Dokumentace k programu *Caesarova šifra* – Filip Raasch

Zadání

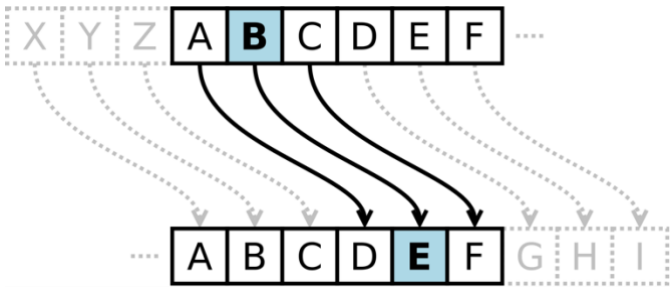
Na vstupu text uložený v textovém souboru. Pro zvolenou hodnotu klíče proveďte zašifrování textu a export do textového souboru. Následně proveďte zpětné dešifrování textu a výsledek uložte do textového souboru.

Úvod

Caesarova šifra patří mezi substituční šifry. Tím jsou myšleny šifry, kde pro danou množinu znaků dochází k záměně za jinou množinu znaků. Svůj název nese Caesarova šifra dle Julia Caesara, který tuto šifru používal při komunikaci ve svých vojenských taženích. Princip této šifry popsal ve spisu Zápisy o válce galské. V dnešní době se tato šifra v kryptografii nepoužívá, jelikož je velice jednoduše dešifrovatelná. Stačí projít všech 26 možností klíče pro rozluštění šifry.

Princip

Při Caesarově šifře se každé písmeno zašifruje tak, že se zapíše písmeno o hodnotu klíče (k) pozic dále. Rozšifrování probíhá tím způsobem, že se místo zašifrovaného písmena zapíše písmeno o k pozic dříve. Stěžejní je tedy po zašifrování hodnota klíče. Pokud hodnota klíče pro nějaké písmeno přesáhne konec abecedy, šifrování počíná opět od začátku abecedy.



Obrázek 1: Princip Caesarovi šifry (Zdroj: https://cs.wikipedia.org/wiki/Caesarova_%C5%A1ifra)

Příklad:

Máme slovo „AHOJ“ a chceme ho zašifrovat s klíčem 3 (jak používal Caesar):

A -> D

H -> K

O -> R

J -> M

V Caesarově šifře by slovo „AHOJ“ znělo „DKRM“. Rozšifrování by proběhlo nalezením písmena o 3 pozice dříve.

Existující algoritmy:

Algoritmů pro Caesarovu šifru je pro většinu programovacích jazyků napsáno již větší množství.

Dle stránky tutorialspoint.com je algoritmus pro zašifrování napsán následovně:

```
def encrypt(text,s):
result = ""
   # transverse the plain text
   for i in range(len(text)):
      char = text[i]
      # Encrypt uppercase characters in plain text

      if (char.isupper()):
         result += chr((ord(char) + s-65) % 26 + 65)
      # Encrypt lowercase characters in plain text
      else:
         result += chr((ord(char) + s - 97) % 26 + 97)
      return result
```

Algoritmus používá pozice písmen v ASCII tabulce pro zašifrování. Rozšifrování autoři algoritmu provádějí následovně:

```
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

for key in range(len(LETTERS)):
   translated = ''
   for symbol in message:
      if symbol in LETTERS:
         num = LETTERS.find(symbol)
         num = num - key
         if num < 0:
            num = num + len(LETTERS)
         translated = translated + LETTERS[num]
      else:
         translated = translated + symbol
```

Jiná implementace algoritmu vynechává použití ASCII tabulky. Tato metoda je k nalezení na stránkách w3resource.com:

```
def caesar_encrypt(realText, step):
      outText = []
      cryptText = []

      uppercase = ['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']
      lowercase = ['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 eachLetter in realText:
            if eachLetter in uppercase:
                  index = uppercase.index(eachLetter)
                  crypting = (index + step) % 26
                  cryptText.append(crypting)
                  newLetter = uppercase[crypting]
                  outText.append(newLetter)
            elif eachLetter in lowercase:
                  index = lowercase.index(eachLetter)
                  crypting = (index + step) % 26
                  cryptText.append(crypting)
                  newLetter = lowercase[crypting]
                  outText.append(newLetter)
      return outText
```

Popis zvoleného algoritmu

Mnou zvolený algoritmus na zašifrování nevyužívá ASCII tabulku a je podobnější poslední zmíněné variantě šifrování. Šifrování je v mé verzi jinak strukturováno a rozloženo do tří funkcí.

V první funkci je vytvořena nová, posunutá abeceda pomocí klíče. Index písmena z původní abecedy v nově vytvořené abecedě odpovídá jeho zašifrování.

```
def zaklicovani(klic):
    abc_velka = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    abc_mala = "abcdefghijklmnopqrstuvwxyz"
    nova_velka = list(abc_velka)
    nova_mala = list(abc_mala)
    # zaklicovani predni casti nove abecedy
    for i in range(len(abc_velka)-klic):
        nova_velka[i] = abc_velka[i+klic]
        nova_mala[i] = abc_mala[i+klic]
```

```
    # zaklicovani konce nove abecedy
    for k in range(klic):
        nova_velka[len(abc_velka)-1-k]=abc_velka[klic-k-1]
        nova_mala[len(abc_mala)-1-k]=abc_mala[klic-k-1]
    return(abc_velka, abc_mala, nova_velka, nova_mala)
```

V druhé funkci je provedeno samotné zašifrování textu s pomocí klasické abecedy a zašifrované abecedy:

```
def do_cesara(vstup, klic):
    abc_velka, abc_mala, nova_velka, nova_mala = zaklicovani(klic)
    vystup = list(vstup)
    for i in range(len(vstup)):
        if vstup[i] in abc_velka:
            kdeje = abc_velka.find(vstup[i])
            vystup[i] = nova_velka[kdeje]
        elif vstup[i] in abc_mala:
            kdeje = abc_mala.find(vstup[i])
            vystup[i] = nova_mala[kdeje]
        else:
            vystup[i] = vstup[i]
    return(vystup)
```

Třetí šifrovací funkcí je rozšifrování textu, funkce je velice podobná funkci k zašifrování.

Struktura programu

Program se dělí na dvě části – na definování použitých funkcí a hlavní část programu.

Definované funkce:

Název: `nacist_text`

Použití: Funkce načte data z textového souboru vstup.txt.

Název: `zaklicovani`

Použití: Funkce vytvoří z původní abecedy novou abecedu posunutou o hodnotu klíče. Vrací jednak původní, tak novou abecedu. Toto provede jednak pro velká, tak malá písmena.

Název: `do_cesara`

Použití: Funkce zašifruje vstupní text pomocí zadaného klíče. Vrací zašifrovaný text. Zašifrované jsou pouze znaky bez diakritiky, znaky s diakritikou, nebo další znaky jsou vráceny nezměněné.

Název: `z_cesara`

Použití: Funkce rozšifruje vstupní text pomocí zadaného klíče. Vrací rozšifrovaný text. Rozšifrované jsou pouze znaky bez diakritiky, znaky s diakritikou, nebo další znaky jsou vráceny nezměněné.

Název: `uloz_sifru`

Použití: Funkce uloží zašifrovaný text z funkce do_cesara do požadovaného souboru.

Název: `uloz_rozklicovane`

Použití: Funkce uloží znovu rozšifrovaný text z funkce z_cesara do požadovaného souboru.

Hlavní část programu

V hlavní části programu je nejprve uživatel dotázán na hodnotu klíče. Ta se musí pohybovat v rozmezí 0-26. Následně se načtou data ze vstupního souboru vstup.txt. Text je zašifrován a uložen do souboru zaklicovane.txt. Ze souboru zaklicovane.txt se poté načte zašifrovaný text. Je rozšifrován a uložen do souboru rozklicovane.txt.

Vstupní a výstupní data

Vstup:

Ve složce, ze které je spouštěn .py program, je nutné mít soubor vstup.txt, ve kterém je uložen text k zašifrování. Text může obsahovat písmena s diakritikou, čísla a další symboly, tyto však nebudou zašifrované.

Výstup:

Do složky, ze které je spouštěn .py program, je uložen soubor zaklicovane.txt. Do tohoto souboru je uložen zašifrovaný text. Do složky je dále uložen soubor rozklicovane.txt. Do souboru je znovu uložen rozšifrovaný text. Pokud soubory neexistují, jsou vytvořeny nové. V případě, že ve složce již existují, jsou přepsány.

Problematická místa a možná vylepšení

Slabinou šifrování pomocí této šifry je její jednoduchost – Caesarova šifra je velice jednoduše rozšifrovatelná. Podporovat tuto slabinu může program tím, že nešifruje písmena s diakritikou. Při použití velkého množství diakritiky tak může být i po zašifrování jasné, o jaké slovo se jedná.

Vylepšit by se program také dal tím, že by uživatel zadal, z jakého souboru chce načíst data, a také jaká data chce rozšifrovat. Tato verze programu neumožňuje rozšifrovat požadovaný text.

Shrnutí

Přestože je Caesarova šifra snadno rozšifrovatelná a nemá již praktické místo v dnešní kryptografii, je zajímavá především svým historickým využitím. Je také snadno implementovatelná, a proto využitelná pro výukové, případně herní aktivity.