Can you crack Vigenère cipher?

Matlab Challenge — Summer Term 2022/23

May 19, 2023

1 Motivation

The Vigenère cipher (Wikipedia) has been a classical cryptographic technique used for centuries to encode sensitive information. It was considered unbreakable for a long time due to its polyalphabetic nature, but with the advent of modern computing power and cryptographic techniques, it can be cracked. This project aims to explore and demonstrate the method used by modern computers to break the Vigenère cipher.

2 History

The Vigenère cipher was invented by a French diplomat named Blaise de Vigenère in the 16th century. It was initially called the "le chiffre indéchiffrable" or "the indecipherable cipher" because it was believed to be unbreakable. The cipher gained widespread popularity during the 19th century and was used extensively by militaries and governments worldwide. It was used by the Confederate Army during the American Civil War and by European powers during the Franco-Prussian War.

The Vigenère cipher remained unbroken for centuries until the 19th century, when cryptanalysts began to develop techniques to crack the cipher. In 1863, Friedrich Kasiski, a German cryptanalyst, discovered that the Vigenère cipher was vulnerable to attacks based on the repeated use of the same key. This technique, known as the Kasiski examination, was one of the first successful attacks against the Vigenère cipher and can be implemented in this project.

3 Task

Your task is to crack a long text ciphered by the Vignère code, *i.e.*, and you must determine the keyword used. Several methods can be implemented:

1. *Brute-force attack*: This involves trying every possible combination for the keyword until the correct one is found. This method can be time-consuming, but it is feasible for short keywords with modern computers.

2. *Computer-assisted attacks*: These include hill climbing, simulated annealing, and genetic algorithms, which can be used to find the optimal keyword more efficiently than brute-force methods. These techniques rely on algorithms and advanced computing power to find the solution.

3. ***Kasiski examination***: This involves looking for repeated patterns in the ciphertext that can be used to determine the length of the keyword. Once the length of the keyword is known, the cipher can be broken using *frequency analysis*.

The Kasiski examination is implemented by the teachers and will be used as a reference for the students' implementation. However, the student can implement any cracking technique. The necessary **computation time** and accuracy of the cracked keyword and the text will be used to determine the winner among participants.

4 Details and Hints

4.1 Vigenère cipher and the alphabet

The alphabet in this project also contains some special characters besides regular letters. For convenience, the letters are always in upper case. The alphabet has 37 characters, including blank space, and reads:

```
alphabet = ␣!'(),-.:;?ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

where ␣ is a space. The characters can be further represented as labeled positions in the our alphabet (analogously as in ASCII table). The Vigenère cipher uses a series of interwoven Caesar ciphers, *i.e.*, it is a polyalphabetic substitution cipher, which based on the keyword, shifts the letters in the original text to the ciphered letters. Let's work out the example summarized in the table below:

plaintext	M	A	T	L	A	B		I	S		F	U	N	!
keystream	M	T	B	?	M	T	B	?	M	T	B	?	M	T
textMyASCII	24	12	31	23	12	13	1	20	30	1	17	32	25	2
keystreamMyASCII	24	31	13	11	24	31	13	11	24	31	13	11	24	31
cipherMyASCII	11	6	7	34	36	7	14	31	17	32	30	6	12	33
ciphertext	?	,	-	W	Y	-	C	T	F	U	S	,	A	V

The goal is to cipher the plaintext with the key `cKey = 'MTB?';`. Firstly, the keyword is extended to match the length of the plaintext. Characters in both, keystream and plaintext, are represented by the index position in our alphabet, *e.g.*, `'M'` has 24th position, `'A'` has 12th position, etc. The ciphered text is then obtained by the shift of the characters determined by the keyword according to the formula

$$\texttt{cipherMyASCII} = \mathrm{mod}\left(\texttt{textMyASCII} + \texttt{keyMyASCII},\ 37\right), \tag{1}$$

where mod() is the modulo operation, *e.g.*, for the first column in the table, $\mathrm{mod}(24+24, 37) = 11$, *i.e.*, the first character in the ciphered text is `alphabet(11)='?'`. The coded character should be `Z` if (1) equals 0, *i.e.*, if the $\texttt{textMyASCII} + \texttt{keyMyASCII} = 37$.

4.2 Kasiski examination

The Kasiski examination involves searching for repeated sequences of three or more characters in the ciphertext. These sequences are likely to result from the same plaintext letters being encrypted using the same part of the keyword. By calculating the distance between these repeated sequences, it is possible to determine the likely length of the keyword. Afterward, we can separate the ciphertext into sections with the same length as the guessed key length and compute the histogram of each character in each position in each section. The obtained frequency distribution can further be compared to the frequency table of the English language to determine the possible shift in the distribution.

Pseudocode of the Kasiski examination:

1. Collect the ciphertext.
2. Find as many sets of repeating sequences of n same characters (n-graphs) as you can, and the distance between each of them, *i.e.*, you compute the distance between the first character of each sequence ($n = 4$ should be sufficient).
3. Calculate the greatest common divisor (GCD) of the distances between the repeated sequences.

4. Calculate the histogram of the GCDs and guess all possible key lengths. Refer to the string example:

   AOASSPQUCKMRAQAQQQAOASURQGIQDUGRNICJQUCKRAUAXSSR,

   where the gap between "AOAS" pairs is 18, suggesting 18, 9, 6, 3, 2 long key, and the gap between "QUCK" pair is 30, suggesting 30, 15, 10, 6, 5, 3 or 2 long key.

5. Choose a guessed key length and initiate the frequency analysis:
   (a) Divide the ciphertext into sections of the length of the chosen key length.
   (b) For each section, calculate the frequency distribution of the letters in that position of each section.
   (c) Compare the obtained distribution to the pre-computed frequency table (see section 4.3) and evaluate the most likely occurring shift, *i.e.*, to determine the character at that position of the key.
   (d) Repeat for each position of the section.

6. Decipher the ciphertext with the cracked key.

4.3 Frequency table

In order to attack the Vigenère cipher using the Kasiski examination algorithm, a frequency table of common English is required. However, due to the nature of our alphabet, the frequency table needs to be precomputed. For the construction of the table, we can use "Sonnets" by William Shakespeare, which is imported into MATLAB 2022a, and some special characters are removed as

```
sonnets = fileread('sonnets.txt');
sonnets = upper(sonnets);
sonnets = strrep(sonnets,newline,'');
```

Variable `sonnets` now contains the whole book. In the figure below, the frequency analysis is performed:

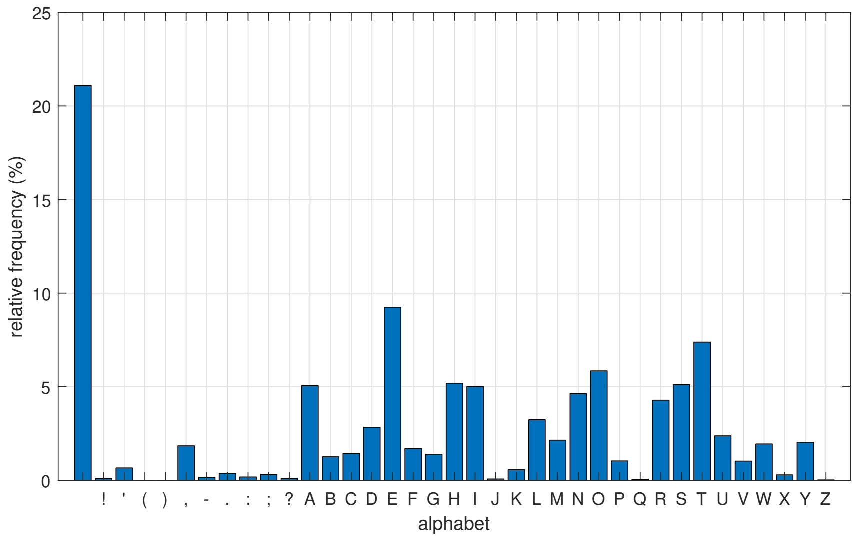

Figure 1: Frequency table determined from "Sonnets". The most commonly occurred character is ␣.

For MATLAB 2023a and newer, the documentation is no longer part of the installation package, and the supplementary file 'sonnets.txt' has to be loaded by the command:

```
openExample("matlab/AnalyzeTextDataExample",...
"supportingFile","sonnets.txt","workDir","C:\CompTask ");
```

which copies required file to the directory `"C:\CompTask"`.

5 Supplementary materials

- https://crypto.interactive-maths.com/kasiski-analysis-breaking-the-code.html
- https://medium.com/for-the-love-of-ciphers-vigenère-cipher
- https://www.cipherchallenge.org/Five-ways-to-crack-a-Vigenere-cipher.pdf
- The teachers can also provide any hints.

6 Criteria

Implementation

- Implement the competition task as a function

  ```
  function [crackedKey, crackedText] = crackVignereCipher(cipherText)
  ```

  which takes ciphered text as an input string and outputs the cracked keyword with cracked text.

General

- An unlimited number of students can select this project. However, no collaboration between students is expected.
- Complete the challenge till **May 21, 2023, 23:59** and submit it via BRUTE.
- Contact `matlab@fel.cvut.cz` with any questions.
- The project should be submitted, including short documentation for each file describing how it works.
- Like regular projects, a short presentation (a couple of minutes) is expected.
- No external MATLAB toolboxes or third-party libraries are allowed.
- It is possible to always withdraw from the competition and select one of the regular projects. This decision should be discussed with the teachers, and their approval is required.

7 List of Awards

By participating in the competition, you will be rewarded with some of the unique MATLAB merchandise, see Figure 2.

The three best solutions will also be rewarded with Humusoft vouchers for MATLAB courses, where you can learn some of the advanced functionalities of MATLAB language, such as Simulink, parallel computations, and embedded coding.

8 Disclaimer

Small changes both in the challenge assignments and in the organization of the contest are reserved.

= A\b;
MathWorks
MATLAB Voucher
Školení MATLAB dle vlastního výběru zdarma
PLAYING CARDS
PLAYING CARDS
www.humusoft.cz/matlab/voucher

Figure 2: Professionally arranged demonstration of MATLAB merchandise that you can win. 🤯