Part IX
Miscellaneous ciphers

madness's book on classical cryptography
unit 105: symbolic substitution
last modified 2021-01-18

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*, New York: Bobbs-Merrill, 1939, 147.83.93.163/cops/fetch.php?data=4538&type=pdf&id=2942, 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.

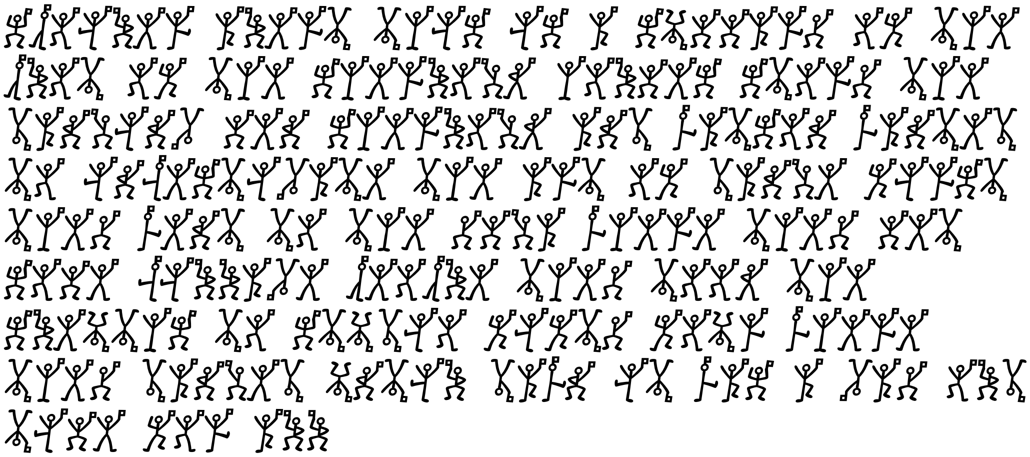
Martin Gardner, *Codes, Ciphers and Secret Writing*, New York: Simon & Schuster, 1972, sections 4, 6, and 7.

Exercises

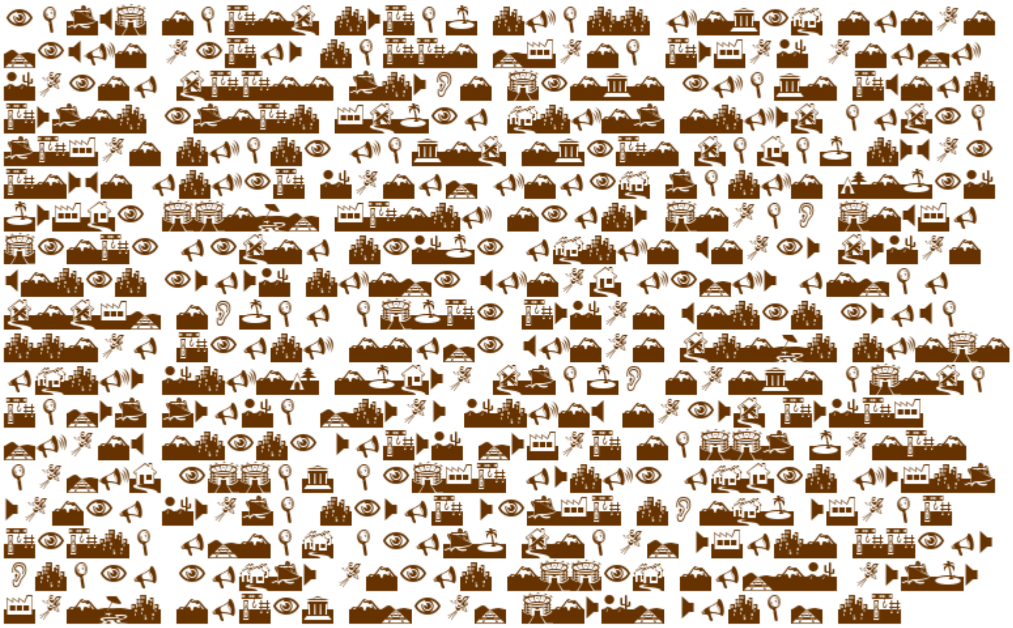
1. This is the ciphertext from Edgar Allan Poe’s “The Gold-Bug” (the original version). Break it.

```
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;‡?;
```

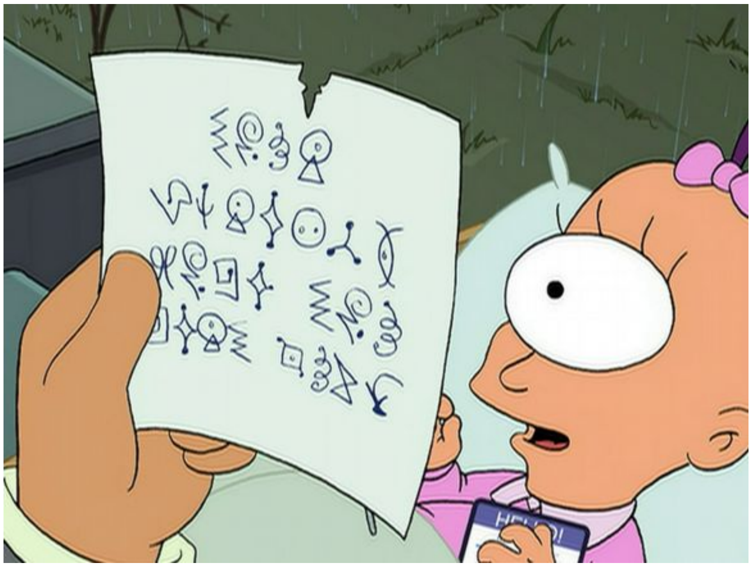
2. This ciphertext uses the symbols from the Sherlock Holmes story “The Adventure of the Dancing Men.” Some symbols did not appear in the story and were designed by the font’s creator. Break the ciphertext.



3. This ciphertext is from the 2002 British National Cipher Challenge. Break it.



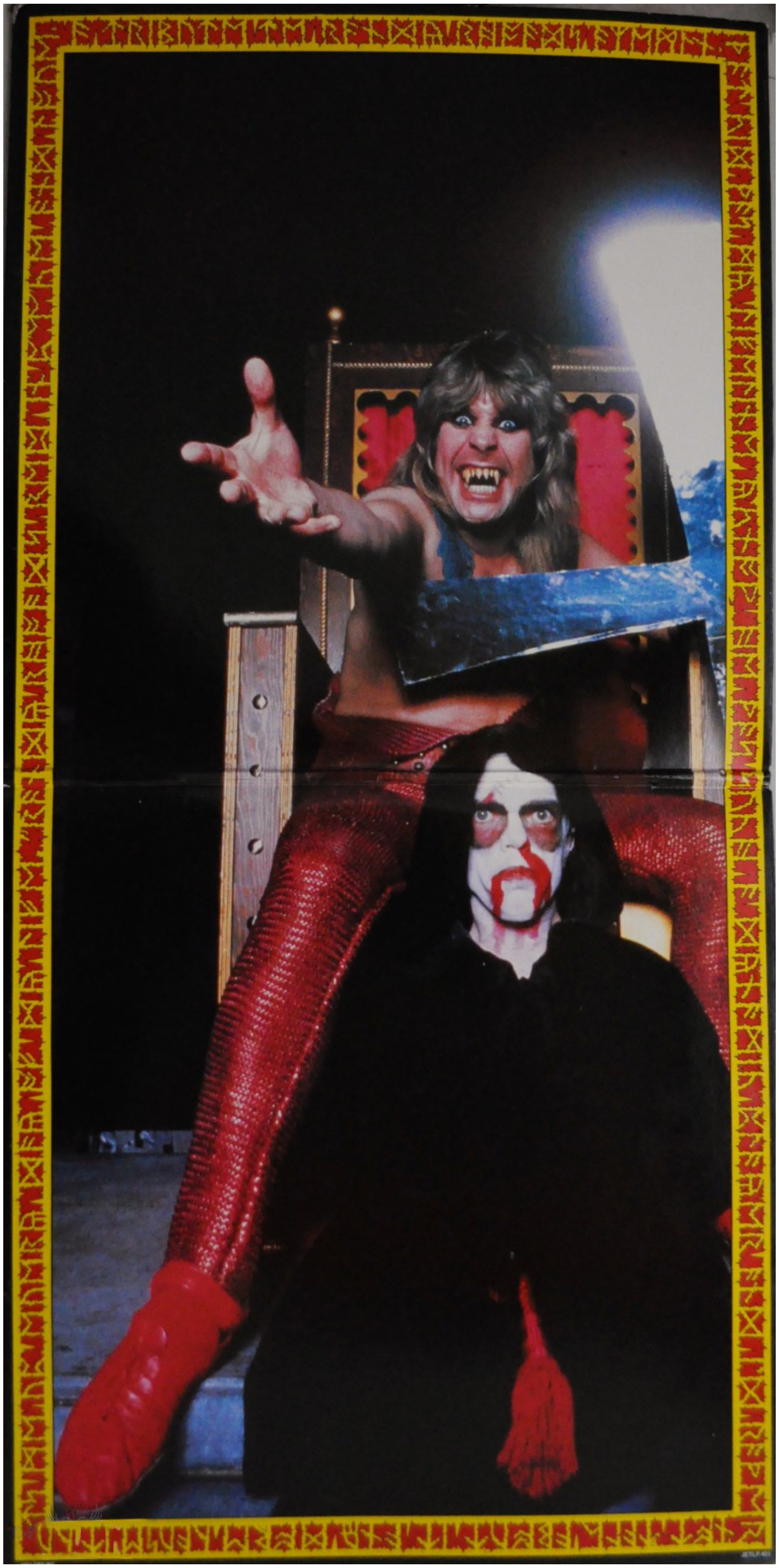
4. What does the note say?



5. >⊓ΓV ΓV >⊓□ ·⌉Γ⌉·⌉□⊡ LΓ·⌉⊓□Γ· ⌋L·V⊏ ⊔·⊡⊏·V·⊡ ⌋V >⊓□
⊏Γ·□□⊐·⌋V⊏·⊡V LΓ·⌉⊓□Γ· ⊏·Γ· >⊓□ Γ·⊏·VΓLΓ·<LΓ⌋⊡V LΓ·⌉⊓□Γ·

⋖⊏< ⌝⌜⊏⊔⌟⊔⌞⋖ ⌟⌞⌜□⌟⊐⋖ ⊔⊡⊏V ⌟⊔⊏<> ⌜> V⌜⊡⌞□
⌞⊓⌜⌞⊐⌜□⊡ <V□ ⌜> ⊏⊏>□⊡ ⌜>V V⊓⌟⌝□V ⌟⌜□ ⊏⌜⊏⊐ ⌟
>⌜⌞>⌟⌞>⊏□ ⊔⊏⌟⌜⊐ ⌟⊡⊐ ⊏⌜⊏⊐ ⌟⊡ ⋗ ⊔⊏>⊓ V⌜>⊓ ⌟⊡⊐
V⌜>⊓⊏<> ⌟ ⊐⊏>

6. What is Ozzy trying to tell us?



madness's book on classical cryptography
unit 106: one-time pad
last modified 2020-10-16

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
```

madness's book on classical cryptography
unit 107: slidefair cipher
last modified 2020-10-03

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
```

madness's book on classical cryptography
unit 108: nicodemus cipher
last modified 2020-10-28

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
```

madness's book on classical cryptography
unit 109: fractionated morse
last modified 2020-10-28

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.

Reading and references

Practical Cryptography, practicalcryptography.com/ciphers/fractionated-morse-cipher

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

Programming tasks

1. Implement an encryptor.

2. Implement a decryptor.

3. Implement a dictionary attack. Remember that there are many ways to construct a mixed alphabet from a keyword.

4. Encrypt your textual corpus that contains spaces but no punctuation with the key `ABCDEFGHIJKLMNOPQRSTUVWXYZ`.

5. Compile a table of tetragram frequencies from the encrypted corpus.

6. Implement the hill-climbing attack described above.

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
```

madness's book on classical cryptography
unit 110: hutton cipher
last modified 2020-10-28

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
```

madness's book on classical cryptography
unit 111: scrabble cipher
last modified 2020-10-28

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

   K E Y W O R D A B C F G H I J L M N P Q S T U V X Z

2. Shuffle the alphabet with a keyword. Here is our example with the keyword `KEYWORD`. We begin with the standard 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

   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’:

   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
   ↕
   L M N O P Q R S T U V W X Y Z K A B C D E F G H I J

   Then we take the result and apply a swap around the next key letter, ‘E’:

   L M N O P Q R S T U V W X Y Z K A B C D E F G H I J
   ↕
   F G H I J E L M N O P Q R S T U V W X Y Z K A B C D

We continue and use the remaining letters of `KEYWORD` to get this keyed alphabet:

Y F G H I J E L M N R P Q D S T U V O X W Z K A B C

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’:

```
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
                                    ↓
Y F G H I J E L M N R P Q D S T U V O X W Z K A B C
```

We get an ‘O’. Next, shuffle the alphabet around the letter ‘S’:

```
Y F G H I J E L M N R P Q D S T U V O X W Z K A B C
                              ↕
      T U V O X W Z K A B C S Y F G H I J E L M N R P Q D
```

This new alphabet will be used to encrypt the next letter of the message.

```
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
        ↓
T U V O X W Z K A B C S Y F G H I J E L M N R P Q D
```

We get ‘X.’ Another shuffling, this time around the plaintext letter ‘e’:

```
T U V O X W Z K A B C S Y F G H I J E L M N R P Q D
                                    ↕
                      L M N R P Q D E T U V O X W Z K A B C S Y F G H I J
```

This continues until the entire message is encrypted.

```
s e n d h e l p
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
O X W D Z M S A
```

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...
```

madness's book on classical cryptography
unit 112: homophonic substitution
last modified 2020-10-28

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.

Python tips

The `choice()` function from the `random` module chooses an element from a list or string randomly.

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*, New York: Bobbs-Merrill, 1939, 147.83.93.163/cops/fetch.php?data=4538&type=pdf&id=2942, 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!"
```

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
```

madness's book on classical cryptography
unit 113: polyphonic substitution
last modified 2020-10-28

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*, New York: Bobbs-Merrill, 1939, 147.83.93.163/cops/fetch.php?data=4538&type=pdf&id=2942, 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
```

madness's book on classical cryptography
unit 114: polyhomophonic substitution
last modified 2020-10-28

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
```

madness's book on classical cryptography
unit 115: pollux cipher
last modified 2021-02-08

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

```
32505074465102238606055614121828842422198749200477528129481
24352943365835636123046203371568604447658946933208557031471
5868418
```

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) ↔ (−, ×, ·, ·, −, ×, ×, ·, ·, −).

```
08212864285502991999172916039593104067495506377812459 1
```

4. Break this ciphertext with your brute-force attack.

5. Break this ciphertext with your hill-climbing attack. You will find that it is possible despite the brevity of the text.

   ```
   194925739853721143540660461564856257706111381886975538
   ```

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.

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

madness's book on classical cryptography
unit 117: duplicitous ciphers
last modified 2020-10-28

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
110011101000110100012000111001110111112222
```

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
```

```
6103861938175185498550455127250790908570156858918507O7
38433086863834198515383452512798503034477474 74
```

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
```

```
2312000001000010223010002120012302000000020223122000000000
2302200111001113220100000000000022331000010000232020000000000
0230120000000100300010000000001123002000000000003201200000001
0030001000001000020123000100101200010000100001022012000000000
2003230301100120023112010000120002101200001001113001200011 2
0010221100100000111032020101012000312031110012122212110110 0
0000123033110012002111202000001111210231010011220020223011 0
1011200121211100121022212001000010010123000100111012333201 0
1001012120211100100200101230011110001230320000011020112020 0
0110010030102000100121202021101000011000223000001100021012 0
000000111300120000012222230333
```

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
```

madness's book on classical cryptography
unit 118: combination-lock cipher
last modified 2020-10-28

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	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
```