

# Izbirni projekt pri predmetu Informacija in Kodi

## Tajno kodiranje z Vigenerjevim kriptografskim sistemom

David Blazheski

Fakulteta za elektrotehniko, I letnik MAG, Smer: Avtomatika in informatika  
E-pošta: db1120@student.uni-lj.si

### 1 Uvod

Vigenerjeva šifra je simetrična metoda šifriranja besedila, pri kateri je vsaka črka besedila šifrirana z različnimi Cesarjevimi šiframi, pri čemer je inkrement določen z ustrezno črko drugega besedila, ki predstavlja ključ. Cesarjeva šifra je ena izmed najpreprostejših in najbolj znanih metod šifriranja. Gre za vrsto nadomestne šifre, kjer je vsaka črka v besedilu nadomeščena z novo črko, ki je premaknjena za določeno število mest po abecedi. Vigenerjeva šifra uporablja zaporedje več Cesarjevih šifer z različnimi vrednostmi premika.

Za šifriranje se lahko uporabi tabela abeced, imenovana tabula recta oziroma Vigenerjeva tabela prikazana na sliki 1. V njej je abeceda zapisana 26-krat v različnih vrsticah, pri čemer je vsaka vrstica ciklično pomaknjena v levo glede na prejšnjo, kar ustreza 26 možnim Cesarjevimi šiframi. Abeceda, ki se uporablja na vsaki točki, je odvisna od ponavljajočega se ključa.

Dešifriranje poteka tako, da poiščemo vrstico v tabeli, ki ustreza ključu, nato pa v tej vrstici poiščemo položaj črke šifriranega besedila in uporabimo oznako stolpca kot odprto besedilo.[1]

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

Slika 1: Vigenerjeva tabela. Vir slike: [1]

Šifriranje lahko predstavimo tudi z modularno aritmetiko, tako da črke najprej pretvorimo v številke, po shemi  $A \rightarrow 0, B \rightarrow 1, \dots, Z \rightarrow 25$  [1].

Šifriranje z uporabo ključa  $K$  za posamezen znak  $M_i$  lahko predstavimo kot:

$$C_i = E_K(M_i) = (M_i + K_i) \mod 26, \quad (1)$$

Dešifriranje se izvede z uporabo ključa  $K$ :

$$M_i = D_K(C_i) = (C_i - K_i) \mod 26, \quad (2)$$

kjer je  $M = M_1 \dots M_n$  sporočilo,  $C = C_1 \dots C_n$  šifrirano besedilo, in  $K = K_1 \dots K_n$  ključ, pridobljen s ponavljanjem ključne besede  $\lceil n/m \rceil$  krat, kjer je  $m$  dolžina ključne besede.

### 2 Algoritem za šifriranje poljubnega besedila

Začetni del algoritma omogoča šifriranje in dešifriranje besedila, ki ga uporabnik vnese preko tipkovnice. Pri zagonu programa uporabnik vnese poljuben stavek. Pred začetkom šifriranja program zahteva vnos ključa, pričemer se ta ne izpisuje, temveč so prikazane zvezdice namesto vnešenih znakov, s čimer je zagotovljena varnost vnosa. Poleg šifriranja program omogoča tudi dešifriranje tajnopisa.

```
def read_binary_file(filepath):
    with open(filepath, "rb") as file:
        return file.read()
def write_binary_file(filepath, data):
    with open(filepath, "wb") as file:
        file.write(data)
def load_code_table(filename):
    with open(filename, "r", encoding="utf-8-sig") as file:
        return [int(line.split()[1]) for line in file.split()[0]]
```

Funkcija `read_binary_file(file)` prebere vsebino binarne datoteke in jo vrne kot bajtno zaporedje. Klic funkcije je primeren za nalaganje podatkov iz binarnih datotek. Druga funkcija sprejme bajtno zaporedje `data` in ga zapise v binarno datoteko. Tretja funkcija `load_code_table(filename)` naloži kodno tabelo iz datoteke, kjer vsaka vrstica vsebuje kodo in znak. Funkcija vrne slovar, kjer so ključi znaki, vrednosti pa ustrezne celotne številke kode iz datoteke. To omogoča enostavno iskanje kode za določen znak.

```
def vigenere(data, key, mode='encrypt',
              code_table=None):
    key_length = len(key)
    if key_length == 0:
        raise ValueError("Napaka: Ključ ne sme biti prazen.")
    result = []
```

```

if code_table:
    reverse_table = {v: k for k, v in
                     code_table.items()}
    for i, char in enumerate(data):
        if char in code_table:
            char_code = code_table[char]
            key_code = code_table[key[i % key_length]]
            shift = char_code + key_code if mode == 'encrypt' else
                char_code - key_code
            result.append(reverse_table[
                shift % len(code_table)])
        else:
            result.append(char)
    return ''.join(result)
else:
    # Za binarne podatke
    return bytes((byte + key[i % key_length] %
                  256 if mode == 'encrypt' else
                  (byte - key[i % key_length]) % 256 for
                  i, byte in enumerate(
                      data)))

```

Funkcija *vigenere* je implementacija vigenerjeve šifre, ki omogoča šifriranje in dešifriranje datotek (besedilnih ali binarnih), pri čemer uporablja ključ.

- *data*: Podatki, ki jih želimo šifrirati ali dešifrirati (besedilo ali binarne podatke).
- *key*: Ključ, ki se uporablja za šifriranje ali dešifriranje.
- *mode*: Način delovanja funkcije, ki določa ali bo funkcija izvedla šifriranje ali dešifriranje. Vrednost je lahko encrypt ali decrypt.
- *code\_table* : Kodna tabela, ki je slovar, ki povezuje znake z njihovimi številskimi kodami. Če ni podana, funkcija deluje na binarnih podatkih.

Funkcija najprej preveri dolžino ključa. Če je ključ prazen, vrne napako, saj je za delovanje šifre potreben ključ. Če je podana kodna tabela, funkcija deluje na besedilu. Vsaka črka besedila je povezana s številko s pomočjo tabele, ki določa njen numerični kod. Ključ se uporablja tako, da za vsak znak besedila izračunamo numerični premik glede na ustrezni znak v ključu. Če je način *mode* nastavljen na encrypt, se numerični kod za znak v besedilu poveča za kodni vrednosti ustreznega znaka iz ključa. Če je *mode* nastavljen na decrypt, se ta vrednost zmanjša. Če kodna tabela ni podana, funkcija predvideva, da se deluje z binarnimi podatki. V tem primeru za vsak bajt v podatkih uporabimo ustrezni premik iz ključa in izvedemo šifriranje ali dešifriranje tako, da na vsak bajt dodamo ali odštejemo vrednost iz ključa.

### 3 Algoritem za šifriranje datoteke

Program je bil dopolnjen tako, da je sposoben šifrirati in dešifrirati poljubne binarne datoteke. Ključ za šifriranje in dešifriranje je lahko podan bodisi neposredno preko terminala bodisi kot dodatna vhodna datoteka. Za šifriranje in dešifriranje datotek se uporablja funkcija *vigenere* iz poglavja 2.

### 4 Rezultati

Delovanje programa smo želeli preizkusiti na čim bolj interaktivnem način, kar je prikazano na priloženih slikah 2, 3, 4

```

Vnesite 1 za šifriranje/desifriranje besedila ali 2 za šifriranje/desifriranje datoteke:1
Vnesite besedilo za šifriranje: David Blazheski Projekt pri predmetu IK.!
Vnesite ključ: ****
Šifrirano besedilo: %FPEoDKB_BDTPCi1WIIFPNQWC1Qw?CNJNTk.%wl
Dešifrirano besedilo: David Blazheski Projekt pri predmetu IK.!

```

Slika 2: Testiranje šifriranja besedila

Vneseno besedilo za šifriranje je bilo David Blazheski Projekt pri predmetu IK.!, ključ pa kodi. Rezultati testa so pokazali, da algoritmom deluje pravilno, saj pravilno prikaže šifrirano besedilo in uspešno obnovi prvotno besedilo po dešifriranju.

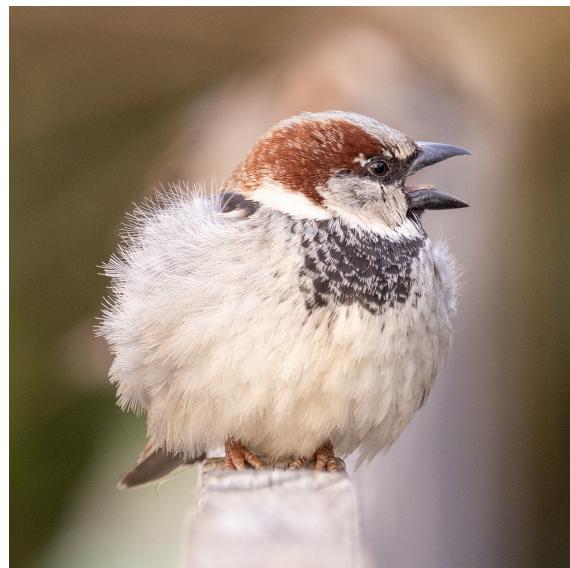
```

AG/2. Informacija o kodi/Projekt/Projekt.py"
Vigenere cipher za besedila in poljubne datoteke
Vnesite 1 za šifriranje/desifriranje besedila ali 2 za šifriranje/desifriranje datoteke:2
Vnesite ime vhodne datoteke: šifrirana.png.vig
Izbira ključa: 1) vnesi datoteko; 2) vnesi ključa preko tipkovnice: 1
Vnesite način (encrypt ali decrypt): decrypt
Vnesite ime izhodne datoteke: dešifrirana.png
Vnesite ime datoteke s ključem: ključ.vig
Operacija 'decrypt' je bila uspešno izvedena. Izhod je v datoteki 'dešifrirana.png'.

```

Slika 3: Testiranje šifriranja datotek

Dešifrirali smo vhodno datoteko *šifrirana.png.vig* z uporabo ključa, shranjenega v datoteki *ključ.vig*. Rezultat dešifriranja je bila datoteka *dešifrirana.png* prikazana spodaj.



Slika 4: Dešifrirana slika.

## 5 Dodatna naloga

Glavna pomanjkljivost vigenerjeve šifre je, da se ključ začne ponavljati, če je krajši od besedila. To omogoča, da z analizo vzorcev v šifriranem besedilu določimo dolžino ključa.

Najprej smo dolžino ključa določili s pomočjo Kasiski metode, ki je namenjena analizi šifriranega besedila.

Njeno delovanje temelji na iskanju ponavljajočih se vzorcev znakov v šifriranem besedilu in izračunu razdalj med njihovimi pojavitvami. Na podlagi teh razdalj se izračunajo možne dolžine ključa z iskanjem največjih skupnih deliteljev. Najpogosteje dolžine ključa, pridobljene na ta način, so ključne za uspešno dešifriranje šifriranega besedila [2].

Rezultat Kasiski metode je bil dolg seznam možnih dolžin ključa: Suggested key lengths: [(113, 45352), (2, 22782), (226, 22640), (3, 15049), (339, 14964), (4, 11312), (452, 11245), (5, 9270), (565, 9212), ...].

Na podlagi rezultata smo sklepali, da je dolžina ključa enaka 113, saj ima ta vrednost najvišjo frekvenco pojavljanja.

```

import re
from collections import defaultdict
def read_file(file_path):
    with open(file_path, 'rb') as file:
        return file.read()
def kasiski_method(ciphertext):
    distances = []
    for i in range(len(ciphertext) - 3):
        pattern = ciphertext[i:i+3]
        for j in range(i + 3, len(ciphertext) - 3):
            if ciphertext[j:j+3] == pattern:
                distances.append(j - i)
    gcds = []
    for dist in distances:
        for i in range(2, dist + 1):
            if dist % i == 0:
                gcds.append(i)
    freq = defaultdict(int)
    for gcd in gcds:
        freq[gcd] += 1
    common_lengths = sorted(freq.items(), key=
        lambda x: x[1], reverse=True)
    return common_lengths
ciphertext = read_file('tajnopus')
key_lengths = kasiski_method(ciphertext)
print("Suggested key lengths:", key_lengths)

```

Po pridobitvi dolžine ključa 113 smo poskusili razbiti šifro z uporabo naslednjega algoritma, ki vključuje funkcije za iskanje ključa, dešifriranje in shranjevanje rezultatov.

- *decrypt(data, key)*

Ta funkcija dešifririra podatek s pomočjo vigenerjeve šifre. Vsak bajt šifriranega besedila se dešifririra s tem, da se od vrednosti bajta odšteje ustrezni bajt iz ključa, pri čemer se uporabi modul 256, da zagotovimo kodiranje po UTF8.

- *find\_key(data, key\_length)*

Funkcija poišče ključ na podlagi dolžine ključa tako, da podatke razdeli na več delov, pri čemer vsak del vsebuje vsak drugi bajt, ki je bil šifriran z istim delom ključa. Za vsak del izračuna najpogosteji znak in ga primerja z vrednostjo presledka, saj je presledek eden najpogosteje uporabljenih znakov v slovenskih besedilih.

- *break\_vigenere(file, key\_length)*

Ta funkcija združi prejšnje funkcije za razbijanje vigenerjeve šifre. Prebere šifrirano datoteko, izračuna ključ s funkcijo *find\_key*, nato dešifrira besedilo s pomočjo *decrypt*, in na koncu shrani dešifrirano besedilo v novo datoteko.

```

from collections import Counter
def write_to_file(file_name, content, encoding=
    'utf-8'):
    with open(file_name, 'w', encoding=encoding
    ) as file:
        file.write(content)
def read_file(file):
    with open(file, 'rb') as file:
        return file.read()
def decrypt(data, key):
    decrypted = bytearray()
    for i, byte in enumerate(data):
        decrypted.append((byte - key[i % len(
            key)]) % 256)
    return decrypted.decode('utf-8')
def find_key(data, key_length):
    common_char = ord(' ')
    parts = [data[i:key_length] for i in range
        (key_length)]
    key = bytearray()
    for segment in parts:

```

```

        most_common_byte = Counter(segment).
            most_common(1)[0][0]
        key_byte = (most_common_byte -
            common_char) % 256
        key.append(key_byte)
    return key
def break_vigenere(file, key_length):
    encrypted_data = read_file(file)
    key = find_key(encrypted_data, key_length)
    decrypted_text = decrypt(encrypted_data,
        key)
    write_to_file("decrypted_cypher.txt",
        decrypted_text)
    print("Vaše besedilo je v datoteki
        decrypted_cypher.txt")
    return decrypted_text
file = "tajnopus"
key_length = 113
decrypted_text = break_vigenere(file,
    key_length)

```

Po izvedbi algoritma je bilo dešifrirano besedilo shranjeno v datoteko *decrypted\_cypher.txt*.

Sifrirano besedilo je izvirno besedilo *Med dvema stoloma* avtorja *Josipa Jurčiča*.

## Literatura

- [1] Wikipedia contributors. (2025, January 11). Vigenere cipher. In Wikipedia, The Free Encyclopedia. Retrieved from [https://en.wikipedia.org/wiki/Vigen%C3%A8re\\_cipher#Friedman\\_test](https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher#Friedman_test)
- [2] National Cipher Challenge 2024. (2025, January 11). Five ways to crack vigenere cipher. <https://www.cipherchallenge.org/wp-content/uploads/2020/12/Five-ways-to-crack-a-Vigenere-cipher.pdf>