TSWY SYVX SZTX VZSW
TXZS VUWY RTZX RXYT ZRTX YSTW VRZW RTWY UVYX WYTS TRXY YRXT
ZTXR WTYS TRYX TXUY TXZS TUXZ TZWU TRYW TYUW XVSZ WTYS VUYW
TRXZ STWY YTXR RTZX WTYU VUZW UVYX ZTXR TYRX UZTX VSZX YTWR
XTZR TUZX ZVXS TRWY UTXY YTWU UVZW VSXZ TWYR XYTR ZTXR TYSW
TYUX TWSY ZTWR XTYU TSYW YUVW XRTZ SYVX TSYW VSXZ TXYR VUWY
YWRV VRWY YRWV
```

Then we tabulate the occurrences of symbols in the same block. In this grid we see an ‘X’ when two symbols appear together in at least one four-symbol block.

	R	S	T	U	V	W	X	Y	Z
R			X		X	X	X	X	X
S			X		X	X	X	X	X
T	X	X		X		X	X	X	X
U			X		X	X	X	X	X
V	X	X		X		X	X	X	X
W	X	X	X	X	X			X	X
X	X	X	X	X	X			X	X
Y	X	X	X	X	X	X	X		
Z	X	X	X	X	X	X	X		

We can see that R, S, and U never appear in the same block. Therefore, they must be the symbols for the ternary digit. Likewise, T and V do not appear together, so must represent one of the binary digits. The complete set of sets is

{T, V}, {W, X}, {Y, Z}, {R, S, U}

We can now map combinations of these symbols to the 24-letter alphabet. The choice of mapping is irrelevant, since we will be analyzing the result as a monoalphabetic substitution later. So, without loss of generality, let us use this mapping:

TWYR → A	TXYR → G	VWYR → N	VXYR → T
TWYS → B	TXYS → H	VWYS → O	VXYS → U
TWYU → C	TXYU → I	VWYU → P	VXYU → V
TWZR → D	TXZR → K	VWZR → Q	VXZR → W
TWZS → E	TXZS → L	VWZS → R	VXZS → X
TWZU → F	TXZU → M	VWZU → S	VXZU → Y

With this mapping, we obtain this intermediate ciphertext:

```
LGUMABMFLEBWSLGBKSXVIQGCFCCYMGGKBULRLPKGKBQAVBGGKB
GILMFACXBPKBGKCSVKGMXAKMXAICSXAGKBIBDIBPKUBXGPNNN
```

Using our hill-climbing attack on the monoalphabe monoalphabetic substitution cipher recovers the first plaintext (clearly, the trailing XXX is padding):

```
IT MADE ALICE QUITE HUNGRY TO LOOK AT THEM. I WISH
THEY'D GET THE TRIAL DONE, SHE THOUGHT, AND HAND ROUND
THE REFRESHMENTS. XXX
```

The substitution key is M?EABDVKL?YFUXC?WIPGS?RNQ?, where '?' denotes unknown parts of the key because the plaintext does not contain 'B,' 'J,' 'P,' 'V,' or 'Z.'

To decrypt the second plaintext, we need to identify the permutations used to reorder the symbols in each four-character block of the ciphertext. To that end, we need to order the sets of symbols, so we will say that they should be in the order {T, V}, {W, X}, {Y, Z}, {R, S, U}. The first block of the ciphertext is ZTXS, so its permutation is (1, 2, 0, 3), because it takes TXZS to ZTXS. The permutations for all blocks of the ciphertext are

(1, 2, 0, 3), (1, 0, 2, 3), (0, 3, 2, 1), (3, 0, 1, 2), (3, 1, 2, 0), (0, 3, 2, 1), (0, 3, 2, 1),
(0, 1, 2, 3), (0, 3, 1, 2), (1, 0, 3, 2), (1, 0, 2, 3), (0, 3, 2, 1), (0, 2, 3, 1), (1, 2, 3, 0),
(1, 2, 0, 3), (1, 3, 2, 0), (1, 3, 2, 0), (1, 0, 2, 3), (0, 3, 2, 1), (2, 3, 0, 1), (0, 2, 3, 1),
(2, 0, 1, 3), (0, 3, 2, 1), (1, 3, 2, 0), (0, 3, 1, 2), (2, 3, 0, 1), (0, 3, 2, 1), (1, 2, 0, 3),
(0, 2, 3, 1), (1, 2, 3, 0), (1, 2, 0, 3), (1, 3, 2, 0), (0, 2, 3, 1), (2, 3, 1, 0), (2, 3, 1, 0),
(0, 3, 1, 2), (0, 1, 2, 3), (0, 2, 3, 1), (1, 3, 2, 0), (3, 1, 2, 0), (2, 3, 0, 1), (2, 3, 0, 1),
(0, 3, 2, 1), (1, 2, 3, 0), (1, 3, 2, 0), (2, 0, 1, 3), (0, 2, 3, 1), (3, 2, 0, 1), (1, 2, 0, 3),
(1, 0, 2, 3), (0, 3, 2, 1), (0, 1, 3, 2), (0, 1, 2, 3), (0, 2, 3, 1), (0, 2, 1, 3), (0, 3, 2, 1),
(0, 3, 1, 2), (1, 0, 3, 2), (1, 0, 2, 3), (0, 3, 2, 1), (0, 2, 3, 1), (1, 2, 3, 0), (1, 2, 0, 3),
(1, 3, 2, 0), (1, 0, 2, 3), (0, 3, 2, 1), (1, 3, 2, 0), (1, 2, 0, 3), (0, 3, 1, 2), (2, 3, 1, 0),
(0, 3, 2, 1), (1, 2, 0, 3), (1, 0, 2, 3), (0, 3, 2, 1), (1, 2, 0, 3), (0, 2, 3, 1), (1, 2, 3, 0),
(1, 2, 0, 3), (1, 3, 2, 0), (0, 2, 3, 1), (0, 1, 2, 3), (2, 0, 1, 3), (1, 2, 0, 3), (0, 3, 1, 2),
(0, 3, 1, 2), (0, 1, 3, 2), (1, 2, 0, 3), (1, 0, 2, 3), (0, 3, 2, 1), (2, 3, 0, 1), (2, 0, 3, 1),
(2, 3, 1, 0), (0, 3, 2, 1), (0, 2, 3, 1), (0, 1, 2, 3), (0, 2, 3, 1), (3, 1, 0, 2), (0, 2, 3, 1),
(3, 2, 0, 1)

The mapping of these permutations to the 24-letter alphabet is arbitrary. Let us take a canonical one.

(0, 1, 2, 3) → A	(1, 0, 2, 3) → G	(2, 0, 1, 3) → N	(3, 0, 1, 2) → T
(0, 1, 3, 2) → B	(1, 0, 3, 2) → H	(2, 0, 3, 1) → O	(3, 0, 2, 1) → U
(0, 2, 1, 3) → C	(1, 2, 0, 3) → I	(2, 1, 0, 3) → P	(3, 1, 0, 2) → V
(0, 2, 3, 1) → D	(1, 2, 3, 0) → K	(2, 1, 3, 0) → Q	(3, 1, 2, 0) → W
(0, 3, 1, 2) → E	(1, 3, 0, 2) → L	(2, 3, 0, 1) → R	(3, 2, 0, 1) → X
(0, 3, 2, 1) → F	(1, 3, 2, 0) → M	(2, 3, 1, 0) → S	(3, 2, 1, 0) → Y

We get this intermediate ciphertext:

```
IGFTWFFAEHGFDKIMMGFRDNFMERFIDKIMDSSEADMWRRFKMNDXIGFBADCFEHG
FDKIMGFMIESFIGFIDKIMDANIEEBIGFROSFDADVDX
```

Analyzing it as a monoalphabetic cipher yields

```
THE QUEEN OF HEARTS SHE MADE SOME TARTS ALL ON A SUMMERS DAY.
THE KNAVE OF HEARTS HE STOLE THE TARTS AND TOOK THEM CLEAN AWAY.
```

with key `D?ONFH?G??BSRAE?TKMIWCV?X?`, where again '?' denotes unknown elements.

Having recovered both plaintexts, we could stop at this point. However, if we desire, we can try different ways of assigning the symbols `R`, `S`, ..., `Z` to the four mixed-radix digits and search for a recognizably keyed alphabet. It is easier to begin with the second plaintext. If we take the mapping of permutations to letters and apply the substitution key that we found above, we have

(0, 1, 2, 3) → N	(1, 0, 2, 3) → H	(2, 0, 1, 3) → D	(3, 0, 1, 2) → Q
(0, 1, 3, 2) → K	(1, 0, 3, 2) → F	(2, 0, 3, 1) → C	(3, 0, 2, 1) → ?
(0, 2, 1, 3) → V	(1, 2, 0, 3) → T	(2, 1, 0, 3) → ?	(3, 1, 0, 2) → W
(0, 2, 3, 1) → A	(1, 2, 3, 0) → R	(2, 1, 3, 0) → ?	(3, 1, 2, 0) → U
(0, 3, 1, 2) → O	(1, 3, 0, 2) → ?	(2, 3, 0, 1) → M	(3, 2, 0, 1) → Y
(0, 3, 2, 1) → E	(1, 3, 2, 0) → S	(2, 3, 1, 0) → L	(3, 2, 1, 0) → ?

By reassigning the symbols 1, 2, 3, and 4 we can try to find a mapping that corresponds to an easily recognized keyed alphabet. In this case, it is simple: we merely exchange 2 and 3.

(0, 1, 2, 3) → K	(1, 0, 2, 3) → F	(2, 0, 1, 3) → C	(3, 0, 1, 2) → ?
(0, 1, 3, 2) → N	(1, 0, 3, 2) → H	(2, 0, 3, 1) → D	(3, 0, 2, 1) → Q
(0, 2, 1, 3) → A	(1, 2, 0, 3) → R	(2, 1, 0, 3) → ?	(3, 1, 0, 2) → U
(0, 2, 3, 1) → V	(1, 2, 3, 0) → T	(2, 1, 3, 0) → ?	(3, 1, 2, 0) → W
(0, 3, 1, 2) → E	(1, 3, 0, 2) → S	(2, 3, 0, 1) → L	(3, 2, 0, 1) → ?
(0, 3, 2, 1) → O	(1, 3, 2, 0) → ?	(2, 3, 1, 0) → M	(3, 2, 1, 0) → Y

This gives us `KNAVEOFHRTS?CD??LM?QUW?Y`. The missing letters are obvious at this point, and the keyed alphabet is `KNAVEOFHRTSBCDGILMPQUWXY`, so that the keyword is `KNAVE OF HEARTS`. We also learn from this procedure that the ternary digit must come third in the mixed-radix numbers, rather than fourth as we had used earlier, since we had to swap the last two elements of each permutation.

We now turn to the task of recovering the keyed alphabet that was used on the first plaintext. If we apply the substitution key that we found, we get a new mapping:

TWYR → D	TXYR → T	VWYR → X	VXYR → ?
TWYS → E	TXYS → ?	VWYS → ?	VXYS → M
TWYU → O	TXYU → R	VWYU → S	VXYU → G
TWZR → F	TXZR → H	VWZR → Y	VXZR → Q
TWZS → C	TXZS → I	VWZS → W	VXZS → N
TWZU → L	TXZU → A	VWZU → U	VXZU → K

Since in recovering the second keyword we had to swap the last two elements of each permutation, we will do the same to each block above, and then reorder.

TWRY → D	TXRY → T	VWRY → X	VXRY → ?
TWRZ → F	TXRZ → H	VWRZ → Y	VXRZ → Q
TWSY → E	TXSY → ?	VWSY → ?	VXSY → M
TWSZ → C	TXSZ → I	VWSZ → W	VXSZ → N
TWUY → O	TXUY → R	VWUY → S	VXUY → G
TWUZ → L	TXUZ → A	VWUZ → U	VXUZ → K

We now look at various ways in which to assign the values of 0, 1, and 2 to the symbols representing digits. If we take T=0, V=1, X=0, W=1, U=0, S=1, R=2, Y=0, and Z=1, and reorganize the mapping accordingly, we get

TXUY → R	TWUY → O	VXUY → G	VWUY → S
TXUZ → A	TWUZ → L	VXUZ → K	VWUZ → U
TXSY → ?	TWSY → E	VXSY → M	VWSY → ?
TXSZ → I	TWSZ → C	VXSZ → N	VWSZ → W
TXRY → T	TWRY → D	VXRY → ?	VWRY → X
TXRZ → H	TWRZ → F	VXRZ → Q	VWRZ → Y

We see that the keyed alphabet is `RA?ITHOLECDFGKMN?QSU?WXY`. Again, the missing letters are obvious, and the full key is `RABITHOLECDFGKMNPQSUVWXY`, giving a keyword of `RABBIT HOLE`.

Finally, we would like to say that this cipher was dubbed the “MadHatter” cipher by a participant in the online forum for the BNCC. A ciphertext encrypted with the cipher was offered as a bonus challenge in the 2019 competition.

Reading and references

Merle E. Ohaver, “Solving Cipher Secrets,” *Flynn’s*, October 2 and November 13, 1926, toebes.com/Flynns/Flynns-19261002.htm, toebes.com/Flynns/pdf/Flynns-19261002.pdf, toebes.com/Flynns/Flynns-19261113.htm, toebes.com/Flynns/pdf/Flynns-19261113.pdf

Chris Christensen, “Lester Hill Revisited,” *Cryptologia* 38:4 (2014) 293-332, DOI: 10.1080/01611194.2014.915260

Thomas Kaeding, “MadHatter: A toy cipher that conceals two plaintexts in the same ciphertext,” Cryptology ePrint Archive, report 2020/301.

Grant A. Niblo, “The University of Southampton National Cipher Challenge,” *Cryptologia* 28:3 (2004) 277-286. DOI: 10.1080/0161-110491892935. The current (or most recent) challenge is at www.cipherchallenge.org. The 2019 challenge is archived at 2019.cipherchallenge.org.

Programming tasks

1. Implement an encryptor and a decryptor for Levine's simple cipher. Try them on a some plaintexts of your own.

2. Implement an encryptor and a decryptor for the first example of a duplicitous cipher.

3. Implement the attack described above for the first example of a duplicitous cipher.

4. Implement an encryptor and a decryptor for the mixed-radix cipher from the BNCC.

5. Implement the attack on the fractionated mixed-radix cipher from the BNCC.

6. Implement an encryptor and a decryptor for the second example of a duplicitous cipher. Feel free to call functions you wrote for Exercise 3.

7. Implement the attack described above for the second example of a duplicitous cipher. Feel free to call functions you wrote for Exercises 3, 4, and 5.

Exercises

1. Finish breaking the example of cryptanalyzing the first duplicitous cipher.

2. The following ciphertext was encrypted in a way similar to the first example of a duplicitous cipher. Break it and find both plaintexts. What are the keywords?

```
272515275805153890580751688591850745386172034985703851
260783437019851583584778683025071945178586232758058530
078907833443836186890738156085038787508387052530518525
342568688370864515831692278307255186520972516845250778
301717257090512785505803055878853887868509850715688519
850734273889257168252685503861079803608361154303541703
709883078654913071261527851917588649857130511903711316
850530895871512785508503058578588387688534273898837083
151758581903701772259027851751034983923015258370250778
836825717252070951278523868338053425157215728552506083
831571512758508503055887583887865809850751868591850743
728350852361718515381730541572031551278545305085273087
784530709885072583545207903023168686682532853472580723
835052071715030792581572582505150583161785501771855178
508387585086541527585058030558788583788685232507308668
454327838952178652268598037127250709030798437225508625
070903608316153283050783830625582951323043072507900307
986825922625070951725889167115030789300638498503868690
850751868519580778832625709015275825050738715817432785
505851275845030585708351430370158589254958518368984583
```

```
610386193817518549855045512725079090857015685891850707
384330868638341985153834525127985030344774747 4
```

3. Encipher these two plaintexts with the same cipher and keys as in the previous exercise.

   #1: `CRYPTOGRAPHY IS`
   #2: `FUN`

4. This ciphertext was encrypted with a cipher similar to the second example above of a duplicitous cipher. Break it and find both plaintexts. Are there any keywords?

```
586951737184523778144879184684968641895641866185342641
876489741886597258493669584936428731654187572384616895
537247181487486282478157275372857428569887525187586913
574871175372853649894681474861824731569856537171486253
864132574136753157314186537274814618487231568659573141
873157864159686985418757238497418773258417513748615693
864274827143659881744186486189564862689581646135852752
376325528793464869714874896418649864894286698565984826
428771435186418765894879748147188427598673154178365278
156148698469858416649824368471841772341846487261536985
351748717285695835174187327595867948728584974871257341
874396498642867482471368245986486189648641643134694869
531764282587728472358417498689568642742874315869753147
186284352741872436658187524789649878257482486265894618
728435166598614842365862648284723147752869534871237564
286589896451377148614851366539847152377394694384276413
285752361578528773256985841651866324418718575872725369
853517418735617825523753627285469386497148859671537285
364931473157895616483642648234614871728584274236739441
684986428669856498648198463426874147813427624884964936
342743725728698478148642598656137285741846892487698562
358426428763518956153781745689486186154236714874988641
984648616243648172354713781452874872817486516439856958
695173718459867315537281478147794884175237714848613642
462361347523852749876315714851368956527814377481895651
377148725318465986782542874871314682577352487278147235
752851786284859642877258537282755287562378245287537282
757528653289561753874147936413352764287841513687526934
695889741753471878245986537181744379648946187489641862
346284341641876314528756238624874274318569715387415723
741864897234324687418724598643975287815775237582857236
526489846162846513486164326428613484174286782463514286
731472486418748269584623739441864872316586491573418748
698461689551365872798482748642748246186234641835727413
487164815723235643613715418684263752816458697582472871
484187428713568527587169853517487168596841518664327148
```

5. Encipher these two plaintexts with the same cipher and keys as in the previous exercise.

#1: WELCOME TO
#2: MY TEA PARTY

6. This is the final challenge from the 2019 British National Cipher Challenge. Break it. What is the keyword?

```
11101 00101 00001 21122 10001 10111 00101 10001 01111
00210 11011 11111 01011 01001 01011 21001 11000 01011
01000 00012 21222 10011 11010 01000 01001 10201 00000
11101 10010 01011 00012 00212 00011 01000 01101 01110
00000 20021 10010 00100 11010 11011 00002 10000 10000
11010 00001 00022 00110 00101 10101 10001 01100 00210
01111 10010 01001 00000 10111 02020 00101 00101 01001
11000 01022 21210 00001 00110 10101 00010 21020 02111
00100 00110 11101 10000 21012 01110 11001 10011 00000
10021 10011 01101 01100 00011 00002 10011 11100 10010
01001 00101 10020 00021 00001 01010 01010 01001 10002
21011 00011 10111 10000 00120 12102 01001 10001 10010
10110 01021 02011 01111 00101 00000 10001 10022 20000
00100 01010 01100 01110 21021 00101 00100 11101 10000
21002 12111 10100 01100 01011 10002 01202 01010 11001
00010 01111 00011 00101 00010 11110 10000 00012 01122
00111 00001 11110 10000 00021 12210 01001 10100 10100
10010 22002 00001 10000 01101 11101 00100 21001 01000
11110 10000 00001 01202 20200 00011 10110 01000 01002
22100 21101 11000 00100 01110 00200 12011 00000 11010
10010 00111 10221 00100 01010 11110 10001 00102 21220
10000 00110 10100 10000 02100 22101 00000 11010 10001
01012 00200 00100 00001 11101 10100 01021 10001 00010
11000 11010 01101 02111 10001 01110 11001 00000 01002
11002 10110 11110 10011 11011 01000 00210 00010 11101
00110 00001 22221 20000 11010 00110 11100 00100 02210
01101 10000 01100 01011 11122 10001 01101 01000 00110
10021 10021 10101 11100 11101 10110 01021 20010 10010
01110 11101 00100 21220 01010 00011 00010 00100 11010
12201 00100 10011 10000 11000 21112 00111 00011 10010
00100 00001 20010 11101 00011 10000 11001 01112 00211
10011 01011 01011 10001 10200 21100 10000 11010 00001
00020 20010 01111 00111 11001 11011 20112 10101 00111
11010 00100 01001 12202 11100 10100 00100 01001 10002
11010 11110 11010 01100 00002 00202 00100 00110 01011
01000 01200 02111 11101 01001 10000 00100 20001 20001
11101 00011 10000 11001 01112 10011 10000 01110 10010
10021 10101 11111 11001 01000 01011 11002 11110 11110
10101 10110 00021 02220 11100 11010 11000 01000 00202
01100 00100 11011 00111 10021 00200 11000 10100 11000
00001 12020 20100 01110 11001 01100 01011 11002 11110
```

```
11111 11011 10101 01001 10220 11100 01010 00100 01001
02002 11101 1111
```

7. This is the bonus challenge from the 2019 BNCC. Break it and recover both plaintexts. What are the keywords? (This ciphertext used the *inverse* permutations, so you will have to adjust accordingly when you look for the second keyword.)

```
JADGGBJDEJGAJGEBJDHADLHBGJDADLGBDBJHJADHDAGKAJEGDKAHAH
KDAJEGAKHDKAGDALEGDKBHDKGAHJDAEAGLLADHHJDADLGAELHBELGA
EAGJDGLAHAJDJEHABHKDJBEHGBLDEAGJALEGEAGLGJEAHDAKGKDABD
JHLBHEBKDGDJAHLADGGBLDEJGABDLGLAEGHAJDDLGAHKDBDAKHALEG
DKGBDHJABHDJDAGJAKDHEKAGDAGJADHKAHJEDHKBEJHBBGDKAJDGDL
HAJEGAKDHAGAKDAHKDDHKBGALEKDGBEAGJDAKHKDGAGALEBHKDEHJA
HBKDBHKDAKDHEBGJHDAJBDKHGDALELGABLDHJAEGEGBJDJHADGLBDB
GKJAHDDBKGDJGAAEJHAKEHDJHAADJHHAEKJADHHAJEEJGAHDAKGALE
DBHKELAGDKBGJAHDHDBJDKHBHDBKEGAKGKDBADJHAGLDDLBGEJGAGL
DBELGADJAHADLGBLDHGAJDBGLDDALGDHJAADJGAEHJAEGJDAHKGKDA
DBHJBLEGELAGJAEGGALEDKGBADJGJAEHHAKDDKHBHKEAEJAGEJGBEL
GABGLEDALGJAHDGDLBHDKBADLGEBJGDKHBKADHHALDDJHALGDALDBG
DJGAELGAEAGJDHKBHAKDBLGDAEJGKADHJAEGLEGADJGADKHAEJAHBD
HLDKGBGJDAAELGJAGEBLDGELAGDBGKJADHLBEGGBLDDJHADHBKEBHJ
DGJABHDJBDKHHBDKKAEGGALEDKGBDHKBBELGADGKBDGKEAGLGJDADB
JGGAJEEJGBDAJHDBHLAGJEELAGKBDGBDKHGBLEELGAHKEAEAJGEJGB
AEGLGBELLADGHAJDDLGBKHDBDALGJBEGBDKHADKHDLAHDAJHADLGDB
LGJADGAGLEAEGJBDHKKADHGBLDDBGLKBDHDBLGGBKDDJHAHLBDJAGD
BGLDJBEGDKHBHADKBDLGGJEAAJHEADJHLADGJAEGHAKDDKGAJGEAKD
HADKGBHJDADLGADBHKBDHLAHKDKEBGEJGADAHKAJEHDJAGDBLGHBLD
DJHAJDGBDBGJDAJGBGLDLBGDDBKGJADHBJGEBDKHBGLEBJDGAJHEHJ
EBBDKHLAGDALEGDKBGJADHBDKGDBKHELGAEJHAAGJDHBELKBEGGAJD
EJHAJDHABDKGHJDADLAGJBEHJDHADJHADJGBDLAHADHJDLGALEHBDB
GLDBJGJDHAHJDAAHKELBEHJADGDKHAAJEHDBGLLAEGLBEGHAKEEJGA
EHAJDBHLDGKBAHDJAELGGBDKDJHADAGLEGLAGBKDDHJAAEKHBDGJAD
HJDAGJGLDBLEGBGALDDJHABKDHEBJHGBKEDJGAKEGAEAGJDAKHDAKG
JADGEAHJGJDAJAEGBDGLELHBEJBGDGBKJGDAEAGJKDAHBDLHBHKDEG
LBDJGBEHJABDJHHDAJBHLDBKDHALGDGAELGKDBAELGGDBKJADHGALE
DLGABDKHBLEGDJHBGDBJDAHJHKDBBHEJDKGADJAHELAGLAGEGEAJAH
KDGADKLGEBAEKHJAGDKADHHAJEAHKEAJEGBGEJKEGAEJGAHKDADKGA
LEGADBGKJHADJEHAADJGGJEABGDLEAJGHAJDDLGBDBLHDBGKAHJDDK
AHLGDBBELGJAHEAEJHJAHDAKHDGBJDLBEHJADGBDLHGDKBEAGJLGEA
JHDAADLGJAGDBHDJBJDHAEJGLEGALDHBEAGJLGAEKDGBAEKHGJEAAH
DKEAKGJDHAELHBHJDADLGBALDGJADGHDAKEBGJDJBGDKHBLBDGADJH
DHBJELHBBDGKADJHADLGAGELDKGBDJAHDAGLJADHHBDLADJGGLDBAK
DHBDKHLAEGGBKDAGJEAKDHAKGDBLDGBDKHHALDADJHLDGAEHLBADLG
AJDHKBEGGJDADAGLGKEAAGJDJDHBBGJDDAJHEJGADAHKELAGDKBGDJ
GAELAGKADHHBKDDLGAAEJHAJEGEJHAGDAJDBGJGJEABGEJDJHAELAG
DKBGEJGADKHAAGKEGAEJLGEABDLGKBHDLADHHAJDAGLDBLEHBGEKLE
GBEJGBGKDBDKHBEBGLAEGLBHKDJEBHELGAKGDBADJHLBHDADJGLBHE
ALGEHBKDKBDGJADHADJGDALGGELADKGBDAHJAGDLAGJDHDBJDBJHJA
GEGLEADLGBAGJDHEBLLEAGDKHBGJEALGEADBLGADJHGJDBHJEBKHDB
```

KDGBEAHJDAJHDAGJADGLBDHKDBGKAEHJAHJDDGJADLGAEGJABDLGKB
GDADGJDJBGDBJGDBHJJHDAGAELBDKHHKDBJBDGGAJDELGAJDHABDLH
GKDBDJAHDAKHBDLGGBDKJADHGALEDKGBBDKHBEGLDKAGDBKGAELGAJ
EGALGEBHDKDLHAHDJALADGGJDABHJEALEGAJHDAGLDBLDHAJGDLADG
HAJEGLDBAEJGLAGEBKHDGBJEGBJEELGBGLDAGALDDJHAEAHJELAGBD
HKBGKDAHDJDLGALAEGGBKDDJGALAEGBGLDDGKBAHDJBHKDBLEGAKGD
BDKGGAELEAGLDHKBBDGKADJGDHLAAHDJDBHLBKDHDKAHEAJHHAJDDL
GADHJAAEJHJAGDAEGLELAGGBKDEAGJGLDBDJBHBEGLLGEAGAJDELGA
AEGLGDKBDAJHEAGLJGEABGKEAJDHAJGELAEGGAJDDJGBGDBJBDLGJA
HDJADHBEGKADJHJAHEEKBHLBEGJEGAELGADJHADKAHADGJELGAELBG
LAGDGDAJDBGJBJDHELBGLAEGLDHBDKGBDJHADKAHAEGLDKGBDJAHDL
GADJGADBHJBJHDGJEAELGADJGAEJBGLAEGBELGGAJDDBGJJDBGELHB
EAGLDHBKDBHKKAEGJADGBDHLDJGAGAELBEJGGKDBKBDHLEGBELGADK
HBEJBHAEGJELGADLBGLBHDGDAJEAJGDBGLAGLEEJBGKBDHJDGAELGA
EKHADKBHBEGJEKGADJAHELGADJGAHAKDEAHJDJBGDBKHHDKBAEKGAH
JDAEHJDJGAELAGGAJEAGLEDAJGAKDHAHJEEGLADKGBAJDHDKHAKDGB
GLEBDAGLLADGJAEGADHJEJHABHKDKAHDDAJGJBDGJAEGGBJEDJHABL
DGGAELADGJDLGALEGADJAHAEJHLEGABKDHDBGKHDAJADLGBJEHDAJH
JADHGLEABEJHKBHDDALGEGJAGLEABEJHJDGBDAGJDBLGDGBKADHJEA
HJDGJABGEJDAGLDBKHDBGLLBDGDBGKHJDADGALEBGKGJEAHKDAEJHA
AEGLKDGBGAJDAGLEDGLBGBKDHADJGKDBAGJDAJHEHAKDJADHDLHAAJ
DHDLGADJBHDHAJBEJHKBDHGALDJADHGBLDDJHAHAJDDKHAGJDAADLG
DJGADBJHDBJHAEJGLAEGHBLDEJGALGEAGKBDJAHDAGJELAEGDKGBHA
DJADLGGJDABLHDADJGJAEGLBDGGALEEJGBBHKDGJDAGLAEKAHEDBHK
GJEBGKEADJHAAEGLBHKDAGDLAJDGBDLHJDGALEGAGBJEDBGKLAEGKH
DBAELGJAGDGKEADJHAKBDHGBLEAGLEBKDHBHEJAGJEALEGAJGDDAHK
EHAJBDJHLBGELADGDAKHEAGJAHKDDKAGLBDHJEGAELGADKGBEJBGBE
GLDLGAEJAGKBHDGDBLEGAJLAGELBEHBGLDKBDGAHJDDALGAJDGKADH
DAJGBEJGALDGBDKHGBLDBDLGLEGADGKBADJHBJEHJAEGHJDADBGJHJ
EAAGJDJEHBAGLEDAJHDLGAEGJAAELGAJDGKADHAHJEBLDHJAGDDLGB
EJAGLBEGLDGBELGAEJGADKAHAEGLEJGAEKBGDJHAELGABHKDGBDLAH
JDJADHJEGAEAGLEAKHDBKHAEKHAHJEKBDHHDBLAHKDAJGDGBJDJADG
DGALDKGADAHJALDGDJAGBHJDHBDJAGJELAEGKDGBDKHBGJDBDJHAEB
LGADKHAHJEJADHLDGAAELGKBGDADHJDKBGDHAJEJHADGAKJAHDJAEG
DAHKDAJGAHKDKBDHAGLEDBKGAJDHLADGBGKEKBDHEBGKAHJDAKDHLA
EGJAEHHBKDDBHLKADHDBLHADHJADKHALGEAGJDDGJBAGJEJEGBDAHJ
DAJGJBEHGALEAHJDALDGAGEJAGLEAKDHAJHDDAHLADHJAGLDDHKBDK
HAEBJGDAHJBGJEDKBHKDAHDLGBGJEAJHEAHDJAADLGAEGJAHKDKADG
DKGBBDHKLHDBJEGAHAKDAGLEDGKBHAJDBHLDDBKHLADGGBJDEJHADG
BLDKGBDAJHLBDHJDGADBGLLGEABDKHKAGDJADHAELGHBDKLBEGGALE
AGJDDAKGJDGAAEJGADKHELAH

Challenge 1

```
10000011002233001011010121201100000010002232000010010121201
00000000111221230010021120012220100021220123120000120001223
22110011100101032000000000032023010000202020222000010000230
12000011200112201200000200000022300000111201230000001000112
```

```
231200000100001022301000212001230200000002022312200000000000
230220011100111322010000000000022331000010000232020000000000
023012000000001003000100000000112300200000000003201200000001
003000100000100002012300010010120001000010000102201200000000
200323030110012002311201000012000210120000100111300120001120
001022110010000011103202010101200031203111001212221211011000
000012303311001200211120200000111121023101001122002022301100
101120012121110012102221200100001001012300010011101233320100
100101212021110010020010123001111000123032000001102011202000
011001003010200010012120202110100001100022300000110002101200
000000111300120000012222303333
```

Challenge 2

```
AFKGOWTPY472DKCGMTXQ1805DCKHNTVRY852FALHNTVPY186HDCKSNXP720
5GEAKNQTVY681EKHBMSPX81Y6HFCJROTV248YCDJHPVMTY681FGKAOTWP53
Y7DJBGOQVT2608BGJEXSNR9Y16BJGDWQOT27Z4DBHKVUMP168YKECGNVTPY
681ECGKSPMW5Y27JFAITROVY481FKAINPWT14Z8AGKEWMPS0762LDBHPVNT
1Z59GFAKNUWPY274GBKFMXQS2074KEAINWUP167YBJDHNUVQ7Z24BDJHMWU
PZ482CJDHPOTV5Z29GBJEVROT61Y8CHKEWPNU274YDIJBMQXU51Z8DHKCNV
RTY671GFBKNPVU4Y72ALEHQTVM1Y67GFBKTNPV159YBJDHOTWR24Y7FAHKV
TNQ167YFKBGPNTV347YAFHLNWPT7Y24EBGJNPVT24Z9AHLFTNPV167ZKEBG
PMVT34Z8DAGKWSMP0681JFIAOSWPY836DBGKUMQW61Y7JEAHWNRS25Y7AFK
ISPMW167ZDBGLTPMV6Z38KGBDPUVNZ482BIJDPTNW724YAHLFTMQWY274GA
FJTQOV91Y4FHJAXOSR5207JFAIVRTOY716JFBGVQUN259ZKFAIPWMU2067J
AHENTXR2Y47JAFHOPXTY752CGLERNWS961ZEGKBWPUM0186AJHDPSNVY274
IDBKTNVP1Y48JHCDQTVO0176BDHJVNUQ248ZBJDINTWPY742AIKFWPSN106
8JHCDSPMW83Y6DGKBMVQTY276HFALTMWP347YJDBHTMPW72Y4DBLHPXSN15
8YEIJAWPNT247YCKGDQXUM4Z27IBJENUWQ72Z5KBDGVMQT169YBJDHOWSP9
1Z6EKHBSMWQ0716EHJAOVQUZ472JGCDUQOW72Y5JDGBPTVN91Z6FGJANVQU
Z592GDBJOTXPY284HCKDMWPS1Z96JDBGNXSR3Z48EKBHNPWU35Y7BGJEPMT
V924YFCHKVTPM2Z47IDBJSOQV2Z95BGJFTOVP86Y1KDCHPVNTY671FJBGQO
SV259ZBFGKMWPSZ671LGBDPTVMZ836BGKDTRMW724YEAGKXSNP1690BKDGM
TWR248YADJHPWOSZ582JDHANVPTY671HFAKWPSMY681FAHKOTRVY418AJID
OUVQZ247AFKGPOSW357ZAGJDUNPX247ZAKFGOTWP24Y7CDKGPUWN257YDGB
LMTVP29Z6EAHLQTWM0247AKFIPOSX41Z8GAEJPTVMY924KEBGPVTM4Z72AJ
DIPTWN824YAFJIQUWMY725BGDKMVUQ71Z4LBDGVPMS196ZGFAKTMPV0752B
GKDVUMQZ427BGJEMPSV159ZJDBHWOPS168ZCKDHMSWP38Z4BGJDSQNX916Y
BJEGWPNTZ274GFAJTQVOY572HDCJRSXNZ481GCEKQMTW5207GBDLNTPWY48
2AKGDWNPUY691BDGJPSXOZ472HADJORWT247YGKFATWPO581YGKECTXPM48
2YHJDCSXPM4720
```

Unit 118
Combination-lock cipher

The *combination-lock cipher* was introduced in the 2020 special coronavirus edition of the British National Cipher Challenge. It is a fractionating cipher modeled after a combination lock with a number of numbered wheels (think about bicycle locks, not safes). Letters in the plaintext are each expressed as three trits (base-3 digits). Since there are 27 characters possible with three trits, spaces are included as character zero (in ciphertexts they are written as ‘@’). The most significant trits (MST) (the first digit of a three-digit base-3 number, including leading zeroes) are shifted some number of characters to the right, with wrap-around to the beginning of the text. The least significant trits (LST) are shifted by a different number, and the trits in the middle are shifted by a third number. Hence, the key is a set of three numbers that represent the shifts. Each set of shifted trits works in analogy to one of the wheels on the combination lock. After the shifts, trits are converted back into characters.

As usual, let’s work a short example for clarification. Suppose our key is (2, 5, 7), and that our (very short) message is

```
COMBINATION LOCK
```

First, we decompose each character into trits. To help with that, here is a table of the plaintext symbols and their trits:

```
      _ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
     ---------------------------------------------------
MST:  0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
      0 0 0 1 1 1 2 2 2 0 0 0 1 1 1 2 2 2 0 0 0 1 1 1 2 2 2
LST:  0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
```

The decomposition of the message is:

```
C O M B I N A T I O N _ L O C K
-------------------------------
0 1 1 0 1 1 0 2 1 1 1 0 1 1 0 1
1 2 1 0 0 1 0 0 0 2 1 0 1 2 1 0
0 0 1 2 0 2 1 2 0 0 2 0 0 0 0 2
```

Now we execute the shifts. Trits that roll off the end are brought back to the beginning.

```
2→   0 1 0 1 1 0 1 1 0 2 1 1 1 0 1 1
5→   0 1 2 1 0 1 2 1 0 0 1 0 0 0 2 1
7→   0 2 0 0 0 0 2 0 0 1 2 0 2 1 2 0
```

Trits are converted back into letters to obtain the ciphertext:

```
0 1 0 1 1 0 1 1 0 2 1 1 1 0 1 1
0 1 2 1 0 1 2 1 0 0 1 0 0 0 2 1
0 2 0 0 0 0 2 0 0 1 2 0 2 1 2 0
-------------------------------
@ N F L I C Q L @ S N I K A Q L
```

An automated attack on this cipher that is better than brute force works as follows. We fix the first number in the key to 0. We vary the second key number from 0 to the length of the ciphertext. For each choice of second key number, we decipher the text and calculate the index of coincidence of the plaintext. For those key numbers that give an IoC in the top 10%, we also vary the third key number and again calculate the IoC of the resulting plaintext. The best overall IoC wins. Unfortunately, it is difficult to know where the beginning of the plaintext is located, so there is an overall shift of all three key numbers that will not be found by this algorithm and must be found by hand afterward.

A participant in the BNCC suggested an extension of the combination-lock cipher in which two or more characters are grouped together. During encipherment, the number of wheels is three times the number of characters in each group. With more wheels, it is possible also to represent the key as a keyword.

Let’s rework our example with two characters per group, and use the keyword `CIPHER`. The plaintext has an even number of characters, so we do not need to pad it with ‘X’ or ‘_’. First, break the groups into trits:

```
C M I A I N L C
O B N T O _ O K
---------------
0 1 1 0 1 1 1 0
1 1 0 0 0 1 1 1
0 1 0 1 0 2 0 0

1 0 1 2 1 0 1 1
2 0 1 0 2 0 2 0
0 2 2 2 0 0 0 2
```

The shifts are (since space is zero) ‘C’ = 3, ‘I’ = 9, ‘P’ = 16, ‘H’ = 8, ‘E’ = 5, ‘R’ = 18. Some of these are longer than the number of groups, so we must take them modulo that number.

```
 3→  1 1 0 0 1 1 0 1
 9→  1 1 1 0 0 0 1 1
16→  0 1 0 1 0 2 0 0
```

```
 8→  1 0 1 2 1 0 1 1
 5→  0 2 0 2 0 2 0 1
18→  0 2 0 2 2 2 0 0
     ---------------
     L M C A I K C L
     I H I Z K H I L
```

The ciphertext is

```
LMCAIKCLIHIZKHIL
```

To break the extended cipher, we can partition the ciphertext into sets that are enciphered with the same key using the original cipher. If the characters were grouped into twos, then every other character forms a partition that was enciphered with the original cipher and three key numbers. The remaining characters form the other partition; they were enciphered with another set of three key numbers. We can then break each partition separately, then inteleave the resulting plaintexts to get the complete plaintext. Since each partition may have a different overall shift, we need to adjust them relative to each other until the textual fitness is acceptable.

Programming tasks

1. Implement an encryptor for the original combination-lock cipher.
2. Implement a decryptor for the original combination-lock cipher.
3. Implement the attack described above for the original cipher.
4. Implement an encryptor for the extended cipher.
5. Implement a decryptor for the extended cipher.
6. Implement the attack for the extended cipher.

Exercises

1. Encipher this text with key (9, 22, 3).

   > It was a very ill time to be sick in, for if any one complained, it was immediately said he had the plague; and though I had indeed no symptom of that distemper, yet being very ill, both in my head and in my stomach, I was not without apprehension that I really was infected.
   >
   > (from *A Journal of the Plague Year* by Daniel Defoe)

2. Decipher this ciphertext with key (15, 7, 11).

```
JIDJGFH@@IL@TRTJYCSBRFE@KWZ@WEACHBFQJILO@DNAETAKTFIDCU
BTKZRG@OHVIRRX@@KTSBUFCYAIHDIKNB@RHGBUFRUDJFHKXDKBIFCA
IWTSCAF@W@CIJFKV@JLGUS@CIRGKRTTGERHETFVCADIK@BVACREWCR
BZSRMAACBFFNLBLL@GKMO@IHERBTHLXBBTGRAASCRBRBJYCZFR@AZN
UA@SBUJP@GU@I@ACXKP@PU@KTHIYC@RXD@WNN@UBAANHUGJRHDLJOG
HUJICGALQBUCTCKB@HUHX@@GZNUB@YB@QM@BBTCR@EORHWVUI@ZBHI
FCRBTBISCASRAU@K@BJYCZFR@GTQU@CRDCRJKG@GL@KBHIYC@UXG@U
JJF@SIRURBRDJWEKCZS@Z@HCILXDCWNKBRBCIEC@JVB@BLE@AXRCDM
IBBAU@B@EPVCR@VBFIQHCLKUL@AQ
```

3. Break this ciphertext that was encrypted with the original cipher.

```
SFI@DWDBHJRJO@CACNKRDIHBMWCQDAKUEBRSCJTDFNR@DCRC@KXUSK
@FJTHCTAAEIKQZT@YWSFJIEUW@ZBW@CLQAOKRFXSFRFCS@@HR@DD@L
B@HJFF@JLCERXJA@OT@TALAIH@HXHBEC@KB@KD@BSCD@OZCET@RBOT
BKO@VDIXDLWJJCNKHQLUIBVCWAQVNI@MNEJORLCTBEVCA@NUBBTC@C
S@FNRARVFNICTDIBH@SFA@ECIOKV@A@W@BFBQBILM@F@VT@ANJRB@C
@LSBXCBSHCRDCJ@RBXBRU@CH@LOIJQLIZ@U@WZAFAISKVCD@EXDFBL
R@BSCB@HWWROKA@HWQ@JIVACUL@HTBCDORNOE@EIYDTI@OLR@@LOCB
TECAIIC@N@AHUBA@CLUKRHECCIRHAUAEU@UEDC@RI
```

4. Encipher this text with the extended cipher and keyword GARDEN. Feel free to use ‘X’ to mark the end of a sentence, and ‘XXX’ to mark the end of the message. All other punctuation should be discarded.

> Every afternoon, as they were coming from school, the children used to go and play in the Giant’s garden. It was a large lovely garden, with soft green grass. Here and there over the grass stood beautiful flowers like stars, and there were twelve peach-trees that in the spring-time broke out into delicate blossoms of pink and pearl, and in the autumn bore rich fruit. The birds sat on the trees and sang so sweetly that the children used to stop their games in order to listen to them.
>
> (from “The Selfish Giant” by Oscar Wilde)

5. Decipher this ciphertext with the extended cipher and keyword VEGANS.

```
@RDLLB@@QLFRCUFWTFLKIOBFIXAZJVN@FJBLO@IK@FUJJGECEARGDO
CCHMSCSXEJHIZC@QO@CUFCQTSLI@RZCNLSRLECSRHGFAC@KFHILR@X
STHCURI@BHXSOBTBCU@JQEFIIVETIBKZLSCWIBCIFUNA@DTG@XIBGX
KU@D@R@JISPPRAOKDGAYOSFLAXBRABVDJGBGOE@@E@BJNPUKKCJLK@
@AGHRAUKTKH@DNL@BDXF@AVQSCL@WRTIERFEB@BLIASQWBRXNLORAN
H@MWRRH@UHKE@@FJCBHHJEACEINXELATYBLL@EEK@EHIEBKWDAYUAI
WADKCTJBHAFMP@GR
```

6. Break this ciphertext that was encrypted with the extended cipher. What is the keyword?

```
BCHD@ETTW@BNOATNU@@Q@KXKCOBTOWDTL@SCHKLXAFQVSFC@CLEJFA
ABCORZWHKAD@YFHCCDIRKTCW@IISS@UBFZTWCB@FUEICXA@FIRWTTL
```

```
W@FXAIEHD@QBNITCA@SURL@IFSUCBFZLACRKNIQC@USRME@VZVHLBZ
SUAD@TB@FU@DOSDDSBZR@TYC@@CO@EWKTC@AGEKUUBRSBIC@EXA@EB
I@XE@EBWTCKAMDDUYEKUR@@BTFDEEXARGCBWBU@@KZ@UULAQKTXIBB
EORUDCZ@ZO@BYMENRFJHCNCAXAACMBCX@IKSIRC@BSINR@HUXBIELJ
ODNSUTL@FRCRITRIDRLH@K@BM@@P@DKD@@XCDRGB@USJP@ASLBDE@@
IFICDSLGCJIT@USAJHEWRBTBBD@MDKEBBFCBO@AZCIOBFAJFCIIIUT
@JISI@@RH@UIJLXRZK@VEGFHCKHIARDECURZAS@UCWAFCE@NZTTSVD
CISEU@WBM@BXF@NT@@@D@SACQBSQLEBRI@QKLFHQTBDCS@RJD@KXAL
ITAK@FGHPYKOCRCICAPEAL@DG@E@WFIRSK@RHCEFKHLRLI@RAWAAIN
GBLNDX@@KAXCCMLDL@HGRK@GFIWDTRUYT@FDEEBGJ@DPJWIVIIACIC
UTEIRCAIDTL@AE@TICCATRKHCYYGDNCWT@YFR@ANUCTTJ@I@VUCCC@
ARJCCCIY@SIC@KRT@B@KR@L@AGJ@CNWBF@KEIZKEEIJZNABCYC@JMD
XRFATBKLC@@TDLTUBMTVCM@@ICUWMARIN@SUCEESRXJNLAVJ@YIE@B
YBITTALNEGRGUAJ@CTCACCVBFSTIRRS@XRAW@BHFBFEKCBBZTBUKOD
EEIHENCFC@XLBETICEHJSLWSRTKCFTVYEHWIKFAW@TFAOCSFCIQ@BC
BJCHK@TVKLSRDDAWBNZYT@QEERBDTTBC@QKLE@AXHC
```

7. I seem to have made a mistake, and the following ciphertext was encrypted with a broken cipher that only has only one wheel. Can you break it?

```
GSRFTUDURFTRORFTSXAR@OARDQOOSXCAXICDRFTSRFLZI@MSIGDSJX
@FTSRGROSX@@MSICDRFTSREM@MXXIGTUABRBOEGRH@@YFRFTSRAEGD
SCD@EB@FJBWD@OFIBROIFILAIFTRORFTSXAR@OARDW@DQOOSXCAXI@
MSIFTSRAEGDSCD@EB@FJBWD@OFICVWGXI@MD@FTUDTOIH@@MSIGSRD
ARODSSIDPABRBFEDIEGF@FSRJBSXFIH@GSRDARODSSIFTRORFTSRAR
OFTRCFIBD@OR@MSIFTRORFTSRFPDRFAYNDGJXI@OEGDSICFRGTUABR
A@PFJXIFTSRAIQI@MSIFTSRDUBKXRH@GSRDARODSSIFTSRDRMJXIEB
@@MD@FTSRGUDSOIGTUABR@VEGREGJX@FTSRFJROI@MSIEPFKRFTSRF
ILDFIEB@EGF@BOARORFKUEOIH@GSRDARODSSIBWGRFW@@VASSIDJVR
FW@AGFARFTSVIEB@@MD@@LDDJVXXIHHH
```

Challenge 1

```
CY__ALHYBXLDBJQGTQWASQUWBCYADMTQFAD_Y_XGAFDXWPTELAROTCIN_MI
WFI,OBKIBH_AMCHAQDHE_JDFBLKDQFEUKUJPDGADODVDQXBIQ_LIPEYFFRR
K_XSA,AQB'I_TNW_PIPNG'NTB_UQQB'X_D!G__._,GW_UO_TYU_!QFDHQ_!
WDJPBFQGM_DEC,AEREQCPDSIAVBMLQTQFWPMC_DSX__DYTALBD_FGP'AQRI
ZBPDEMPYEBGIG,CLSCPDAMECAE_U_IED_DQ!GXN?_.GTDIJ_ACISDEBC_EG
PEHE_JADRNMPK'U_Q__.A_FTXKGFOI._QF?GH,AT_YED_DQNGXLIBYARIBD
EBC_QCDA_AFEFAPED,CQ!G_YEGBEL_TAFWHIYFOEHXTD_?!_'_DDITH_HQI
FMACDDGPAJDCEFT_HGQMHD_K__P,C,EAT_IFF'NDB_AQANUXJPVEMMPITCW
HRNWNYKPAF?GHLA_TIA_DNQFNUHHET_UDBXADNGGCDHQFT?XBJU_DAFGHMQ
PILAPOELXD
```

Challenge 2

```
KFHASWBKFRCOORNDFA@SIMDCC@EAPWAHCSCJR@@BFALE@SII@CQIGZ@MFUK
T@INAKNBU@SBN@EDFHRO@I@COORWRW@UBERE@FOXCSI@KVBHAREJ@SSX@PQ
ID@N@FBT@TOAESEB@PTQCRRI@A@Z@RBETEI@FUITKLE@GICTBI@QVBENFLI
C@BFBREOFTFJ@CRSB@UASABLRQ@NA@BKMAXXUNM@THJSIFI@C@A@SETY@CR
IIIUR@LKTTNE@MAEBTJRCDAKLING@LN@INC@UDARAQDAYZOUKIKG@ROBTOC
AYBI@SWEW@ALD@RHELFILOR@GBD@O@ZFMO@TIEALRTBSNI@REW@KLCHWR@U
HOLDCB@QEAU@BE@FIW@CSLOUT@RRCRTCBGGBANFBODZ@KMOWRCBGGTBMGBQ
ANBIKSIRUMOBCRTIMWJJLEARF
```

Challenge 3

The ciphertext alphabet has been mixed with a keyword, but the plaintext alphabet has not.

```
QRTANTXWUPNORQJR@QRAQSBIINEFTRTBCNGWCYKUFZTRCOOOBOCRCTD@TNT
PGTQCCHIERKIOVRXETCGUSCYYRBBSETFNTDXNTSURYNUQTZCJRRFOGAOYOD
JNRDONKOOTQMSRUSTEXOCWORTZECU@TAHOTDN@FCINU@DNNJTM@P@Q@GCTF
@IIDZXOGBGEONTOKT@OURFJTRW@INPRLNTOUGNEEMNUVUGSRGZNTNERKGQB
CZNREQTDIAGTNFHTUSTG@EDKGRFRINRMZONNJRTFTX@RGQCZNT@KNPGRJTP
KSNTIRCBEQRKRTREKLTVYNXTTONTOPUHHUQDOIUNRTZPTGSGORG@DT@NOVN
VSRQEOBRCRAIDXACITRJNNOTNNCCXJOLMDTERBNUKYOGNFRTFCXKQBQFBCO
C@PXU@WNRNTUXFONNRTQWWTVGPFSRD@XNTHQLTDSC@PAU@INVNOPTNHC@R@
A@IRSA@VNTUTMCTEFBSEDFUCT@VQKBUEN@RSTKXGRSSPTSNEFGQRCSTOAQU
CNQ@QJSPTMYHTCBESUFNFCOSQPHREQINNNQURNODMWRSWRFGRQOSRSIQCVD
CF@TREWKOMKORKMVTRFFKEXBWH@SDSXT@XSQTNVCVNTFOT@OPRCESQRTNRC
GTGOXZDJGUPCN@OTCZZTELNOCYWR@XSQTNJTVNONNTCCYWOOJRCQGGFFEPB
TFURMQX@CRTDJNXCCNRHZ@SCKQSJO@VOVKRUNOFINPTZCTREP@DXOUUOQOT
NSRNFRVKSNTAZNQBGRCUXSTVGRCGRRONGPAQNTPWPTVNRHJECXKFNAJDNXC
TSCJNEUXOGNTQWFFJ
```

Challenge 4

Both plaintext and ciphertext alphabets are mixed with the same keyword.

```
@CLE@EEEEBBMBSYMCTLBVJKQULICZ@CT@EVOYEBHNTLEBUEH@EWJERRJUKZ
R@IE@DBFRDUN@ILALZMA@EGPEEAYUAJC@@M@@EDUAEAXKIBYPESJLMGB@LS
G@QM@MVL@CDAKSUBANNXRDRMPRYMLIGNEBXEJL@NILVQRBYOKARUKEUZAOL
TFB@BVJIYLAFHLRCTRXENK@UKMVBAMVRLMHNTE@RHFMP@BVFKHAFWJEIJBL
NGXLOU@IA@MIGL@CERUENJYTJDHNM@XUFCYUVNGNDMALHG@FTDETBRJ@MAT
LGRHLAGLRQANRWUZMLL@MJBIEULN@MATLELKX@EUUJEABBTUAS@THEAJLTN
XHZULBAMGRMDXVZ@MYTEGRLOLMHLENOGWRLMBUG@MUXGX@M@OLBKIUBHTLE
IJJZT@MSLRUW@ELMGRLZJHFMP
```

Challenge 5

Plaintext and ciphertext alphabets are mixed with different keywords.

```
PFVHOIBDHTDMSKC@EKZOGLUYKTX@TLYVAKJBRYBY@YESMOAKVEYXYWTAAZL
DCUBMAACWKBMYYBEASVAZYAEKYFZBTDJHLNUDXKEFKAYYEMBCYABLBYQE@Y
```

AMCAAUIMOFGDAJWCTRDPBAPDAPKVDEM@XLBFMAETDBMFXJEAYAECLIYZLKY
HH@KEBYSDBNDO@BX@HRFUMBXPSNBRGEABGPMUFFIA@CAMBEY@HEPAKBEQIL
MCDBCXX@@ODEUOJEESQAXI@CHESEAAZFAVMZZELIARHF@SLKCDBYIAESLKB
LZJERFBIAQJFBUOO@EYZBEMJCIASGAKGCU@ACNAAYICAPBRYUPNAHFIBEBZ
CVMOZBE@QL@EMYELEFJDYKLESYTDYLXVBCOBROENFDUKXSVIGLPAS@RFV@K
XXBFADHFYQJFEAMJHUSFIUQCDVVDXLKSDAKMRYSIUDTIJYUIBKHWDMRFYYA
QKEQBEHLEKDBKHGNKLWEYG@@RECMOMKRHBBCKTARMMFO@IAABFQVIEFBLFZ
@@KMEEJECLYAGYAFPASYKAZODVMKDURFGX

madness's book on classical cryptography
unit 119: chase cipher
last modified 2020-12-04

Unit 119
Chase cipher

The *Chase cipher* is an invention of Pliny Chase from the 1800s. It begins with a 3×10 grid in which we place the alphabet, mixed perhaps with a keyword. The remaining four spaces are filled with other symbols; we will use the digits `2`, `3`, `4`, and `5` (`0` and `1` look too much like `O` and `I`). The coordinates of the items in the grid are written above and to the left in this example:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	K	E	Y	W	O	R	D	A	B	C
1	F	G	H	I	J	L	M	N	P	Q
2	S	T	U	V	X	Z	2	3	4	5

We take our secret message:

```
MY SECRET WORD IS OBFUSCATE
```

We need to decide on a period. We can take the entire message at one go, or break it into words, or break it into blocks of the same length. For security, we should either use the entire message at once or use blocks whose lengths are all the same. For this example, we take blocks of length five, and pad the last block.

```
MYSEC RETWO RDISO BFUSC ATEXX
```

For each block, we fractionate by writing the coordinates of the letters in two rows, like this:

```
MYSEC
10200
62019
```

We take the bottom row and treat it like an integer; in our case, the integer is 62019. Then we perform some mathematical operation on that integer. Here there is some flexibility, but both parties to the message must agree on what operation to use. The only requirement is that it must be reversible, so that

the recipient can decipher the message. For example, we might multiply by 7. Our new set of coordinates is

```
010200
434133
```

Notice that we add a zero to the beginning of the upper row, so that both rows have the same length. We could add any of `0`, `1`, or `2`, and should choose randomly, for security. The new coordinates are converted back into characters, according to the table.

```
010200
434133
OIOTWW
```

One possibility for the full ciphertext is

```
OIOTWW VRA5WB WCOGUB ZDGX2W XC4KS4
```

The Chase cipher is very flexible in how it is set up. The length of the blocks can be changed, as can the mathematical operation used in the enciphering. With another choice of operation, such as taking the first few digits of the logarithm after the decimal point, the blocks in the ciphertext can be the same length as the blocks in the plaintext, rather than requiring the addition of an extra digit, as in our example.

Python tips

In the `random` module is the function `randrange()`. It returns a randomly chosen integer in the range from 0 to one less than its argument.

Reading and references

Pliny Earle Chase, “Mathematical Holocryptic Cyphers,” *The Mathematical Monthly* 1:6 (1859) 194-196, books.google.com/books?id=SVNLAAAAMAAJ&pg=PA194

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 203-204.

Programming tasks

1. Implement an encryptor and a decryptor for the Chase cipher. Separate the mathematical operations as their own functions, so that you can choose which operation with each use of the program. Implement multiplication by a constant, as in the example, and any other operations you would like.

2. Implement a dictionary attack.

Exercises

1. Apply a dictionary attack to this ciphertext. Can you find another way to attack it?

```
U5EFA3 KEBPU4 AVMIJM KIGBJU UOHKI3 2UUIFE 2AHIUP WTORBU
UUJAK4 V5EFA3 2AJTMA DXXLBY MOIFMC BUTOJM DBTPUP V2BGKI
IHQTOR NKODQI DSSGBG ZCALQI UIFMGS NCHTAO GGINIS VOHPIO
ZLIPAR NKODQI ASWFAS JUTAHY TBGNTH DQTBAG GUENJB AEB2NI
YJSTBU XTRIQI V2FUBY KEFUWI AGBPLY JOIG2F DUBINI NRATVM
GUGTIF NAJTOH MRAVMA NCBMTR ZOTWHE NAHMSA IQIAYG GTIKIG
VK2PAG GSTBBG SYTNTH VIFUIE WLRUFU WSTPTS ALZIGE 2TRUU4
MIPJUU SENKIG VKINIS GBLAIF DMNTSL SZIJIG DBMAIF 2AIGDL
DWAPTA ZMIVMA JAOTWF OMAHMS WUQINI NBTUHM 2NWREM VITRLM
GPTHMG YINDOT MJDBOE BIWURF GT2COO GHIJFU 2JBUNT VI5INI
2RATSA GNIADA MGJJTY U4OIFS WOEMJ4 IBTUHM NPLIWU SQINLY
GLPCJT YHHTKE NAHMMG AF4OIF SWOCOO WGUQQI AVCLWI IK52NI
IRATSA JPE3EI BMBAHM VEBOLM GUSEKJ ZIVMWU OOLIKT YHNUQM
WNLIKT YHJVIF DUIET4 MAIGBG NAIHIF IUOONF FIUQ2F AUIFAS
MUTAGD GBPL2F IAJBPI DZJIOK AWTFIG 2CQIAS YIHRFU TSTKEA
NBKIPI NK4MGD MIGDLS WAK5IF ITRTBU XOIVUR SBHEMM VOQEUP
AIBBCD OOLMMC BOEOWU 2JIUQY TLPCJT JH2KEO THDQIA VMPIMM
GLPLIS F4LOCD 2POWIA FYT5IF BEARUI DVINIP ZUOSTH DVINCB
YEHDSA MGFINI GDRTAG VFEACD ASMGFM WFUBIF NAH5IS GFBMUF
DUWIUY GLPCNO WMOFUS ZIGURE KLMBUE AIBKIG ZISMGD TFCEMG
TEIBKT YPIV4T UIFBOC WUQIFA VETGBG WTMTDH ITAOIF MIKNIF
SUBUNC KISV5I ORJT2E GLQIOB TUEIEA 2BKIPI ASMGFM TPAEHL
AZLUNC ZISV5I MALBCI 2JUGKI 2HVMMG AFINEE NTBFMG MEOBOR
TLJEMU YJTGOT VGESTO AMNJTM TFUWLM WNLEME BLILAG ODO5IF
ZEARUI ASMGFM WNCCEH ADSMGD FINLFC WOSTHS SMGFIF ZEDRUI
YHFEAC SDWXXV
```

Challenge

```
JQQJYJ A4OJ5F NDUAPF OCQGEF HOKVXC REOAXH E3RM5P CFRQG3
JZ2PT3 RZIDPT QGNNUR JJDOXH RSDLS3 WJ5FTU D23QG3 PQZS3U
TWIDPT DGKYY3 JCJDOJ AP3URT QGKC5V CNSCHT NJY2IW C3Y3Y3
GURAAC RCEA5V 5NSKVQ YFZJTU H3XCGT DGNS3U TWJINP T5IO5V
Y3GRMA OMU5NJ AAHA5V CEFQG3 RAHSYJ ZS25ZF NCXCHM OYFKNF
GYUXCG TSOQG3 JYQZJT CAFUDP KX23RF RSDPYM HC5EAT YFY3UX
RSFHFT DHXCKP OMRQG3 GR54AC QT2XCG LD55QG BHEA5F QYH2LJ
Y3ULAC 5AC3TT NXCKOJ YEA5FT QG2WD3 AROAXH D3RFXU R3DXVD
JILJ5V EOUUSF O2ZTMU GXSOMA YAY2LJ YEAQG3 TNQA5J QYCQG3
YZMLPF WMHQCV ZX23RF 35JWAA YKCWJU BHAQG3 RNQA5J QXKCWP
5FRQEP OMSTDP BHHWY3 KSMLMT JJY2JT 5VUTXU KJWFFR TUTI3U
WJLRX3 CHKGRU LNCTRF D2EYVM TPJDOJ GSZRTF BMQG3R ZIGHXF
OHD3Y3 YUSP5V 53YHX3 GY3QRS C2OAEO OHHWNQ TA5J5V RULP3U
```

RRQGKY GYNSKC RWPYNJ EHZKHA NQYCHT ZPDODT QG3GRU NMDUZJ
L23RFX NLJU5F HCQGNP TDYCWJ Y3UWJX EAY3XC TGUTPQ ZPICRW
YEAUTT QG2WD3 LRDMT3 PXCGFH TMD2MQ LVUDFQ 35XAHA WFECQQ
BCKV3R 3MJTRU OCCCAC CPDENF JFFFRT GMSIKM BHD3RT QGNXHR
BMJXEO QOAIOD LAEFYU NQZIDP YCPNSV 53RGXU HENHWU HHD3Y3
YAQAWU GYAQG3 BHKTSF HZ3OZF GYCHXU NLXYJT D3YMCV NX2S3U
GA23DP WIOZEF HHHG3R CCPNSS RPQCVT YKVXCG LDLIG3 LKAMCV
GJ5QCV RQDP5V CESDAW OHCPEA RNQATW E3RVMA QSYC5V 5NPYNJ
QHOANY RPJU3T EGKYY3 NCJDOJ BXDH3T DG3XUT R53RIF QA2EAU
ZIGCLY RI2LJU OHPPPQ CKHAI3 PLXYIW EGKCGR RFN2YQ TZJWJU
DSG3ZJ T5KUDY EYCQG3 OYXUWP CFPD5F GYCN2Q ZVHG3R JPYHCO

Part X
Proto-mechanical ciphers

Unit 120
Disk ciphers (cipher clocks)

Consider a device that looks like a pocket watch. Around the outer rim are symbols like letters and digits. Inside that is another ring of symbols. The outer ring has *m* symbols equally spaced around it, while the inner ring has *n* symbols, also equally spaced. The device has two hands like a clock. When the longer hand moves from one symbol on the outer ring to the next, the shorter hand moves from one symbol on the inner ring to the next. In other words, the hands are geared in the ratio *m*:*n*. The device can be used to encipher a text by rotating one of the hands clockwise until it points to the first plaintext symbol; the ciphertext symbol is the one to which the other hand points. The first hand is advanced to the next symbol in the plaintext; the second ciphertext symbol is indicated by the other hand. This continues until the full text is enciphered.

The *Wadsworth cipher disk* uses the outer ring for ciphertext symbols. It holds 33 symbols, the letters A-Z and digits 2-8. They can be removed and replaced in any order. The inner ring is for plaintext symbols, of which there are 26 (just the letters). Rather than having hands, in this device the two rings that rotate in the same direction under a single pointer. To encipher a text, the inner disk is rotated until the plaintext letter is under the pointer. The gearing rotates the outer ring to bring the ciphertext letter to the pointer. This operation is equivalent to the “cipher clock” described in the previous paragraph. The disks are always rotated in the same direction, presumably counterclockwise.

There is no evidence that there was ever more than one Wadsworth disk in existence. We do not know if it was used more than once. It was invented by Decius Wadsworth in America in 1817, more than fifty years before Charles Wheatstone reinvented it in England (too slow, Britain!).

The *Wheatstone Cryptograph* is better documented, especially by Wheatstone himself. Its outer ring holds the plaintext symbols, which are fixed in place. There are 27 of them, the space and 26 letters. The inner ring has 26 ciphertext symbols, which are the 26 letters in any order. The initial position before the first letter is enciphered is for the longer hand to point to the space and the shorter hand to point to the first letter of the mixed ciphertext alphabet. The hands are always rotated clockwise. To encipher a text, the hands are rotated clockwise until the longer hand points to the plaintext letter. The shorter hand points to the ciphertext symbol.

Wadsworth cipher disk. Photo from NSA.

Wheatstone Cryptograph. Photo from eBay.

In his paper describing his device (citation below), Wheatstone introduces a novel way of mixing the ciphertext alphabet from a keyword. It goes like this:

1. Write the keyword in a row.
2. Take the 26-letter alphabet and delete any letters that appear in the keyword.
3. Add the remaining letters in the alphabet under the keyword in rows with the keyword letters at the top of each column.
4. Duplicate letters in the keyword are deleted.
5. The mixed alphabet is read off in columns.

Here is an example using the keyword `WHEATSTONE`:

```
W H E A T S T O N E
B C D F G I J K L M
P Q R U V X Y Z
```

The mixed alphabet is

```
WBPHCQEDRAFUTGVSIXJYOKZNLM
```

The procedure for encipherment on the Wheatstone disk is

1. Remove all punctuation from the plaintext. Keep only letters and spaces.
2. Either:
   a. Put an ‘X’ between any double letters in the plaintext. If the double letters are already ‘XX,’ then put a ‘Q’ between them.
   b. Replace the second letter of doubles with ‘X’ or ‘Q.’ Be sure not to accidentally get a doubled ‘X’ or ‘Q.’
3. If the plaintext does not end with space, then add one. The last ciphertext letter serves as a sort of checksum.
4. Position the longer hand of the device so that it points to the space on the outer ring of symbols and the shorter hand so that it points to the first letter of the mixed ciphertext alphabet.
5. For each plaintext symbol (letter or space):
   a. Rotate the longer hand clockwise until it points to the plaintext symbol. Do not go all the way around. Let the shorter hand move on its own, according to the gearing.
   b. Read the ciphertext symbol, which is indicated by the shorter hand.

Step 2 is necessary because there is no combination of ciphertext letters that the Wheatstone disk can decipher to a repeated letter, since the ciphertext alphabet is shorter than the plaintext alphabet. This is not true for the Wadsworth disk; for double letters on that disk, we would simply rotate 26 more steps for the second letter.

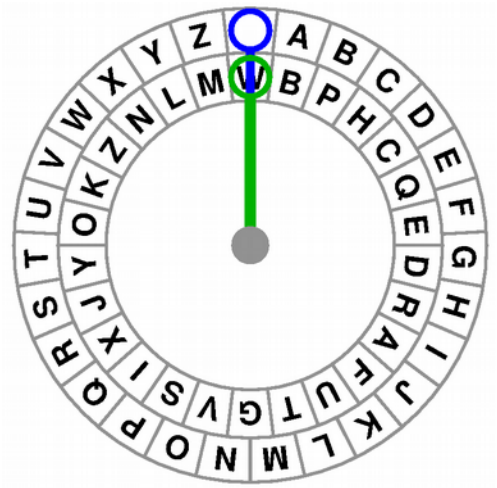
Let’s look at a short example and encrypt this message with the Wheatstone disk. We will use the mixed alphabet from the example above.

```
SECRET MESSAGE
```

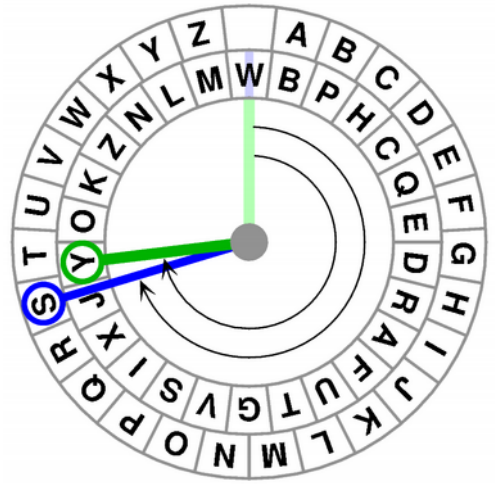
First, we have to prepare the plaintext:

```
SECRET_MESXSAGE_
```

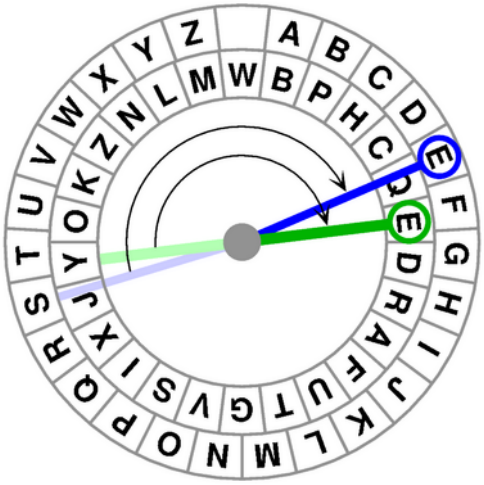
We begin the encipherment by putting the hands of the device in their initial positions.



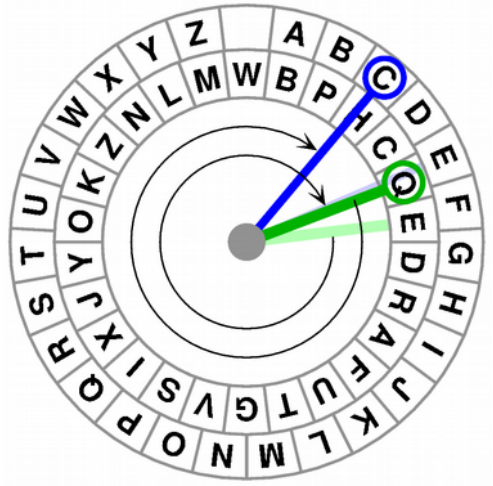
The first letter of the plaintext is ‘S,’ so we turn the large hand clockwise 19 steps until it points to ‘S.’ At the same time, because of the gearing of the device, the short hand also moves 19 steps and lands on ‘Y,’ which is the first letter of the ciphertext.



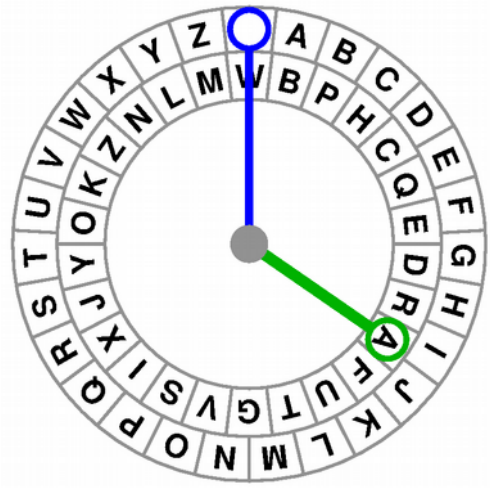
The second letter of the plaintext is ‘E,’ so we turn the large hand 13 steps clockwise to ‘E,’ while the short hand also moves 13 steps and lands on ‘E.’



Since we always rotate clockwise, to encipher the next letter, ‘C,’ we have to turn the large hand most of the way around the circle. The short hand indicates that the next ciphertext letter is ‘Q.’



The process continues, always clockwise, until the entire text is enciphered. By the time we get to the space at the end of the plaintext, the short hand has moved so far that it points to ‘A’ in the inner ring.



The full ciphertext is

YEQORNCXFLHMRVGA

The *Pletts Cipher Machine* was another cipherclock based on the Cryptograph. Its innovation was to mix both the plaintext and ciphertext alphabets, to make breaking its cipher more difficult.

The astute reader may have noticed that these three devices give us stream ciphers. Let's consider the Wheatstone disk. The internal state can be thought of as the positions of the two hands, (h_{long}, h_{short}). The initial state is (0, 0). The cipher is factored into a stream cipher that corresponds to the disk using the key ABCDEFGHIJKLMNOPQRSTUVWXYZ, followed by a monoalphabetic substitution cipher M that uses the mixed alphabet as its key. For each character p_i from the plaintext, which we think of as a series of integers (space = 0, 'A' = 1, 'B' = 2, ...), the action of the encryptor is

$$\begin{aligned} x &= p_i - h_{\text{long}} \mod 27 \\ h_{\text{short}} &= h_{\text{short}} + x \mod 26 \\ h_{\text{long}} &= p_i \\ c_i &= M(h_{\text{short}}) \end{aligned}$$

For the Wadsworth disk, exchange the rolls of h_{long} and h_{short}, and replace 26 with 33 and 27 with 26. For the Wadsworth disk, we also need to specify that if x is ever zero, it should be changed to 26; this allows for the encipherment of repeated letters, which is never done with Wheatstone.

Let's now redo our example encipherment with the Wheatstone disk as a stream cipher. Start with an internal state of (h_{long}=0, h_{short}=0). The first plaintext letter is 'S'=19, so x is 19 − 0 = 19. The internal state changes to (19, 19), and the output of the stream cipher is 19. Applying the monoalphabetic substitution gives us the first ciphertext letter, which is 'Y.'

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

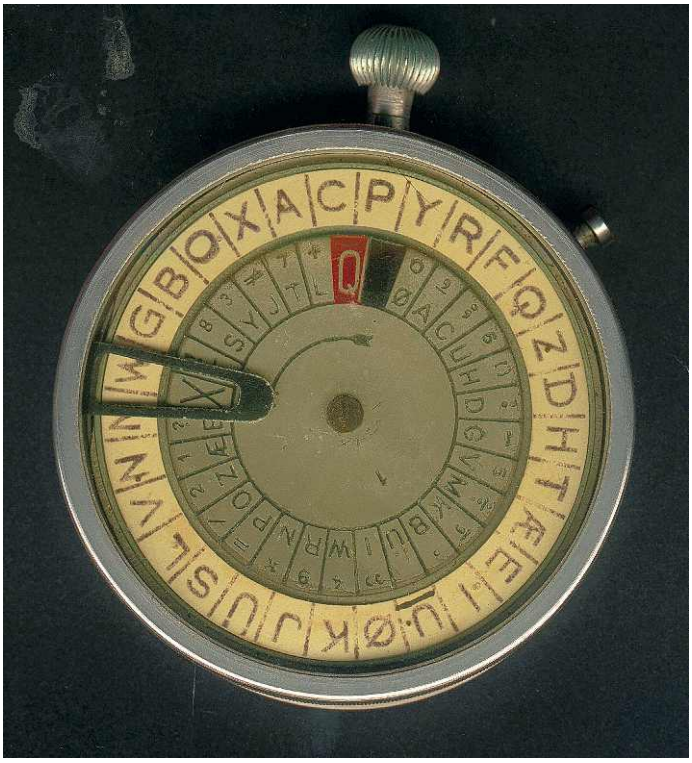
Next is 'E'=5, so x = 5 − 19 = 13 modulo 27. The internal state becomes (5, 19 + 13 mod 26) = (5, 6). The output is M(6) = 'E.' Then comes 'C'=3. The internal state becomes (3, 5). The next ciphertext letter is M(5) = 'Q.' Etc.

A more interesting way to define the internal state S is as the total number of steps taken. From this point of view, the encryptor does these things for each character that it enciphers:

$$\begin{aligned} x &= p_i - S \mod 27 \\ S &= S + x \\ c_i &= M(S \bmod 26) \end{aligned}$$

During World War II, the Danes used a derivative of the Wadsworth and Wheatstone devices, called *Urkryptographen* (Danish for “clock cryptograph”). Like the Wadsworth disk, it had two concentric rings of characters that rotated counterclockwise under a single fixed pointer, and the key was written on the outer ring. Like the Wheatstone Cryptograph, the plaintext alphabet contained a space which the ciphertext did not; therefore, double letters had to be disguised with ‘X.’ The letter ‘Q’ was used to indicate numbers, which appeared with some letters on the plaintext disk. The two letters ‘X’ and ‘Q’ are not used in native Danish words. Unlike either of the other two devices, Urkryptografen had three additional letters in each alphabet: ‘*Æ*,’ ‘*Ø*,’ and ‘Ü,’ so that the rings had 29 and 30 characters. The plaintext alphabet was also mixed, and there was a number of preprinted disks. One such plaintext alphabet was (where ‘_’ denotes the space)

```
_ØACUHDGVMKBÜIWRNPOZÆEXFSYJTLQ
```



Urkryptografen. Photo from Museum of Cypher Equipment in Fife, Scotland.

Reading and references

Charles Wheatstone, “Instructions for the Employment of Wheatstone’s Cryptograph,” *The Scientific Papers of Sir Charles Wheatstone*, The Physical Society of London, 1879, pages 342-347. archive.org/details/scientificpaper00londgoog (the last two pages of the article were completely ruined by Google in that copy), books.google.to/books?id=CtGEAAAAIAAJ

William F. Friedman, Several Machine Ciphers and Methods for their Solution, Riverbank Laboratories Department of Ciphers Publication No. 20, 1918, www.campx.ca/Several_Machine_Ciphers.pdf and www.marshallfoundation.org/library/methods-solution-ciphers

James Stanley, “The Wheatstone Cryptograph,” incoherency.co.uk/blog/stories/wheatstone-cryptograph.html

William F. Friedman, Six Lectures on Cryptology, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/ACC15281/41785109082412.pdf

“Ciphers and Cipher-Writing,” Macmillan’s Magazine, XXIII, Feb 1871, pages 328-338, babel.hathitrust.org/cgi/pt?id=mdp.39015004979913;view=1up;seq=340

NSA file 41788379082740: www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/patent-equipment/FOLDER_515/41788379082740.pdf

Basic Cryptography, Dept. of the Army Technical Manual 32-220, April 1950, section 142 in chapter 11, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/FOLDER_238/41748889078809.pdf

Niels Faurholt, “Urkryptografen (The Clock Cryptograph),” *Cryptologia* 27:3 (2003) 206-208, DOI: 10.1080/0161-110391891874; this article is available also at www.jproc.ca/crypto/crypto_watch.html

Greg Mellen, “Cryptanalyst’s Corner,” *Cryptologia* 8:1 (1984) 55-57, DOI: 10.1080.0161-118491858773

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 195-198.

Auguste Kerckhoffs, “La cryptographie militaire,” *Journal des sciences militaires* IX (1883) 5-39 and 161-191, www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf, www.petitcolas.net/kerckhoffs/crypto_militaire_2.pdf, section IV.

Donald W. Davies, “Charles Wheatstone’s Cryptograph and Pletts’ Cipher Machine,” *Cryptologia* 9:2 (1985) 155-160, DOI: 10.1080/0161-118591859870

Programming Tasks

1. Write a function that mixes a ciphertext alphabet using the prescription of Wheatstone.
2. Implement an encryptor for the Wheatstone disk.
3. Implement a decryptor for the Wheatstone disk. Think about how to handle double letters in the ciphertext.
4. Implement a dictionary attack on the Wheatstone cipher.

5. Implement an encryptor for the Wadsworth disk.

6. Implement a decryptor for the Wadsworth disk. You do not have to worry about double letters in the ciphertext, since they cannot appear with this cipher. To handle double letters in the plaintext, advance the device 26 steps between them; the shorter hand will return to the same place, but the longer hand will point to a different ciphertext character.

7. Learn Danish and implement a simulation of Urkryptografen. Brug det for at skrive i sifer din favorite aventuren av Hans Christian Andersen. Skift din bakkeopstigningangreb på Wheatstone-kryptografen (Unit 123) at tilægge de tre nye bogstaverne, og brug angrebben din at løse siferteksten din.

Exercises

1. On average, how many characters are enciphered before the Wheatstone disk passes through its initial configuration? Use your monogram frequency table in your calculation.

2. Encipher this text with the Wheatstone cipher using keyword `CONVOLUTION`.

   > Viewing the Wheatstone Cryptograph as two rotating pointers is only one way of seeing this cipher. Another way is through the lens of modular arithmetic. The modulus for the plaintext is twenty-seven, while the modulus for the ciphertext is twenty-six.

3. This is the example ciphertext from Wheatstone's article. It was enciphered with the keyword `FRANCE`. Each paragraph is enciphered separately; i.e., the disk is reset to the starting position at the beginning of each paragraph. Decipher the message.

   ```
   PZLSPQREQAJDITFBUFZOHQOSUQUDIKITORTWEZACMTPLERAUESKGSO
   FGFDKHLSJIRKHFHMFADAYIVUOHAOBLNOGREJAIBKMPJZTMJABQCNFP
   OMYHYRCZDCWBXUBZ

   ZBILIJTEJYSPFDLCXETKQASOXOUNNODQJCWECLXPUYIEMMCMSYVCFP
   OKWCDEDVDAGLPEEKNAGVKMNUULSHXYXYVGFQPUYIORQKLPTCZHHK

   ZBKUPVSWZWXAQXDREKTKQASOXOUIRSKOMFSTIIXGWTQJJVDYFNAHLS
   IIXIAGQLZXVOGNHGRBUOHYZOOPWVYDDMQJKFMOBJPDYVRBAWKGWSJU
   RJGITOWTVEZBHSOSLVUNBCHQSOTEIEBDQMGWHGJAMISXFIFBBPAVPE
   SVCJUTAD

   PZLPTYVXQXDTGLTTAFCVMHOMBINJKWVYAZOCQLAIUKFEGFNCNFIZHH
   KVZYQUGLIVENKAHTRVFVEBWHWLRYCMLXWSYSYJHUFSFPOKEGZRXLUB
   FT
   ```

4. Break this ciphertext with a dictionary attack. The keyword is a ten-letter English word. The cipher is Wheatstones's.

```
ZUKMNXQCDJKEADWCJVQHKCWNGGFVMAGQFEOCOWOIXTVIAKJPNIJJXU
UQKCLBXVABOFIAATPQHIVKJQYUWZQRZKXWXYZVILALHDSSIGGMQQVK
ZVYCVZUPMHRTCTJXCFXBIJGLNUVWEBTJZTOQLIKPRYEDCXIGDMETEY
YPOWFEBARSZZCOHWPABUIZCLWJJYQQKGPTTLWAIZUKXNGWDVHVDXUX
EXEBJWJPPSXPNEPUHRHTILXRKSGXKWMCZFZMBTVMNUDCDVLAFCQRRM
GGFOUWAHVXHNHABCUPXKZOABAWMMZGFVNOQCKYIOVGXDBUARIYSTJG
YVGWLPLKEWXHJSZVIMSYFPL
```

5. Decipher this text with the Wadsworth disk cipher in which the ciphertext alphabet is not mixed.

```
TABLV3BTJWYCVKNRCOXFLSD2GTL6JUBUJNQ7CUXZG4AK4NT2LBO7DM
XYLQDQVCK8OW3AL6T2GWOU3MCPR4ODKVZK56APHABIZ36JUBUN2J8Q
5M7XM3B2MUC2K8YH4TAK6BMXM6DXGL24EF7XHLV7KBE2M3FMZMOERU
3CZG5OSVDHZ46LAFPAYBVG4IOH4IL3S6WYADMXQ3CILMUKYOYQHULP
SAEWZ3I6CM6RBKXZN5P2DEHL56LBMOEJ6CHYGKPVCZQBZ2O5RARJQG
U78K6OBZDK6CKT4BMEUBYP38ZNFY27LYH26P3GKY7YG7OCPBPSDSFT
6XYFHMNSCQR7V8BY5RW3L2LPAOQ5FOUIU8H7K2LAFU8YRBZJYL5V5T
HVL5HITGXKRFRGJW
```

madness's book on classical cryptography
unit 121: attacking cipher clocks with cribs
last modified 2020-10-28

Unit 121
Attacking cipher clocks with cribs

If we know some part of the plaintext (a “crib”), we can use it to break the ciphertext. To make it easier, we will encrypt the crib with an unkeyed cipher clock before we compare it to the ciphertext, since the difference between a ciphertext and a text that was encrypted with an unkeyed device is a monoalphabetic substitution.

An example will make the technique more clear. This ciphertext was encrypted with a Wheatstone Cryptograph:

```
DGBRHDRYDURNFTWLOMIBLSPDZCIJNSMBZXCSHDYCVMGOAZSAEGNVQLJRFTX
CWJHACZZMPEHMZNPZJYKMRXQGOUIUCRBOYQLOTOBUNZGJTEQBOFYLEHBOBZ
RRGWOHWJKCUKPZCIQELLOUEJVAYVYPQWHEOBJNVGMAXRGLYYHNAHYCYFTGR
ILIJNSMHQFUWYYUEJVNFIXNSDMXZSAHCNFVDVXVZKMCJWLXTBKJJGHHDRSK
CIMACCWTUKZVSRDGZSKCNUQQGCDKWKHVHIIFIXUMYIPWYSOPCGGSGHZBFWW
XPCEWEQMISKVXEMIZXSHAZEWOMRGQUTKCTKMPEUQJYLHTIDHFPERYUKSKON
NCEPLARCSBLMKOBCBSEFGXN
```

We have this short bit of its plaintext:

```
never wear out no matter how long it may stand in times
```

Before we use the crib we will encipher it with an unkeyed Cryptograph. The ciphertext alphabet for this process is just the unmixed alphabet. Before encipherment we need to disguise the double `tt` by replacing the second with `q`.

```
never wear out no matqer how long it may stand in times
```

Enciphered this is

```
OGXHUDAJGXGVBBIWXJWLECRENVCKOADDXRALSFUSUNOWJAXGLYSIMFT
```

Notice that we did not add a space to the end before encipherment, as Wheatstone had recommended, since we do not know if the next character after the crib is a space.

Now we compare the enciphered crib to the ciphertext, one position at a time. Start at the beginning:

```
DGBRHDRYDURNFTWLOMIBLSPDZCIJNSMBZXCSHDYCVMGOAZSAEGNVQLJRFTX...
OGXHUDAJGXGVBBIWXJWLECRENVCKOADDXRALSFUSUNOWJAXGLYSIMFT
```

Notice that B in the crib matches up with F *and* T in the ciphertext. Also, A in the crib matches up with R, S, C, and Z. These things cannot happen with a monoalphabetic substitution, so we know that this is not the correct position.

The only acceptable line-up we can find is at position 107:

```
...TEQBOFYLEHBOBZRRGWOHWJKCUKPZCIQELLOUEJVAYVYPQWHEOBJNVGMAXRG...
   OGXHUDAJGXGVBBIWXJWLECRENVCKOADDXRALSFUSUNOWJAXGLYSIMFT
```

Now we can begin to reconstruct the key by tabulating the substitutions for the letters of the enciphered crib:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ERCLKABFGHIJMPQ??UVXYZWON?
```

The missing letters are D, S, and T. Missing three letters means that there are six possibilities. If the key is mixed in some less obvious way, then we would have to try all six. But in our example, we can guess immediately that the key is

```
WONDERCLKABFGHIJMPQSTUVXYZ
```

and the keyword is WONDERCLOCK. Usually, we also have to try all twenty-six ways of rotating the key before we find the correct key, since we do not know the position of the cipher clock's hands when the plaintext was encrypted at the point that the crib begins.

Deciphering with the reconstructed key gives us the plaintext, which you should recognize as a paragraph from *The Wonder Clock* by Howard and Katharine Pyle.

```
CLICK BUZQ WENT THE WHEQLS AND THEN TICK TOCK TICK TOCK FOR
THE WONDER CLOCK IS OF THAT KIND THAT IT WILQ NEVER WEAR
OUT NO MATQER HOW LONG IT MAY STAND IN TIMES GARQET DOWN I
SAT AND WATCHED IT FOR EVERY TIME IT STRUCK IT PLAYED A
PRETQY SONG AND WHEN THE SONG WAS ENDED CLICK CLICK OUT
STEPQED THE DROLQEST LITQLE PUPQET FIGURES AND WENT THROUGH
WITH A DANCE AND I SAW IT ALQ
```

Now let's work through a more complicated example. This ciphertext was also encrypted with a Cryptograph:

```
VDZYHDCDOFCLZSCPWTKEIVTTRNVGSYSCPUVEUJOWVFTZBLXDYXNZAU
NICZVYFHIKQFZLEMNLQIJHRCGOJSRODYLLCKDTWQUAOXTAVJOOYOGU
ZGBRZXZIRDXAGYBLHRGUINGCCNQHJSUWPLEUVDDAWVORXIRJBUOOYW
FQOTXDFRDYUJYEVXUAITGMLIIHKGAEFFVDAKXEHGSYXAVVAQCNECYK
ZOXWFXRRQIJYAZVCPZDUWSHHPAWLVFKDYNLRBTSCYREOBLPVVXZMCF
DUNQSXBXSRKAPMWCYHTBUERKAVCCARWYDNEHXSWXBPHRACQIWIOYBX
```

```
PFMNUKGEFFDNBXASYSXDHQFLDNWYTDGMCWLTPUXZFUIIALEKEILREZ
JHNPAEIBYHRNVGDLKUFECLXCGEWMEFVLCZBUONKUISJCSOFCLZTNES
DUWSXNLROKNNGWCHBOUIZZATRKNNPQHXJMONSMZDWKGTBZKPDQFTIJ
NVXJRKZP
```

This crib is known to be part of the plaintext:

```
information exists of course but it is scattered
```

After we replace the double `tt` with `tq` and encipher with an unkeyed Cryptograph, the crib is

JOHQTPEXNTTGLEQABBJYQLOAGEFSOQJJQZKRAKSLWVOMBOCC

The first position that does not give a contradiction in regarding the monoalphabetic substitution is 177. This is actually the only position that works.

```
...YEVXUAITGMLIIHKGAEFFVDAKXEHGSYXAVVAQCNECYKZOXWFXRRQI...
   JOHQTPEXNTTGLEQABBJYQLOAGEFSOQJJQZKRAKSLWVOMBOCC
```

We begin to reconstruct the key:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
EFR?GSHU?VCKWLXTANYI?OZMDQ
```

There are three missing letters (`B`, `J`, and `P`) in the key, so there are six possibilities. Plus, there are twenty-six ways to rotate the key. That makes 26 × 6 = 156 total possibilites. But we don't need to try them all. Let's just try one and see what happens. Start with

```
EFRBGSHUJVCKWLXTANYIPOZMDQ
```

There are twenty-six ways to rotate this key, and we try them until we find something useful. Deciphering the ciphertext with the key as written above gives gobbly-gook:

```
IXUPCUFTPVDGPYCMDGBQIYDCPDVPQCPUDQSIPQCTPGUAHRSBVQTYR...
```

Next we try `FRBGSHUJVCKWLXTANYIPOZMDQE` and get more nonsense:

```
HWTOBTESOUCFOXBLCFAPHXCBOCUOPBOTCPRHOPBSOFT GQRAUPSXQ...
```

We continue to rotate the key, and soon we arrive at `TANYIPOZMDQEFRBGSHUJVCKWLX`, which gives this plaintext:

```
THE NEQD FOR INXORMATION OF AN EOACT AND RELSBCMFADIB...
```

Aha! We are getting close. Most of the key is correct, but now we need to find the correct placement of `B`, `J`, and `P`. By swapping `B` and `P` we get the true plaintext, which is from *The Modern Clock*, by Ward L. Goodrich.

```
THE NEQD FOR INFORMATION OF AN EXACT AND RELIABLE CHARACTER
IN REGARD TO THE HARD WORKED AND MUCH ABUSED CLOCK HAS WE
PRESUME BEQN FELT BY EVERY ONE WHO ENTERED THE TRADE THIS
INFORMATION EXISTS OF COURSE BUT IT IS SCATQERED THROUGH
SUCH A WIDE RANGE OF PUBLICATIONS AND IS FOUND IN THEM IN
SUCH A FRAGMENTARY FORM THAT BY THE TIME A WORKMAN IS
SUFQICIENTLY ACQUAINTED WITH THE LITERATURE OF THE TRADE TO
KNOW WHERE TO LOQK FOR SUCH INFORMATION HE NO LONGER FEQLS
THE NECESQITY OF ACQUIRING IT
```

Notice that instead of trying 156 possibilities, with this method we only need to try at most 26 + 6 = 32 possibilities.

Now that we have the plaintext, we can stop. But, no, we can't stop. Never stop solving puzzles. Let's go onward and get the keyword. The key we found was

```
TANYIBOZMDQEFRPGSHUJVCKWLX
```

It looks random, but notice the sequence `Q`, `R`, `S`, (`T` is missing), `U`, `V`, `W`, `X`, with one or two letters between them. Also see `F`, `G`, `H`, `J`, `K`, `L`. So the key was likely constructed with Wheatstone's prescription. We proceed by breaking the key into columns and getting the sequences each to appear on one row.

```
T I M E P     C
A B D F G H J K L
N O Q R S U V W X
Y Z
```

If we search our dictionary for `TIMEP??C?`, we find the only choice is `TIMEPIECE`.

Programming tasks

1. Implement a boolean function that detects whether two (short) texts could be related by a monoalphabetic substitution.

2. Implement the attack that uses a crib to break a ciphertext encrypted with the Wheatstone Cryptograph.

Exercises

1. Break this ciphertext that was encrypted with the Cryptograph. Use the crib "the average collector would be bored." What is the keyword?

```
XRIHAOZJMWGRABONYAJXBGPMWMFDHLGBDBYDPVHTBKATCDNUIXEENO
LSUHVCLRCFHRVUQFKBMLMSFGLHMAIAYSWAMNEOPTSJQJBPGITLRYLA
YCRJTGPWAKWAZXSZKONBVNHLXFQDKSPUONMFEGQGBQBCNXNUHUXXTM
```

```
RVXPVUQFKCQLMNYXQUIKNGPHANTBVKIVDBAVWONAZGVEJAMSJKXLHQ
TDZFFPJMSFLKMVXGYSKBTBIWTYDNGXOCNUYOHEDACVHXYPKUGNDOIJ
GQRLSFQEOMMXHQSPRYUMBIFYCVFYGAQJWSJOISLFVESKJNIVMXQXEK
YMYXRVELATVXPNBOMEABVQNUUBVIVUYOZQXYETDKKGGWWDEIPXNEXT
MUFXNESNZXUNJIHVAPMFKYZBKWSXIZVMSCDZEQVCBIGGWUAPMGWMZK
BIZPJURNLOKMUAQJGSNVKAODZMYJGJIMCNUCYKMVVLZCQFSLNWWDYA
KQHQGQRVELDBYSGONJEREYNPKEXSAEUCRQJREGWOIOIUKLADDEZQRO
NHZALWJPEYHMDKIYVYDPFZBSFNMDVEOSQTKDNYIAXUDYAKNQECUULN
LCJICNOOHMVBTCYQLSXSIWBNESKTPOYCXOCOLMKHXYFKCNWIPRDGMW
Q
```

Challenge

crib: `NOT A SOUND BROKE THE SILENCE OF THE STILL NIGHT`

```
DOW#GLGHUJ#VOWNQJLPNWBDZQDAQ$R@CVZIXRMEFP#HIGNPAMGCIKGJTPCI
DQTCSLF$UHAXNE#VNGJUK@WDITNJXNHYRKIYQGEQ$NE@IUVHRT#$KCWLVKS
JEOXCLESNBFKFMJAS$LIOSCNVKLFKMWN@#GHPJOXEYVLOST#RHGOMFASYRT
F#AVPHFGUS@TQHUBIXYB#JLTCJRDUS#QND$UEKBTCTAZPWLCXQYPSLUPG#J
@AMPXB@JI$VYJRTZAFRZU$MNOKASVKABVTBHK#ELPYF$KXQSUJDOCHFLESG
CBZLDHXCRZFENO#BC@YPGW@TGJVTANFVIYFKGYOIZSV@JCSLUPG#BUV#LNS
GHILJZTYNVZUFCPQHRKCESL$ATUZE@IUOUKCWYKBHVOAUPF$PSGSADHP@ZQ
M#TGJ#H$N$UQRHDFN$JXE@IOQLMKRYJNWNQBSW@DROQT#VPENJ@YJIU$R@C
VHUXRIYABOYQZGJDKZJ@QIDQTCH#PHXAHKQSTEUO#N@RSTYPBZTFWLZ$CSZ
@CLVDP@RXE$ZGKXJADKWITE@JMJPCTLYOAZXAQRBOWRIJSNDJ@YRXEFEHXO
HYR@Y$ZCDKMRLAGJFTIEBNK#LIWNCXQYP#YJNWZKTBGOPTGBHPLI
```

Unit 122
Digram-counting attack on cipher clocks

This attack only applies if the length of the ciphertext alphabet is at least two more than the length of the plaintext alphabet. The reason that it works, if the text is long enough, is that when we encipher the next letter of a text, then the hands on the cipher clock move at most *m* steps (remember that m is the length of the plaintext alphabet, and *n* the length of the ciphertext alphabet). So if *n* is one greater than *m*, there are no double letters in the ciphertext, but all other digrams are possible. If *n* is two or more greater than *m*, then the missing digrams indicate what letter(s) is found on the key just before the last ciphertext letter that we wrote down.

To show how it works, we will work through three examples.

example 1: n = m + 2

Below is a ciphertext. It was encrypted with a cipher clock that uses this plaintext alphabet:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

and this ciphertext alphabet, before it is mixed:

ABCDEFGHIJKLMNOPQRSTUVWXYZ◊○

And here it is . . .

XAYIHVBS○URMXE◊○WBQPSTGXECXCXPA◊OCATLXVBEQSZYB◊DTAJSNEKGJDV
WEAXA◊GBNGQWJQXOJEGPXJLFXFESFMWDOWEKEJFPWE◊CABSYUWSKOLZDMH○
BIBYTJQYUECVAUEOCSPKWLGM◊HIEWNEJKCUBSXMSEQVALMNTUBSNBMD○OJE
QGADBPGYPHYQJVHBHIBTDKTAPCEUMI○WEHNGZIDVQXZENWAQBPFHJKMSYOB
UORLGNBLZ◊○QAWPRIEOVY○◊RVYUFCHO◊UZJ○DVUNAT○NF◊UPXKOPYRSZ◊ST
JHKNSMPRDVXTGEKITAPLUEJ○OVR○SUPJPZKGNGDWCXIPJDIXCPBYOQMSLZU
JVNXED○QG○WENUGCN◊OFEPATLMPDWLCDEOATAP◊D○Z○CFORQLUZPXQ○ZGOK
TA◊NKMPG○MQLZUJVNB◊ADBNALQ◊E◊KLND○XUF◊BL○E◊GOBHDA◊DTAGXPXIY
UVBCYOJ◊TQD○WDLIO○NJKB○TDRUZRNG◊H○JW○PSE◊NRD○XUSLHZSRNXSDXZ
GKIJEUYXCPVOMHGHSLNULRZPU○UNO◊EGEAPEXUFWEACWJYIEMLHJXNOWLIQ
YXKVXV○GRCMPYKOJDP◊DJZGVPOVUBIQGAZIHUYX◊RLWBSL◊DSGTBEV○PSKE
DLGRLORQYVJQWBGBZMDAWCUQ○IKET◊PISD◊QPVSZGTLTDZODTUO○MEKO◊FX

OMXCOGZIDVUNAXKRENOQ◊FHVAW◊UIVPSKGJVRNGQVMDJYBPQBQ◊FHOMSFEC
WORYQSZLMK○ZFPSIJQCEHXSEDHNJQJGZOBPSBQMGUJOMSFECWORYQSZLMK○
ZFPSIJQCEHXUJ◊MHMPAEGBNXEIDUWCDK○OQYXKMSIVOPBZKQDRGWZSHBYTA
PLXB◊CSF◊SMAYMEJPTHJWGNBYN○UCZMBL○WCSMTGYJOUBHJQZF○ILIESECK
◊FXIVRXMSENJGQJQDVXECSJU○UKSPAUFBOYVJVJMRVJ○BWCUQKXEJPA◊ELV
JS○QR◊VSEASOGUIMCGDEUYXVYJYLNETQEZO◊PQRO○NKMXZOSVASNKLZJFAP
WCE○FJVGMKMXDZ◊RXMSNSMJNB◊XB○JEMNDI◊YSKWAP◊SRU◊H○JABHNQEOLE
DASYTCPYOSGMIVGOJQAKUCLXJRNUCPAIHVBDU◊LMR○OTLGIZUF○KGXNRDLU
MIDZLCOBGZIJHGCYPEGTJ◊DUAVAVH○MJRJEYQDRGLHXT○BVF◊KSREJCXC◊R
V○RLGBQNBYTAPDNDKMXSLHUJHKHJBDGIUNTWBV○KTAJVGMK◊AUYFZSNOGKR
HBTXUJNKDT◊JRFEUMIHOBXAEILWTCZSJ○KTAPGIFI◊GD◊FS○VZLZGJBTCXB
○JEMNDUPKXEVL○E○MDWDGHIEWTAPGAZVR○SUGL○EYSWTBI○JZFAVFV◊N◊R◊
FSFPJOWJ◊ILMIXVZOPBPS◊PA◊○KBSZ◊GPELOROXOHYNEHTJZ◊T○J◊TPKVZD
BPD◊OJHEJDHDGQEAWGNBJ◊OPCMKBR○IVSCOHYQJUGQ○XZKDGEGXGAZVR○SU
XAEYGZIULT○XNXBPJNFUI○UTGLGKOYBLJWMKDFJWBVDRE◊FHVYBLSCSZSNK
◊XHQDEAGNBTJDVZUZF○EUVYPKBSDJXAGAYSLUZG○VIAOHESYJT○HIWOXBCL
VQNR◊AXUXZNPBRQ◊XFHIEWDA◊QUDHIXYFZS○JYSPKBDHJSNOACE◊KECUIHV
YDNHVJDWXHMKDYPGEN◊FHJBEVCXUOELYJWMKDTETDSAUCUIECXF◊CXPHSZL
V○KVUXBJ○EKPKDHLH○PX○TGWTK○HBLXECTJO◊VZSVAWQJAUSZKXHSIJOZF○
EW◊UNKP○XQJUJEYLMNPSWIWBUSPFJUBDVZVFTXCFEGWP◊SNBMDWBC○QNPHO
BJRTBPUZJHCGQJBZ○DTQYXCKMXDGRMPYJRLRNG◊DJZVLICVHEJKPAVF◊EM○
VYQDUBDXUJWBYDUGYXFCOFEBLGJHUYQJHLT◊DTETPRI◊PJFC◊VF◊RVXAHUY
◊WQEUGWCJ◊EIAGVB◊AIDORQYOQZLGMLVQXYXTXBQYNRDJ◊BZ○FZSY◊LZOTL
MDAPILIPYUI◊HBFSR○RVAV◊L○OMKVUZENUAQBPFHDIVXDJSUYFOROSXMATL
GXPZ◊DSCLUXUIMPKECUGTLRXSTYGZIF○PEWPIFYGOIJVAWDVL○UZPOWDLHE
XZT○EQLXCZDJUVCOPBHYMBZBZHXGEUGPNDW◊HSE◊MXJRBSRNDGZH○JQCPE○
SUIMNDKVEXUFCPTLDHMB◊V◊GW○UAPXNSZPGSAYPVHYXJWMLFZF○◊Y◊○CUZI
○ACVBSMAYQJVUGQMDXBPECUQ○NJVXVGBCP◊LIUWSZYXTQYUJ◊LXCVCWOUZT
○IZJBNYSBDIPSXCZTXLEMNYQJVLIEANOKEBQYXFCXYVQWPDOALM○NGTJFPY
BLIPGCKMJDWEALHPHOROVGHIEWDYJNDUZSYOS○DLXMAJHJL○OMK○CDECD○K
TFPDXCOMPVHC◊YSMOKSZRXLTHJOE○UPKRUCGLUXBEAORXPHEQSHRXMKL◊IS
◊TDUG○PCDIVAI○WZ◊BMJ◊LSROUKL◊I◊LOXQTJUGCX◊SR◊CPYTLQDUBZDEVL
I◊FHFLFAEBSTPY○UK◊OWJLTDJZF○QP◊LUEHXVMCWJYQJNB◊○WGBCTQGKUGL
PFXOASABRZ◊KMKDZTXSTAPYXJFXAEBSNGZIB◊IBFPSIPQUIJBJ◊HFCPNVEJ
◊YJHSEXBEQEQ○RVKXYVFSY○U○OQRMVWDQBWQUQPY○LNR◊FXEOJWJALFJGT◊
SL○TABIE○FNREJKVXPIV○RTXNOHJLSZJBDKVHIBTAIUQNUIPA◊ORQRZLDUB
THZOXSMJQYXCZPIEOJWJALSXHDHMGCJQZF○ME○BIFECUSYOTQXAVUZDTAG◊
HTBTCUKOQGDLFZC◊SMNEQSHFJMKL◊○XGWJUXSURVKX◊CEGMAUILWOSGAWOW
CENO◊FHJKPYR◊EMZDVZ◊○NOVB◊HIBY○UY○PFNA◊○SY

The first step in the attack is to tabulate all of the digrams in the ciphertext. Remember to count *all* of them. The first four letters are XAYI; from them we get XA and AY and YI; don't forget that one in the middle. We don't need to count the digrams, just make a table of all the ones that exist.

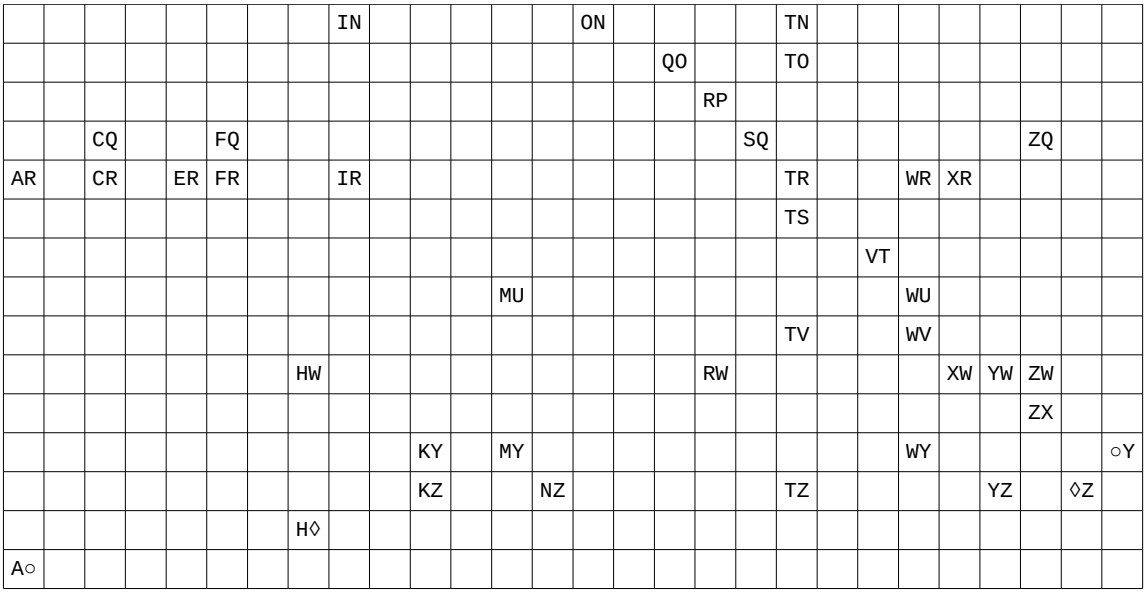
AB			DB	EB	FB	GB	HB	IB	JB	KB		MB	NB	OB	PB	QB	RB	SB	TB	UB	VB	WB	XB	YB	ZB	◊B	○B
AC	BC			EC	FC	GC	HC	IC	JC	KC	LC	MC		OC	PC	QC	RC	SC	TC	UC	VC	WC	XC		ZC	◊C	○C
AD	BD	CD		ED		GD	HD	ID	JD	KD	LD	MD	ND	OD	PD	QD	RD	SD	TD	UD	VD	WD	XD	YD	ZD	◊D	○D
AE	BE	CE	DE		FE	GE	HE	IE	JE	KE	LE	ME	NE	OE	PE	QE	RE	SE	TE	UE	VE	WE	XE		ZE	◊E	○E

	BF	CF	DF				HF	IF	JF		LF		NF	OF	PF		RF	SF	TF	UF	VF		XF	YF	ZF	◊F	○F
AG	BG	CG	DG	EG			HG		JG	KG	LG	MG	NG	OG	PG	QG	RG	SG	TG	UG	VG	WG	XG	YG	ZG	◊G	○G
AH	BH	CH	DH	EH	FH	GH		IH	JH	KH	LH	MH	NH	OH	PH		RH	SH	TH		VH		XH	YH	ZH	◊H	○H
AI	BI		DI	EI	FI	GI	HI			KI	LI	MI		OI	PI		RI	SI		UI	VI	WI	XI	YI	ZI	◊I	○I
AJ	BJ	CJ	DJ	EJ	FJ	GJ	HJ	IJ			LJ	MJ	NJ	OJ	PJ	QJ	RJ	SJ	TJ	UJ	VJ	WJ	XJ	YJ	ZJ	◊J	○J
AK		CK	DK	EK		GK	HK	IK	JK			MK	NK	OK	PK	QK		SK	TK	UK	VK		XK	YK	ZK	◊K	○K
AL	BL	CL	DL	EL	FL	GL	HL	IL	JL	KL		ML		OL	PL	QL	RL	SL	TL	UL	VL	WL	XL	YL	ZL	◊L	○L
	BM	CM	DM	EM	FM	GM	HM	IM	JM	KM	LM			OM		QM	RM	SM		UM	VM	WM	XM	YM	ZM	◊M	○M
AN	BN	CN	DN	EN	FN	GN	HN		JN	KN	LN	MN			PN	QN	RN	SN		UN	VN	WN	XN	YN	ZN	◊N	○N
AO	BO	CO	DO	EO	FO	GO	HO	IO	JO	KO	LO	MO	NO		PO		RO	SO		UO	VO	WO	XO	YO	ZO	◊O	○O
AP	BP	CP	DP	EP	FP	GP	HP	IP	JP	KP	LP	MP	NP	OP		QP		SP	TP	UP	VP	WP	XP	YP	ZP	◊P	○P
AQ	BQ		DQ	EQ		GQ	HQ	IQ	JQ	KQ	LQ	MQ	NQ	OQ	PQ		RQ		TQ	UQ	VQ	WQ	XQ	YQ		◊Q	○Q
	BR		DR			GR	HR		JR	KR	LR	MR	NR	OR	PR	QR		SR		UR	VR			YR	ZR	◊R	○R
AS	BS	CS	DS	ES	FS	GS	HS	IS	JS	KS	LS	MS	NS	OS	PS	QS	RS			US	VS	WS	XS	YS	ZS	◊S	○S
AT	BT	CT	DT	ET	FT	GT	HT	IT	JT	KT	LT	MT	NT	OT	PT	QT	RT	ST		UT		WT	XT	YT	ZT	◊T	○T
AU	BU	CU	DU	EU	FU	GU	HU	IU	JU	KU	LU		NU	OU	PU	QU	RU	SU	TU		VU		XU	YU	ZU	◊U	○U
AV	BV	CV	DV	EV	FV	GV	HV	IV	JV	KV	LV	MV	NV	OV	PV	QV	RV	SV		UV			XV	YV	ZV	◊V	○V
AW	BW	CW	DW	EW	FW	GW		IW	JW	KW	LW	MW	NW	OW	PW	QW		SW	TW	UW	VW					◊W	○W
AX	BX	CX	DX	EX	FX	GX	HX	IX	JX	KX	LX	MX	NX	OX	PX	QX	RX	SX	TX	UX	VX	WX		YX		◊X	○X
AY	BY	CY	DY	EY	FY	GY	HY	IY	JY		LY		NY	OY	PY	QY	RY	SY	TY	UY	VY		XY		ZY	◊Y	
AZ	BZ	CZ	DZ	EZ	FZ	GZ	HZ	IZ	JZ		LZ	MZ		OZ	PZ	QZ	RZ	SZ		UZ	VZ	WZ	XZ				○Z
A◊	B◊	C◊	D◊	E◊	F◊	G◊		I◊	J◊	K◊	L◊	M◊	N◊	O◊	P◊	Q◊	R◊	S◊	T◊	U◊	V◊	W◊	X◊	Y◊	Z◊		○◊
	B○	C○	D○	E○	F○	G○	H○	I○	J○	K○	L○	M○	N○	O○	P○	Q○	R○	S○	T○	U○	V○	W○	X○	Y○	Z○	◊○	

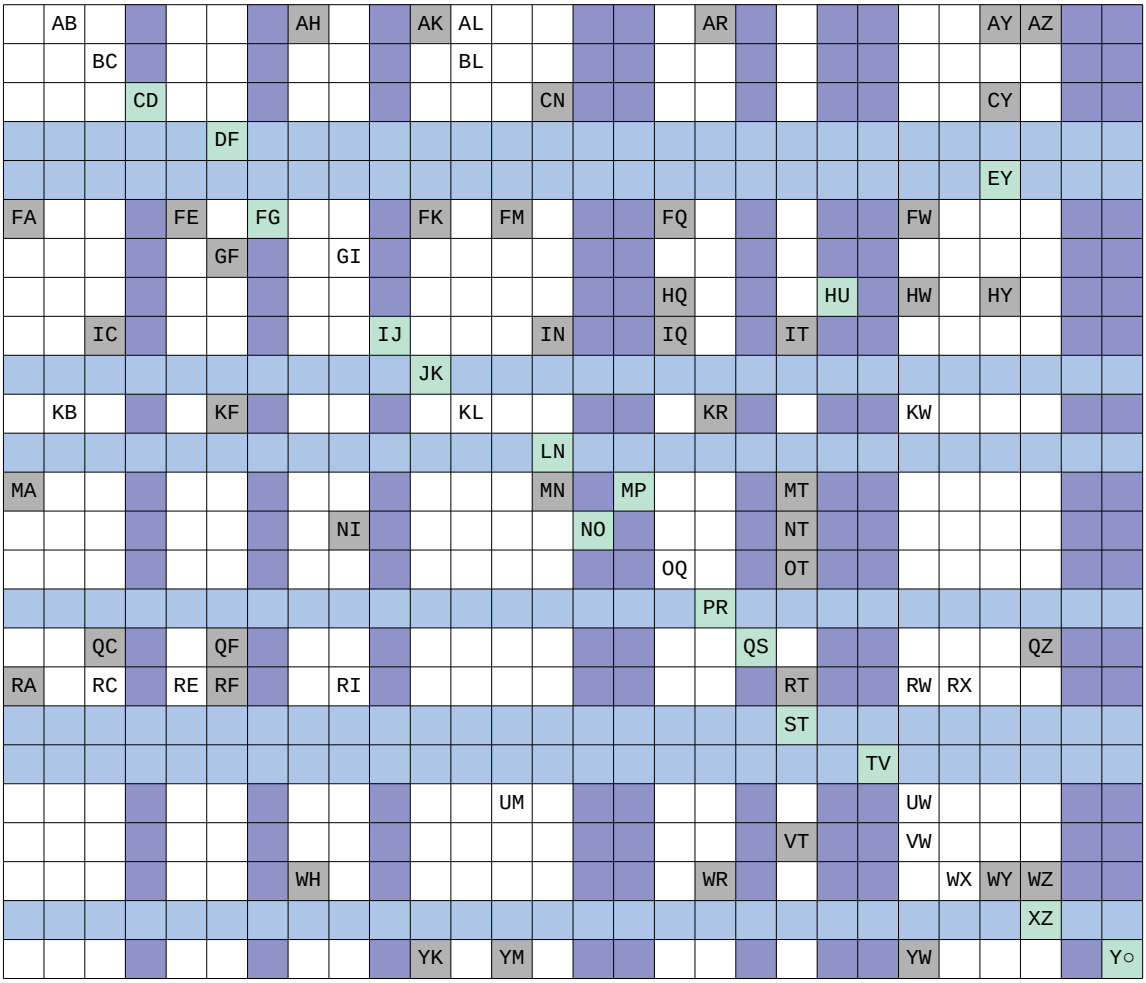
What is important to us now is which digrams are *missing*. Of course, all of the diagonal entries are missing, since double letters cannot occur in the ciphertext for a device with this configuration. But look at the `J` column. The digram for `JI` is the only other one missing. Therefore, we know that `I` comes immediately before `J` in the key.

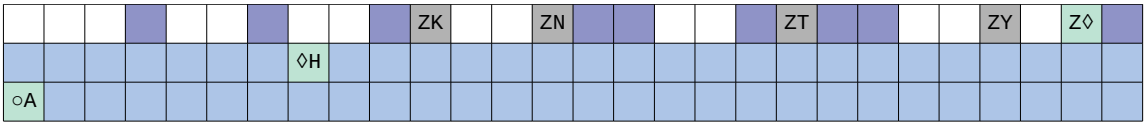
Here are all of the missing digrams, except for the doubles:

	BA						HA			KA	LA						RA							YA	ZA		
		CB									LB																
			DC										NC											YC			
					FD																						
																								YE			
AF				EF		GF				KF		MF				QF						WF					
					FG			IG																			
																QH				UH		WH		YH			
		CI							JI				NI			QI			TI								
										KJ																	
	BK				FK						LK						RK					WK					
													NL														
AM													NM		PM				TM								

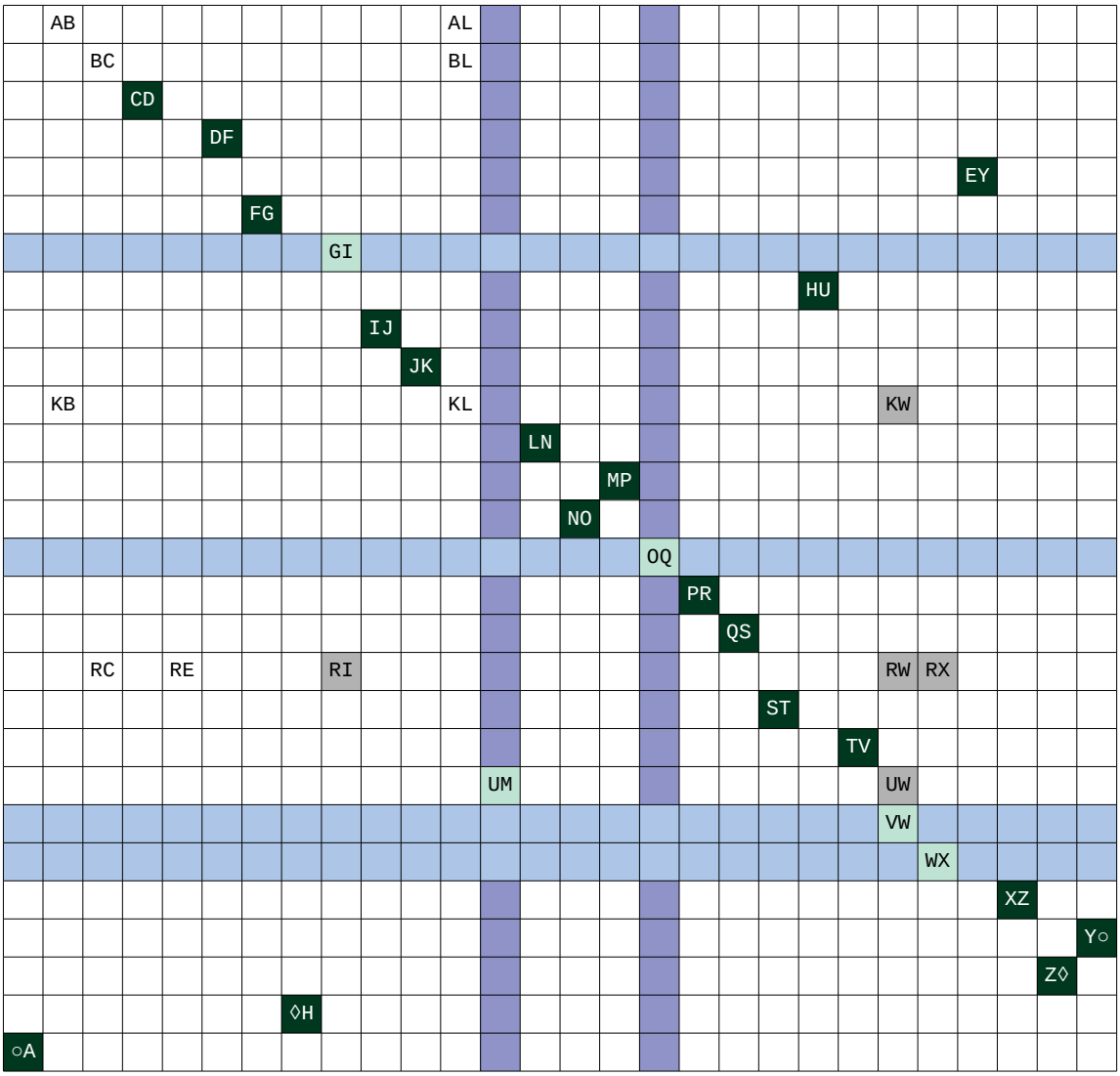


Now don’t get confused, but to make things easier, we reverse each missing digram so that it is written in the way it might appear in the key. Here is the new table:

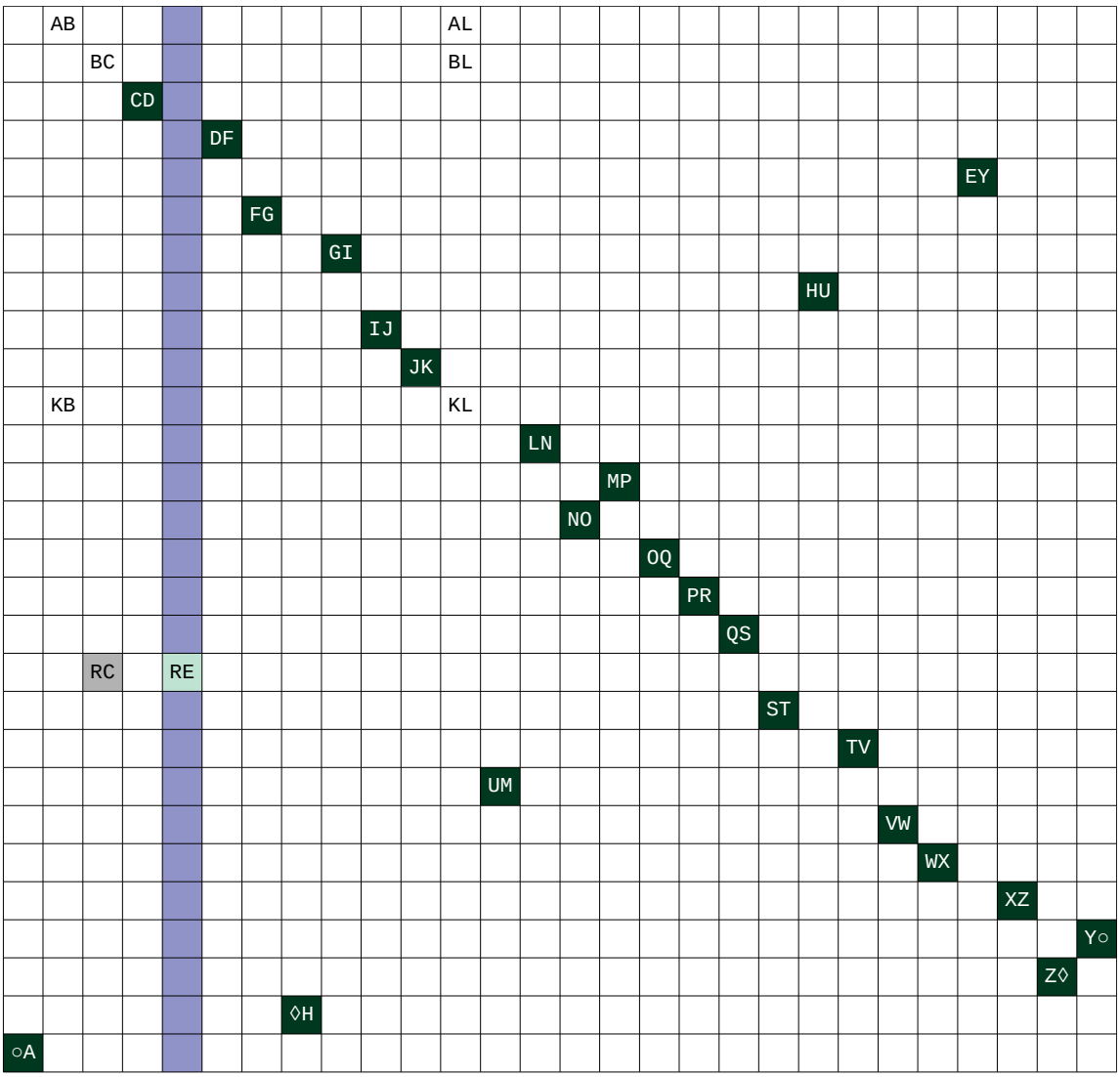




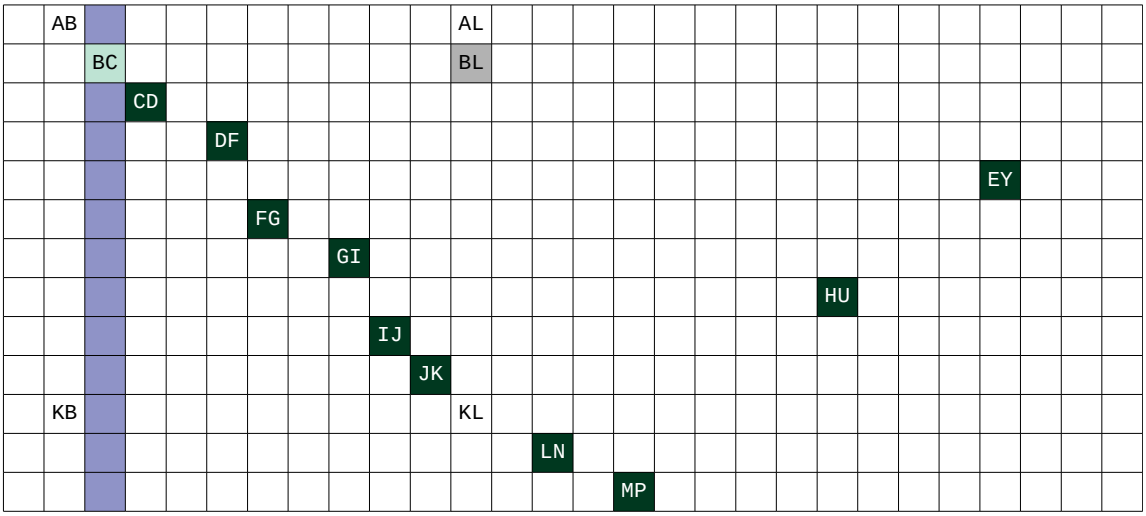
Now we look for any digram that appears alone in a row or column. Such a digram must be in the key. We find CD, FG, IJ, NO, MP, QS, HU, TV, Z◊, Y○, DF, EY, JK, LN, PR, ST, XZ, ◊H, and ○A. Once we decide to keep a digram in a row, we can eliminate the other digrams in its column; similarly, once we keep a digram in a column, we can eliminate the others in the same row. We can also eliminate the reversal of any digram that we keep. Here is what is left after we make those deletions:

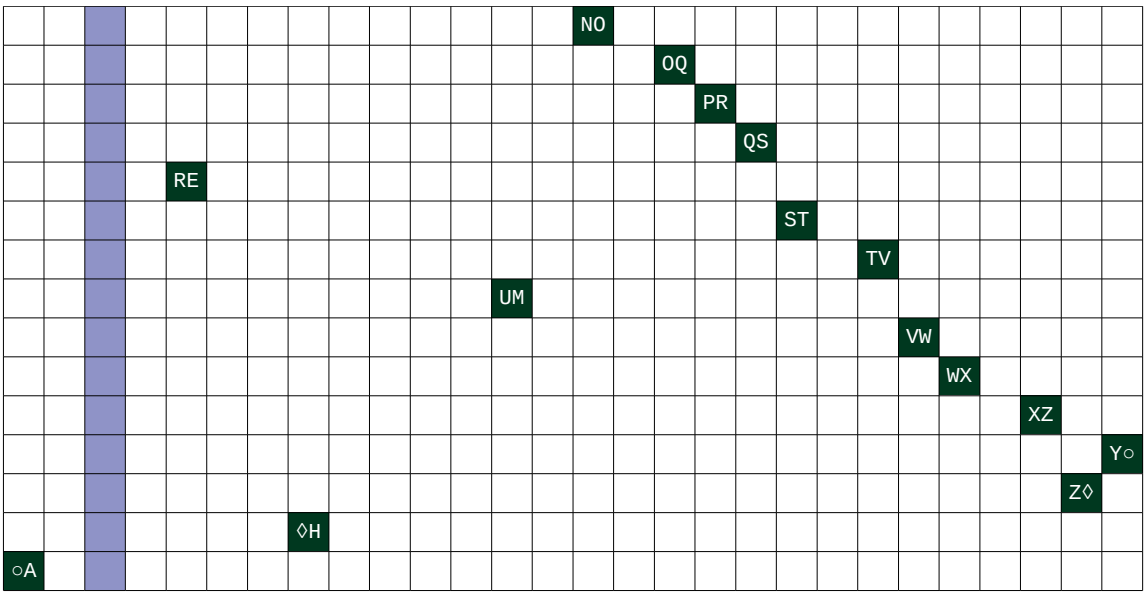


Now we play the same game and look for digrams that are alone on a row or column in what remains in the table. We find GI, OQ, UM, VW, and WX. Eliminate those rows and columns to get an even smaller table:



Now we get RE and eliminate RC. One more time:





Now we keep BC and eliminate BL. One more elimination won't gain us anything, so we stop.

Here are all of the key's digrams that we have found to definitely belong:

```
BC CD DF EY FG GI HU IJ JK LN MP NO OQ
PR QS RE ST TV UM VW WX XZ Y○ Z◊ ◊H ○A
```

But remember that AB, AL, KB, and KL are still possibilities.

If we chain the definte key digrams together, we get these key fragments:

```
BCDFGIJK
LNOQSTVWXZ◊HUMPREY○A
```

There are two ways to combine them, but it doesn't matter, since the key goes on a ring. So now we know that the key is something like

```
LNOQSTVWXZ◊HUMPREY○ABCDFGIJK
```

To find the final key, we need to roll this key to each possible position on the ring and decipher the ciphertext. When we find a recognizably English plaintext, then we have found the correct key. It turns out that the correct key is

```
◊HUMPREY○ABCDFGIJKLNOQSTVWXZ
```

and the plaintext is (broken into words)

```
ALL THESE SORROWS ARE PAST MY GLANCING AT THEM MAY NOT BE
WITHOUT ITS USE FOR IT MAY HELP IN SOME MEASURE TO EXPLAIN
WHY I HAVE ALL MY LIFE BEEN ATTACHED TO THE INANIMATE
OBJECTS THAT PEOPLE MY CHAMBER AND HOW I HAVE COME TO
```

```
LOOKUP ON THEM RATHER IN THE LIGHT OF OLD AND CONSTANT
FRIENDS THAN AS MERE CHAIRS AND TABLES WHICH A LITTLE MONEY
COULD REPLACE AT WILL CHIEF AND FIRST AMONG ALL THESE IS MY
CLOCK MY OLD CHEERFUL COMPANIONABLE CLOCK HOW CAN I EVER
CONVEY TO OTHERS AN IDEA OF THE COMFORT AND CONSOLATION
THAT THIS OLD CLOCK HAS BEEN FOR YEARS TO ME IT IS
ASSOCIATED WITH MY EARLIEST RECOLLECTIONS IT STOOD UPON THE
STAIRCASE AT HOME I CALL IT HOME STILL MECHANICALLY NIGH
SIXTY YEARS AGO I LIKE IT FOR THAT BUT IT IS NOT ON THAT
ACCOUNT NOR BECAUSE IT IS A QUAINT OLD THING IN A HUGE
OAKEN CASE CURIOUSLY AND RICHLY CARVED THAT I PRIZE IT AS I
DO I INCLINE TO IT AS IF IT WERE ALIVE AND COULD UNDERSTAND
AND GIVE ME BACK THE LOVE I BEAR IT AND WHAT OTHER THING
THAT HAS NOT LIFE COULD CHEER ME AS IT DOES WHAT OTHER
THING THAT HAS NOT LIFE I WILL NOT SAY HOW FEW THINGS THAT
HAVE COULD HAVE PROVED THE SAME PATIENT TRUE UNTIRING
FRIEND HOW OFTEN HAVE IS AT IN THE LONG WINTER EVENINGS
FEELING SUCH SOCIETY IN ITS CRICKET VOICE THAT RAISING MY
EYES FROM MY BOOK AND LOOKING GRATEFULLY TOWARDS IT THE
FACE REDDENED BY THE GLOW OF THE SHINING FIRE HAS SEEMED TO
RELAX FROM ITS STAID EXPRESSION AND TO REGARD ME KINDLY HOW
OFTEN IN THE SUMMER TWILIGHT WHEN MY THOUGHTS HAVE WANDERED
BACK TO A MELANCHOLY PAST HAVE ITS REGULAR WHISPERINGS
RECALLED THEM TO THE CALM AND PEACEFUL PRESENT HOW OFTEN IN
THE DEAD TRANQUILLITY OF NIGHT HAS ITS BELL BROKEN THE
OPPRESSIVE SILENCE AND SEEMED TO GIVE ME ASSURANCE THAT THE
OLD CLOCK WAS STILL A FAITHFUL WATCHER AT MY CHAMBER DOOR
MY EASY CHAIR MY DESK MY ANCIENT FURNITURE MY VERY BOOKS I
CAN SCARCELY BRING MYSELF TO LOVE EVEN THESE LAST LIKE MY
OLD CLOCK IT STANDS IN A SNUG CORNER MIDWAY BETWEEN THE
FIRESIDE AND A LOW ARCHED DOOR LEADING TO MY BEDROOM ITS
FAME IS DIFFUSED SO EXTENSIVELY THROUGHOUT THE
NEIGHBOURHOOD THAT I HAVE OFTEN THE SATISFACTION OF HEARING
THE PUBLICAN OR THE BAKER AND SOMETIMES EVEN THE PARISH
CLERK PETITIONING MY HOUSEKEEPER OF WHOM IS HALL HAVE MUCH
TO SAY BY AND BY TO IN FORM HIM THE EXACT TIME BY MASTER
HUMPHREYS CLOCK MY BARBER TO WHOM I HAVE REFERRED WOULD
SOONER BELIEVE IT THAN THE SUN NOR ARE THESE ITS ONLY
DISTINCTIONS IT HAS ACQUIRED I AM HAPPY TO SAY ANOTHER
INSEPARABLY CONNECTING IT NOT ONLY WITH MY ENJOYMENTS AND
REFLECTIONS BUT WITH THOSE OF OTHER MEN
```

You might recognize the text from *Master Humphrey's Clock* by Charles Dickens.

example 2: $n = m + 3$

For this example, we need to add another character to the ciphertext alphabet. This time, for each character in the key, we should find two missing digrams for the two characters that follow it. In which

order those two occur can be deduced from the choices of what characters might follow them. For example, if we have missing digrams AB, AC, BC, and BD, then we know that B and C are after A. Since C is after B, the order is ABCD.

Here is a ciphertext for us to break:

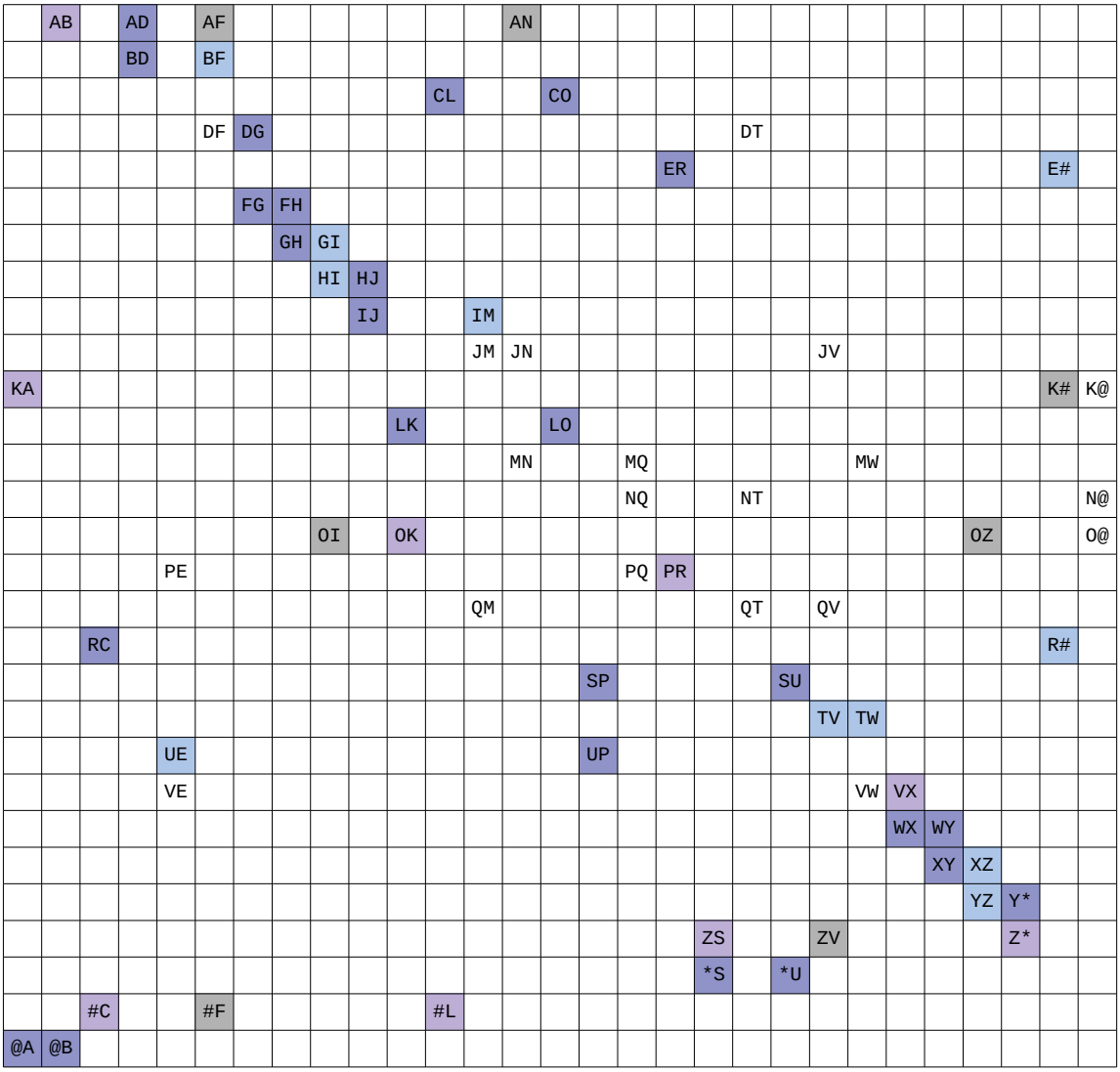
```
JEY@VHRIECWIBIRMBSVECN*LOTHSDIZ*DLPAF*LHUN@XCFXLACYUTHBPTCT
KVDENVWEXBVKMVGK#WG*GJVYSI@WAHOMY@YDZQ#JTERJBLHMRBQCXGUAMUB
HAPZG@AV#WSPDINPX#HEKEHVLSQID#HVLV@WJZMUHYIMTCWH@CXKVDYOWUE
QZGRN#CA#QVECN*ISCVBL@FME#AWOA#TZVMPOZR*CHXKICFSEHXZL@WELWK
VW#W@SYOQV*HPIPQAFTXGJV*#MDHVSWLXLU@PHNKIJPBPXPZDJOTIXLRD#H
CEIZAOJEBGSOBOXFYPOMERWYSTFMKXRLYSP#@PBZQIB*B#GL*IJXGBVE*QY
RSAXYJCGQGWOWSWEO#YUAHVNFSA#HDPQFMRG*DHSKPGPIXGXLGERSCNF*SA
TJWNHEWXJPJFSI@CUR*VFKQXLQ*#IJUZBCWDHLXSMTJUXFSQRI#FUPM*BMA
XYPQOBUZNCLSMROPRW@WJYAWRGTLPG@RIYUCBYN#DXFIWUXOYKN#BOXCVOT
HUDOIN#DQSDS@AHZLVIUNG*GHYLBPGUEDUEYMANSDFZRSQIE#Z@BOF*MDR#
BXBITBYJSDSHZ#PXUNTRLJOZBCQCLBCFLWZKW#IQEIAWRLUFR*BCBLENRKN
*NX@SMA#VWK@IJ*TEJBO@DTFODY@DYRAVYNHZF*NQYFIJY#FGQR#XQWEJPQ
CBJD*NGS@U#HMOADZWXC@QTXGUTYZ#TWN*VRK*MEFCNQYHLYRGBN#*UBXJO
JEHKQVAFXQA#*W*K@PX#*BEMXHBRKGLBLFWSMEWKRVEOZIBQS#MDV#IYS#D
@HWEHETUZRB*@DX*XQBKNWKEADIJAY@WNGESGMOBCN#*BXFTCMBXLBIAORI
U@XZPLD*EQGJRLGBQZBPQU@DZCTCTKVIPRV#QZSJWBUX*VCP@V*FHBI*PCW
X@RJN@LOD@WJDTX@JZBPQW*BJSXAVNWSMEW#QZK#PKIPQAYN#MK*OIN#DLY
BKWRQIRXFKWPJT*HJSDLWH#IBWNLKUMNBGIPXTYQT#MYFCMAHQTSDMG#OH*
@DPOXFKXKPBLJFSVHRCDYD@GOBVKMCAIPED@RVENPVA#@AFEA#LZ*N*@WXU
TRSHQZKAT@QYJETBRSIXZUVGN@*GQRWF*TCDSCSFCJAFZSIAILVGPOTZRJB
V#VN*#SCUAOFULUBIJQ@WOIUFUWSFXKFJCJONZDRNYNKYKCJE#ALQSXJKSH
THURK*VLWSCNFOR#*MPT*@BCUVRHXYP*NOINCUHD#P#IJPRJBLMPDNW*NST
BUH#CJZV@RLB#QVCIPF*TKZDSDNIESDSYZKNO@TP#LQG*OHMU#SNQ*JR@CZ
ENOIUIL*OP#PAFJEFXJMSM#YBDOFCLGZJUO*HZ*MWXCLQGVX*BHXT@Q*JRK
*BRL@SCKZTGJOTUNIPXAERSILVSVWJRUYNRNLQGWUT@DJ#WCSKNLDLNDGNE
IAXYRVKGAIPQAHEKGEHPHY@LMVSIMUFGOXIUQ*#*UTMC*HLNL@BROKRZQJR
GJFRK*EDGKTSD#MDPDNFRTYQVLVNFLFTB#CMXQDMGAZMNCNFOQJACAUKYVY
IM#BVMFMKTFJACWP#MFXMAHTZ#J#QCSKGLAGMXTIUBLWOG#MHCMCPOS#MDW
IET#FIKEFK@XULTYGUTYKCQCSG@MZETUZRSLXKJQVOPBIPXKNSGRUVA#FIP
@JMRZT#SY*WX*OMU#SNWH@CXJZUGVPEIAPRQ#OZGLETLRHXYEGHRNY*AHEM
RZCFCILRJBVXMNCMLBJSDITRAMEMUYBFWZSEGCOUVP*#MCBRK#PH@JVACLM
Y@YAWYA*FRGUGKIPNYDMZTRUWCDYD#QZHKR#IRG@AJLAMYVPYRXINRCNOVD
WI*PZFXQZMFUXGVDKFULHZ@QJYABTFGNEGKRJPDRHVFIZDJNXAWRJZIDXOP
YASBQYKMLPMGNBRKVIBQWKIBVMPKSMCBFJPO#YF@LFTLYHMUN@QKJPVOEBP
IFSKNLDYHOIDWNOVOPGEZ*AOTV*J@RFXEHEMRVDR#J*FYG@#JPQJXZEGLIR
@#ZGLWRTBGB#S#WOSNPFVYCMUNRMNFL*KMPZDG*QK#QCXGNLDENOIUDZARH
JCUBEOZJPJFYTZ@TYLHTAWQOFNOQUPLITEAYVM@GQE*NLDMOYKN@SYA#NC@
*EDQ#QOTHJNSNY@QFOUGMZ#P@WJDTX@JDJWKE*RKUCXLUZT*LPTUIQZGI*B
LJLO#OY@SXUIVSIU@AHYZTW*BQIBCZBDVSHTKBDQGSMKH#DX@SYOQGESQVA
YOZCUP@DZNGRTRZAWJQL@JGWQDJC*B*GYCZEKVIFPANBILZAZBXRTUHMENX
G*DETZFN@AHPTWNFEADPXDMXNULHZH#ABMEJNCN@SYA#SKBY#IQ@E*VLFVY
*RJAHVMUGKNK@TPOPJXPXY#AMSHTARLXRIMTPWRIQE*ANOQLKPHWOWKTLSC
SADZCMCXF@YZAZ#DXLI*GSHZDISNE@UALAON#P@BQWKIYAUNEBGYKRLSVFK
```

```
*@CVD*ALGWR@XPZATHZE#J#DLPEGSVHRCIDJV@ATOIR#FSPYKGEQYLTSCVB
*DFMDCOQUBO*R@CL*AWCZBDJOYEJ*RFWLFVLNHQYEIACAQGABLXACVFKNIA
HRYGVCWOQLTYGYHKRC@Q@STFVO*LGNEWOALOTKJPZTBOISXUIK@VLEYGVBW
PTFHRMWFLMGTB*DH#ZKW#JRQJQKBIV#JPHIWOWDJCVHPLIDLSISA*NEIUPZ
DGRGTBP#*U@DZCGSMKVIR*ALVGMZTIQ@EN*SJYVKXISDWALWQAHJSRJMVSB
M@ECUXHAVYNKDPEIAQIBX@WOI*CBPJMER#YVKFX@RJY#YPDHNKIUJODGTMK
JWRDHUKXKXLENCN@S#QZCJWBUIPKBCBWXSDJMUHXBSQIDZIDJ#ADIULJFUT
GJMZ@QJ@SMAN#CLHQF@GYTMXHI*LQ#CBORNBDYJACLMZUZNKX*BYMSMK#SV
RGZL@ZEOKHSGINFEQZGRMZTI*TPV#UPYFHLQ#OPHZWRIUIRJBG#J#NF@UCB
KUCKIYEOTGNZV#MDWDFXET@M*NTXY#I*FWEMVSIMUFHOW*VWLQTEADZPOHQ
Z#JRCNCLFQRGSBYHCGWZBNTUXYCOQVUKFKTPQDTWY@M*MNWKTCKZEHTSCUA
JZADZQH#*STXEMENEHNSDVDKCNFHZARVEOZGLQTCBOZ@FVYAMFWATJFZ#*I
J*J#W@BXZ#DJWS@QWDHCFLESUXFSFQLYOZI*HXK
```

Here is our table of digrams present in the ciphertext:

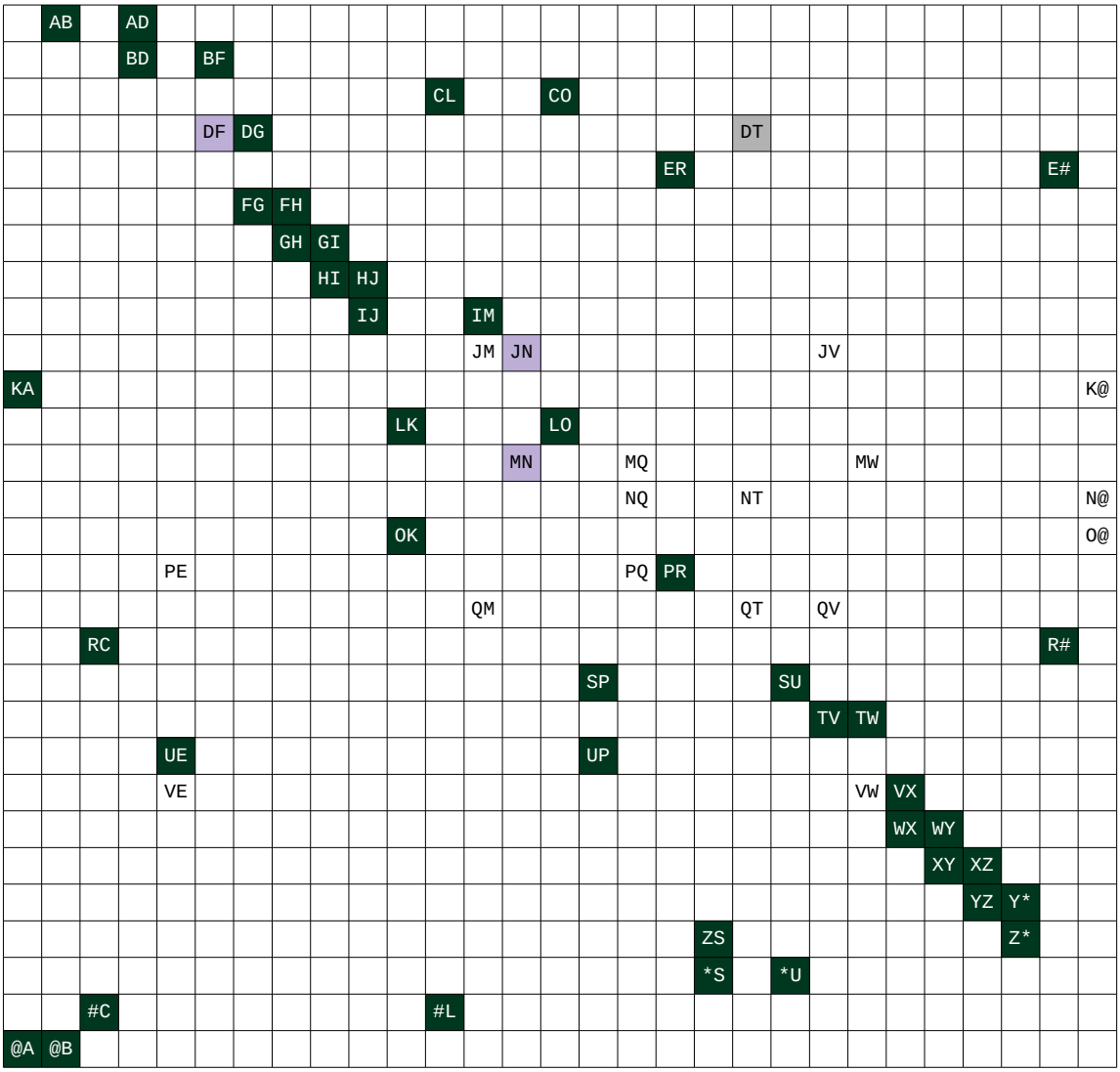
		CA		EA		GA	HA	IA	JA	KA	LA	MA		OA	PA	QA	RA	SA	TA	UA	VA	WA	XA	YA	ZA	*A	#A	@A
AB		CB		EB		GB	HB	IB	JB	KB	LB	MB	NB	OB	PB	QB	RB	SB	TB	UB	VB	WB	XB	YB	ZB	*B	#B	@B
AC	BC		DC	EC	FC	GC	HC	IC	JC	KC		MC	NC		PC	QC	RC	SC	TC	UC	VC	WC	XC	YC	ZC	*C	#C	@C
AD	BD	CD		ED			HD	ID	JD	KD	LD	MD	ND	OD	PD	QD	RD	SD		UD	VD	WD	XD	YD	ZD	*D	#D	@D
AE	BE	CE	DE		FE	GE	HE	IE	JE	KE	LE	ME	NE	OE	PE	QE		SE	TE	UE	VE	WE	XE	YE	ZE	*E	#E	@E
AF	BF	CF	DF	EF				IF	JF	KF	LF	MF	NF	OF	PF	QF	RF	SF	TF	UF	VF	WF	XF	YF	ZF	*F		@F
AG	BG	CG	DG	EG	FG				JG	KG	LG	MG	NG	OG	PG	QG	RG	SG	TG	UG	VG	WG	XG	YG	ZG	*G	#G	@G
AH	BH	CH	DH	EH	FH	GH				KH	LH	MH	NH	OH	PH	QH	RH	SH	TH	UH	VH	WH	XH	YH	ZH	*H	#H	@H
AI	BI	CI	DI	EI	FI	GI	HI			KI	LI		NI	OI	PI	QI	RI	SI	TI	UI	VI	WI	XI	YI	ZI	*I	#I	@I
AJ	BJ	CJ	DJ	EJ	FJ	GJ	HJ	IJ		KJ	LJ			OJ	PJ	QJ	RJ	SJ	TJ	UJ		WJ	XJ	YJ	ZJ	*J	#J	@J
	BK	CK	DK	EK	FK	GK	HK	IK	JK		LK	MK	NK	OK	PK	QK	RK	SK	TK	UK	VK	WK	XK	YK	ZK	*K		
AL	BL	CL	DL	EL	FL	GL	HL	IL	JL			ML	NL		PL	QL	RL	SL	TL	UL	VL	WL	XL	YL	ZL	*L	#L	@L
AM	BM	CM	DM	EM	FM	GM	HM	IM	JM	KM	LM			OM	PM		RM	SM	TM	UM	VM		XM	YM	ZM	*M	#M	@M
AN	BN	CN	DN	EN	FN	GN	HN	IN	JN	KN	LN	MN		ON	PN		RN	SN		UN	VN	WN	XN	YN	ZN	*N	#N	
AO	BO	CO	DO	EO	FO	GO	HO		JO		LO	MO	NO		PO	QO	RO	SO	TO	UO	VO	WO	XO	YO		*O	#O	
AP	BP	CP	DP		FP	GP	HP	IP	JP	KP	LP	MP	NP	OP				SP	TP	UP	VP	WP	XP	YP	ZP	*P	#P	@P
AQ	BQ	CQ	DQ	EQ	FQ	GQ	HQ	IQ	JQ	KQ	LQ		NQ	OQ	PQ		RQ	SQ		UQ		WQ	XQ	YQ	ZQ	*Q	#Q	@Q
AR	BR		DR	ER	FR	GR	HR	IR	JR	KR	LR	MR	NR	OR	PR	QR		SR	TR	UR	VR	WR	XR	YR	ZR	*R		@R
AS	BS	CS	DS	ES	FS	GS	HS	IS	JS	KS	LS	MS	NS	OS		QS	RS		TS		VS	WS	XS	YS	ZS	*S	#S	@S
AT	BT	CT	DT	ET	FT	GT	HT	IT	JT	KT	LT	MT	NT	OT	PT	QT	RT	ST		UT			XT	YT	ZT	*T	#T	@T
AU	BU	CU	DU		FU	GU	HU	IU	JU	KU	LU	MU	NU	OU		QU	RU	SU	TU		VU	WU	XU	YU	ZU	*U	#U	@U
AV	BV	CV	DV		FV	GV	HV	IV	JV	KV	LV	MV	NV	OV	PV	QV	RV	SV	TV	UV				YV	ZV	*V	#V	@V
AW	BW	CW	DW	EW	FW	GW	HW	IW	JW	KW	LW	MW	NW	OW	PW	QW	RW	SW	TW	UW	VW				ZW	*W	#W	@W
AX	BX	CX	DX	EX	FX	GX	HX	IX	JX	KX	LX	MX	NX	OX	PX	QX	RX	SX	TX	UX	VX	WX				*X	#X	@X
AY	BY	CY	DY	EY	FY	GY	HY	IY	JY	KY	LY	MY	NY	OY	PY	QY	RY	SY	TY	UY	VY	WY	XY				#Y	@Y
AZ	BZ	CZ	DZ	EZ	FZ	GZ	HZ	IZ	JZ	KZ	LZ	MZ	NZ	OZ	PZ	QZ	RZ		TZ	UZ		WZ	XZ	YZ			#Z	@Z
A*	B*	C*	D*	E*	F*	G*	H*	I*	J*	K*	L*	M*	N*	O*	P*	Q*	R*		T*		V*	W*	X*	Y*	Z*		#*	@*
A#	B#		D#	E#		G#	H#	I#	J#	K#		M#	N#	O#	P#	Q#	R#	S#	T#	U#	V#	W#	X#	Y#	Z#	*#		@#
		C@	D@	E@	F@	G@	H@	I@	J@	K@	L@	M@	N@	O@	P@	Q@	R@	S@	T@	U@	V@	W@	X@	Y@	Z@	*@	#@	

Here are the missing digrams, reversed:

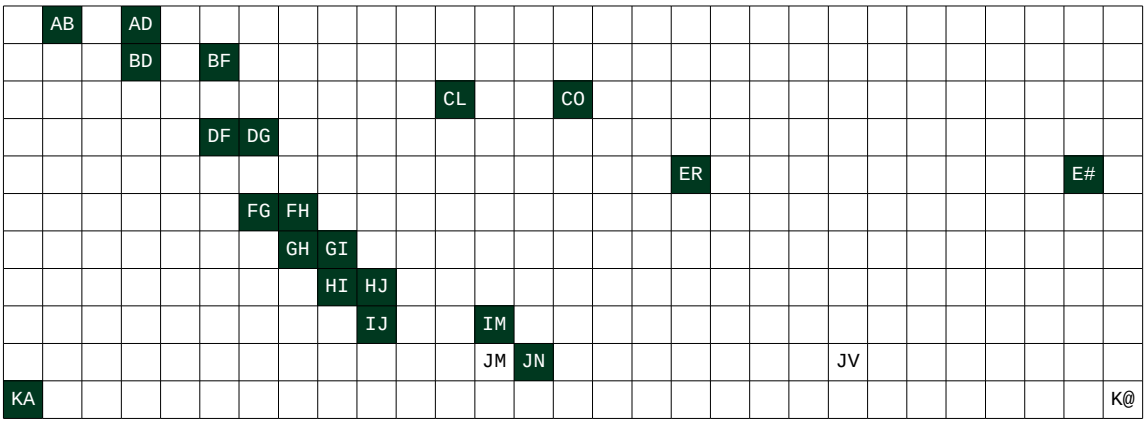


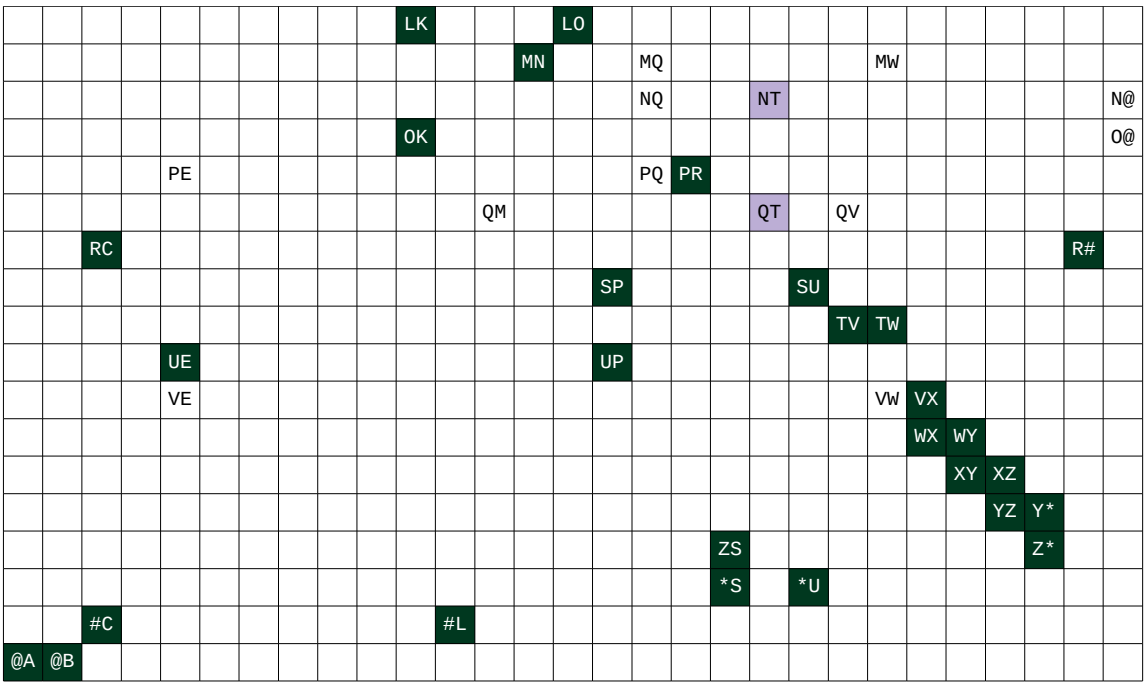
This time, we are looking for rows and columns that have exactly *two* digrams (some are in both), since there are two characters that we cannot reach when we encipher the next letter. From rows we get BD, BF, CL, CO, ER, E#, FG, FH, GH, GI, HI, HJ, IJ, IM, LK, LO, RC, R#, SP, SU, TV, TW, UE, UP, WX, WY, XY, XZ, YZ, Y*, *S, *U, @A, and @B. From columns, we add AB, #C, AD, DG, KA, OK, #L, PR, ZS, VX, and Z*.

We can eliminate all non-listed digrams from rows and columns that contain two of the digrams in our list, since we know that any other digrams in such a row or column is accidentally rather than necessarily missing. One such column contains GI, HI, and OI; since GI and HI are in our list, we know that OI cannot be. The table becomes



Now there are some new rows and columns with only two digrams. This allows us to add `DF`, `JN`, and `MN` to our list. Moving onward, we can eliminate a few more from rows and columns that already have two listed digrams, such as `DT`.





Now we can also add NT and QT. We have reached the end of this path. The key diagrams we have are

```
AB AD BD BF CL CO DF DG ER E# FG FH GH GI HI HJ IJ IM JN KA
LK LO MN NT OK PR QT RC R# SP SU TV TW UE UP VX WX WY XY XZ
YZ Y* ZS Z* *S *U @A @B #C #L
```

We see that A is followed by B and D, and B is followed by D. So the order must be ABD. Using this logic, we can construct this key:

```
@ABDFGHIJMNQTVWXYZ*SUPER#CLOK
```

We have to roll the key and try each option until we get a readable plaintext. The true key that we find is ultimately

```
*SUPER#CLOK@ABDFGHIJMNQTVWXYZ
```

and the plaintext is (with spaces added)

```
THERE ARE TWO KINDS OF CLOCKS THERE IS THE CLOCK THAT IS
ALWAYS WRONG AND THAT KNOWS IT IS WRONG AND GLORIES IN IT
AND THERE IS THE CLOCK THAT IS ALWAYS RIGHT EXCEPT WHEN YOU
RELY UPON IT AND THEN IT IS MORE WRONG THAN YOU WOULD THINK
A CLOCK COULD BE IN A CIVILIZED COUNTRY I REMEMBER A CLOCK
OF THIS LATTER TYPE THAT WE HAD IN THE HOUSE WHEN I WAS A
BOY ROUTING US ALL UP AT THREE OCLOCK ONE WINTERS MORNING
WE HAD FINISHED BREAKFAST AT TEN MINUTES TO FOUR AND I GOT
TO SCHOOL A LITTLE AFTER FIVE AND SAT DOWN ON THE STEP
OUTSIDE AND CRIED BECAUSE I THOUGHT THE WORLD HAD COME TO
```

AN END EVERYTHING WAS SO DEATHLIKE THE MAN WHO CAN LIVE IN THE SAME HOUSE WITH ONE OF THESE CLOCKS AND NOT ENDANGER HIS CHANCE OF HEAVEN ABOUT ONCE A MONTH BY STANDING UP AND TELLING IT WHAT HE THINKS OF IT IS EITHER A DANGEROUS RIVAL TO THAT OLD ESTABLISHED FIRM JOB OR ELSE HE DOES NOT KNOW ENOUGH BAD LANGUAGE TO MAKE IT WORTH HIS WHILE TO START SAYING ANYTHING AT ALL THE GREAT DREAM OF ITS LIFE IS TO LURE YOU ON INTO TRYING TO CATCH A TRAIN BY IT FOR WEEKS AND WEEKS IT WILL KEEP THE MOST PERFECT TIME IF THERE WERE ANY DIFFERENCE IN TIME BETWEEN THAT CLOCK AND THE SUN YOU WOULD BE CONVINCED IT WAS THE SUN NOT THE CLOCK THAT WANTED SEEING TO YOU FEEL THAT IF THAT CLOCK HAPPENED TO GET A QUARTER OF A SECOND FAST OR THE EIGHTH OF AN INSTANT SLOW IT WOULD BREAK ITS HEART AND DIE IT IS IN THIS SPIRIT OF CHILDLIKE FAITH IN ITS INTEGRITY THAT ONE MORNING YOU GATHER YOUR FAMILY AROUND YOU IN THE PASSAGE KISS YOUR CHILDREN AND AFTERWARD WIPE YOUR JAMMY MOUTH POKE YOUR FINGER IN THE BABYS EYE PROMISE NOT TO FORGET TO ORDER THE COALS WAVE AT LAST FOND ADIEU WITH THE UMBRELLA AND DEPART FOR THE RAILWAY STATION I NEVER HAVE BEEN QUITE ABLE TO DECIDE MYSELF WHICH IS THE MORE IRRITATING TO RUN TWO MILES AT THE TOP OF YOUR SPEED AND THEN TO FIND WHEN YOU REACH THE STATION THAT YOU ARE THREE QUARTERS OF AN HOUR TOO EARLY OR TO STROLL ALONG LEISURELY THE WHOLE WAY AND DAWDLE ABOUT OUTSIDE THE BOOKING OFFICE TALKING TO SOME LOCAL IDIOT AND THEN TO SWAGGER CARELESSLY ON TO THE PLATFORM JUST IN TIME TO SEE THE TRAIN GO OUT AS FOR THE OTHER CLASS OF CLOCKS THE COMMON OR ALWAYS WRONG CLOCKS THEY ARE HARMLESS ENOUGH YOU WIND THEM UP AT THE PROPER INTERVALS AND ONCE OR TWICE A WEEK YOU PUT THEM RIGHT AND REGULATE THEM AS YOU CALL IT AND YOU MIGHT JUST AS WELL TRY TO REGULATE A LONDON TOMCAT BUT YOU DO ALL THIS NOT FROM ANY SELFISH MOTIVES BUT FROM A SENSE OF DUTY TO THE CLOCK ITSELF YOU WANT TO FEEL THAT WHATEVER MAY HAPPEN YOU HAVE DONE THE RIGHT THING BY IT AND THAT NO BLAME CAN ATTACH TO YOU SO FAR AS LOOKING TO IT FOR ANY RETURN IS CONCERNED THAT YOU NEVER DREAM OF DOING AND CONSEQUENTLY YOU ARE NOT DISAPPOINTED YOU ASK WHAT THE TIME IS AND THE GIRL REPLIES WELL THE CLOCK IN THE DININGROOM SAYS A QUARTER PAST TWO BUT YOU ARE NOT DECEIVED BY THIS YOU KNOW THAT AS A MATTER OF FACT IT MUST BE SOMEWHERE BETWEEN NINE AND TEN IN THE EVENING AND REMEMBERING THAT YOU NOTICED AS A CURIOUS CIRCUMSTANCE THAT THE CLOCK WAS ONLY FORTY MINUTES PAST FOUR HOURS AGO YOU MILDLY ADMIRE ITS ENERGIES AND RESOURCES AND WONDER HOW IT DOES IT I MYSELF POSSESS A CLOCK THAT FOR COMPLICATED UNCONVENTIONALITY AND LIGHTHEARTED INDEPENDENCE COULD I SHOULD THINK GIVE POINTS TO ANYTHING YET DISCOVERED IN THE CHRONOMETRICAL LINE AS A MERE TIMEPIECE IT LEAVES MUCH TO BE DESIRED BUT CONSIDERED AS A SELFACTING CONUNDRUM

```
IT IS FULL OF INTEREST AND VARIETY I HEARD OF A MAN ONCE
WHO HAD A CLOCK THAT HE USED TO SAY WAS OF NO GOOD TO ANY
ONE EXCEPT HIMSELF BECAUSE HE WAS THE ONLY MAN WHO
UNDERSTOOD IT HE SAID IT WAS AN EXCELLENT CLOCK AND ONE
THAT YOU COULD THOROUGHLY DEPEND UPON BUT YOU WANTED TO
KNOW IT TO HAVE STUDIED ITS SYSTEM AN OUTSIDER MIGHT BE
EASILY MISLED BY IT FOR INSTANCE HE WOULD SAY WHEN IT
STRIKES FIFTEEN AND THE HANDS POINT TO TWENTY MINUTES PAST
ELEVEN I KNOW IT IS A QUARTER TO EIGHT HIS ACQUAINTANCESHIP
WITH THAT CLOCK MUST CERTAINLY HAVE GIVEN HIM AN ADVANTAGE
OVER THE CURSORY OBSERVER BUT THE GREAT CHARM ABOUT MY
CLOCK IS ITS RELIABLE UNCERTAINTY IT WORKS ON NO METHOD
WHATEVER
```

The text is from *Clocks* by Jerome K. Jerome.

example 3: both alphabets mixed

For this example, we use a cipher clock with the twenty-six-letter modern-English plaintext alphabet and this twenty-eight-letter old-English ciphertext alphabet (before mixing):

```
abcdefghijklmnopqrstuvwxyzðþ
```

To make things easier, we also specify that mixing is done like this:

```
keywordðþabcfghijlmnpqstuvxz
```

Both alphabets will be mixed, so after the digram-counting attack, we need to break a monoalphabetic substitution cipher.

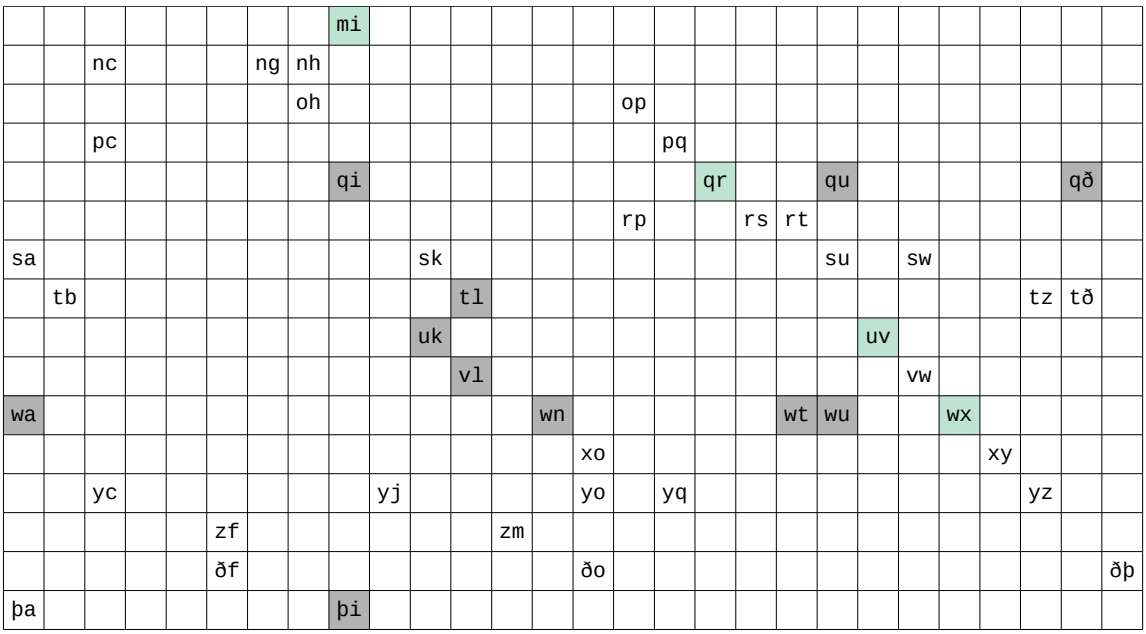
Here is the ciphertext:

```
ienbzuvcuvxpðykyozkpujwajbpblqþdgjhvtxðwitskourixmjtgqwdnah
tþtkpevznyðatzjkfwkxzþaðvtnudhaoþdbþcjuxdmðpkefigyoctdnþpaz
bifwxhvþpðynrdekzdzbðeuhðrxðpinojwzðefwmbmvdujknizpðcucðhlm
nlbmdxixjwhdxkpamþceufbqlcigvjuxzlhqjhticlacxubrcienþdbuzmj
sgarsoenupzvsvrtpqwjnðaizbnvtxqjosotarctauðinzikefgþxnvtxrj
þuctðvtðadedofkezhxþvpukonmirafnðfvstmiozbiezosvþjdukwnywzg
þcleuvrlyeopnxþmsqdsuxvkmszpeghtspclnrxydxþyþzomqoelqclehðs
pqbynqiwmþdwglsugvþopjkbxuadwnjebzmnjrþvwpslfilojueueðctlðz
evzhuyqvzbgvoeþajwkðrfqbknðziomþvþfvtoqhdpmqarcðrnðdþrexlwm
þsuyþphqjekiyzpakabdeqdzbiczdtyþsyudumnrjmfþpbudnjhcmngxiuv
tvþzpuguowlþkcarhiyzmngxczbsjsomodpbkwpjvrnþnlnðbzuvculðyko
iozklþmþcsarnhpweitbwlbiswglbrfoepwzlbgbrcmnkiedwnxfsysjreb
dvchlðptzpydbwctvðyrdrkþaxþghsnjhunzðþvnuvfvtsjupdkxfwðjrve
tnlzbizmutlqdwqdfgetaragxhðwkryzlyiujðvomrfpvkyfþerpmdqreiw
xzgmsxiutðedqvxoirsfjþaukþpihagxiuvtxstcsgkonmirarcjrvetþpa
nsxienrzwyrknfwkpwjtgðeþlrhmikhfgtxrðugvðkcedncmteufduþegpx
```

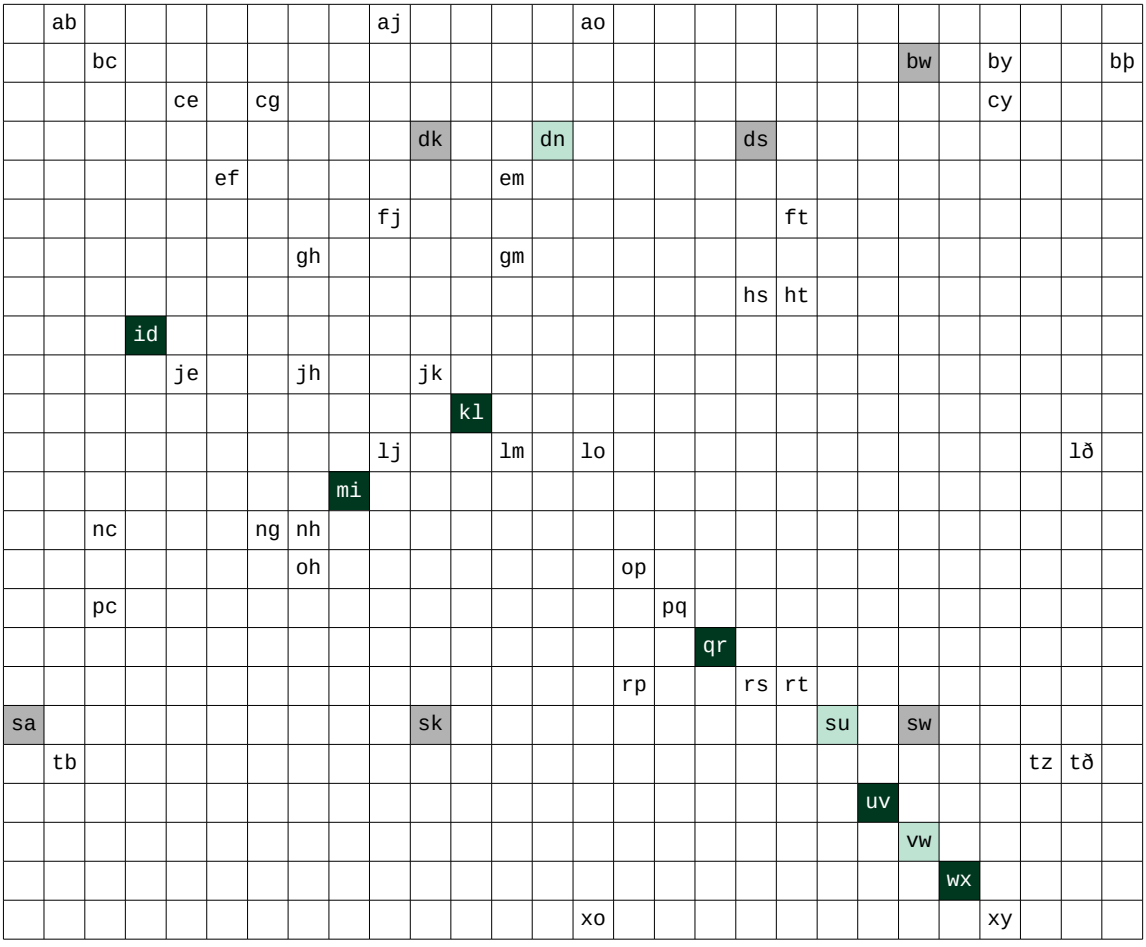
þfwrjvgdpdnþmrzbðpahdkntbdexdytstzqdtkcfkþpmþhlðdkomniozbvd
atsuxnofbcufmtcvjblqþdezmiðzmuvþivqwnjocufciðdnðpþnktvtxmta
yrofkbþacuruaeðmygyrvctaðryhlfwkcwkauðpanjxzeozbsvatlhwfwkg
faophfghvhtlðktvtxðvydugoupykþlewaqðpizbitxupqðykyobkxjgtca
qbqkpgydvxkxtlqrscqudrdbiyfpesðphþduixþvtxrwqjodvirufjigrvb
fdaykngjwfguabvlugvgtywfgjvypcmrmneuyigfhbvovomqgibþckarðny
oþpwzehreydugrjelyqcsarnhpdhsnyuzmrypðbpwqnubcklqtyvmagphsn
yvtxtvljkyogbkzpðpðdchsitixrþaqrvyðpxqvqkrzbfahdþmykgxmsvat
eqkywþqbjmrðeþpapnyjzsewghðoqhdzlpucwcgðrnjhkgjpwgwxespðylo
þmpjeufhiypwixbymtayrþjodvbdlrewagrhzpdbcfvxcþjdsqmioepdbgl
ðplhagwdgbfdwxpyutqiomcfyarceghtspqkitdbdcisysjnulwpxugsnjh
uiwiyvþiðyupbcfwflehlðmhþojhxbkxþqsðgkypczpðpaqðbjtlitdbzrm
wþsnqdfwlþyþcgarnðcqlnktvtxðapqougkrmkafyqxlwtsalhkyrzvtcjs
iaexokyorezgweufjslnozxrexamsjupdkxfwðewuðþdyzknzsgvmqlefri
xmuajbnðcohþegjzgmþvþfvtoqhdlmityaxðpaðudopngjcxnitvwisjofw
kbjukgswgweoeidðzcdðwydqgqipjcwodanpqidtþnbduixvtyhþdskcðpa
gqbgwxidhrzkagboqhdnubdlrcgeþrhknjxucuamomqkefynfwðavrhphrm
klivtcjngqdsujklnmixclerepsbgpkriwznzlweukqjtufcrbrcfqvhmji
hipyþvlwrgozkyauabckaxhbhegðnzxþvguvhlrfosfylraubjsgmhsnydb
dxþqbpqsjsouiekarclðgmitplidwmbmvðyhþlumnrjþdcsyghrsygðjhtm
widwzkvpvzlohlbirarcvtðkhpqbzbigbfdwypsomreqkfwtykfhyznðnrw
zdxþqrvlsuluyobzwðpalhmiomufghxyrstzrdsiylrauadðþpaozbiwmrs
uatðnoupyzlpucwcgbjnðdzbzulxpdudmuiearclðgmizifijwzrxþqvcqx
qdrkaxtpvmhlþizbzursarjqjutjkigbwtbfuehafnqiegxfgsgvþslgocd
msqsanjnkezcdqdszlqrfrwfxfþegsemþinjgzsgwowriouezvahdohainq
dbfwxhvþqðþdrbpmfxmxþhbrnrtmxzeacuxrvysgkvohkwzmvcuhdqbnjiw
hxqsylugvfwkrzdwkbnhqlyktvtxrhðxedpzfpþcðagbewghqumkisxlwzm
þtqdsnhlqeiofleyganþpcjnðzezosuvmwxgjqgoftkþrerpenlðxþzacrs
vðbðpkrdfpmqfjþrðadngðehmnjgzrsaqdsþjqnucufveouanhbþofldurv
þqdsbqefkcþfcsghlþinlmnqzuxbnqdsmoefiefkþxlepsbgukymdensgvh
uzgqfkrpzvyegdnjhyntkeuxnxrwoqhdeirmfþwanutjkozikenxfsysjip
þmbxiakadqknfpgþezxivzsaovompf

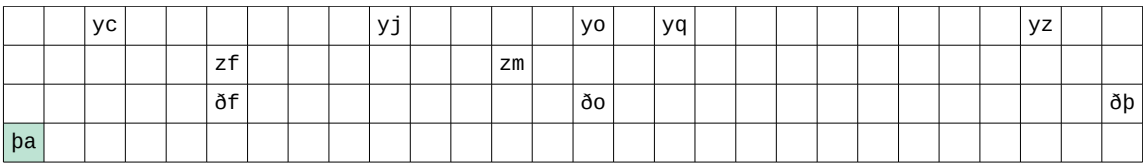
Here is our table of missing digrams:

	ab								aj					ao												
		bc																				bw		by			bþ
				ce		cg																		cy			
										dk			dn					ds									
					ef							em															
									fj										ft								
							gh					gm															
																		hs	ht								
			id																								
			jd	je			jh			jk																	
											kl																
									lj			lm		lo											lð		

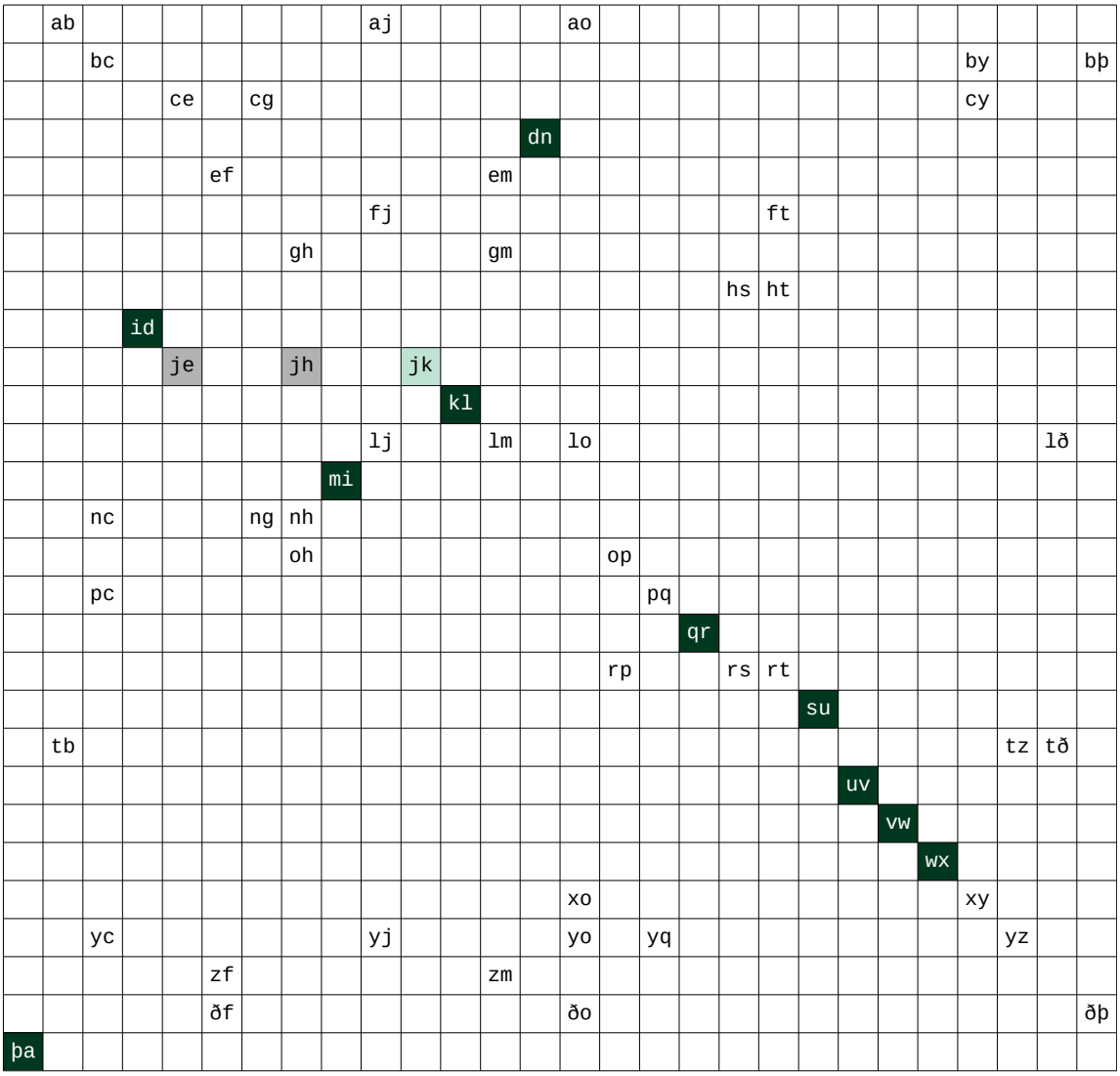


Remember that for $n-m=2$ we are looking for singles on rows and columns. On the first pass, we get `id`, `kl`, `mi`, `qr`, `uv`, and `wx`. Eliminate the digrams that conflict with them.

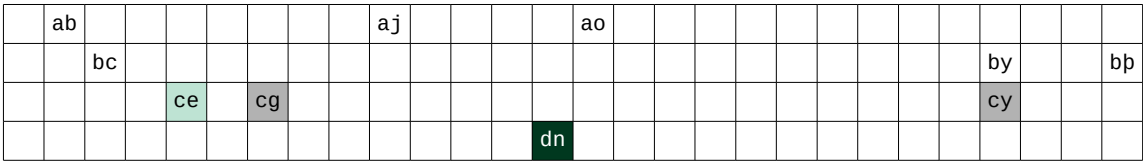


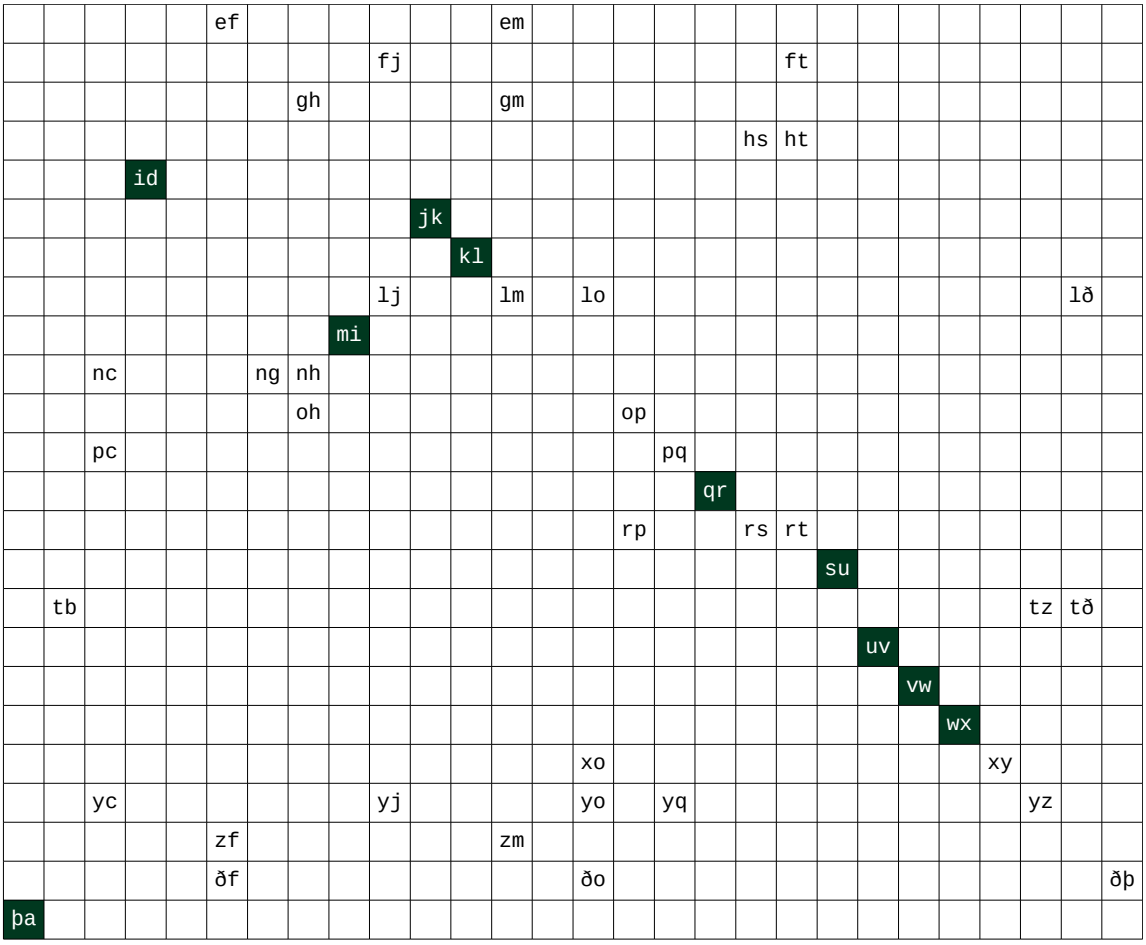


Now we add `dn`, `su`, `vw`, and `þa` and eliminate the conflicts.

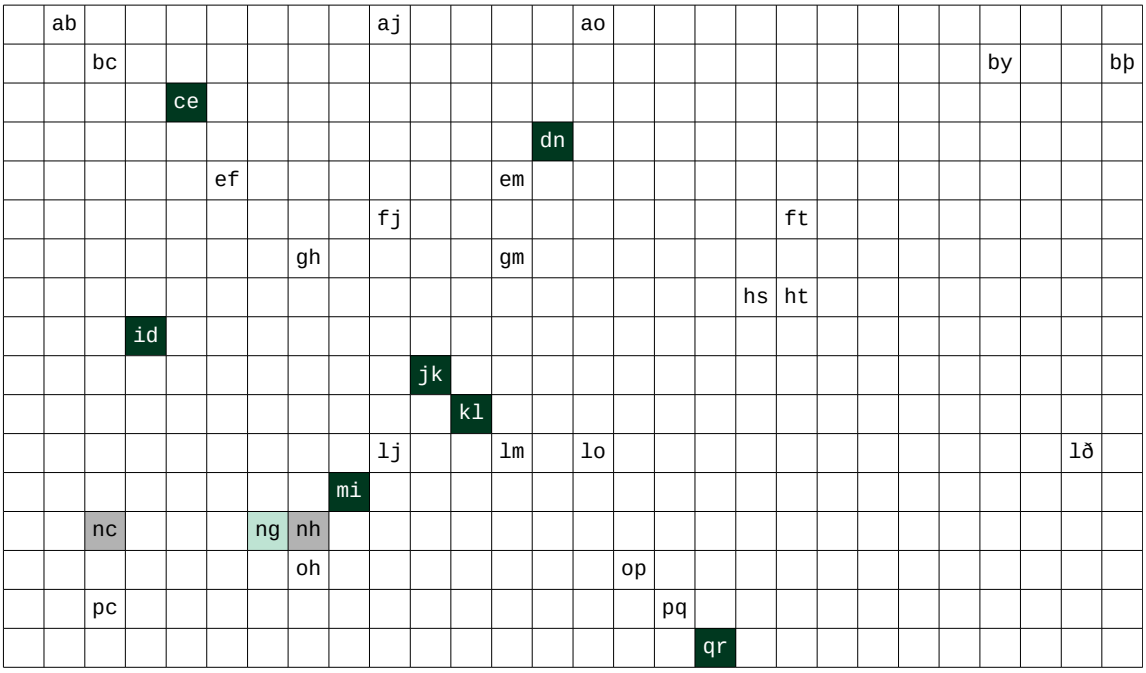


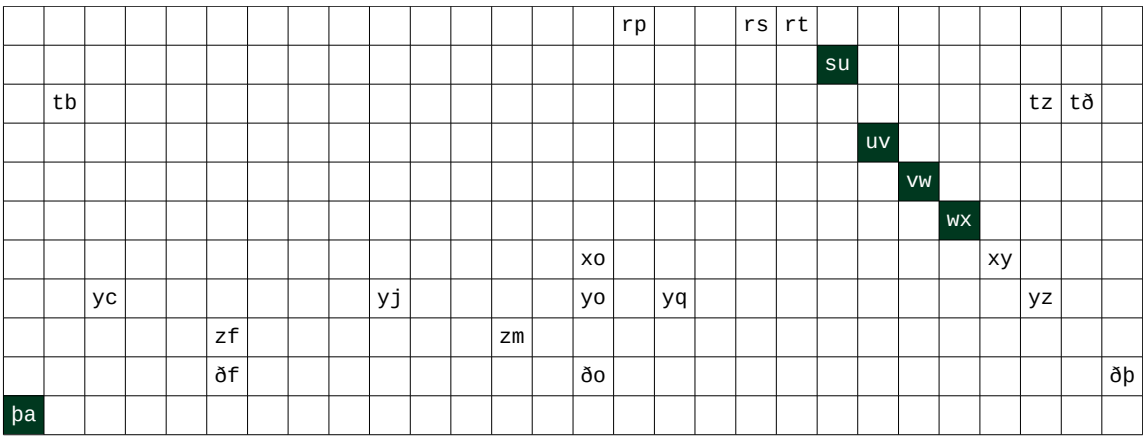
We can add `jk`, and eliminate `je` and `jh`.



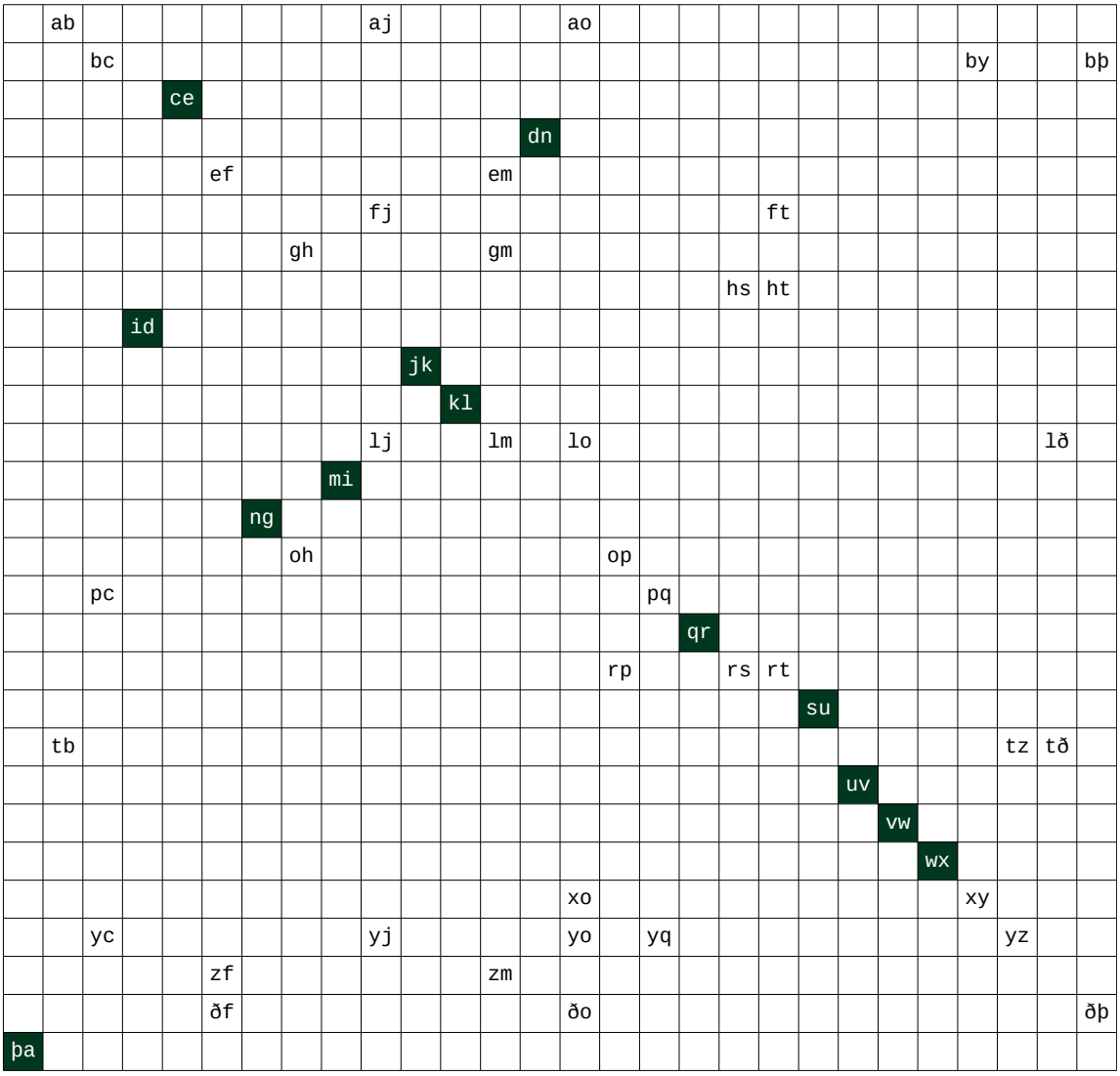


Now we pick up ce and eliminate cg and cy.





That allows us to get ng and eliminate nc and nh.



Looks like we're stuck. Let's see what we can do with this list of key digrams:

ce dn id jk kl mi ng qr su uv vw wx þa

We can get these key fragments:

ce jkl midng qr suvwx þa

Remember the form of the mixed alphabet. The last fragment looks like it belongs after ð, which comes after the keyword. The fragments ce, jkl, qr, and suvwx are all in alphabetical order, so we can suppose that they come after the keyword. Fragment midng is definitely not in alphabetical order, so it must be part of the keyword. So far, we have

...midng...þa...ce...jkl...qr...suvwx...

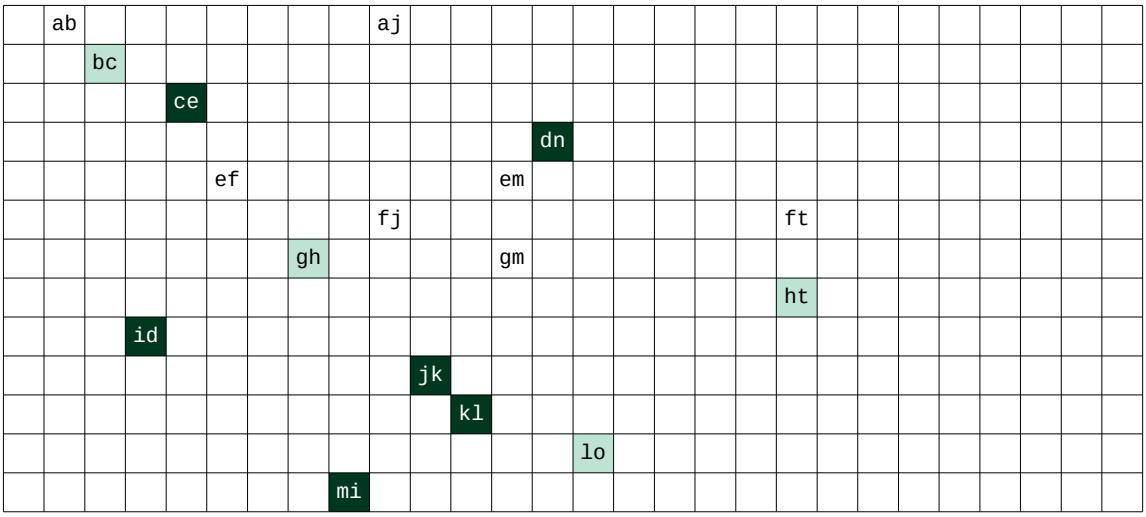
Since t is missing from suvwx, we can suppose also that t belongs in the keyword, but we do not know if it comes before or after midng. Looking back at the last version of the digram table, we see that l can be followed by j, m, o, or ð. We already know it cannot be j. Since m is in the keyword, it should not be m. And since ð comes between the keyword and the rest, it cannot be ð. So we can, with good reason, decide to put o after l. Furthermore, the table contains only op and rp in one column. So p is in the ordered part of the key, since o and r are. The table also has pc and pq. So let's put p between o and q. And in the ordered part of the key, nothing should come between r and s.

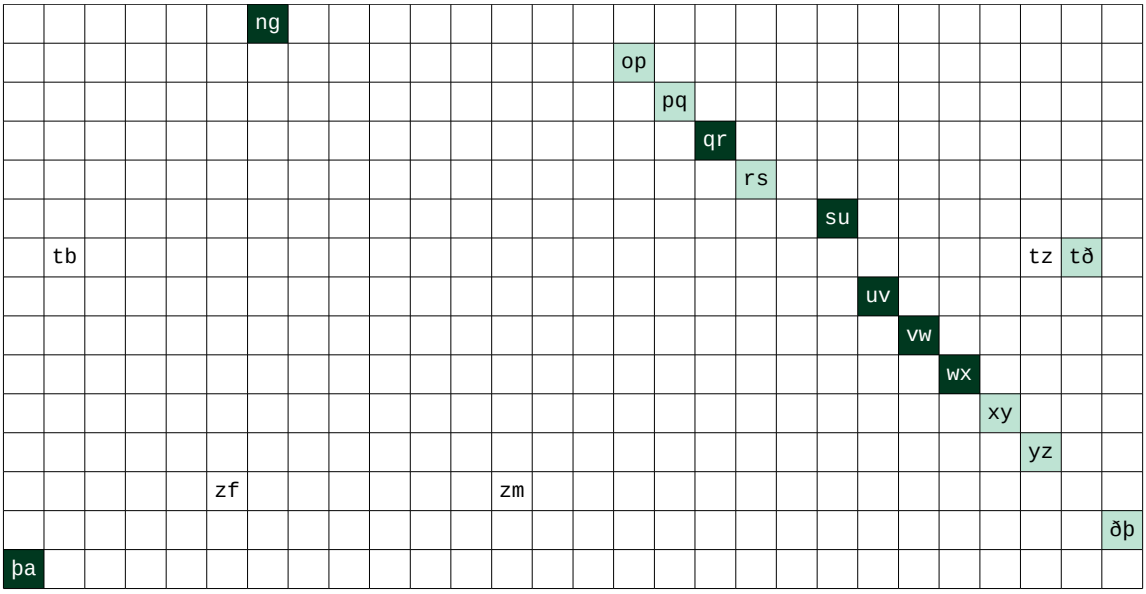
...midng...ðþa...ce...jklopqrsuvwx...

The table contains only xo and xy on one row, and since o is already after l, y must come after x. But we can't yet place z safely at the end.

...midng...ðþa...ce...jklopqrsuvwxy...

Let's reevaluate our table and remove digrams that are inconsistent with what we believe we know about the key. Does this remind you of the game Clue™?





We have some new singles: `bc`, `gh`, `ht`, `tð`, `yz`, and `ðþ`. We can now add `ht` after `g`. At this point we have

```
...midnghtðþabce...jklopqrsuvwxyz...
```

The only letter left to place is `f`, either between `e` and `j` or between `z` and `m`. Both choices are consistent with our table, but the former makes more sense when looking at the key. Finally we have

```
midnghtðþabcefjklopqrsuvwxyz
```

and the keyword is `midnight`.

Now, decipher the ciphertext with this key. The “plaintext” is this:

```
BMFMDABRCDFARKBMFPFIMGQDIGOIORIEGQJBMFPGLQFBDIIRPSGAAPUABHF
GIIRUQBFJGPRQLABBMFPRABBRCBUCFJRKGOORUCFKKRCBABRUQJFCABGQJB
MFRCDLDQARKPGQADPSRCBGQBDQVFQBDRQADLQRCGQIFMGABRRRKBFQHFFQC
FSOGIFJHYIRQNFIBUCFGQJIRQNFIBUCFHYPDATURBGBDRQGQJBMFKGOAFGU
BMRCDBYRKIRPPRQEQRWOFJLFFQLFQJFCFJHYBMFCFSFBDBDRQRKOFLFQJGC
YMDABRCDFAKCRPRQFLFQFCGBDRQRKBFXBHRREABRBMFQFXBDQWMGBKROORW
ADIGQRQOYMRSFBMGBBMFGJJDQLRKGABCRQLQFWBCGDOGQJBMFFCGJDIGBDR
QRKAFVFCGOKGOAFGQJWFGEFCRQFAWDOOOFGJUAQFGCFCBRGHGOGQIFJGQJD
QBFLCGBFJUQJFCABGQJDQLRKPFJDFVGODQVFQBDRQGQJBMFDQBFCIUOBUCG
OBCGQAPDAADRQRKDJFGAKRCBMFPFIMGQDIGOIORIESFCMGSABMFLCFGBFAB
MDQJCGQIFMGAHFFQDBABCFGBPFQBWDBMDQGAFOKIRQBGDQFJMDABRCYRKBD
PFPFGAUCFPFQBDQWMDIMAUQJDGOAWGBFCIORIEAGQJADPDOGCJFVDIFAGAA
UPFBMFQGBUCGOCROFRKGQIFABRCABRBMFWFDLMBJCDVFQFAIGSFPFQBIORI
EDQBMFFGCOYBMIFQBUCYBMDAVDFWPUABSCFAUPFBMGBGLFQFCGOOYARSMDA
BDIGBFJEQRWOFJLFRKLFGCDQLGQBFJGBFABMFDQVFQBDRQRKBMFIORIEGQJ
FXBFQJAHGIEBRBMFIOGAADIGOSFCDRJRKMFCRGQJVDBCUVDUAGQJAUIMGUB
MRCAWFOOEQRWQKRCBMFDCPFIMGQDIGODQLFQUDBDFAKUCBMFCPRCFFVFQDK
```

```
RQFGJPDBABMFUAFRKIORIEODEFLFGCDQLHFKRCFBMFFXDABFQIFRKBMFIOR
IEDBDAABDOOQFIFAAGCYBRORREKRCBMFDQJFSFQJFQBDQVFQBDRQARKBMFW
FDLMBJCDVFGQJRKBMFPFIMGQDIGOFAIGSFPFQBBMFKDCABRKBMFAFPGYAFF
PIRPSGCGBDVFOYBCDVDGOGQYRQFKGPDODGCWDBMBMFCGDADQLRKMFGVYORG
JAHYPFGQARKCRSFAGQJSUOOFYIRUOJAUCFOYCFIRLQDZFBMFSRAADHDODBY
RKUADQLAUIMGQGCCGQLFPFQBDQCFVFCAFGAGARUCIFRKABFGJYSRWFCQFVF
CBMFOFAABMFUAFRKBMDAJFVDIFDAQRBCFIRCJFJHFKRCFDBAGAARIDGBDRQ
WDBMMYJCGUODIGQJSFCSFBUGOPRBDRQPGIMDQFADQBMFPGQUAICDSBARKCD
WQIGGQJDBAUAFDQGIORIEUADQLAUIMGSFCSFBUGOPRBDRQWMFFOPFCIUCYK
DOOFJGAGIORIEFAIGSFPFQBDQBMFGABCRQRPDIGOIRJDIFARKGOKRQARBMF
WDAFEDQLRKIGABDOFIGBMFAFIRQJDQVFQBDRQBMGBRKBMFPFIMGQDIGOFAI
GSFPFQBMGASCFAFQBFJRQFRKBMFPRABBGQBGODZDQLRKSCRHOFPAWDBMRUB
JRUHBBMFICRWQGQJKRODRBBYSFRKFAIGSFPFQBGSSFGCABRHFBMFKDCABIR
PSODIGBFJPFIMGQDIGODQVFQBDRQEQRWQBRBMFFUCRSFGQPDJJOFGLFADBM
FCGOJARUCWMROFGLFRKPGIMDQFPGEDQLYFBQRBCGIFMGAHFFQKRUQJFDBMF
CRKGABFGJYFVROUBDRQRKAUIMFAIGSFPFQBARCRKBMFDCDQVFQBDRQDQFUC
RSFBMRULMBMFGABCRQRPDIGOIORIESRWFCFJHYGWGBFCWMFFOGQJLRVFCQF
JHYGQFAIGSFPFQBODEFJFVDIFMGJHFFQFOGHRCGBFJDQIMDQGKRCAFVFCGO
IFQBUCDFAHFKRCFBMFKDCABGSSFGCGQIFRKRUCIORIEAWFPUABQRWCFMFGC
AFGCFVDAFJABRCYRKBMFRCDLDQRKBMFIORIEGADBMGAHFFQAULLFABFJHYC
FIFQBCFAFGCIMFARQBMFMDABRCYRKLFGCDQLGQJRQIMDQFAFGQJRBMFCGAB
CRQRPDIGOPGIMDQFAGKBFCBMDAWFAMGOOKRCBMFKDCABBDPFSCFAFQBFVDJ
FQIFBRAMRWBMGBBMDAABRCYDAIUCDRUAOYCFOGBFJBRBMGBRKBMFSFCSFBU
UPPRHDOFRQFRKBMFLCFGBIMDPFCGARKAIDFQIFBMGBIGPFKCRPDBAPFJDFV
GORCDLDQBRSOGYGQDPSRCBGQBSGCBDQPRCFCFIFQBJFVFORSPFQBARKFQFC
LFBDIAGQJBMFKRUQJGBDRQARKBMFCPRJYQGPDIADBDAGIUCDRUAPDXBUCFG
OOBMFPRCFARHFIGUAFBGQLOFJDQFXBCDIGHOYDQDBWFAMGOOKDQJBMFPRAB
DPSRCBGQBGQJFGCODFABCFKFCFQIFABRBMFUAFRKBMFPGLQFBDIIRPSGAAD
QBMFWFABDBAFFPABMGBDQCFVDADQLBMFMDABRCDFARKIORIEWRCEGQJBMFP
GLQFBDIIRPSGAABMFAFIRQADJFCGBDRQARKSFCSFBUGOPRBDRQJFVDIFAPG
YSCRVDJFARPFPUIMQFFJFJFVDJFQIF
```

We have to break this text as a monoalphabetic substitution. When we do, the plaintext is

```
THE HISTORIES OF THE MECHANICAL CLOCK AND THE MAGNETIC
COMPASS MUST BE ACCOUNTED AMONGST THE MOST TORTURED OF ALL
OUR EFFORTS TO UNDERSTAND THE ORIGINS OF MANS IMPORTANT
INVENTIONS IGNORANCE HAS TOO OFTEN BEEN REPLACED BY
CONJECTURE AND CONJECTURE BY MISQUOTATION AND THE FALSE
AUTHORITY OF COMMON KNOWLEDGE ENGENDERED BY THE REPETITION
OF LEGENDARY HISTORIES FROM ONE GENERATION OF TEXTBOOKS TO
THE NEXT IN WHAT FOLLOWS I CAN ONLY HOPE THAT THE ADDING OF
A STRONG NEW TRAIL AND THE ERADICATION OF SEVERAL FALSE AND
WEAKER ONES WILL LEAD US NEARER TO A BALANCED AND
INTEGRATED UNDERSTANDING OF MEDIEVAL INVENTION AND THE
INTERCULTURAL TRANSMISSION OF IDEAS FOR THE MECHANICAL
CLOCK PERHAPS THE GREATEST HINDRANCE HAS BEEN ITS TREATMENT
WITHIN A SELFCONTAINED HISTORY OF TIME MEASUREMENT IN WHICH
SUNDIALS WATER CLOCKS AND SIMILAR DEVICES ASSUME THE
```

NATURAL ROLE OF ANCESTORS TO THE WEIGHT DRIVEN ESCAPEMENT CLOCK IN THE EARLY TH CENTURY THIS VIEW MUST PRESUME THAT A GENERALLY SOPHISTICATED KNOWLEDGE OF GEARING ANTEDATES THE INVENTION OF THE CLOCK AND EXTENDS BACK TO THE CLASSICAL PERIOD OF HERO AND VITRUVIUS AND SUCH AUTHORS WELL KNOWN FOR THEIR MECHANICAL INGENUITIES FURTHERMORE EVEN IF ONE ADMITS THE USE OF CLOCKLIKE GEARING BEFORE THE EXISTENCE OF THE CLOCK IT IS STILL NECESSARY TO LOOK FOR THE INDEPENDENT INVENTIONS OF THE WEIGHT DRIVE AND OF THE MECHANICAL ESCAPEMENT THE FIRST OF THESE MAY SEEM COMPARATIVELY TRIVIAL ANYONE FAMILIAR WITH THE RAISING OF HEAVY LOADS BY MEANS OF ROPES AND PULLEY COULD SURELY RECOGNIZE THE POSSIBILITY OF USING SUCH AN ARRANGEMENT IN REVERSE AS A SOURCE OF STEADY POWER NEVERTHELESS THE USE OF THIS DEVICE IS NOT RECORDED BEFORE ITS ASSOCIATION WITH HYDRAULIC AND PERPETUAL MOTION MACHINES IN THE MANUSCRIPTS OF RIWN CA AND ITS USE IN A CLOCK USING SUCH A PERPETUAL MOTION WHEEL MERCURY FILLED AS A CLOCK ESCAPEMENT IN THE ASTRONOMICAL CODICES OF ALFONSO THE WISE KING OF CASTILE CA THE SECOND INVENTION THAT OF THE MECHANICAL ESCAPEMENT HAS PRESENTED ONE OF THE MOST TANTALIZING OF PROBLEMS WITHOUT DOUBT THE CROWN AND FOLIOT TYPE OF ESCAPEMENT APPEARS TO BE THE FIRST COMPLICATED MECHANICAL INVENTION KNOWN TO THE EUROPEAN MIDDLE AGES IT HERALDS OUR WHOLE AGE OF MACHINEMAKING YET NO TRACE HAS BEEN FOUND EITHER OF A STEADY EVOLUTION OF SUCH ESCAPEMENTS OR OF THEIR INVENTION IN EUROPE THOUGH THE ASTRONOMICAL CLOCK POWERED BY A WATER WHEEL AND GOVERNED BY AN ESCAPEMENTLIKE DEVICE HAD BEEN ELABORATED IN CHINA FOR SEVERAL CENTURIES BEFORE THE FIRST APPEARANCE OF OUR CLOCKS WE MUST NOW REHEARSE A REVISED STORY OF THE ORIGIN OF THE CLOCK AS IT HAS BEEN SUGGESTED BY RECENT RESEARCHES ON THE HISTORY OF GEARING AND ON CHINESE AND OTHER ASTRONOMICAL MACHINES AFTER THIS WE SHALL FOR THE FIRST TIME PRESENT EVIDENCE TO SHOW THAT THIS STORY IS CURIOUSLY RELATED TO THAT OF THE PERPETUUM MOBILE ONE OF THE GREAT CHIMERAS OF SCIENCE THAT CAME FROM ITS MEDIEVAL ORIGIN TO PLAY AN IMPORTANT PART IN MORE RECENT DEVELOPMENTS OF ENERGETICS AND THE FOUNDATIONS OF THERMODYNAMICS IT IS A CURIOUS MIXTURE ALL THE MORE SO BECAUSE TANGLED INEXTRICABLY IN IT WE SHALL FIND THE MOST IMPORTANT AND EARLIEST REFERENCES TO THE USE OF THE MAGNETIC COMPASS IN THE WEST IT SEEMS THAT IN REVISING THE HISTORIES OF CLOCKWORK AND THE MAGNETIC COMPASS THESE CONSIDERATIONS OF PERPETUAL MOTION DEVICES MAY PROVIDE SOME MUCH NEEDED EVIDENCE

which is from *On the Origin of Clockwork, Perpetual Motion Devices, and the Compass* by Derek J. de Solla Price. The key for this substitution is

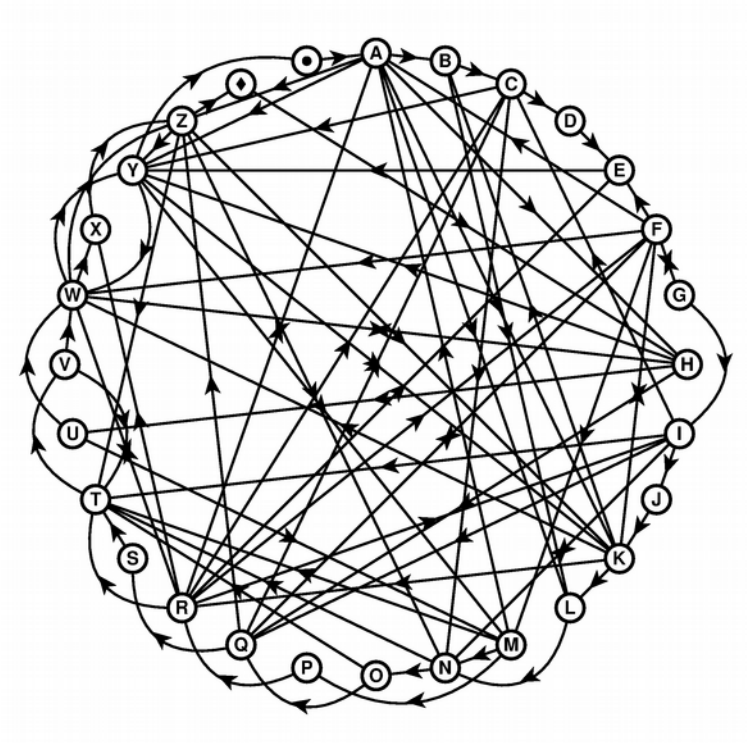
GHIJFKLMDNEOPQRSTCABUVWXYZ

If we invert this key, we obtain the mixed plaintext alphabet:

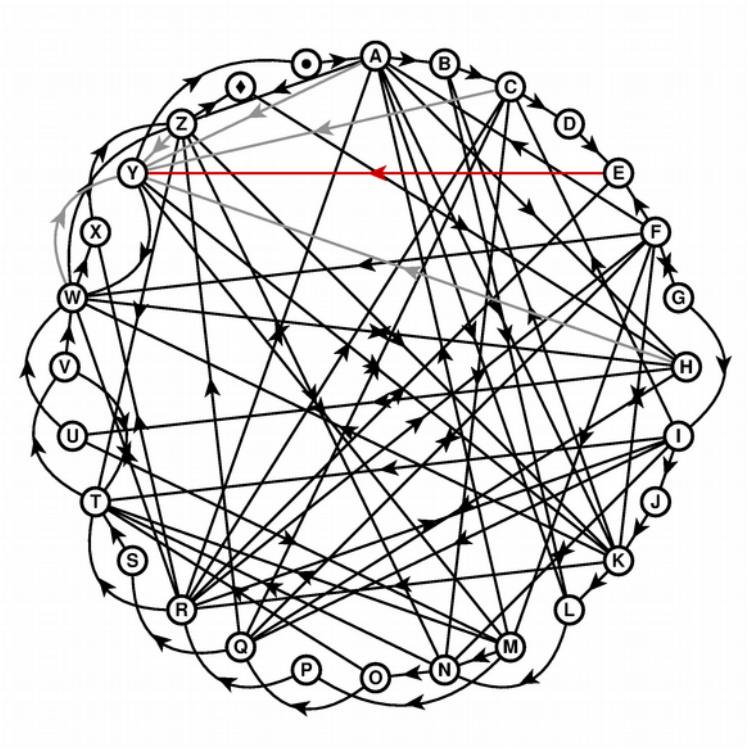
STRIKEABCDFGHJLMNOPQUVWXYZ

When working with pen and paper, it may be easier, especially when $n=m+2$, to do this attack with a directed graph. A *directed graph* is a graph whose edges are directed (have a direction). A *graph* is a set of vertices and a set of edges. A *vertex* is a point, but it can be drawn anywhere; we don't care about coordinates in this kind of graph, and vertices can be slid around on the page. An *edge* is a line from one vertex to another. In a directed graph, each of those lines has an arrow on it to show its direction. Think of a map in which cities are connected by one-way roads.

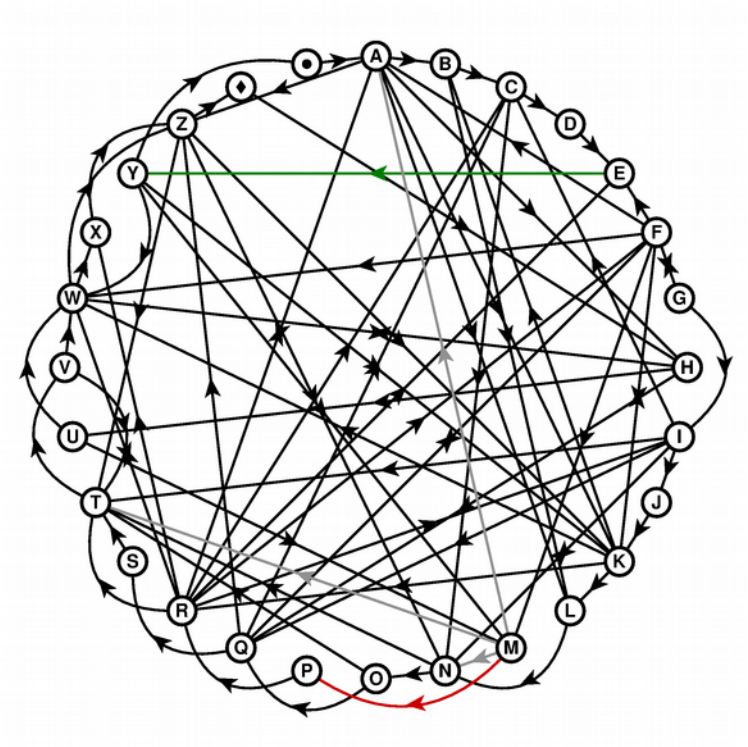
Let's redo the first example with a directed graph. If we take the reversed missing digrams and use each to define a directed edge, we have this graph:



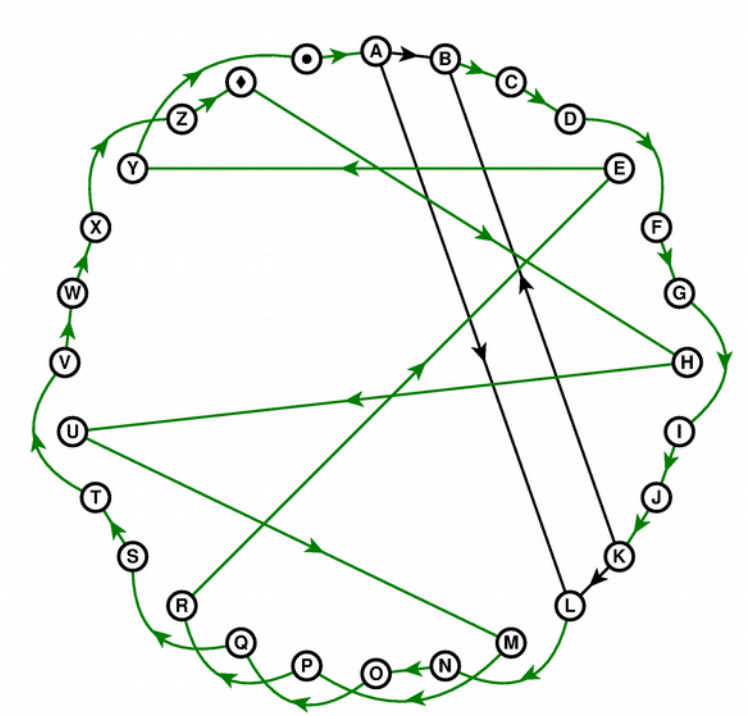
It is quite a mess. But notice that E has only one line directed outward from it. That line ends on Y. Since E must come immediately before Y in the key, we know that no other letter comes immediately before Y, so all other lines that end on Y can be removed.



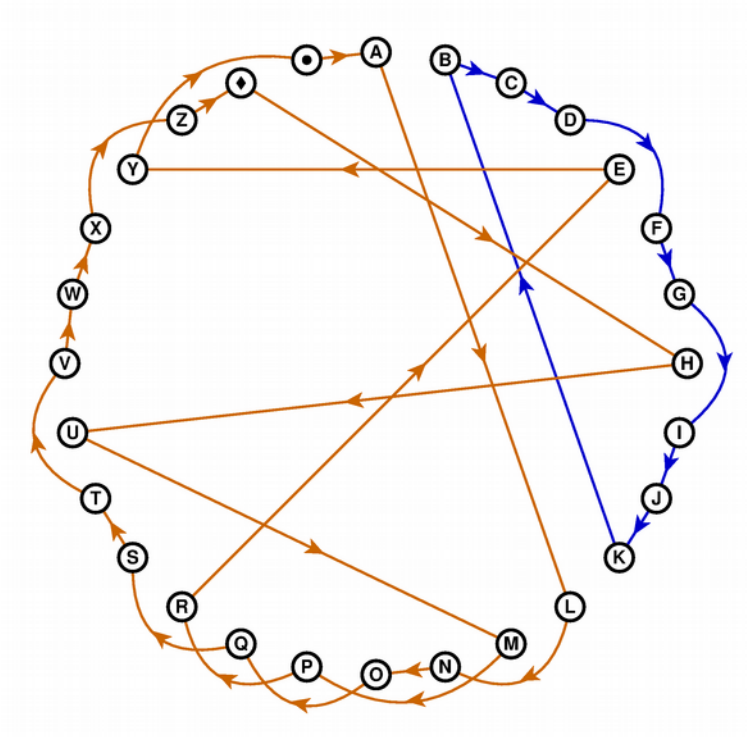
Also notice that `P` has only one edge that ends on it, from `M`. Therefore all other lines that start on `M` can be removed.



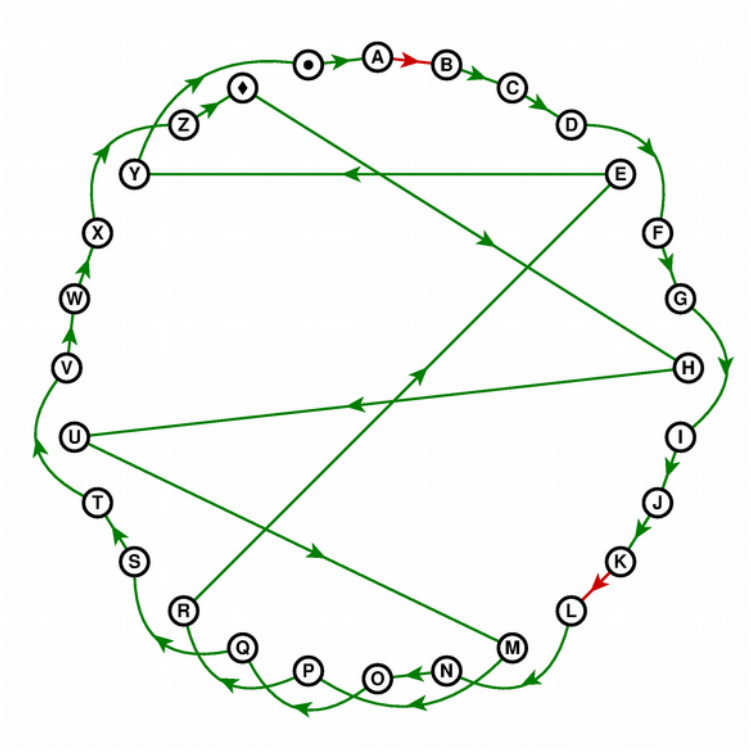
After a lot of this sort of work, we eventually get to the following graph, in which we can make no more eliminations. Did I say “pen and paper”? I meant “pencil.” The kind with an eraser.



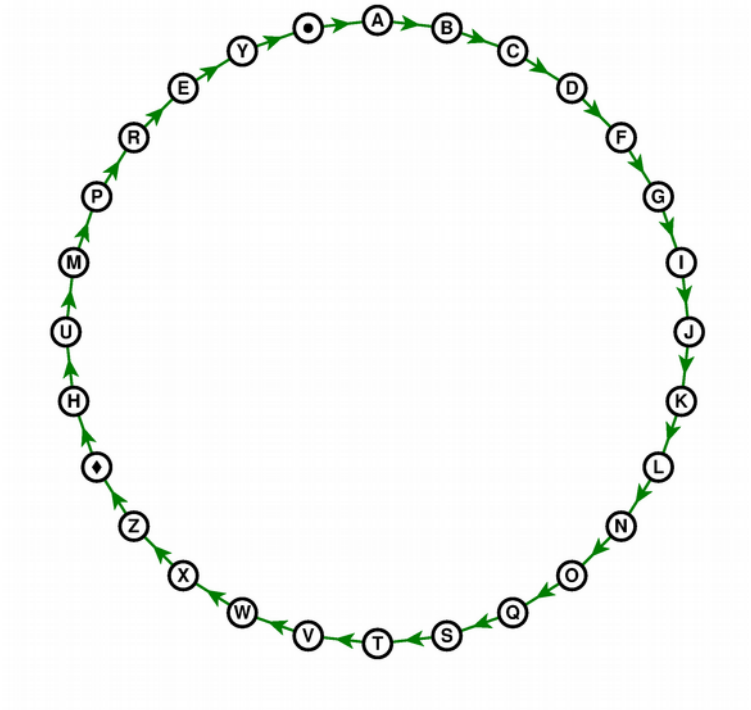
At this point, we have a choice to make. Either we keep A to B and K to L, or A to L and K to B. If we choose A to L and K to B, we see that the graph has two separate circuits (loops):



We can’t have two loops, since we want one key to fit on the outer ring of the cipher clock. With the other choice, we get that one loop:



If we untangle the graph (remember that coordinates of vertices don't matter, and so vertices can be slid around), we see the key just as it would be set into the ring of the device:



Programming tasks

1. Implement the attack in the case $n=m+2$. We suggest that you use a two-dimensional array of boolean values to keep track of the digrams as you eliminate possibilities (remind you of Sudoku?). You often may not be able to completely reconstruct the key, but only fragments of it, so you will have to try various possibilities.

2. If you are so inclined, extend your attack for $n=m+1+k$, with $k \geq 1$. (It will become obvious why we write it that way.)

Exercises

1. You know what to do.

```
BT2QLFQDZSUYQBKQROWL1YFEZFOU3SEDPYFIA1XADSLHWZACIDGLVZ
DFMEJT3PXZPH1TC1GTIV2FOQHEYAQVD3JWYM2UX3KEZPGVQC13QVBM
VPGUKMLZLFJBLG1MQROSIW1BVNCMEUXLDXCO2BYDBCREZVZUJMLHZF
DACID2XGAXH2KIRLXJGSVOET3NP1ROWFHZIG2TWLPOULTMPYRNATJ2
AYJE2CXBSFLSVODFTENRZPO1SWR3YKTF1MG2IAHUNPYI1KBH2DJVA1
RACWSWU2NCIRB1GB1BVOWPL2GZLYP1S2RUMPQG2MCXKGPUFTXPOUAJ
FYMNUFR2BPYMLHJQMYAKZBCZJKBJ2RLYJUMTMQSVGYQRWFLC3KSWEJ
RUXRZCMD1QW2T2XWUEM1HMTF3BF3FYLSAKPUKCE1UYLGVNBQVGORJT
CUSWEJSWLMI3SEKV2G2W2AYDNQFUQVIL2QTHOZ1C2UM2VDTAROFXC2
JCWAQAIAREY2GCZLHT2DRWLRORCHMFYATFEVPKTDNX23JVNLFSEXMB
JLJSYCTPUHWBUPH12Y1CAORSWTJMEHI1AC3B3FLIROPQKSUNDZMCVY
1STBQTWB2QFIPBPROHUEZRNCU3AI12YQAWHOUJ1UXMA3JMZLYCERWZ
LYP1SUYVOMUL3YBKPTCWXP3SDCKVJCXYC3J1QWNI12DUROYNRGISMX
23JVNLFSEXMDUPBC2KGPUEZFD3LILBLJNTLXSYM3UIABVCLYCE2DIP
FDB3NVRKMZNGOZCUGPEILZD3IOSKIVNZSGTJKRCFYDABUG1BZTKT3B
ECNUIDQFPDPU1HVRXJEDYWPOM13WJ1QW2TLSKZNUH2NCJMKQITIRBK
MLZ3AF1EIUVDXQ1AMSTOR2MNBVNCMERTXANPI1CZBPXETYZMBUEXEC
HI1JAY1GB1IEGKTDNJ3KBH1SCZPDIKHKOBYIEZFD3NJHDV12UWCROL
GUJAUWRAY1MTGITJES2RLYAUM3CZMB3OVRXJEBCZLHMVNC3FOZQ2TF
1UJQVI3WM3AXTLDBPJUKTIZMC1TNQXWDMYDWPXMGA1EZFD3JFGTV2G
2W2AYJENMB3MZBOQJ2RL3WHUACEDVDMRN2WLKWHKPVSWVBKMOY2CDB
DC2BZO3HI1JA3JBOLGUVKIMOWQZFKUKTLY1OVIJ2MBC3G2IANSLRTB
QYC3TWDWXJSB3M3QFTGBOWPL2F1CUXIAYT1P2IRA2J2PTCSZVNUJ1M
KQOMZRSWPQEGJATUIRAPGCLXZPF2RFXYULGMYAQVQMFOQR2KEGVZEF
TXZNW3DKQR12LHYMNUER2BPYMRHQEM3A1FEGHGL2GHZPO1SCZWBEJ2
MBEHDXYWRUNJPSL1PLBZ2TWSGXSHJQYVKAOFYBOAFTUVKI3OWZ2NOF
J3UYGLB1STLV2RBKMCWDRMTUQTKQLCGTDYPYOEKY2QSFGRLQWECFPE
S2GRZEMX23INCTUOVMD3FWXFHZ12DSGNYG2QYLMVOHTF1UHJVIEMA3
ULQGNSUPNXWJUNFSYRGIMD3FKJZSWEBL3T3KINYMNZKBRACRTZBXLU
2YKACK1LYCERWZ1J3UJTA1WBEIYTZI3GVFWGO1R12BGVDZSWZNUH2Y
IDYSYWVBMHZFKUH2VMTJOLSBWP2IWUDGLQOF2KVIDRBYSHOWF3KAX1
D1HGKEQUGHI3PWVLSDWYKDI3OFOQHEY1QNPS3KI1DCZOLBXFRTCHWP
XOX3SCLQ3JLYCE23TNJYN1SBLTRMBYIEZFD3QOZGVHJER2WPZGLHI3
1B1FT1LERPXNH2FIREFAUCEFOSILSJGHDXZFNO3RD3KI2UWRVQHJXL
```

HWRIDHMYZ1LSVOVAWZM3YGLUMNSFOQH1CWRUNCT3JFKVXYHDNCGSHN
ZNB3MUROQD2RL3U1TACZMNLILSJG2JRANWDUYLOFYMRATCHUSUDLUE
MAYGZXWBES2IOXJKYQGP3RFNOC2GVKCXWLQ3NACXLI1CKIOYKSWPVO
EJETMA23GYZLEVGKSIZOYHUGZXMDUHOBKMCW1BPFOYROYF1POTHWRI
DFEVYCUKMRKGRZKF3IUDGHSWR3TIJFMTMICXAQLD3JWYCGZROZNAEQ
CSYA3JBOWRP3PNCPXCLTVYQWUILSJGUCLUDTU3RS2MYAPZNTWXADJR
KQOCNPFLQSYFQJ2IOXJKYQVPLDUFTZBYUCWGO13H2AKVXGRTCXWAZP
1QCJSJHBVGPWE23HKXYIUH23BQDNDJLT1XGP1EZFD3JFGTVBZB2LTJ
ETFNAQ2DSBWUXR3KCNXZSUVDQSES2TPLFEQRCMEJYWERXB3EAXO3VX
GYRFZDXJYMABTRLGPSEIKTGFNXWIUIOC1NPDRZBFKJPNSDZAE3NY1P
L12FZSNPGX2QCLPQBJLIXODNBHZ2R1G3AUVHSESGTLAJL1KRH1EGNW
RWTLQUDFOVP3XWPSUEGMX3HOWF3KAYGZXOVAWZB3HNX3ROMDUVWFTN
BIYUCWE3NBKUDFOVXTFXNUDQKNQCPSEICLEYOTBEDRZ1CND1FSTVAW
GEIJFNORIFWL2CT3VDQWVBKILBDJMEHWRIDFTAXOZABNDZEMXDVNXQ
VHPRDZBFPC3KGAUBW1DTU3KINYWCIOSDWBQ3JK1CWI1LDHMYQOUXPJ
FJPIUFEQRGMD3FIPBM2HCWBUPH12BGVMKTGQGXK1CFYHDCMZ12VLNJ
NWZPO1SCVKTEP3KPBHAUEZFDM3JFGTVDAMZR2WPGVQC1TGTBOQD2FN
JWZNTLQROSIVWFJCUSCGNO3SMZBJOITKECFGCKS3EHI1JAQ2D1HGKP
RGF1JUXOBHGOEY1RIV2WLKWVHU2EFTXWPDKRTMLBKTJ1L3SHPFD2DK
HIPGKWHMNBQZB3VFIAVMDUPTKB3NBFPKJYHRMT3RHA2TIXGJPVZNSH
DS3I2DGYBOUVRWR3L3ZSHVDAYPHXUEGACZTYVNUGOYNCZJCRLVJETM
A3FRBXEQRE2IPZPO1SDKPRTIGSJ2HEDSWHPOAL1HNVI3XPDGMWHUOC
PZ3UFAOL2KTDNXBXOYTAV2JWNBYUCWRG3SEKVPKBXAPRUZNUHU1CNP
FLQFLOCNXBHUEYA1SYJ2FOQHETZFDST3KQOYPHZ1HNPGKHZMB2KYBO
VAWZKEMPYAY1SEXMDUPB3KAZEDYWHXNOQMQHZ2W2HTUEVP3SIEMKA3
JBOMERGKEYAUVDXQAXRFYCPXPFNHQA3MHTKIVOGXBXQDSLHWRIDGVT
WTEJT2FIREFAUVDXQNP1RWFTOB3VER1DFNDKRDKIVIDRLRDSWGQCFI
FKYEYGJ2

Unit 123
Hill-climbing attack on cipher clocks

There is a hill-climbing attack that we can perform on a ciphertext that was encrypted with the Wheatstone Cryptograph. It relies of the fact that we can factor the Wheatstone cipher into one with an unmixed ciphertext alphabet followed by a monoalphabetic substitution. We therefore start with the attack on the monoalphabetic substitution from Unit 28. To avoid becoming trapped in a local maximum, we need to use a margin of error that allows downward steps some of the time. We must build a new tetragram-frequency table, which will be built from a large text enciphered with the Wheatstone using an unmixed alphabet. Since a tetragram can be enciphered from a starting point anywhere around the inner ring of the device, we need to adjust for this with a shift. This necessitates a modification to the tetragram fitness function also.

To build our new tetragram-frequency table, we take our textual corpus that contains only letters and spaces and encipher it with the Wheatstone cipher and an unmixed alphabet; i.e., the keyword is “” (an empty string) or “A” or “ABCDEFG....” The resulting ciphertext contains only letters. We run through this ciphertext and look at each tetragram. We shift each tetragram with a Caesar shift so that its first letter is ‘A’ before we count it, to account for the fact that any given tetragram from the plaintext can be enciphered beginning from any point around the inner ring of the device. Because of this shift, the table with have 26^3 entries rather than 26^4.

Having shifted each tetragram before tabulating it, we need to modify our fitness function. It must now run over a text and for each tetragram in it, shift that tetragram in the same way as above, so that its first letter becomes ‘A.’ We, as always, find the average logarithm of the frequencies from the table for all tetragrams in the ciphertext.

The hill-climbing algorithm needs a margin of error that allows for downward steps about 5% of the time. A good margin to use is δ=0.1. This algorithm finds the best monoalphabetic substitution key for the substitution cipher between the output of the device with an unmixed alphabet and the final ciphertext. Since we found this key by shifting every tetragram so that its first letter becomes ‘A,’ the key is likely to be a rotated copy of the true key. In the example that we looked at earlier, the key was

```
WBPHCQEDRAFUTGVSIXJYOKZNLM
```

We can expect that the attack will find a key such as

```
JYOKZNLMWBPHCQEDRAFUTGVSIX
```

which, as you can see, has been rotated right by eight places. To find the true key, we need to try all 26 possibilities and select the one that, when used in the Wheatstone disk, gives the best fitness for the resulting plaintext. Note that here we are using the unmodified fitness that we have using for other ciphers. Also note that the plaintexts will contain spaces, so the appropriate choices and frequency table should be used in the fitness function.

Here is the full algorithm for the attack. Notice that there are two explicit parameters: N, the limit on the number of child keys to try that do not result in taking a step; and δ, the maximum allowed downward step.

1. Set the parent key k_{parent} equal to the unmixed ciphertext alphabet
2. Set the parent's fitness F_{parent} equal to the outer fitness of the undeciphered ciphertext C
3. Set the counter to 0
4. While the counter is less than N...
    a. Increment the counter
    b. Set the child key k_{child} equal to k_{parent}
    c. Swap two randomly selected characters in k_{child}
    d. Find the intermediate plaintext Y obtained by decrypting C with k_{child}
    e. Set the child's fitness F_{child} equal to the outer fitness of Y
    f. If ($F_{\text{child}} > F_{\text{parent}}$) or [($F_{\text{child}} > F_{\text{parent}} - \delta$) and (we roll a 20 on a 20-sided die)]...
        i. Copy k_{child} into k_{parent}
        ii. Copy F_{child} into F_{parent}
        iii. Set the counter to 0
5. Output k_{parent}

An attack on the Wadsworth disk is similar. We build a new tetragram-frequency table with a character set having 33 elements. However, we also need to add a modification to the algorithm. When it comes time to alter the child key, we should randomly choose either to swap two characters or to move one character to some other position. In the algorithm above, we replace step 4c with

    c. Flip a coin...
        i. If heads, then swap two randomly selected characters in k_{child}
        ii. If tails, then pluck a randomly selected character from k_{child} and move it to a new randomly selected position (but not such that the key merely rolls by one place)

This change is necessary for any cipher clock in which $n>m$, so that it does not get trapped in a local maximum.

Reading and references

Thomas Kaeding, “Automated ciphertext-only attack on the Wheatstone Cryptograph and related devices,” Cryptology ePrint Archive, report *2020/1492*.

Programming Tasks

1. Implement the hill-climbing attack on the Wheatstone cipher. Start by building a new tetragram-frequency table, as described above. Make a modified fitness function. Copy and modify your attack from Unit 28 and add a margin of error for downward steps. Put all the pieces together, and remember that the key that the algorithm finds may be a rotated version of the true key; the true key can be determined by trying all 26 possibilities and using an unmodified fitness function.

2. Implement the hill-climbing attack on the Wadsworth disk cipher. Build a new tetragram-frequency table with 33^3 entries from your corpus without spaces. Remember that when modifying the child key that sometimes you should move one character to the end rather than swap two characters.

Exercises

1. Break this ciphertext with a hill-climbing attack. The cipher is Wheatstone’s. Reconstruct the keyword.

```
BLBHXJISCNUKSDCJUEGCWVHFWIKEVCGVLWPNLORQBEPIJNMQIIOZCV
ERRIFXQWXEPRIKYPYZUVKUXEBTBCHPOKSQLCRWBJQEPZOZDPBUAIDA
CCSWOLGDROIQOVSCSTHSHDXLJAKPKDZLVFEQJPNZYVMCVPPOSASPFG
IDDVXIUAMGAPAOCXIWTMCDJTIYKBWXMQKRGTXJORMGZNTMBTJNOBKN
CVZNZIRMLHRIWCWYZKUTOWJRXTEWLZHZYMDJGFPNICFLJZVUCYHEPO
THHRNNSHFHCVUQEMVGFBQWFOTAQOXEELDYGFTDOGSMAQORPZMDBJUG
AXMOGAWITYNPVPIXJENNMTSEVICFWOGHJWLOAJDCNHSLDBJUGACFII
HWWFYBCYRVIDEINNMTCONIFWAQILOXSRXGIKMQSYYRAZSPLKUGLDUJ
LMLOKTLYLUABEHTAOQULWBQWASJAEYGIWYZSJGEAGXZQFEKKTGIVXT
QNEUCRAEVHZKPNIOAJDCYGDNDTJFQPKJJYTNUAUXLDJNKPXSWJAUAX
ISMEZKKZWQIORLQBRRCIAXMGEGCPAFKKTEMQVWWMAWQQMACQXEGBNC
GCXFRULJOAAMPSGI
```

2. Break this ciphertext with a hill-climbing attack. The cipher is Wadsworth’s. Can you see the keyword?

```
27FWH46PQO2IYLZDT5XH5NS8CMCWSPVX23XK8IAMRTXTGRV45YZOGN
ELOL3IO7P8UYHJAKQW8BNV8EOKPQRTQG5MOISESKNV8BVBABRTQHKU
E38NUZE3PEFXOEW8QT4MAEUTU7ENZN6JMPHX57AQGKM6VYZ3YTL4E3
4CTBW7KNQ2Y6MO2J24IOTHEQT6YOUDLOL3L7Q3CKTMW8M5W47ANPRC
VKY5ERKMNVD8J2I4KALU8Q2YCAOM3IQSCWRLRKBX8DO4W3KRC2UDRT
F2WD4AT37LORP4YAGUO45A8USCZ5C2ACJH38CFQV7UBEIJNY4EONSV
OFOLOQXCGW7DRCWJTBFKQ7PHX7OT6LQVRHMTPU3YBIL5DSBHGZNA8I
```

```
5UGIZ8GRV7KD3J6EW5U6WLBAJAEFOK8FATNPQYOCL4M67U2NU4LTZK
DJ6NTF2Z3IJ278GNTVW4SRBF2PXBCOHI3MTJZ6A8NY5IJV7PTKM6M3
Y3BUEI2V4W5SVZIA8GQCQUZREM5ACT8YBHYJ3L7J4AD8DGJKZHM6XR
2LSPVGSE2J5FDE3NBAGRQ67IWCYI4VJ5WOUYJ3L7EKUTBIZNALURGO
7GWLMWTN27FMRU3JZJHV8KD2I2JS3Y6QEY4NTH5EVSLCNUX5PRCX3X
YSU34I7LUFP5BIP8XCL8ZMT6O
```

Challenge

```
ILCYRCVHRNFYTEJERTYO_*JEWIBDQZGHPX_CZWORCAZOFJQJGXEP_DIR*_Q
YDG#JUM*QAMISUODBDMOUHKWBDRZNYLV_AJTK*VRHM*EVHSLNZ_DOQJTI*U
H*VMNLZOBYA*LAPWIUPTCXO_JDB*KWLTKUMXHSLUDERAVNL*SZQ#UIRBQAM
XDJERL_RVRFIXAJRCMXDP*_DNYAOVRPXRXMSY*_HCUMI_IOUWKAUIHBPTSX
OXOCEXOAFKWBDRZMYELC_*QCLOUHREHQIYUSRCESRPG*#HDWSB_IZCVJTCH
CLMIRPIKWOXDPIOXEMVJ#WOVMPRHIMHFGTBSDCY*QE#U#E_MRNWMSUOB#MO
ULRFI_FREZNYLVHYWCUQHKYAZGQN#CATMGYNOJUGHT#UMTFDCOI#OVRKEPN
#RHXJ*DUBVODRHWRXHTESLUDERAVNL*_HYOCZRBGI#QCGCNQUOHJYKZBEJZ
FVXMIAL*#GAOXM_PFEBVFGXBSDC#IBYOBUVRKVDODOU#W_BYAZ*NJEYS#AR
OXJ*DLKR_OFVWE#M#PVJYEV_K#FA#YRFYT#JEZBDMFKSCJL#DJIXG*UWYZH
IOKNREPZKFQYEAWHZBQCGODTSIW*#VYRFSODK_VYTC_BXJNRHABU#_*OXEM
HTVYKIOQNDWMXA#LBCPQOQKPMKOZRBZDTDOQGHYJ_ZFAOSKZPFVDCMYQCUL
MECE#BVBGEXFIT*QE#U_CUMQ*IOXGNF#_Z*LPMIC#N*SPK#LKQHP_PVPIFG
IAUESBLNPE_TEMJOYBUW#VGPV*QZSM#GJPQKVBVN_AUALAPWI_VPGEMOGYQ
CLOBRKFHL*YR*SME_KODRIZLZ#SRUWF#EXM#GKOWM_*TKMQYS#EUCGXMIYF
WO#MENHXRUJSRCERLE*IFJ#_
```

Unit 124
Cylinder ciphers

The first cylinder cipher is the *Jefferson cypher wheel*, invented by Thomas Jefferson. It had 36 disks, each with a mixed alphabet of 40 letters printed around its edge. It was intended for the encryption of French correspondence, since French was the language of international diplomacy at the time.

Somewhat later, Bazeries reinvented the cylinder cipher, so it is sometimes called a *Bazeries cylinder*. His had 20 disks of 26 letters each. The key for this device is the ordered list of the disks on the cylinder. Bazeries proposed a method for deriving the key from a keyword, and this method was adopted later by the United States military (we explain it below).

Bazeries cylinder. Photo courtesy of Étienne Bazeries.
The cylinder is set to encipher the phrase “I am indecipherable.”

Below are the mixed alphabets on the disks of the Bazeries cylinder (the French have no need for `W`, it seems). Notice that many are built from French phrases.

disk	mixed alphabet
1	`ABCDEFGHIJKLMNOPQRSTUVXYZ`
2	`BCDFGHJKLMNPQRSTVXZAEIOUY`
3	`AEBCDFGHIOJKLMNPUYQRSTVXZ`
4	`ZYXVUTSRQPONMLKJIHGFEDCBA`

5	YUZXVTSROIQPNMLKEAJHGFDCB
6	ZXVTSRQPNMLKJHGFDCBYUOIEA
7	ALONSEFTDPRIJUGVBCHKMQXYZ
8	BIENHURXLSPAVDTOYMCFGJKQZ
9	CHARYBDETSLFGIJKMNOPQUVXZ
10	DIEUPROTGLAFNCBHJKMQSVXYZ
11	EVITZLSCOURANDBFGHJKMPQXY
12	FORMEZLSAICUXBDGHJKNPQTVY
13	GLOIREMTDNSAUXBCFHJKPQVYZ
14	HONEURTPAIBCDFGJKLMQSVXYZ
15	INSTRUEZLAJBCDFGHKMOPQVXY
16	JAIMELOGNFRTHUBCDKPQSVXYZ
17	KYRIELSONABCDFGHJMPQTUVXZ
18	LHOMEPRSTDIUABCFGJKNQVXYZ
19	MONTEZACHVLBDFGIJKPQRSUXY
20	NOUSTELACFBDGHIJKMPQRVXYZ

The *M-94* (also known as *CSP-488*) was a cylinder with 25 disks. The disks of the M-94 were identified by a number or by the letter that followed ‘A’ in their mixed alphabets. Here is a list of them. Notice that disk 17 begins with “ARMY OF THE US.”

disk	mixed alphabet
1 or ‘B’	ABCEIGDJFVUYMHTQKZOLRXSPWN
2 or ‘C’	ACDEHFIJKTLMOUVYGZNPQXRWSB
3 or ‘D’	ADKOMJUBGEPHSCZINXFYQRTVWL
4 or ‘E’	AEDCBIFGJHLKMRUOQVPTNWYXZS
5 or ‘F’	AFNQUKDOPITJBRHCYSLWEMZVXG
6 or ‘G’	AGPOCIXLURNDYZHWBJSQFKVMET
7 or ‘H’	AHXJEZBNIKPVROGSYDULCFMQTW
8 or ‘I’	AIHPJOBWKCVFZLQERYNSUMGTDX
9 or ‘J’	AJDSKQOIVTZEFHGYUNLPMBXWCR
10 or ‘K’	AKELBDFJGHONMTPRQSVZUXYWIC
11 or ‘L’	ALTMSXVQPNOHUWDIZYCGKRFBEJ
12 or ‘M’	AMNFLHQGCUJTBYPZKXISRDVEWO
13 or ‘N’	ANCJILDHBMKGXUZTSWQYVORPFE
14 or ‘O’	AODWPKJVIUQHZCTXBLEGNYRSMF
15 or ‘P’	APBVHIYKSGUENTCXOWFQDRLJZM
16 or ‘Q’	AQJNUBTGIMWZRVLXCSHDEOKFPY
17 or ‘R’	ARMYOFTHEUSZJXDPCWGQIBKLNV
18 or ‘S’	ASDMCNEQBOZPLGVJRKYTFUIWXH
19 or ‘T’	ATOJYLFXNGWHVCMIRBSEKUPDZQ
20 or ‘U’	AUTRZXQLYIOVBPESNHJWMDGFCK
21 or ‘V’	AVNKHRGOXEYBFSJMUDQCLZWTIP

22 or ‘W’ AWVSFDLIEBHKNRJQZGMXPUCOTY
23 or ‘X’ AXKWREVDTUFOYHMLSIQNJCPGBZ
24 or ‘Y’ AYJPXMVKBQWUGLOSTECHNZFRID
25 or ‘Z’ AZDNBUHYFWJLVGRCQMPSOEXTKI

The key for the M-94 is a list of 25 disk numbers, indicating their order on the cylinder. As we mentioned earlier, the key can be generated from a keyword. The procedure, taken from Bazeries, is as follows: First, write the keyword as many times as necessary to have 25 letters. Then, in alphabetical order, from left to right, number them. For example, if our keyword is CYLINDERCIPHER, then we proceed as below. Since ‘C’ is alphabetically first, we number the ‘C’s first. Then the ‘D’s, etc.

```
C  Y  L  I  N  D  E  R  C  I  P  H  E  R  C  Y  L  I  N  D  E  R  C  I  P
1 24 15 11 17  5  7 21  2 12 19 10  8 22  3 25 16 13 18  6  9 23  4 14 20
```

The key is 1, 24, 15, 11, 17, 5, 7, 21, 2, 12, 19, 10, 8, 22, 3, 25, 16, 13, 18, 6, 9, 23, 4, 14, 20. Notice that the lowest number is not zero.

To encipher a message with the M-94, the 25 disks are placed on the cylinder in the order given by the key. The message is broken into blocks of 25 letters. Each block is enciphered by rotating the disks until the block is written in a straight line down the cylinder. Then, some other line of letters is taken as the ciphertext block. It is required that that line not be immediately above or below the plaintext on the cylinder. The offset between the plaintext and ciphertext lines should change from one block to the next in some random fashion. This makes the cipher probabilistic, but also makes decipherment nondeterministic. During decipherment, since the offset is not known, we must choose the best line on the cylinder.

Let’s continue with our example, and encrypt the message

THIS MESSAGE WAS ENCRYPTED WITH A CYLINDER CIPHER

The first block is THISMESSAGEWASENCRYPTEDWI. We line up this block on the cylinder by rotating the disks:

```
 1 24 15 11 17  5  7 21  2 12 19 10  8 22  3 25 16 13 18  6  9 23  4 14 20

U  S  P  A  N  Y  V  E  R  F  I  Z  G  Y  J  I  R  Q  V  E  Q  X  Z  F  X
Y  T  B  L  V  S  R  Y  W  L  R  U  T  A  U  A  V  Y  J  T  O  K  S  A  Q
M  E  V  T  A  L  O  B  S  H  B  X  D  W  B  Z  L  V  R  A  I  W  A  O  L
H  C  H  M  R  W  G  F  B  Q  S  Y  X  V  G  D  X  O  K  G  V  R  E  D  Y
T  H  I  S  M  E  S  S  A  G  E  W  A  S  E  N  C  R  Y  P  T  E  D  W  I
Q  N  Y  X  Y  M  Y  J  C  C  K  I  I  F  P  B  S  P  T  O  Z  V  C  P  O
K  Z  K  V  O  Z  D  M  D  U  U  C  H  D  H  U  H  F  F  C  E  D  B  K  V
Z  F  S  Q  F  V  U  U  E  J  P  A  P  L  S  H  D  E  U  I  F  T  I  J  B
O  R  G  P  T  X  L  D  H  T  D  K  J  I  C  Y  E  A  I  X  H  U  F  V  P
L  I  U  N  H  G  C  Q  F  B  Z  E  O  E  Z  F  O  N  W  L  G  F  G  I  E
R  D  E  O  E  A  F  C  I  Y  Q  L  B  B  I  W  K  C  X  U  Y  O  J  U  S
X  A  N  H  U  F  M  L  J  P  A  B  W  H  N  J  F  J  H  R  U  Y  H  Q  N
S  Y  T  U  S  N  Q  Z  K  Z  T  D  K  K  X  L  P  I  A  N  N  H  L  H  H
```

P J C W Z Q T W T K O F C N F V Y L S D L M K Z J
W P X D J U W T L X J J V R Y G A D D Y P L M C W
N X O I X K A I M I Y G F J Q R Q H M Z M S R T M
A M W Z D D H P O S L H Z Q R C J B C H B I U X D
B V F Y P O X A U R F O L Z T Q N M N W X Q O B G
C K Q C C P J V V D X N Q G V M U K E B W N Q L F
E B D G W I E N Y V N M E M W P B G Q J C J V E C
I Q R K G T Z K G E G T R X L S T X B S R C P G K
G W L R Q J B H Z W W P Y P A O G U O Q A P T N A
D U J F I B N R N O H R N U D E I Z Z F J G N Y U
J G Z B B R I G P A V Q S C K X M T P K D B W R T
F L M E K H K O Q M C S U O O T W S L V S Z Y S R
V O A J L C P X X N M V M T M K Z W G M K A X M Z

The ciphertext for this block can be any of the rows other than the ones immediately before or after the plaintext. Let's just pick IQRKGTZKGEGTRXLSTXBSRCPGK. In the same way, we encipher the remainder of the message (we only need 17 letters for the second block). The full ciphertext can be

IQRKGTZKGEGTRXLSTXBSRCPGKLIIBEVOOJNUBBYNAS

The M-94 cylinder cipher. Photo from robbo@ev1.net.

Reading and references

Thomas Jefferson, “The wheel cypher” or “Project of a cypher,” Thomas Jefferson’s Papers, volume 128 item 22138, volume 232 items 41575 and 41576, U.S. Library of Congress, www.loc.gov/item/mtjbib025756, founders.archives.gov/documents/Jefferson/01-37-02-0082

Étienne Bazeries, *Les Ciffres Secrets Dévoilés*, Paris: Charpentier et Fasquelle, 1901; books.googleusercontent.com/books/content?req=AKW5Q...; pages 37-38, 132-135, 244, 277.

Friedrich L. Bauer, *Decrypted Secrets: Methods and Maxims of Cryptology*, 4th edition, Berlin: Springer-Verlag, 2007.

Wikipedia:
en.wikipedia.org/wiki/Jefferson_disk
en.wikipedia.org/wiki/M-94

www.jproc.ca/crypto/m94.html

ciphermachines.com/jefferson

Crypto Museum, www.cryptomuseum.com/crypto/usa/jefferson/index.htm

maritime.org/tech/csp488.htm
maritime.org/tech/csp488man.htm

William F. Friedman, Several Machine Ciphers and Methods for their Solution, Riverbank Laboratories Department of Ciphers Publication No. 20, 1918, www.campx.ca/Several_Machine_Ciphers.pdf and www.marshallfoundation.org/library/methods-solution-ciphers

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 192-195, 247-249, and 325.

Programming tasks

1. Implement an encryptor for the M-94.
2. Implement a decryptor for the M-94. Use tetragram fitness to choose the best offset for each block.

Exercises

1. What is the size of the key space for a cylinder cipher with *n* disks with 26 letters on each disk, if the offset is not specified in the key (i.e., the offset can be different for each block)? What if the offset is specified and has the same value for each block?

madness's book on classical cryptography
unit 125: hill-climbing attack on cylinders
last modified 2020-10-28

Unit 125
Hill-climbing attack on cylinders

We can build a hill-climbing attack on the M-94 by using the parent/child key paradigm that is familiar to us already. The key is a permutation of the numbers 1, 2, ... , 25. To generate a child key from its parent, we randomly swap two of those numbers. If the tetragram fitness of the best decryption (remember that the offset is not determined a priori) exceeds that of the parent, the child takes the parent's place. To avoid being trapped in a local maximum, we can allow downward steps some of the time. A good maximum for the allowed distance downward is 0.15 or thereabouts. If we cannot find a child with a higher fitness that its parent in the last 1,000 children, then we output the parent key and quit.

Programming tasks

1. Implement the attack on the M-94.

Exercises

1. This ciphertext was encrypted with M-94. Break it. It may take a while.

```
UHMBVYKDKYSFBEQSVLZRQURQNPCJNKKBHSNIQUYJDSKSHQJPBKBTAG
JYPLKHGBMAOYUPYGPKVUFKWDBFBUJKWGVAYXKDJWONHFCYPIXPSBQV
FCRFYTKGJNWJDIDEPHVUWWVUJOTNKXMUCQZXLOMWUMYEDNWEWUJJFX
DLUSQPTFCCHUVBABRECUCQIPHJAPFGMGBYBOEMWXXZJUKSVLVVMNGG
UQVWSNKFQTYXOAUHWYNDSUQWSOMSAFQJMTWPOMWTAVSYDEXMDZUXPB
ZOXNNUEKZDMMGGMAFMUOZOGSAWIQMHJVQUYYYUNXTEULNWIRINTIVZ
CALTXXHUOUGBNUYFHZHFGCBNOIJWRWSXLILBGCTDWQWTNPHXNUQEHU
OMPNKWDRBRWYSWFILNYTDOUKDKIFIQDLQIJTLBRQKNTPEZUZPODHBL
LGKHZWKYJPZSJSURDSSOHCUXZVIKOUYEADDOXILPQQAHRAOVSBETQR
ZAGXRRDTYSVIKBTIQBNXJKHVSZJJZSWNJZMKXJKIHJIFCMWKIXXQZG
YRPPEFTXOUDQOVRCXABHKHDENXJIHHCAWFWXDEFAMCCKHQMTWPHXYK
AILDRIWBWLPSURPYDSYKWVWORCHMJJPPHJLLYEYNRHIQZLKASYAQJO
HIOKDZJLHBCAWTOSMFLVSDXHZKVOELRXBGPWXHAGPZCJUKNZCOGLNT
NVLKVFIOUIZCPXEGZHDLMVFYDZHMMKMOLGOVSYYVHMQNREOYPTCVLX
RCSAVPNHYPXHIVIKOSNFQQHCANYJDHNZHACDUIHCAWYETOOAQLYEUU
```

```
VTCHSYPGIZCRBVLTQBHRLKTPTXMPYBNTOHAGKOQIZRCRAOTSYCUOYX
HVZAYSPNHAKYAJCSFWLHHNNKORVYHPFCXQAAYJJSERNPDOSSBINUIL
ASPUDPRNWGWRDNYUCBSSJPMAKSKQQXFEPYWMMEDVCKMBWUXMJXOCAB
YHYODHCIVTRIXOXRSNUNKSJXSWKWVVASXMNQOHCSVMGSJXRLCZQVCC
TDUJXIXRRZTMXMBGTGHTCOSOFZAFJMIXRZZTIVKMLMQPQUNOGGXYKR
DLYOEAHNHVEORHVIKEGMVMXDWBBMXUNBIPBBIJAMRQFOQCRRUDWVXR
MOWJCLXUFMFEYGPMUVKBUUUZPRFUJWQEOOWYMEGXNMWNDIWNJTNDPZ
FYXAGXSVGQWKJRSPERDGYEPRBPZUAIFXRQIMGWABUWGCWWGRIIQJXL
XLQTRQTICGIZRCLIJSMWXWWCYQHAGKYUGTETXPNGAPCKABBUDFWGZT
CMWBRBQKUVMUYSGTNSCLVCJBMKY
```

2. This ciphertext is from the 2009 British National Cipher Challenge. It uses the set of 26 disks listed here, and the same offset is used for each block. The first two words of the plaintext are in French, but that should not stop you from decrypting it.

disk #	letters
1 and 14	HIMQEBNYULWTASCVOJPRXZFGDK
2 and 15	BLAXCVSHIRWPFYQODETMNJKZGU
3 and 16	EADZMIJYKSRXVQTLPUCHNWGBOF
4 and 17	LHRTSOUAZKXEDIBWYVPFNGMJCQ
5 and 18	AOEBMUNWJYCGXQZVHLSTKRDPFI
6 and 19	HBTWOSIVQRDPYGNXUZLAFCKEJM
7 and 20	OBQZGRLKXATPNSYEWVDFIJUMCH
8 and 21	IMRWVULBDKOQHAETPGZSYFCNJX
9 and 22	RUHCESJAXMNZOGVBQTKDWILPYF
10 and 23	SAZORCIETFYKXMDPGWQVUHLBNJ
11 and 24	QUAIHJMCEZTKYFVLPBXOSGRWND
12 and 25	MUCATPJZOXEWIFQHBSRGLYDKNV
13 and 26	QWLCKUETDIVPHFAMZOYBRSGXNJ

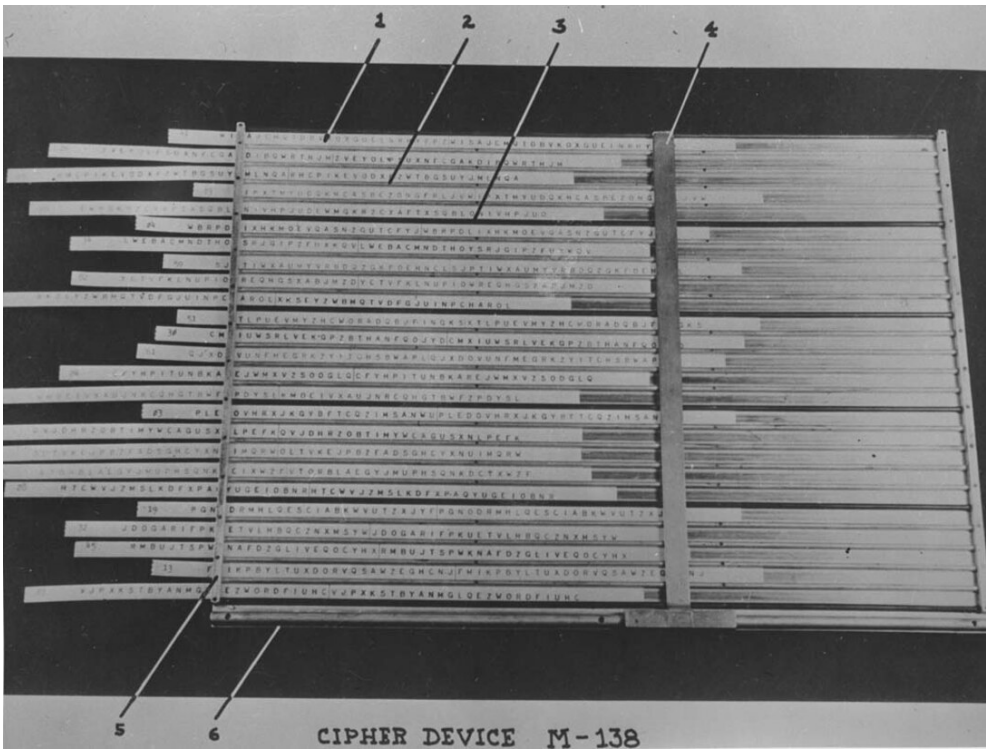
```
WUUHB SSCGG YRMIK JLNPL ZBGCI UJDUK FBHCU KZNVA EBXGA
AGBGD GLZAJ ELSDV EPTMI USOSW WARIZ PBRCA LZJNI CNFAF
VXLCO CMMOF LCTII HEEWY IZAKZ MEAGV XXOOA ZYULB CIFEW
DJPWJ MYGJB VVGPE SGKCP MAGZG OQKDG JKIML DTXXT CDDVZ
XRJDP FIFKD GJKIV AUAJM NBZXD VKEYD DTJAF CBXAI JUSPM
PPAAW LZZNJ USTCK DFERT CJCMJ IUVQQ LNCTL CBAFI YIPWI
LBEZG JAMWG ECDCR CWUGK JDQNH FAKAP AYCBH EUNDY PODNG
KCASC GPMEA DYJYB HKURJ KCRIZ YHYZM DIITU ACGAV FXRUZ
YVFIL ZDIND KCBZL IODDN XXXRU EJOIL CSVWS QQUOJ YLVAI
YDNXR USBVO LCUQE ASYWB ICIRA RBVHK ESDTZ XLRTK XCHXM
FGUXC NFBQK IMOFZ FDTDA GKCBG VFDUJ OSKIQ ORABK BNDCA
WCRCL MUWZI KJEQO JZUEQ EZAMI LZDEY BYEWJ VRMDC CLWMC
BXJVG WRRKW ADJZT PNWLR JNDNP ZUDNX IGLBA EFSKC EBUYS
IKZLN PESGK CPMZF JUENI XHFAB SCJNG XXRUX JMPID HUWZL
QDKTX MFFGC OBIAA EGNKR JDVGZ BDCND VKJPK PMXYA QWXCL
```

IUUVE FSMWA KHIGC RCWSP EQFYW FYIUD UFRGE EPXNA EJBMF
YAJUY ARTIU SUFQQ KNPCL ANQZV ATEBW GNDEX ROBGH QVOUN
RLLVC IFKEY IHHDJ BWIWI HXZKF MTGCK VOOAO SRTZX WHWCW
SQGNO OPGOJ BZXEU ESFZN XBOIB FQGAF PCTUA XDMYW RARWW
CFMAQ GGIHC ZXVKJ KRJTM FYAEI KNORL EEIXA GNCII REBQZ
MGTEG WKFNR CBSES BRTXD CUALD NXXVC LFDYF IROKC FBHCS
RTJCE JBVKP IWRLM DUCDY GSBIN EQVBL AUCLM HAYFZ MDBCZ
YEFSD AXIKJ KWSSK CMAIW GCLBC WUIXS CYBGP JOBVJ DCLAA
UZRBI LZDSC DKZUJ PGOJB YBZUF EDSVK JRDDB CDXKW IKIJI
GFMCH BEICO PVZBJ GXAMP ODXWC CKSMG AGECB ULIDW GECVF
NFKTE DOQBM CXGBE QCFFH EJLBN VNOEZ JPRFB MCDAR GSHPI
UULSJ FACHK HPOCM YIFYO QNSHD EATQE FASZA JYRGR UOXKP
GANPC JLJRK RMSVH WZHIJ LDOMX LAMED YPLIF JXBSJ JWICR
EFKIX LPVWU SFITU UCUYH YCBNZ RCWSL KBRAT IKNOL ZISEO
HMNWT PEFIK OKOXF ERNUN OXFAG ZSAAF GYRQU SPMKU MJFDQ
KJROK JJZRZ KOKRL

Unit 126
Strip ciphers

Strip ciphers involve paper strips that are laid in horizontal tracks on a tray. Each strip has two copies of a mixed alphabet. The strips can be slid relative to each other, and a block of plaintext is enciphered by lining up the strips so that the plaintext letters appear in a column. The ciphertext is then read from a different column. Essentially, a strip cipher is a flattened version of a cylinder cipher. We can see now that the reason each strip has two copies of its mixed alphabet is so that we do not run past the end of one when using the device.

The *M-138* is the flattened version of the M-94 and was used by the United States in World War II. It had 25 strips in 25 tracks. We cannot say with confidence at this time whether the strips had the same mixed alphabets as the M-94 or whether there were more strips (but only 25 were used at one time).



M-138. Photo from U.S. Department of Defense.

M-138-A was an improved version of M-138 with 30 tracks and 100 strips (only a subset of 30 are used for any one message). The U.S. Navy renamed it *CSP-845*.

To attack a strip cipher which has more strips than tracks, we can modify our hill-climbing attack on the cylinder cipher by extending the key. Suppose the device has *m* tracks but *n* strips. Then the key is a permutation of the integers 1, 2, ... , *n*, but only the first *m* entries in that permutation are used in decipherment. When we generate a child key from a parent key, we randomly choose one of those first *m* entries and swap it with any other entry. If the offset is the same for all blocks, then it is possible, for a sufficiently long ciphertext, to break it, even without allowing for downward steps. If the offsets are random, then this attack is likely not to converge to a solution.

Reading and references

Wikipedia, en.wikipedia.org/wiki/M-94#M-138-A

Greg Goebel, *Codes, Ciphers, & Codebreaking*, chapter 5, vc.airvectors.net/ttcode_05.html

Klausis Krypto Kolumne, scienceblogs.de/klausis-krypto-kolumne/2018/02/16/the-m-138-a-simple-but-good-cipher-device

Operating Instructions for CSP-845, maritime.org/tech/csp845inst.htm

Christos military and intelligence corner, chris-intel-corner.blogspot.com/2012/10/us-military-strip-ciphers.html

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, page 325.

Programming tasks

1. Copy your routines for the M-94 and rename them for the M-138.
2. Implement an encryptor for the M-138-A. Allow it to read the strip set from a text file.
3. Implement a decryptor for the M-138-A. Allow it to read the strip set from a text file. Use tetragram fitness to find the best offset for each block.
4. Implement the attack on the M-138-A. Allow it to read the strip set from a text file.

Exercises

1. Encrypt this text with M-138-A and key 55, 37, 85, 10, 30, 9, 73, 91, 61, 4, 17, 64, 81, 94, 67, 1, 28, 75, 98, 90, 52, 48, 8, 11, 6, 45, 14, 38, 60, 5. Choose the offset randomly for each block, but do not use columns adjacent to the plaintext, as we avoided adjacent rows with the M-94. Use the randomly generated strip set below.

Are the mixed alphabets on the strips of the M-one-three-eight the same as the mixed alphabets on the disks of the M-nine-four? We do not know. We have contacted the museum holding one such device, and await an answer.

2. Decrypt this text with M-138-A and key 11, 93, 98, 52, 74, 89, 43, 41, 3, 58, 28, 23, 31, 14, 90, 82, 86, 44, 62, 34, 22, 12, 35, 91, 60, 13, 48, 29, 78, 40. Use the randomly generated strip set below.

```
SHJRQBJIMRHFLLVACMUMCILVCXYUKVPBVXSBUNBQXSCPGEXCHZNOZU
PKZRJPXCPRDQPMPJWHDTVGVUQYYHWZUCAFXRUWWDFWULXHSDSFUYNW
IVCCSMMLJKGLHEPNNEHZBGNEJGGSCXUTLLSJTIGPRCPFJSSXASCPXJ
VZIJLLIAMXIUQZFKALIBPHIGKOCERQGGQNKNXKVBHGOESHPJUBLZLB
LLXBWMRYXVBSHNVDIASJXAGXQRNAAMPABJDGFOYGIGBAOSGCCUWUQU
OFCYDUHYOUDYTQRPBEULPRJAHYSJDZGLROPGFZFMRCDAEGPQAYKNJP
R
```

3. Break this ciphertext that was encrypted with an M-138-A. The key was randomly generated. The offset is the same for each block. Use the strip set below.

```
KYIVOVMFBHRWPYJWITPDERWOXZJSYRWXIXTVTVRWOLMYIREFXAKNNA
QHTMYPPEMNGVMPTHHZTRDCDEFOUVOEITEZOHVAMGYTXTFOVYZURNJC
KJMKFYBCEUMGPVSHBHYHNVULAIGNWQJZMKJDLFJMKJEHSHMVHVVGVC
DPJMYXRLPKJDXUTTYXPLWOWUVETWYZKSVIJPBBUDRKLQJMVVTFDBMU
HNKYDKCAIILFMKKNNYKZQLNPWJZCVMHJHDQZTBLVFVRKXPPZLPXMVY
WFHGKHLVSGIOAUWAFAIOCCPDOOFYKJEIGLMHXXSLIFCMQCDEFOUVVN
URYBUVSOLWVHPVRPXLAFNBBOJXFJTVJRMYKPLIDNWUKKDHYZPBLVDJ
FDSIXYGPUUUQWOLGGQNFUBMDPZRODJLJMHWJLCSKOHXWVBMIAYVFYH
EBLVDJAXCOMYIGJIIQWXIITJLRKEGOLJGADIHWKJRQGUKQARVVVYSX
JZTTVCZYJIOSLBIGBAGCCUNZOLVNYMEZUSXYPMIIMVGZMDQRTZJULT
PXVBMJKNSYYHEBLVDJSKIPXOXUZCMONLGVIAUYQGOCIPWNRISOUJPM
XTESOJAVSNGMUIQODOQRQIOQAPSVPOIGFYKHURYMJRUVBDQDZRNDIF
ZKEJTMCRPZYQKFGIVKORVHUOWNGEFSPCLWIAGUYRWZVDMVMEMTTVEZ
TRONRGLIECZYIRYNNJSVWHBHUYGYYQTZSMJJLOUVRTQZZAZWAOVBMP
NZFKIGONGSYEKJMJPTKWYHBZKOWGPKJZTQHMWBIVILJTFDBZJKYLMI
PDMGULJZTQHGZRYRMVKKKNNYKZAFFQWOLNGMLFSYMYVAITDIKJMGNE
XUQIWQYLVBKIASNVVBKIIXWYQYCATKBGXUKJYHQODPJEQFOLVVGNDE
PVJPUYQACZQQEVEOKIANFEHEPZRAJGLIECFNQRQUSYIZQMKITINWIN
CPWTGPXCLVTKQATCZJEUVGSMUVVYNLVRITGPIKMXNELVJNJHVDLYCG
HHIPXEKMINDELJMSPALVUUL
```

Challenge

```
KHXPKMDWGKBDHUXWVBHKQOOKVTETXTQSBHDLWRFNPAOAEOPMRLHBIINPKQA
ZGPGXNUXMJBKPROCAFZPYQPARZQLYWUZGUJMYRWAORPFJZFPJQNSMBJZXSL
HYZGRBTDOVPCRZSYEPPPGLKUAFWKLXZCKKRBZXTAKRSPBIVJRKNZXUAFLGS
LEUKYNDNVGLPWAXNPNXYCQRLTWENBKRXUACAEUZBBZCMMIWCLUCKUUNRZGS
GWJABUZSVJWHYXYXDUFTJYYKPWLGNHPRXHESOBCTWFNYTOXSGNQOKJIFVMD
```

```
PEGNEVIYPLPLOMSMXGGZZLPUOPOHXIXTUGCOUATFVUYLTNSYMKWUZSIFCMQ
XUZARVITBEJWOOKVYQAZCFKUQAJQOTETTFBGNGPGPDBZLFSDZHHAMFOYQFB
WIMZKMKMMUOFAYSCQREZWVNDWDRMBXRBIEJIZFXMALJECOIPWFSHUAWISLW
VONCIORJOVVGYINUPLGEVCIXWFNRDXXAOSSUHPAWQEXQNACFBXAKOJGLJTU
KAOCZZFHWXSKNGEVIKQTQYQDLCDQJNANFCJRTFQCHBJSZFHRGMYVYPGQULD
ZCDWGHXNXFHFPXNSDTSWZFKOFHGNTOHPOOZRWXAUEFUMWKLAOTWQZUJZFIE
FPDRLQBBZEJHKSVJQCPZNHNEMVTOQVGXHASBHVSRULXKMMOMJKGLMFIRHCH
OTIKYDSPXSFWNLYJNAELCGVGVXQNIHRNSWJFCNXDCMXSFFHTCVMFVUDASHU
NJJYYMWHNPOBJZVFNHKLYXBVVHIJCHSNDDBBTOEDJIVMCQPXGCMBQPZXXZZ
NOKMJZWMACNNRFFVDUBMEGEVGQDQJVAWYOPJXQACAHFIYZJVZPASQVBIDNR
TALWMGXXALPLTSZQKKYVHBXGHOZOVEPEDRURQEUAPPZBJDAJJDSNNMSJOXR
CTNPCFDCHREAQMJXGGBYKLGSVVOUOBCXMAAQKYYQFJBSKHKPBWKFLAKOHZO
UURKFVFKENXWULACWRESAWYXKKSEPYPKLZWTZGJENDTZJU
```

This is the randomly generated strip set for the exercises and the challenge. It should not be considered an historically accurate list. They were generated solely for the ciphertexts in this unit.

strip #	mixed alphabet	strip #	mixed alphabet
1	EBCNRTKUZOQWMFGAPXVYHDIJLS	51	JSATWPRYNFOUVMZXGDHIEKCLQB
2	HIVQZUYJNSRLEXBPOTAGDWCMFK	52	ABXJMZTQOIKVHEDFYCLPRSWNUG
3	VSHTUPZIRWMGQOEALBYJDCNKFX	53	JBYOXPHGVZQLWSDMTKRFCNIEAU
4	HGYENUCRPZWMIQTKAXJBDLVSOF	54	RLKMQABOWHNFSVCXYPEIDZGJUT
5	ZBPGTUVCOWJLEISDHYNRKAXMQF	55	ETLMUZYWOCGIRDBFPKAVJSQHXN
6	YWXHKRZUOBJSFGNAPLQCEDVMIT	56	VMRIUJFKEBOPWTYSADCXNHQZGL
7	DMBNORVFCSAWGIQYKJTZXHPELU	57	ZAMDJFNUHKYPWVRXLIBGQOCTSE
8	XBQFAGITLEHZUDKSOWPCRYNJVM	58	DXYLJAQKUFRPITVNOHSZMGBWCE
9	RPJSWFBUTHEDQMKGOIANYVCZLX	59	XRMNJATKOBFZCPWUVQELGDYISH
10	FSOCELBPVAQKIXWZGTMDNYUHJR	60	SYPITVMJZKLEUHOBFRQNGDCWXA
11	ACPIDOYSRTUHBKQGVNXWFELJMZ	61	DABHQKMWXLICPZSTNEGFVRYUJO
12	YLXTMNVZEJUKDPBHQAFSRWGIOC	62	QFTJLICAZROBEKNUPSWYHVGMDX
13	TBDECYZPNLGUFOJRKQSIWHXVAM	63	OJELKXUHSPIWQMYDNBGTFVZRCA
14	FLNYCMEWSXHIABGQRPOJUKZTVD	64	IEWDCNOLZJHFGBPUQVMYTSKARX
15	FTHXADWIUNYJOMBCQPVSEZLGKR	65	SGKYVNLIUZEAQJMFXOWTDPRCBH
16	UHNYCAOTLMZDGIERJQXPVWFSKB	66	DRHPWCGYVUEXZQALIONMSJFTBK
17	JDGQBOZCVMHAWYKRSTPFXUNELI	67	FTNZQAOEUXYVPWHKRSIDBJLMGC
18	GMJUBTIAVYRHPOECDNKLXZQSWF	68	DIBHXALWQCNZGKVSFUJMPRYTOE
19	INYVECOHBDAFQZLWJMKGTRUPXS	69	KUDQXRFHLISACBTPYNWVJGOEMZ
20	BZAXDNSYWGHCRMEPLUFJIVTOQK	70	MIWEFSCPATKVLORQHXBYJNZUGD
21	HLZUEXJNRTQCKWOYBDMPFIGVAS	71	ZQTNYKHCVXIUMGAJPODEBRSLFW
22	AUEPVHSGMJFRCWDBXOTYILQKNZ	72	UKMHBAJSFWGCDXQYORVTZPELNI
23	FDIJMAGVWLNQKECZOXUSTRPBYH	73	WXYUVKNMADRGCTHLEFPZQOSJIB
24	HOANLPTXQDWYJRGKFBSCZMUVIE	74	BHEPRZTOVDGKMJQUIYNWFXLSAC
25	XTLSZCDFJEYUOPNBARWHVKQIMG	75	ORVUICMBNDXLAJYZSKGFEPTHQW
26	YCJWQUNIHOVPFSLZMGATDKERXB	76	HVJOKXSRFIQUACYNTMPGEWLDZB
27	RECXJZDGIAPKMYHQNOFLTVWSBU	77	KNGAOIYBHWEQMPJZLDSUFRTCVX

28 JDFSECBXZMWPNRHUGAOVLKQITY
29 EQHMOGUIYVACWTJZNKFPXRLSBD
30 DIFAHKGXWPVTRQSJLBZOEYNMCU
31 FJRBXNIGVOUPKWHYLDQESZAMTC
32 AKEFBDJPUIGVNWZHOSQRYCXLTM
33 ZMPKSAUBLWVQDNTGOFEXHCJYRI
34 JISEFBDYAMRTLPCQHXKGUNVZWO
35 CNDTUSILPVFYHMZWRBAKGOJXEQ
36 MATXNJLHVZUKCDBIPQFEWOGYRS
37 IMVEJAFZHQRGLKUPWTNCSYDOBX
38 NYJQLMDARHGOFKBPETWZXUVSIC
39 ZAVEPUWSOKCHNLRTQMJDYFXGBI
40 MFKTUREJYOCGXIZLQPNBVHDAWS
41 RQICZOHVMNWELSUGXYTJKABFPD
42 CPJMODWQXIFYLRNUKBGZVTESHA
43 LZMACJPOSQTKGINXVFWRHUBYED
44 VNTAYRIPFEDMGKWBQJUXLSOHZC
45 ZIWKNJMSVXFEUDGQACHLYBTRPO
46 FEWRDMLBNATKGVCSZJYOQIXPUH
47 NVEHYGLSTBIFXUZPCQRDAMWOJK
48 ALEGPMUOSDFYCHTWQIVNJXKBZR
49 DJNGOUHQYAELRWBMTXSVCIPFKZ
50 VQSEGULNCBMZJOWRYXPAKDTFHI
78 WIRBXALHKJZYQSVGDUNOPETCMF
79 SVYPXTINHZAKBRGMOLJDCUQEFW
80 OJFWITSPNDAMZXLYERUQBGVCHK
81 UJFQCGPAMTLEKRXBOYSDIVWZNH
82 QTEWPORKXCIZYJVBMFULANGDHS
83 TLCGDRXFSZIAJUPMEVOHKNQWYB
84 XKTJDPWLVOEZGRUCYANQFHBIMS
85 TBUFMIHRAGCWPVJLXZQYDOSNKE
86 WMLHSAGZEPKYJUIXVQBCRNFDOT
87 VLNTRGHDUIKEFMSWZJQOYAXBPC
88 YUVBSKNOPEIQDMCWAHJZFGLTRX
89 LYPFDHWBZNCTXROGKQIEUSJAVM
90 EKAQCSILMFJYUZBTXWDPGVHRNO
91 LRQJSDPTYWGBMAINKEVFOXUHCZ
92 AJNRXTMZKHBGOSLEFVCUPDIQYW
93 TCEWPVDIQBJONKYLRZGMFUAXSH
94 BMAOCFRZDWVTGXSHQJEINKUYPL
95 XKVDFJOPIWZQNHAGLERBSCYUTM
96 HZNUGFQRWCEADVSPMOBJYKLITX
97 BPTLKUHZMFADEVOXGSRQIWNJYC
98 MTCISYWNVKOZJELUFPGQBXRDAH
99 BANGXEZSMJPTDWCOHRYFIVLQKU
100 YVTSFOXPBRMKDIELGAQHWCNUZJ

madness's book on classical cryptography
afterword
last modified 2020-11-04

Afterword

What now? You have learned everything there is to know about classical cryptography. Just kidding. Keep learning. There are still more ciphers we have not covered; many can be found on the American Cryptogram Association's website at www.cryptogram.org/resource-area/cipher-types. When you feel the urge, find one that is new to you. Study it, understand it, and implement it. Find a weakness, then a way to break it. Sometimes, the best you can do is a brute-force attack, but for classical ciphers that is still an achievement when you have modern computers to help you. Furthermore, new classical ciphers are invented still, usually as challenges to other enthusiasts. Several of the ciphers in the section on miscellaneous ciphers were invented for that reason.

After the classical era comes the mechanical era, characterized by rotor machines that use a collection of rotating disks to encipher messages. The disks each contain a wire maze that redirects an electric current as it passes from one disk to the next. Some have reflectors that redirect the current back through the set of disks. Enigma was one such machine. It had three or four disks and a reflector. The disks could be removed and replaced in different order, and the reflector could be swapped out for a different one. By using a reflector, Enigma could never encipher a letter to itself. This weakness helped to break its cipher. The Bombe was a device that, together with a crib, was used to recover the key for messages encrypted with Enigma. Other rotor machines include Lorenz, Purple, and Fialka (which is another shade of purple).

We have not mentioned *Kerckhoffs's principles* yet. He summarized some basic properties of a good cryptographic system, such as its ease of use and portability. His second principle is that the security of the system should not depend on the secrecy of the algorithm (or mechanical device), but rather on the secrecy of the keys. One's enemy can often find and steal the details of the system or device, so we should not rely on keeping them hidden. Instead, the system should be strong enough that an enemy cannot break it without knowing the keys. As you know by now, none of the classical ciphers are secure in this light. Furthermore, when a new classical cipher comes along, we can often figure out the scheme and break it without prior knowledge of it. In the modern era, however, Kerckhoffs's second principle is taken very seriously.

The modern era is characterized by the use of computers. With them comes a level of complexity that makes it impossible to break modern cryptographic systems with pen and paper. But on the other hand, new structures and uses for them arise. The ideas that you should carry forward from the classical into the modern era are substitution, transposition, and fractionation; these ideas are used, albeit in more complicated ways. Here are some of the major differences you will see as you study the cryptography of the modern era:

- Algorithms are far more complex, and therefore...
- We must use computers
- Algorithms are publicly known. In fact, there are competitions for new algorithms. The current standard, *AES* ("*Advanced Encryption Standard*") employs the algorithm that won a competition held by the U.S. National Institute of Standards and Technology (NIST). Such competitions take years to complete, as the community evaluates each algorithm and tries to find its weaknesses. Since everyone knows the algorithms, security rests in the secrecy of the keys (Kerckhoffs's second principle).
- *Asymmetric* (*public-key*) ciphers now allow someone to encrypt a message for a recipient s/he has never met before. Such a cipher has two keys; one locks it, and the other unlocks it. The recipient published his/her locking key (the public key) for anyone to use. Messages meant for the recipient are encrypted with the public key, and only the recipient can decrypt with the other, private, key.
- Public-key cryptography also allows us to digitally sign messages. If a message is encrypted with a private key, then anyone can use the public key to decrypt it, thereby proving that it was written by the individual holding the private key. To make this process easier, we only need to encrypt a shorter text, which is a secure digest of the message; this brings us to...
- *Hash functions*: these are functions that take a message and produce a large number, called the *digest* of the message. These functions are special in that it is easy to find the digest of a message, but nearly impossible to recover the message from the digest. For this reason, they are called *one-way functions*.
- The security of a cryptographic system is demonstrated by reducing it to a hard mathematical problem. For example, the *RSA* public-key system uses the difficulty of factoring large integers as the basis of its security.

The quantum era is now beginning. Quantum computers will allow us to solve some classically difficult mathematical problems easily. For example, once a quantum computer can be built that is large enough (really no larger in terms of memory and processing than computers that we now have), it will be able to factor large integers almost instantly. This will mean the death of RSA. Therefore, we need new algorithms, algorithms that are resistant to quantum computing. There is currently a competition at NIST for such things. Stay tuned.

Just keep learning.

Reading and references

American Cryptogram Association, "The ACA and You," http://www.cryptogram.org/cdb/aca.info/aca.and.you/aca.and.you.pdf, 2005 edition: http://web.archive.org/web/*/http://www.cryptogram.org/cdb/aca.info/aca.and.you/aca.and.you.pdf, 2016 edition: web.archive.org/web/*/http://cryptogram.org/docs/acayou16.pdf

Auguste Kerckhoffs, “La cryptographie militaire,” *Journal des sciences militaires* IX (1883) 5-39 and 161-191, www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf, www.petitcolas.net/kerckhoffs/crypto_militaire_2.pdf

Turing Bombe Tutorial, www.lysator.liu.se/~koma/turingbombe/TuringBombeTutorial.pdf

Whitfield Diffie and Martin E. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory* 22 (1976) 644-654, ee.stanford.edu/~hellman/publications/24.pdf

Exercises

1. Find a cipher you have never used before and learn how it works. A good place to start is www.cryptogram.org/resource-area/cipher-types. Can you find a weakness in the cipher? Can you modify an attack that you have to break this cipher? Give it a try.

2. Repeat Exercise 1 as often as you like.

3. Read about the rotor machines from the mechanical cryptographic era, about modern ciphers and hash functions, and about quantum key distribution.

Index

Numbers refer to units, not to pages. Numbers in italics refer to the location of a term's definition. Items in `monospaced typerwriter font` are programming items.

Bibliography

American Cryptogram Association, “The ACA and You,” www.cryptogram.org/cdb/aca.info/aca.and.you/aca.and.you.pdf, 2005 edition: web.archive.org/web/*/http://www.cryptogram.org/cdb/aca.info/aca.and.you/aca.and.you.pdf, 2016 edition: web.archive.org/web/*/http://cryptogram.org/docs/acayou16.pdf; the pages about ciphers are linked from this page: www.cryptogram.org/resource-area/cipher-types

Francis Bacon, *Of the proficience and advancement of Learning, divine and humane*, London: Henrie Tomes, 1605.

Thomas H. Barr and Andrew J. Simoson, “Twisting the Keyword Length from a Vigenère Cipher,” *Cryptologia* 39:4 (2015) 335-341, DOI: 10.1080/01611194.2014.988365

Friedrich L. Bauer, *Decrypted Secrets: Methods and Maxims of Cryptology*, 4th edition, Berlin: Springer-Verlag, 2007.

Étienne Bazeries, *Les Ciffres Secrets Dévoilés*, Paris: Charpentier et Fasquelle, 1901, books.googleusercontent.com/books/content?req=AKW5Q...

Giovan Battista Bellaso, *La Cifra del Sig. Giouan Battista Belaso* [sic], 1553.

Paolo Bonavoglia, “Bellaso’s 1552 cipher recovered in Venice,” *Cryptologia* 43:6 (2019) 459-465, DOI: 10.1080/01611194.2019.1596181

Paolo Bonavoglia, La crittografia da Atbash a RSA, www.crittologia.eu, 2020.

Paolo Bonavoglia, “Trithemius, Bellaso, Vigenère: Origins of the Polyalphabetic Ciphers,” Proceedings of the 3rd International Conference on Historical Cryptology, 2020, ep.liu.se/ecp/171/007/ecp2020_171_007.pdf, DOI: 10.3384/ecp2020171007

Augusto Buonafalce, “Bellaso’s Reciprocal Ciphers,” *Cryptologia* 30:1 (2006) 39-51, DOI: 10.1080/01611190500383581

Pliny Earle Chase, “Mathematical Holocryptic Cyphers,” *The Mathematical Monthly* 1:6 (1859) 194-196, books.google.com/books?id=SVNLAAAAMAAJ&pg=PA194

Chris Christensen, “Lester Hill Revisited,” *Cryptologia* 38:4 (2014) 293-332, DOI: 10.1080/01611194.2014.915260

Benjamin Church, Jr., George Washington Papers, Series 4, General Correspondence: Benjamin Church Jr. to Maurice Cane, July 1775, www.loc.gov/item/mgw443691

Michael J. Cowan, “Breaking Short Playfair Ciphers with the Simulated Annealing Algorithm,” *Cryptologia*, 32:1 (2008) 71-83, DOI: 10.1080/01611190701743658

Noel Currer-Briggs, “Some of Ultra’s poor relations in Algeria, Tunisia, Sicily and Italy,” *Intelligence and National Security* 2:2 (1987) 274-290, DOI: 10.1080/02684528708431890

Donald W. Davies, “Charles Wheatstone’s Cryptograph and Pletts’ Cipher Machine,” *Cryptologia* 9:2 (1985) 155-160, DOI: 10.1080/0161-118591859870

Félix-Marie Delastelle, *Traité Élémentaire de Cryptographie*. Paris: Gauthier-Villars, 1902, archive.org/details/8VSUP3207b

Whitfield Diffie and Martin E. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory* 22 (1976) 644-654, ee.stanford.edu/~hellman/publications/24.pdf

Arthur Conan Doyle, “The Adventure of the Dancing Men,” first published in 1905, now in *The Complete Works of Sherlock Holmes*, London: Simon & Schuster, 2012.

Niels Faurholt, “Urkryptografen (The Clock Cryptograph),” *Cryptologia* 27:3 (2003) 206-208, DOI: 10.1080/0161-110391891874; this article is available also at www.jproc.ca/crypto/crypto_watch.html

William F. Friedman, “Codes and Ciphers (Cryptology),” *Encyclopaedia Britannica*, 1956, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/reports-research/FOLDER_535/41772109081119.pdf

William F. Friedman, Elements of Cryptanalysis, Washington D.C.: Government Printing Office, 1923, www.marshallfoundation.org/library/digital-archive/elements-cryptanalysis

William F. Friedman, The Index of Coincidence and Its Applications in Cryptography, Riverbank Laboratories Department of Ciphers Publication 22, Geneva, Illinois, 1920, www.marshallfoundation.org/library/methods-solution-ciphers

William F. Friedman, Methods for the Solution of Running-Key Ciphers, Riverbank Laboratories Department of Ciphers Publication 16, Geneva, Illinois, 1918, www.marshallfoundation.org/library/methods-solution-ciphers

William F. Friedman, Military Cryptanalysis, Part I: Monoalphabetic Substitution Systems, Washington D.C.: U.S. Government Printing Office, various years for various editions.

William F. Friedman, Military Cryptanalysis, Part II: Simpler Varieties of Polyalphabetic Substitution Systems, Washington D.C.: U.S. Government Printing Office, various years for various editions.

William F. Friedman, Military Cryptanalysis, Part III: Simpler Varieties of Aperiodic Substitution Systems, Washington D.C.: U.S. Government Printing Office, various years for various editions.

William F. Friedman, Military Cryptanalysis, Part IV: Transposition and Fractionating Systems, Washington D.C.: U.S. Government Printing Office, various years for various editions.

William F. Friedman, Several Machine Ciphers and Methods for their Solution, Riverbank Laboratories Department of Ciphers Publication No. 20, 1918, www.campx.ca/Several_Machine_Ciphers.pdf and www.marshallfoundation.org/library/methods-solution-ciphers

William F. Friedman, Six Lectures on Cryptology, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/ACC15281/41785109082412.pdf

William F. Friedman and Lambros D. Callimahos, Military cryptanalytics, Parts I through IV, Aegean Park Press, 1956, reprinted 1985.

Helen Fouché Gaines, *Cryptanalysis: a study of ciphers and their solution*, New York: Dover, 1956; previously titled *Elementary Cryptanalysis* and published by American Photographic in 1939; archive.org/details/cryptanalysis00gain

Greg Goebel, *Codes, Ciphers, & Codebreaking*, vc.airvectors.net/ttcode.html

Lester S. Hill, "Cryptography in the Algebraic Alphabet," *The American Mathematical Monthly* 36:6 (1929) 306-312, DOI: 10.2307/2298294, www.jstor.org/stable/2298294, web.archive.org/web/20110719235517/http://w08.middlebury.edu/INTD1065A/Lectures/Hill Cipher Folder/Hill1.pdf

Lester S. Hill, "Concerning Certain Linear Transformation Apparatus of Cryptography," *The American Mathematical Monthly* 38:3 (1931) 135-154, DOI: 10.1080/00029890.1931.11987161, www.jstor.org/stable/2300969, www.cs.jhu.edu/~cgarman/files/Hill2.pdf

Parker Hitt, *Manual for the Solution of Military Ciphers*, Fort Levenworth (Kansas): Press of the Army Service Schools, 1916, www.marshallfoundation.org/library/digital-archive/manual-solution-military-ciphers, www.gutenberg.org/ebooks/48871

Thomas Jakobsen, "A fast method for cryptanalysis of substitution ciphers," *Cryptologia* 19:3 (1995) 265-274, DOI: 10.1080/0161-119591883944

Thomas Jefferson, "The wheel cypher" or "Project of a cypher," Thomas Jefferson's Papers, volume 128 item 22138, volume 232 items 41575 and 41576, U.S. Library of Congress, www.loc.gov/item/mtjbib025756, founders.archives.gov/documents/Jefferson/01-37-02-0082

Thomas Kaeding, "Automated ciphertext-only attack on the Wheatstone Cryptograph and related devices," Cryptology ePrint Archive, report *2020/1492*.

Thomas Kaeding, "MadHatter: A toy cipher that conceals two plaintexts in the same ciphertext," Cryptology ePrint Archive, report 2020/301.

Thomas Kaeding, “Slippery hill-climbing technique for ciphertext-only cryptanalysis of periodic polyalphabetic substitution ciphers,” *Cryptologia* 44:3 (2020) 205-222, DOI: 10.1080/01611194.2019.1655504

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996.

Bruce Kallick, “A Modified Simple Substitution Cipher With Unbounded Unicity Distance,” Cryptology ePrint Archive, report 2019/621.

Friedrich Kasiski, *Die Geheimschriften und die Dechiffrir-Kunst*, 1863, digital.onb.ac.at/OnbViewer/viewer.faces?doc=ABO_+Z224431001

Auguste Kerckhoffs, “La cryptographie militaire,” *Journal des sciences militaires* IX (1883) 5-39 and 161-191, www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf, www.petitcolas.net/kerckhoffs/crypto_militaire_2.pdf

Solomon Kullback, General Solution for the Double Transposition Cipher, Washington D.C.: U.S. Government Printing Office, 1934, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/FOLDER_439/41751169079035.pdf

Lanaki, lessons and tutorials, www.cryptogram.org/resource-area/crypto-lessons-tutorials-lanaki

André Langie, *De la Cryptographie: Etude sur les Ecritures secrètes*, Paris: Payot et Companie, 1918, HDL: 2027/coo.31924029486838; translated by James C.H. Macbeth as *Cryptography*, London: Constable & Company, 1922, HDL: 2027/uc1.32106002774104 and 2027/uc2.ark:/13960/t0tq62t29

James Lyons, Practical Cryptography, practicalcryptography.com, 2012.

António Machiavelo and Rogério Reis, “Automated ciphertext-only cryptanalysis of the bifid cipher,” Universidade do Porto technical report DCC-2006-1, www.dcc.fc.up.pt/~nam/publica/dcc-2006-01.pdf

Joseph O. Mauborgne, *An Advanced Problem in Cryptography and Its Solution*, Fort Leavenworth (Kansas): Press of the Army Service Schools, 1914, www.marshallfoundation.org/library/digital-archive/advanced-problem-cryptography-solution

Warren Thomas McCready (“Machiavelli”), “The Twosquare Cipher,” *The Cryptogram*, Nov-Dec 1972, 152-153.

Greg Mellen, “Cryptanalyst’s Corner,” *Cryptologia* 8:1 (1984) 55-57, DOI: 10.1080.0161-118491858773

Marjorie Mountjoy, “The bar statistics,” *NSA Technical Journal* VII (2, 4), 1963.

Émile Victor Théodore Myszkowski, *Cryptographie Indéchiffrable basée sur de nouvelles combinaisons rationnelles*, Paris: Société Française d'Imprimerie et de Librairie, 1902, gallica.bnf.fr/ark:/12148/bpt6k1265620p

Grant A. Niblo, “The University of Southampton National Cipher Challenge,” *Cryptologia* 28:3 (2004) 277-286, DOI: 10.1080/0161-110491892935 (see below for links to the challenge)

Merle E. Ohaver, “Solving Cipher Secrets,” appeared weekly in *Flynn’s*, 1924-1928, toebes.com/Flynns

Seongmin Park, Juneyeun Kim, Kookrae Cho, and Dae Hyun Yum, “Finding the key length of a Vigenère cipher: How to improve the twist algorithm,” *Cryptologia* 44:3 (2020) 197-204, DOI: 10.1080/01611194.2019.1657202

Edgar Allan Poe, “The Gold-Bug,” 1843, en.wikisource.org/wiki/Tales_(Poe)/The_Gold-Bug, www.eapoe.org/works/tales/goldbga2.htm

Giambattista della Porta [Giovanni Battista della Porta] [Ioan. Baptista Porta], *De Furtivis Literarum Notis*, Naples [Neapoli]: Ioa. Maria Scotus, 1563, HDL: 2027/gri.ark:/13960/t37142x6g

Fletcher Pratt, *Secret and Urgent: The Story of Codes and Ciphers*, New York: Bobbs-Merrill, 1939.

Bruce Schneier, “The Solitaire Encryption Algorithm,” Schneier on Security, www.schneier.com/academic/solitaire

Claude E. Shannon, “A Mathematical Theory of Communication,” *Bell System Technical Journal* 27:3 (1948) 379-423, DOI: 10.1002/j.1538-7305.1948.tb01338.x, HDL: 11858/00-001M-0000-002C-4314-2

Simon Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, New York: Random House, 1999.

Abraham Sinkov, *Elementary Cryptanalysis: A Mathematical Approach*, 2nd edition, revised by Todd Feil, published by Mathematical Association of America, 2009, www.jstor.org/stable/10.4169/j.ctt19b9krf

W. W. Smith, “Solution of the Playfair Cipher,” in part IV of André Langie, *Cryptography*, translated by James C. H. Macbeth, London: Constable & Company, 1922, HDL: 2027/uc1.32106002774104 and 2027/uc2.ark:/13960/t0tq62t29

James Stanley, “The Wheatstone Cryptograph,” incoherency.co.uk/blog/stories/wheatstone-cryptograph.html

S. Tomokiyo, “First Codebreaking in the American Revolution — Benjamin Church’s Cipher,” cryptiana.web.fc2.com/code/church.htm, 2009-2014.

Johannes Trithemius, *Polygraphiae libri sex*, Reichenau: Joannis Haselberg de Aia, 1518, www.loc.gov/item/32017914

Blaise de Vigenère, *Traicté des chiffres ou secrètes manières d’escrire*, Paris: Abel l’Angelier, 1586, HDL: 2027/ien.35552000251008, gallica.bnf.fr/ark:/12148/bpt6k1040608n, gallica.bnf.fr/ark:/12148/bpt6k94009991

Charles Wheatstone, “Instructions for the Employment of Wheatstone’s Cryptograph,” *The Scientific Papers of Sir Charles Wheatstone*, The Physical Society of London, 1879, pages 342-347. archive.org/details/scientificpaper00londgoog (the last two pages of the article were completely ruined by Google in that copy), books.google.to/books?id=CtGEAAAAIAAJ

Fred B. Wrixon, *Codes, Ciphers & Other Cryptic & Clandestine Communication*, New York: Black Dog & Leventhal, 1998.

“Ciphers and Cipher-Writing,” Macmillan’s Magazine, XXIII, Feb 1871, pages 328-338, babel.hathitrust.org/cgi/pt?id=mdp.39015004979913;view=1up;seq=340

NSA file 41788379082740: www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/patent-equipment/FOLDER_515/41788379082740.pdf

Basic Cryptography, Dept. of the Army Technical Manual 32-220, April 1950, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/FOLDER_238/41748889078809.pdf

General Solution for the ADFGVX Cipher System, Washington D.C.: U.S. Government Printing Office, 1934, www.nsa.gov/Portals/70/documents/news-features/declassified-documents/friedman-documents/publications/FOLDER_269/41784769082379.pdf, archive.org/details/41784769082379

NOVA Online, “Decoding Nazi Secrets,” www.pbs.org/wgbh/nova/decoding

United States Army, Field Manual 34-40-2, Basic Cryptanalysis, U.S. Department of Army, www.umich.edu/~umich/fm-34-40-2

MysteryTwister C3, www.mysterytwisterc3.org

RingZer0 Online CTF, ringzer0ctf.com

(British) National Cipher Challenge, www.cipherchallenge.org. Recent years’ challenges are also on the site. An archive of past challenges is at github.com/themaddoctor/BritishNationalCipherChallenge