Part VII
Stream ciphers

Unit 90
Stream ciphers

A *stream cipher* enciphers characters of the plaintext one at a time. It generates a *key stream*, which is a pseudorandom stream of characters. Each character of the key stream is combined with one character of the plaintext to give one character of the ciphertext. To decipher, the same key stream is used to recover each character of the plaintext from a character of the ciphertext. If the key stream is generated independently from the text, then the cipher is called *synchronous*. If the key stream depends on characters from the text, then it is called *asynchronous*. If that dependence is on the last few ciphertext characters, then the cipher is *self-synchronizing* (or *ciphertext-autokey*), because the receiver of the ciphertext can resynchronize after reading a few ciphertext characters.

There are three variations on how key-stream characters are combined with plaintext characters. Each corresponds to one of the polyalphabetic ciphers that use unmixed key alphabets.

- Key-stream characters are added to plaintext characters, as in the Vigenère cipher. This is the standard variation.
- Key-stream characters are subtracted from plaintext characters, as in the variant Beaufort.
- Plaintext characters are subtracted from key-stream characters. This is similar to the Beaufort cipher.

Whatever information the encryptor of a stream cipher stores as is goes from one plaintext letter to the next is its *internal state*. Typically, the key is used to generate the initial contents of the internal state. The state changes as the encryptor advances through the plaintext. The key stream is generated from the state.

In contrast to stream ciphers, ciphers which act on a block of text at a time are called *block ciphers*. For example, periodic polyalphabetic substitution ciphers and permutation ciphers take blocks of a fixed number of plaintext characters and encipher each block in the same way.

Reading and references

Wikipedia, en.wikipedia.org/wiki/Stream_cipher

Exercises

1. Decrypt this ciphertext which was encrypted with a synchronous stream cipher, and the key stream is based on the Fibonacci sequence. (The Fibonacci sequence in modular arithmetic is periodic, by the way.)

```
ZPWFFVUVDHXCQRRTYGDYHFWECMWAHGIUWVZBCDLXHTGZNLIVNSVQWM
UVGCTSFPSZOXPHMYKAAFVLJSYBTSIAUMGDUANILUFBRBHDCDCJPJGL
ADGFFSBVLJQWOKGOFSSNWLPSYQLETFJZSXWYQZTCZMLUQEHZLMWSQB
ZPVRUAZTUWNTYVVGWSSFKNFNJYHNSQHSCVUOIKGPFIIGEBLERBYKWW
CQSVTPXVHIBEADNBRQEFDNSQGBZDTSRBKSJSBTHUJAUVALQBNBZDSK
YULFDOSVWGIWIEWABZZ
```

Unit 91
Trithemius cipher

The simplest stream cipher is the *Trithemius cipher*. It is synchronous. We can look at this cipher in three ways. First, we can say that it starts with a shift of zero and adds one to the shift with each character. Shifts are applied to plaintext characters modulo 26. Second, we can say that the cipher generates the key stream `ABC...XYZABC...XYZABC...` which is added to the plaintext to get the ciphertext. Third, we can view it as a Vigenère cipher with key `ABCDEFGHIJKLMNOPQRSTUVWXYZ`.

As explained above for general stream ciphers, the Trithemius cipher has three variations on how key-stream characters are combined with plaintext characters. In addition, there is another variation in which the initial shift is taken to be nonzero.

Like the atbash cipher, the Trithemius cipher has almost no security. Once the adversary knows the cipher, s/he can decrypt any ciphertext.

The internal state S of the encryptor is a simple counter. As we have done before, we can think of the plaintext $P = \{p_i\}$ and ciphertext $C = \{c_i\}$ as each a series of integers. In these terms, the action of the encryptor for the standard version of the Trithemius cipher is

1. Set $S = 0$
2. For each p_i in P
   a. $c_i = p_i + S$ modulo 26
   b. $S = S + 1$

Reading and references

Wikipedia, en.wikipedia.org/wiki/Tabula_recta#Trithemius_cipher

Johannes Trithemius, *Polygraphiae libri sex*, Reichenau: Joannis Haselberg de Aia, 1518, www.loc.gov/item/32017914, book 5.

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 135-136.

Fletcher Pratt, *Secret and Urgent*, New York: Bobbs-Merrill, 1939, 147.83.93.163/cops/fetch.php?data=4538&type=pdf&id=2942, chapter XI, section III.

Programming tasks

1. Write a function or script to encipher a plaintext with the Trithemius cipher. Give your function some optional arguments to allow for the three variations on the method of combining key-stream characters with plaintext characters, and for the possibility of a nonzero initial shift.

2. Write a function or script to decipher a ciphertext with the Trithemius cipher. Give your function some optional arguments to allow for the three variations on the method of combining key-stream characters with plaintext characters, and for the possibility of a nonzero initial shift.

3. Write a function or script to break a ciphertext that was encrypted with some variation of the Trithemius cipher. Use tetragram fitness of the plaintext to determine when you have found the right variation.

Exercises

1. Break this ciphertext:

```
PVPJXYQFHGYKZXGDQPUZSDCXKMZPVJCYAJGIEIEHHCEPUZYIQPIFTK
BXYVDRTJWVAGUWAKCCIDVLRJGIJNISQFHPOVVMBZNBYMDUCXKMZAPN
WZZXGWENLHJVNQWCOXPDRIMLBFSETGVTDLFLCICOYSIUWGIKUYPRCS
QYDLYKZFIMWICSHFAHPJIAMMRTYNPQEKLHIHQZFOASVBMSMSMJNEAU
WHTVSMBVJAFDWBUEYWOLLJDETROQRVQ
```

2. Break this ciphertext:

```
HSUFOWQFNYJLEJOTVJGDYNRPJITVKKLXSFFZIMKWMSCVVLZBRBLXAG
QKQVCPQXZLKAKUMUPZSSMFKIWNTJFRXTGCFLVGIZWWUHVQXDBXHRWF
CUCAKYWWGZZJCUBDPYELCGWIAMWEOPINSXZNKHJPPGCDEJKXMUFQSE
QNYGQYGUOXFFVBVXMWJXDGQEODQTATXBNWILAIEPNQNDAISXUPECGG
OWMYUVILISKSNRYMCMTRWFYDBPGXDWGCESDJQGCBRZFBORLTTVHQZN
YWKKLNRCEFSTSJPTHBKSDRHLZYQFKVOHKSQOSVRPGTVZWPHOQCKZYL
MTVHKWFDLBRKVXNGRTFRMBAYIEDUFHVVQPYEWSUICGRXDVNLPOMCSJ
MBXFQPYSWKGTPOXTTNCREWMYKKEYRCECMMKMYBNWHTJSQCKYICWJKR
TF
```

Challenge

```
TAEVZRVZLPSLZAERFZRVDQMMUNVIOPNGNRCKDWHLOCQHZNFMQMQDWXXZJCQ
JQUNNOZDKKVAHVKSFLDKOAIOVOTBORRESLSWNXBZNAIBPGGZZMKBBUYYZNB
LZEVPAEWBPGYFZSLTGMBFTITXRGRZJNGKVZRTHWASLSASLTFJBIMRFZMPPA
IBUFJKYRKVZNAIBXEMOVXBSLLTGRPLLDQEXEIZDEINBSWWJMDRLYBUBFSWY
```

IWBUBFGUNGRVRRJJYYNRITTUYJFNGNGNRNBSVNQHPCMYCDHHBUCHBMXVKBJ
WPTWKXFYGTEEVBFQUNGKBT

madness's book on classical cryptography
unit 92: autokey cipher
last modified 2020-10-18

Unit 92
Autokey cipher

The *autokey cipher* (or *autoclave cipher*) uses a key stream that begins with a keyword which is followed by the plaintext itself. Hence the name of the cipher. It is asynchronous. In the standard version of this cipher, key-stream characters are added to plaintext characters, modulo 26.

An example will help to elucidate. Let us encipher this short text with the key `AUTOKEY`.

`THIS TEXT WAS ENCRYPTED WITH AN AUTOKEY CIPHER`

The key stream is written under the text, and the ciphertext under that is the sum of the two, modulo 26:

plaintext:	`THISTEXTWASENCRYPTEDWITHANAUTOKEYCIPHER`
key stream:	`AUTOKEYTHISTEXTWASENCRYPTEDWITHANAUTOKE...`
ciphertext:	`TBBGDIVMDIKXRZKUPLIQYZRWTRDQBHRELCCIVOV`

If the length of the key is L, then the internal state $S = (s_0, s_1, ..., s_{L-1})$ of the autokey cipher is the last L characters encountered by the encryption routine. The state is initialized by filling it with the letters of the key. For each letter in the plaintext, the character of the key stream is found by shifting the state vector to the left. The leftmost character is popped off the left end of the state vector to become the key-stream character. The current plaintext character is pushed into the vacancy at the right end of the state vector.

Let’s rework our example in the language of state vectors, just to belabor the point. The state is initialized by filling it with the key:

S = (‘A,’ ‘U,’ ‘T,’ ‘O,’ ‘K,’ ‘E,’ ‘Y’)

The first plaintext character is ‘T.’ We push it onto the right end of S and pop off the first key-stream character, ‘A.’

S = (‘U,’ ‘T,’ ‘O,’ ‘K,’ ‘E,’ ‘Y,’ ‘T’)

k_0 = ‘A’

The next plaintext character is ‘H,’ which we push onto the right, and pop off a ‘U.’

$$S = (\text{‘T,’ ‘O,’ ‘K,’ ‘E,’ ‘Y,’ ‘T,’ ‘H’})$$

$$k_1 = \text{‘U’}$$

You get the idea. This continues until the full keystream is generated.

There are three variations of the cipher that correspond to the three variations of combining keystream characters with plaintext characters: Vigenère (standard), Beaufort, and variant Beaufort.

Reading and references

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

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

Fletcher Pratt, *Secret and Urgent*, New York: Bobbs-Merrill, 1939, 147.83.93.163/cops/fetch.php?data=4538&type=pdf&id=2942, chapter XI, section II.

Wikipedia, en.wikipedia.org/wiki/Autokey_cipher

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

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

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 147-148.

Programming tasks

1. Write a function or script to encipher a plaintext with an autokey cipher. Allow for the possibility of choosing any of the three variations of the cipher.
2. Write a function or script to decipher a ciphertext with an autokey cipher. Allow for the possibility of choosing any of the three variations of the cipher.
3. Implement a dictionary attack on the autokey cipher.

Exercises

1. Encipher the following text with the keyword `STREAM`,

a. with the standard autokey cipher (the Vigenère-like version),
b. with the Beaufort-like variation,
c. with the variant-Beaufort-like variation.

   The deep hollows which separate the hills are thickly covered with fern and heather, over which blocks of granite are scattered in all directions; and, as in all similar districts, each valley has its own clear mountain stream, which receives the innumerable waterfalls descending from the hill-sides.

   (from *Devon: Its Moorlands, Streams and Coasts* by Rosalind Northcote)

2. Decipher this ciphertext with the keyword `RIVERBED` and the standard autokey cipher.

   ```
   KPZWYFIUKVGCZEGIFGUDZEUKFWSFJYLHWJFDQGMVJQNXVFPLVPOXLR
   WRUKXZQPIFLTXCBZVTIAQOEWACAAUNDKNSDFDASIOFUKTISHSBFUOM
   CPVGCWQSSXKASGPVWRYARQFGEOIXGLWEXOYEUSIFSSDRJYLHZTQKIO
   SSAGZXCPYGQCRFNSFVMCNOPWKYSIRFLSVTKVHLOEDPKQLGHXIPBANQ
   ELWZNHNRRSBOGTDSGHRYGLGIRHWOOYJPGFWESFQXMQEYGZGXC
   ```

3. Decipher this ciphertext with the keyword `FUTURE` and the Beaufort-like variation of the cipher.

   ```
   MNPWKEQCQEUXKYKMWTPJLUYDAMVWADNOKYAQMFGIPNLAPFADSFZWBL
   OVBRMAGATHOUBPJJHTPRDUSCZZVDASTVPLXYYMXWEGYHPWAJQWVWLA
   INXQDAYLTZWMERQPBDRW
   ```

4. Decipher this ciphertext with the keyword `ELECTRIC` and the variant-Beaufort-like variation of the cipher.

   ```
   EPIGUCJSLHHZRZUGCXVYPPTMFOFZYDAMAVKWVVKTCRATCJNNWPELOA
   LICUWOMROBYBKMICOVUIVOITZVJVSDAEALTPHTNWFUTSHGAOEFPCEK
   OAVGSNIPZMWKLBOSDAEATQZDNGQQQPBGIBBA
   ```

5. Break this ciphertext with a dictionary attack.

   ```
   IVIDVWYYSWAFNTVFDLJQVDAELSGSMSXEEMPWGRGWPQTPOFEACINUPH
   QYEPXMZSVVHPYXBROOHKVWOXNYIXWWLASKMEYEWXLCUPBKAWSHTALF
   WBMJGJXMDBZWYWDEMZMPAJDVTBQJTHNXQJTHVORWGUIYGKLHSPXKPI
   MUFPNWNLPBYWUXXKXQPRGLGVWATPKGSOKEEROOXAAQQNRWSUZTRWSE
   NRMVPASQHBDJSAFRJYHIEJCHDEEXTBOXFJQRKTGJQQHLWSAPPTWIWV
   ```

6. Break this ciphertext with a dictionary attack.

   ```
   PVALSAXRIUTGBRNFNMPKKRATZLWEMOACVIFCJDVZNABCNIMLXYIAHS
   OGPTEAGKLSKMWGSVBKYKTCSDPIRSSWKPJTFHCEXGSVQTBCHQJVZGEW
   CVCICDXYBQLKRWLXJMKDWRKZVSKXOPVXQQCWKFAWDKHJTDNOULXXNM
   GPVRQUZFBZRLMVGDNPBHTYWQNNAABKFLATGRIGQHTCHGDHDETMWSEP
   SSAJZERDTHXWTMRUQ
   ```

Unit 93
Hill-climbing attack on the autokey cipher

This is a modification of the hill-climbing attack that we used against the Vigenère cipher. Given a key length, we vary each character of a key in turn until we cannot increase the fitness of the decrypted plaintext any further.

The algorithm follows. The purpose of the flag is to exit the loop if it has run through all characters of the key but not improved the fitness.

1. set the best fitness equal to the fitness of the unaltered ciphertext
2. choose a random parent key with the given key length
3. set flag to FALSE
4. while the flag is not TRUE
    b. set flag to TRUE
    a. for *i* running from 0 to key length minus 1
        i. copy the parent key into a child key
        ii. for *x* running over A, B, ..., Z
            - set the i^{th} character of the child key to *x*
            - decrypt the ciphertext with the child key to get a plaintext
            - calculate a new fitness of the plaintext
            - if the new fitness is greater than the best fitness
                · set the best fitness equal to the new fitness
                · replace the parent key with the child key
                · set the flag to FALSE
5. output the parent key

Reading and references

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

Programming tasks

1. Implement the attack. Allow for the possibility that the cipher is one of the three variations. Feel free to copy and modify your code from the analogous attack on the Vigenère cipher.

Exercises

1. Break this ciphertext:

```
KPDWZTEEHVSSJGUUHVOTIWHZLALTAIEKBHSLRAEHFSEDRRRDIWTFWG
SNPKAAGSWGFZEFDQLTNSXZEGKIYRLALTAAJOETUDPJEOSLRLJKPKST
AWAWLVUNUCHOMBMYFSGLAWYVOEETZQSCYEDBRIBBPXSMSQSBLWALMQ
AFLRNRRSHFJZULBIPTSPINDUGWALMQAFQRRVYRZYWVMGYGKHLTODPC
KZDKCSTURCPLYOWWLMIVJHZTBOPMTRPGAHFWKZRYNFYWRONRJVREAJ
KERAGHRFYHBPKFEPRUYEKFEGKIYRLPRBAIYSGAGILWPXZSLTZACVYO
OXHGFYGVAVTJLWFWEEWALMQAFYROLIZPWJHCMRBXYYMPAJKMAHXGHW
HWELMLIHHVEVMNFZEFWSYIXVDNPTUVOPDKQOAGHDFKPHSTRUMACMGU
EVHNUTOCCTONWKAVOECAEDTUHRHHPCMOMBZZXYTAMIKJRZEGWDFCCH
IEKWOHECXJLCQKAAWLASBZOIHTSPRSSAMSSJILSUTWDFDPWKPMJAVP
EVXMETUHPHXFVAFWLALBXCJTOEPNMJVMQOIAWALQVLRQVMOEUPFSXK
VMLACCJKCELHBRMGHZXGGVMNF
```

2. Break this ciphertext:

```
OTDBPBEBOCNBUEMOEGMSNQNPNYUMXWKPPMDSDEHZMDMTRCNAYAMGUJ
UHBTNLMKPQIWJVEYNRMGWMUETBJPVLEQBHPZARJJYUXWZQQYXSAOTP
RZPVEYOAUAXXYDIBZNMDXEPPTJGFLGOFZSCCMETIPZZTTDHHICGDNQ
FSOCYNIWKEHWSTDVMKZMYWGNMLNYLCFSQLCCKDCWRGYKHOHZEJBTQJ
RWUKBODTYIYDTSMNUSHSNLPKFNQCCUVBQORDZSTEVJXZAWOALSILAI
UEDF
```

Unit 94
Attacking the autokey cipher with monogram frequencies

This attack is a modification of the attack on the Vigenère as collection of Caesar shift ciphers. For the autokey cipher, we are unable to find the key length from the index of coincidence, so we will have to try various lengths until we find an acceptable plaintext. We will partition the ciphertext into slices, each of which (if we have the correct key length) has been encrypted as an autokey cipher with a key comprised of a single character. For each slice, we try the 26 letters of the alphabet as the key and choose the one that results in the closest fit between the monogram frequencies of the decrypted slice and the monogram frequencies of English.

Programming tasks

1. Implement the attack. Allow for the possibility that the cipher is one of the three variations. Use the cosine of the angle between the vectors of monogram frequencies. Feel free to copy and modify your code from the analogous attack on the Vigenère cipher.

Exercises

1. Use this new attack to break the ciphertexts in the exercises of the previous unit.

2. Break this ciphertext:

```
PLEKLARLUSGSOSAOWAWLQWGBLSBLUFPAVSGNAPDLIPLHTGBCARDAQE
IVVJGVQHVMQOVGKIPHISELWDWHWFFTWJFRHIPYIGNGOSGFZABUKNIK
FQAIFXBFVRTXKMPBPUPPRVSCZEVQAEUXZBXNHWXZHBOAVHWNKALQDC
GEJWOFRMSRMNFRPVNMGJOIMAKMYAMNHXMOEKTOKWVSISRXUQWJITEJ
PWOSQTGSHDSKNSXMPSDCASLOZNLKFAZTNHJJZEQXIYPFYOGEOVGRYL
ENSSPDNSPKHTWJMNHAMCYMOSLHIRORSMOKSMHBUDMVQGKMAGNLQAQQ
AZOSDEEGPPPZGDMSQEPTPILMWMVVVTVVFVQZSVK
```

3. Break this ciphertext:

```
MZHPWWXIFYIIOPPVGLADUZNZRQWWVJDBJRRNIWAYAKRXRULWRCZLUY
CKLLOWLMLAWNUOECNAKGZMONJAQZEJAQEOOGFZEHEDZPIRGLLACCYZ
```

```
KVZLUYXUAIKLPFXEKKOHZVBDRJUVIPVWEZDCLGOULRRYQIUAWARYME
HWKLWOTAYFOQWJPGYTLQIEFRTOSSWGUVNNGODRTGGJMKPMASCXAWYI
BAQEJXJCBGORUWRDUBPWCGWNKBNPXNEKKRRVRJRDWXRUPLLOWLYQEU
QXVRWFSABAMQFKGBACPMDIVTCHGRXWHMPXASWDLWIZXAXOQZMZARMS
AIXFXQUPHJOTKUFKGVKEXKWMQUWYBDDYXPW
```

4. Break this ciphertext:

```
MEKXVYDNJQTQFAFWUXIUNNLJTADQMAGWBDTNVGONJNOAHNMRDZCEUS
SCSELQTZAETCPALNSBNPLZMYMKXDHZYNBYCOXFBOLFIUZPVWMWNAAP
PKHLOCSFPNHRHIEWAVPJUYEDNIHXMJUQKDTK
```

Unit 95
Running-key cipher

In the *running-key cipher*, the key stream is another piece of text, usually taken from literature. Sometimes, it is a plaintext that you have seen before.

Reading and references

David Kahn, *The Codebreakers: The Story of Secret Writing*, New York: Simon & Schuster, 1967, revised and updated 1996, pages 236-238.

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

William F. Friedman, Methods for the Solution of Running-Key Ciphers, Riverbank Laboratories Department of Ciphers Publication 16, Geneva, Illinois, 1918, www.marshallfoundation.org/library/methods-solution-ciphers

Programming tasks

1. We would ask you to implement the cipher, but you can just use your Vigenère functions with a really long keyword.

Exercises

1. This ciphertext was enciphered with a key that is also meaningful English text. Recover the plaintext and the key.

```
SULTFZHWTFUQRFTVMXCZOAAUARHGPACVPRPOXZEVBNUHANEHMWGKVU
DMCSTUHYVVEVFMNWZMJVJKLMLPALXPRLOFLNWGGYLHGLRXGGNAVTCU
NEMDAEPZRBZAAIBXIPSBUZHRJKYHSXBFWPQPXKAVTCTTBDHTQYUKKM
KOIWXXUHAZPGXYRMJKARNNWPDFJDIIPXBEVMDOWVEWGZBUXNPJZPUN
CMACRDNILMCANTHIAWALGFRBEVRRUZLGTJGOXBCSMLXYEPQRRCEUCA
```

INXSEWXSWTRCJQEZAHXZRWHPOKXLXMEIVVZMBGZJPLWLAWSIIPHKZA
GIMGGLRXKUTGKICNIEBXYSIMML

Unit 96
Progressive Vigenère cipher

The *progressive Vigenère cipher* (also known as *progressive-key cipher*) is a synchronous stream cipher. It is a modification of the Vigenère cipher in which the keyword is shifted by an amount each time it is used, as with a Caesar cipher. The shift is called the *progression index*.

An example will clarify things. Start with our usual plaintext and encipher it with keyword `KEYWORD` and progression index 3. Shifted by 3, the keyword becomes `NHBZRUG`. Shifted by another 3 it becomes `QKECUXJ`. You get the idea.

```
plaintext:     THISMES SAGEWAS ENCRYPT EDWITHA STREAMC IPHER
keyword:       KEYWORD KEYWORD KEYWORD KEYWORD KEYWORD KEYWO
shifted key:   KEYWORD NHBZRUG QKECUXJ TNHFXAM WQKIADP ZTNLD
ciphertext:    DLGOAVV FHHDNUY UXGTSMC XQDNQHM OJBMAPR HIUPU
```

We can attack the progressive Vigenère if we know the length of the keyword and the progression index (or we can try values until we find them). With these two numbers, we can remove the progression. What remains is a Vigenère cipher, which we can break using the techniques of Units 34-38. For the example above, once we know the key length is seven and the progression index is three, we can subtract the progression as follows:

```
ciphertext:     DLGOAVV FHHDNUY UXGTSMC XQDNQHM OJBMAPR HIUPU
progression:    AAAAAAA DDDDDDD GGGGGGG JJJJJJJ MMMMMMM PPPPP
new ciphertext: DLGOAVV CEEAKRV ORANMGW OHUEHYD CXPAODF STFAF
```

Some special cases:

- progression index = 0: Vigenère cipher
- key length = 1, progression index = 1: Trithemius cipher

Reading and references

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

Programming tasks

1. Implement an encryptor.
2. Implement a decryptor.
3. Implement the attack described above. You will need to try all possible key lengths and progression indices, until you find a good plaintext.

Exercises

1. Encipher this text with keyword `TARGET` and progression index 25.

   > We now take it so much for granted, we are so conscious of constantly progressing in knowledge, arts, organising capacity, utilities of all sorts, that it is easy to look upon Progress as an aim, like liberty or a world-federation, which it only depends on our own efforts and good-will to achieve.
   >
   > (from *The Idea of Progress* by J. B. Bury)

2. Decipher this text with keyword `REASON` and progression index 2.

   ```
   ZRHMANGVTIWGZAWPZVOOLMLXQQISKIJGXQRXFQEYAKNFGZTVTYDBLD
   EEFQOYDQIXMCNLQSSWXVQIQCMMCWHBLKPMUHPKLWTVQDKPGXMFQNOZ
   OHSGLZHMAYLCSGHHGQYIWZIVBQFVKXDGNIMYRALFYPGGQIUYIIAKCA
   TTFNXTDVGNWRPSTEJEDMAUNZSSFGYIHTESMDTTXKEUABQBGRDHVZRJ
   LQYRWYJIPVYDICNYOELVEGTGAKUCJJVBMKFWOYSUBBBTIHADHVDGLB
   GLUXAKNLMSKHONVTSZCDWSSUCMPQULATYCEUSDCDQTKLEJFRFKOVUG
   ZLRKJDBKYSLXCZWRZZTWBGB
   ```

3. Break this ciphertext.

   ```
   HKOTUMFYIJMSACEODTFEMRQHXNHOCXUARPBHNCZFIHLFDGCFHFAXBJ
   WYIAEJNQSFGDGUWKZNMXPMOQPPAITGTPJKJSGAYZZIQREBSTEFKIMW
   SWNSCNTRRKXKYORQWGXRVGJUNYYOIMAGFZTFIWSUUOWXGTQFHZMJYQ
   WCQEWACGGASQGHOXMBXCQGVGFVOHRISHXQYYJCDMBQASRAIRQZZZWM
   MAYFHJDBMDGUTCHOETWCKIOPUZAOAFUOOBZNDGZJXQWBVKYQQTLBEQ
   UPUJIUWVGRTCAZQBDZSVDSLVXQKJIXWLZYFMCPXELOYEOWYENKVASN
   AKMOEGPVCGTKFEKBBUHVQRDARSDEJHLVRVMRBMSQQJWCSRXKAEHSUF
   IOUJGMPOSTXCZCEPTEKFXHNALWZVMMXEXNQSUVCXAUIILZUJTXAABI
   XOMHNONIVUZONHYUJECRLJPQJUYTFPVEMDLRVAXACNXVITBOWNENNV
   GREKCDYCSFZPZMCIUDMCKHUUYMBRKIQOESQWUNTVSIVDYGWEDCYORQ
   WUJSVLEKHSJFAVDAGERCJYEJVYXCYWUOWAXDYQSPVTGHXGJNZYLHAO
   OUQPSJIPHKULHDQALOVVMVORRDQMGVCZHYIIKMFPLEDJQQGFKYORKU
   ```

WZUVOGLKJYLEXZMDFQFXUGZEBOKOANKNRTLTXCBOGYQUKMRRIZWQOZ
TLGXARDHYJ

Unit 97
Solitaire cipher

The *solitaire cipher* is a synchronous stream cipher in which the key stream is generated with a deck of playing cards. It was invented by Bruce Schneier for the spy novel *Cryptonomicon* by Neal Stephenson.

Solitaire uses a standard deck of poker cards with two jokers. Usually, one of them is black-and-white and the other is colored. We will call one of them “joker A” and the other “joker B.” We also assign a number to each card in the deck. The clubs (♣) are numbered 1 (ace) to 13 (king), the diamonds (♦) 14 to 26, the hearts (♥) 27 to 39, and the spades (♠) 40 to 52. Both jokers are numbered 53.

We begin with the deck in numberical order, clubs followed by diamonds followed by hearts followed by spades followed by joker A and joker B. To key the deck from a keyword or phrase, we perform these steps for each letter in the keyword:

1. If joker A is on the bottom (the last card) of the deck, put it just after the top card. Otherwise, swap joker A with the card below it.
2. If joker B is on the bottom of the deck, put it just after the second card. If joker B is the second-to-last card, put it just after the top card. If neither of these is the case, move joker B down by two cards.
3. Do a triple cut by swapping the stack of cards above the first joker with the stack of cards below the second joker. The first joker is whichever is closer to the top of the deck. The second is closer to the bottom.
4. Look at the bottom card. If it is a joker, do nothing for this step. Otherwise, take the number corresponding to that card and do a count cut by taking a stack of that many card off the top of the deck and putting that stack just above the bottom card.
5. Convert the letter of the keyword to a number, where ‘A’ = 1, ‘B’ = 2, ‘C’ = 3, ..., ‘Z’ = 26, and do another count cut by taking a stack of that many card off the top of the deck and putting that stack just above the bottom card.

The keyed deck is the initial internal state of the cipher, and from it we generate the key stream. We repeat the following process until we have enough key-stream letters to encipher our plaintext:

1. Same as step 1 above.
2. Same as step 2 above.
3. Same as step 3 above.
4. Same as step 4 above.
5. Look at the top card and take its corresponding number. Take a stack of that many cards off the top of the deck. Look at the new top card and record its number. Put the stack back onto the top of the deck.
6. If the number you recorded in step 5 is 53, throw it away and go back to step 1. Otherwise, if the number is greater than 26, then subtract 26 from it. Convert the number to a letter, where 'A' = 1, 'B' = 2, 'C' = 3, ..., 'Z' = 26.

The key stream is combined with the plaintext in a Vigenère-like manner, i.e., with addition modulo 26. However, it is not exactly the same; there is an offset of one. The cipher uses 'A' = 1, ..., 'Z' = 26 for this process as well. To combine a plaintext letter with a keystream letter, convert both to numbers in this way. Add them. If the sum exceeds 26, then subtract 26. Convert back to a letter.

The solitaire cipher is difficult to break unless we know more than half of the keyed deck.

The official instructions say that the plaintext should be padded to a multiple of five letters, but we are not strict about that.

Reading and references

Bruce Schneier, "The Solitaire Encryption Algorithm," Schneier on Security, www.schneier.com/academic/solitaire

Wikipedia, en.wikipedia.org/wiki/Solitaire_(cipher)

Programming tasks

1. Write a function to generate a keyed deck of cards from a keyword or key phrase. Do not use the optional step in Schneier's description of the keying method on his web page.
2. Write a function to generate a key stream given a keyed deck of cards.
3. Write a function or script to encipher a plaintext with the solitaire cipher and a given key phrase.
4. Write a function or script to decipher a ciphertext with the solitaire cipher and a given key phrase.
5. Study what happens to the deck as it is keyed with a keyword or key phrase. Can you devise a method of recovering the keyword from the state of the deck? If so, implement your idea.

Exercises

1. Encipher this text with the key phrase `LUCKY YOU DO NOT HAVE TO DO IT BY HAND`.

   The best poker hand is the royal flush, followed by a nonroyal straight flush. Then comes four of a kind, followed by a full house. Then comes a flush, followed by a straight, then three of a kind, then two pairs, then one pair, then nothing. You don't want to have nothing.

2. Decipher this ciphertext from the 2010 British National Cipher Challenge. It was encrypted with the key phrase `DIE ALCHEMISTEN RISE AGAIN`.

   ```
   AGXJE SSFKM MJMHX ZGJWB CPCVX EBNDK UQOCE DUTIC NPARQ
   PEDIX ZAVYM WZSVT BBVMT HJIGW XZAPJ HJMYN MXRGO RXOWE
   ULMJS AAENC WVUYI FQUTR XEDEJ BLWAA DFPBW ZAXJD DZOTM
   GSEZG NQWJY MFNWL SLTQD URZVQ RKOTS VDNHY EIITY RRWGC
   CSLKS UKHDR LDBZE DXSGV UGMTB NZQJT CBZTT IBWKQ PXUTQ
   MZDIH HKWZK SJEEH FBFYP GYSIH OKKOB JFJSN XSIKS NBMTN
   IADVT CXYZZ AOKQY WXNIZ JWOFZ VSPQQ GASYU MJDEL MDDHV
   ZTNFH MOOLN XAFPE VBHGS TJFMC IFNHZ YCYGG WAQYB UNNHD
   WHSLP IBFAP PDNQN DOCNU RXEAI RZNLR XAKVX XAMPU ZOPOK
   TJVQK IZDSC NC
   ```

3. Decipher this text with the deck 5♦, 6♦, 7♦, 8♦, 9♦, 10♦, J♦, Q♦, K♦, 9♣, joker A, 7♣, 5♣, J♣, Q♣, J♥, Q♥, joker B, A♥, 2♥, 3♥, 4♥, 5♥, 6♥, 7♥, 8♥, 9♥, 10♥, 10♣, K♥, A♠, 2♠, 3♠, 4♠, 5♠, 6♠, 7♠, 8♠, 9♠, 10♠, J♠, Q♠, 3♣, 4♣, A♣, 8♣, 2♣, K♠, K♣, A♦, 2♦, 3♦, 4♦, 6♣. Can you also reconstruct the keyword?

   ```
   SCYWLWXCACDWWIFZSXIMHVMBRIWHCLNLMZIHHWWCHVOZNJCKPPALVN
   MGFNJRLCQFHDKNHZHKRAGIFXGKQQSLNEDGKOTOFRFNZAJOWWVZAPGG
   MLURKZGQDMHEHKYEBLRUPMRPKFHFFMCIDKGYLOFLQQLSOMYAEGCPDY
   RWWJLTXRKQLOLXGCHCSCQMDUKZWMLMKNTHMJQNVORTTIDRHQZBEIJC
   JNTMRTHYNVGAID
   ```

Unit 98
Hill-climbing attack on the solitaire cipher with partially known key

The approach of this attack is to fix the known keys and shuffle the remainder until the fitness of the first few (between fifty and one hundred) deciphered characters exceeds a threshold; then we begin again with the key we have at that point and try to maximize the fitness of a longer part of the deciphered text. For the maximization, we use a hill-climbing technique with the parent/child key paradigm. To generate a child key from the parent, we randomly swap two of the unfixed cards, or do a three-way swap of three unfixed cards. This is a difficult attack, and it may take a long time or need to be restarted many times. It is

Programming tasks

1. Implement the attack. Use tetragram fitness to rate plaintexts. Experiment with the fitness threshold and the length of text to use in the first stage.

Exercises

1. Break the ciphertext from Exercise 2 of the previous unit. The deck is 8♦, 9♦, 3♥, K♦, A♥, J♣, 5♥, 6♥, 7♥, 8♥, 9♥, 10♥, 6♣, K♥, 4♦, Q♦, 2♠, 10♣, A♦, 5♠, 10♠, joker A, K♣, 4♠, 7♠, K♠, Q♣, 9♠, J♥, Q♥, 2♥, 7♣, 3♠, 6♠, 8♣, ?, ?, ?, ?, ?, ?, ?, ?, joker B, ?, ?, ?, ?, ?, ?, J♦, ?, 7♦, ?.

2. Can you break this ciphertext? The deck is 9♦, 3♥, 4♥, 5♥, 6♥, 7♥, 8♥, 9♥, 10♥, 2♣, K♥, A♠, 2♠, 3♠, 4♠, 8♦, 7♠, 8♠, 9♠, Q♥, Q♠, 6♠, 2♥, 2♦, 4♣, joker A, 3♣, 6♣, 3♦, 9♣, 10♣, 7♦, A♣, 10♠, J♠, K♣, A♦, A♥, ?, joker B, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? Can you also recover the keyword?

```
BHSJKOZKSQZBKVJLAICTFVGPJQJKMJCWBKTGMKUXWWIAPSCHXWUIPG
HSPRDSVSADSBHIAMWWRQCGWPSNIMKZXOXYKGCRHKWCSMGZICJYCJCW
XTBKLTAMPCFAQIIJTIWDLCPXFZKIRSVJSCKQATFLPZYCWJECXZYKID
BINPIFQJHWRECYODKTTYKHDYYULKEQFCGPGVGPQZHQAWFRMYUESBID
PSCIIGUJXXOKRSCEOLCHIDXEQCGKBVNRLNNGIXIYKYGFYUITKKHFVC
```