Vývoj Python knihovny pro šifrování, dešifrování a kryptoanalýzu substituční šifry

Cíl práce:

Každý samostatně vytvoříte vlastní Python knihovnu, která umožní:

- Šifrování a dešifrování textu pomocí klasické substituční šifry.
- Automatické prolomení (kryptoanalýzu) této šifry pomocí statistických metod.
- Aplikujete danou knihovnu na zašifrovaná data (pro tuto část je nutné získat data z internetu)

Hlavní úkoly:

1. **Implementace šifrovací/dešifrovací funkčnosti:**
    - Vytvořte modul (knihovnu) v Pythonu, který bude obsahovat funkce pro šifrování a dešifrování textu pomocí substituční šifry.
    - Klíč pro šifrování bude představovat permutaci abecedy (uvažujme zjednodušenou anglickou abecedu **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 mezerou nahrazené podtržítkem _**). Jinou abecedu neuvažujte.
2. **Prolomení šifry (kryptoanalýza):**
    - **Sestavení teoretické bigramové matice:**
      Vyhledejte na internetu (doporučujeme použít texty z Wikisource) vhodný český text. Z tohoto textu vytvořte teoretickou matici přechodů (bigramovou matici), která bude zobrazovat pravděpodobnosti výskytu páru po sobě jdoucích znaků v češtině.
        - **Na tuto část odporučujeme pouzít text z wikisource, nejlépe knihu, například čapka:**
        - https://cs.wikisource.org/wiki/Krakatit
        - Text musí být ale dostatečně dlouhý a obsahovat alespoň několik stovek tisíc kombinací celkem. Například Krakatit má 452 tisíc znaků což je dostačující.
    - **Implementace Metropolis-Hastings algoritmu:**
      Navrhněte a implementujte kryptoanalytický postup pro prolomení substituční šifry.
      Algoritmus by měl využívat:
        - Vytvořenou bigramovou matici jako referenční hodnotu.
        - Metropolis-Hastings (M-H) algoritmus pro hledání nejpravděpodobnější permutace (klíče), která dešifruje daný zašifrovaný text do textu s pravděpodobnostní strukturou co nejpodobnější referenční matici.
3. **Aplikační demonstrace:**
    - Připravte Jupyter Notebook, ve kterém demonstrujete:
        - Šifrování a dešifrování pomocí vytvořené knihovny.
        - Vytvoření teoretické bigramové matice z vybraného českého textu.
        - Provedení kryptoanalýzy na zašifrovaném textu (text bude poskytnut vyučujícím).

- Notebook by měl obsahovat přehledný popis kroků, vizualizaci (např. grafy, tabulky) výsledků a stručnou analýzu úspěšnosti prolomení.

4. **Export dešifrovaných textů a klíčů**
   Každý text který automatizovaně pak dešifrujete (stačí u každého textu udělat 20 tisíc pokusů o dešiforvání – iterací algoritmu) pak uložíte – i jeho klíč. Kde struktura souborů je: „text_{délka_textu}_sample_{id textu}_plaintext/key.txt“. Tedy pro text o délce 1000 a id 20 pak uložíte následující soubory:
   dešifrovaný text bude uložen jako „text_1000_sample_20_plaintext.**txt“**

   Dešifrovaný klíč bude uložen jako „text_1000_sample_20_key.**txt“**

   Kde bude čistě jen text.

   U textů k dešifrování pak necháváme u prvního souboru (délka 1000) příklad tohoto exportu.

5. **Výstupy a odevzdání:**

   - **Kód knihovny:**
     Zdrojový kód (s povinnou dokumentací knihovny – všech funkcí).
   - **Jupyter Notebook:**
     Exportovaný ve formátu PDF nebo HTML s kompletním průběhem demonstrace.
   - **Stručný report:**
     Shrnutí implementačního postupu, použitých metod a dosažených výsledků.
   - **Exportované klíče a plaintexty**

Technické požadavky:

- Projekt musí být realizován v Pythonu.
- Za použití standardních knihoven (např. NumPy, Pandas, Matplotlib, aj.)
- Kód by měl být dobře strukturovaný, komentovaný a snadno čitelný.
- Zvláštní pozornost věnujte testování a validaci výsledků kryptoanalýzy.

Hodnocení bude založeno na:

- Funkčnosti a robustnosti implementace šifrovacích/dešifrovacích funkcí.
- Správnosti implementace kryptoanalýzy (vytvoření bigramové matice, použití M-H algoritmu).
- Kvalitě a přehlednosti prezentace v Jupyter Notebooku.
- Dokumentaci, komentářích v kódu a celkové úrovni prezentace výsledků.

Popis substituční šifry

Substituční šifra – princip, šifrování a dešifrování

Substituční šifra je jednou ze základních metod kryptografie, která spočívá v nahrazení jednotlivých znaků původního textu (prostého textu) jejich odpovídajícími znaky podle předem definovaného klíče. Tento klíč představuje permutaci abecedy, tedy jiný uspořádaný seznam znaků, ve kterém každé písmeno prostého textu má svou jedinečnou náhradu.

Jak šifrování funguje:

1. **Určení klíče:**
   Klíč může být jednoduchý, jako je v případě Caesarovy šifry, kde se klíč zadává jako číslo určující počet posunů, nebo složitější, kdy se používá náhodná permutace celého abecedního seznamu. Například, pokud máme abecedu s 27 znaky (včetně podtržítka za mezeru) a klíč je daný jako posun o 3, pak se písmeno 'A' nahradí písmenem, které se nachází o 3 pozice dále (tj. 'D'), 'B' se nahradí 'E' atd.
2. **Převedení znaků na číselné hodnoty:**
   Každému znaku přiřadíme číslo (např. A = 0, B = 1, …, Z = 25, _ = 26). Šifrovací proces pak využívá aritmetiku modulo n (kde n je počet znaků v abecedě).
   - Příklad:
     Pokud je klíč b = 3 a hodnota znaku v = 0 (pro 'A'), nová hodnota je (v + b) mod 27 = (0 + 3) mod 27 = 3, což odpovídá písmenu 'D'.
3. **Výpočet šifrovaného textu:**
   Každé písmeno prostého textu je tak nahrazeno svým „posunutým“ ekvivalentem. Výsledný text, tzv. kryptogram, je tvořen těmito novými znaky.

Jak dešifrování funguje:

Dešifrování je proces obrácený k šifrování. Pokud máme správný klíč, pro každý znak kryptogramu provedeme inverzní operaci, tj. odečteme hodnotu klíče a provedeme operaci modulo n.

- Příklad:
  Pokud kryptogram obsahuje písmeno 'D' (číslo 3) a klíč je 3, vypočítáme:
  (3 − 3) mod 27 = 0, což odpovídá písmenu 'A'.
  Tímto způsobem se získá původní text.
- **Šifrování:**
  Pro každý znak se najde odpovídající náhrada dle klíče. U Caesarovy šifry se provádí posun o pevně stanovený počet míst, což lze matematicky vyjádřit jako (v + b) mod n.
- **Dešifrování:**
  Proces je inverzní – odečítá se hodnota klíče a aplikací modulo operace se získá původní hodnota znaku.

Příklad použití:

Původní text: BYL_POZDNI_VECER_PRVNI_MAJ_VECERNI_MAJ_BYL_LASKY_CAS
Klíč: DEFGHIJKLMNOPQRSTUVWXYZ_ABC
Zašifrovaný: EAOCSRBGQLCYHFHUCSUYQLCPDMCYHFHUQLCPDMCEAOCODVNACFDV
Dešifrovaný: BYL_POZDNI_VECER_PRVNI_MAJ_VECERNI_MAJ_BYL_LASKY_CAS

Popis algoritmu pro automatizované prolomení / dílčí částí

Samotný algoritmus pro automatizované prolomení textu (Metropolis-Hastings - M-H algoritmus) popisujeme dále. Zde popisujeme postupně jeho jendotlivé části a pak jdeme až k samotnému algoritmu.

Bigramy
Bigramy jsou posloupnosti dvou po sobě jdoucích znaků v textu. Vižijeme je pak pro konstrukci přechodové matice. Přechodová matice ja ústřendním nástrojem na prolomení textu.

Například v českém slově "KRYPTOSYSTEM" najdeme bigramy jako "KR", "RY", "YP", "PT", "TO", "OS", "SY", "YS", "ST", "TE", "EM". Tedy zopakujeme - bigramy jsou posloupnosti dvou po sobě jdoucích znaků v textu.

Tyto dvojice se často využívají při statistické analýze textů, například při vytváření bigramové matice, která zaznamenává četnosti výskytu jednotlivých dvojic znaků. Tato matice pak slouží jako referenční vzor, který lze použít při kryptoanalýze k ověření pravděpodobnostní struktury dešifrovaného textu oproti očekávanému jazykovému modelu. Bigramy tak pomáhají odhalit jazykové vzorce, což je cenné zejména při prolomení šifer, kde je třeba odhadnout správný klíč.

Přechodová matice – Bigramová matice

Přechodová matice, často označovaná jako matice bigramů, je nástroj, který zaznamenává statistiky výskytu dvojic po sobě jdoucích znaků v textu. Každý prvek matice (i, j) udává, kolikrát se znak j objevil hned po znaku i ve vzorovém (referenčním) textu. Tato matice tak zachycuje jazykové vzorce – například u češtiny může ukazovat, že po písmenu "N" následuje často "A" nebo "E".

V rámci Metropolis-Hastings algoritmu se přechodová matice využívá jako základ věrohodnostní funkce. Když je pomocí algoritmu generován nový kandidátní klíč, dešifrovaný text se vyhodnocuje porovnáním jeho bigramových statistik s těmi, které jsou uvedeny v referenční přechodové matici. Čím více se dešifrovaný text shoduje s jazykovým modelem (tj. má podobnou strukturu bigramů), tím vyšší je hodnota věrohodnostní funkce, což signalizuje, že kandidátní klíč je pravděpodobně správný.

V rámci algoritmu budeme používat relativní přechodovou matici a ne absolutní. Relativní matice je takaová matice jejíchž prvky jsou vydělené celkovým počtem bigramů se kterým se pracovalo (tedy součet všech prvků této matice je 1.

Důležitá poznámka: Prvně sestavte absolutní matici přechodů v počtech bigramů které jste našli v textu. Následně ke všem hodnotám kde je nula přičtěte hodnotu 1. teprve pak počítejte relativní matici bigramů – přechodů. Toto je kvůli tomu aby nenastala situace kdy v následující kroku se budete snažit vypočítat logaritmus z nuly.

M-H algoritmus:

Metropolis-Hastings (M-H) algoritmus je metoda stochastické optimalizace, která se v kryptoanalýze využívá k hledání nejpravděpodobnějšího klíče, jenž dešifruje daný kryptogram do textu s jazykovým vzorem odpovídajícím referenčnímu modelu. Níže je popsán postup, jak M-H algoritmus funguje v kontextu prolomení substituční šifry:

1. **Inicializace:**
    - **Výběr počátečního klíče:** Na začátku se zvolí nějaká počáteční permutace abecedy (klíč). Tento klíč se použije pro dešifrování kryptogramu a získání počátečního odhadu prostého textu. Standardně jej volíme náhodně.

- **Výpočet věrohodnostní funkce:** Pro dešifrovaný text se spočítá věrohodnost (likelihood) na základě přechodové relativní (bigramové) matice sestavné z aktuálního deširovaného textu a z referenčního textu – čím více se relativní četnosti bigramů shodují s očekávanými četnostmi v referenčním textu, tím je věrohodnost vyšší.

2. **Generování kandidátního klíče:**

    - V každé iteraci se z aktuálního klíče vytvoří nový kandidátní klíč malou úpravou, **tato malá úprava spočívá v zaměněním dvou náhodně vybraných znaků**. Tato malá změna umožňuje prozkoumávat prostor možných klíčů. Tedy jen prohodíte dva znaky v klíči – náhodně vybrané.

3. **Vyhodnocení kandidátního klíče:**

    - **Dešifrování:** Kandidátní klíč se použije k dešifrování kryptogramu.
    - **Výpočet nové věrohodnosti:** Na základě dešifrovaného textu se opět vypočítá věrohodnostní funkce. Tato funkce porovnává bigramovou strukturu (či i trigramovou, pokud se využívá) dešifrovaného textu s referenční přechodovou maticí.

4. **Rozhodnutí o přijetí kandidátního klíče:**

    - Vypočítá se poměr věrohodností nového kandidátního klíče a aktuálního klíče. Tento poměr určuje pravděpodobnost přijetí nového klíče:

$\rho = \min \{1, (\text{věrohodnost nového klíče}) / (\text{věrohodnost aktuálního klíče})\}$

    - Pokud je ρ větší než 1, nový klíč se přijme automaticky (což znamená, že nový kandidát vede k lepším výsledkům). Pokud je ρ menší než 1, nový klíč se přijme s pravděpodobností ρ – to umožňuje, že i kandidáty s nižší věrohodností mohou být občas přijatí, což pomáhá algoritmu vyhnout se lokálním maximům.

    - Pravděpodobnost generujte jako náhodný výběr čísla z hodnot 0-1. Tedy vygenerujte číslo a pokud toto číslo je větší než ρ tak přijmeme návrh klíče.

    - V pseudokodu tuto pravděpodobnost máme 0.01. Při praktické aplikaci – používání pak lze ale s touto hodnotou experimentovat.

5. **Iterace:**

    - Tento proces se opakuje po předem stanovený počet iterací. Během iterací se uchovává klíč, který dosáhl nejvyšší věrohodnosti. Postupně se algoritmus „pohybuje“ prostorem možných klíčů a konverguje ke klíči, který nejlépe odpovídá jazykovým vzorcům referenčního textu.

6. **Výsledek:**

    - Na konci iterací je jako konečný odhad zvolen ten klíč, který vykázal nejvyšší dosaženou hodnotu věrohodnostní funkce. Tento klíč by měl dešifrovat kryptogram do textu, jehož bigramová (či trigramová) struktura se nejvíce podobá očekávanému jazykovému modelu.

Metropolis-Hastings algoritmus pro prolomení substituční šifry pracuje tak, že iterativně generuje malé změny v klíči, vyhodnocuje, jak "jazykově správný" je dešifrovaný text (na základě přechodové matice), a na základě poměru věrohodností rozhoduje, zda nový klíč přijmout. Tato metoda umožňuje efektivně prozkoumávat velký prostor možných klíčů a najít ten, který s největší pravděpodobností odpovídá původnímu textu.

Pseudokód (povinné argumenty v tomto pořadí a jménem):

alphabet- obsahuje abecedu písmen, např. list
TM_ref – referenční relatinví matice bigramů /přechodů sestavená z nějakého textu který není zašifrovaný (např. knihy)
iter – počet iterací algoritmu
start_key - dává uzivatel ale pokud ho nedá vygenerujte náhodně počáteční klíč pro prolomení šifry.
Text – zašifrovaný text se kterým pracujeme

```
FUNCTION prolom_substitute(text, TM_ref, iter, start_key)
  current_key ← start_key
  decrypted_current ← substitute_decrypt(text, current_key)
  p_current ← plausibility(decrypted_current, TM_ref)

  FOR i FROM 1 TO iter DO:
    candidate_key ← COPY(current_key)
    indices ← RANDOM_SAMPLE(2 FROM {1, 2, ..., LENGTH(alphabet)})
    SWAP candidate_key[indices[1]] WITH candidate_key[indices[2]]

    decrypted_candidate ← substitute_decrypt(text, candidate_key)
    p_candidate ← plausibility(decrypted_candidate, TM_ref)

    q ← p_candidate / p_current

    IF q > 1 THEN:
      current_key ← candidate_key
      p_current ← p_candidate
    ELSE IF RANDOM_UNIFORM(0, 1) < 0.01 THEN:
      current_key ← candidate_key
      p_current ← p_candidate
    END IF

    IF (i MOD 50) == 0 THEN:
      PRINT("Iteration", i, "log plausibility:", p_current)
    END IF
  END FOR

  best_decrypted_text ← substitute_decrypt(text, current_key)
  RETURN (current_key, best_decrypted_text, p_current)
```

Poznámky:

TM_ref – je teoretická matice přechodů bigramů získaná z nějakého českého textu a je vyjádřená jako relativní (součet všech jejích prvků je 1).

Pseudo-kod popisující funkci pro konstrrkci bigramů (povinné argumenty v tomto pořadí a jménem):

```
FUNCTION get_bigrams(text)
  bigrams_list ← empty list
  n ← LENGTH(text)
  FOR i FROM 1 TO n - 1 DO:
    bigram ← SUBSTRING(text, i, i + 1)
    APPEND bigram TO bigrams_list
  RETURN bigrams_list
```

Pseudo-kod popisující funkci pro konstrukci přechodové matice (povinné argumenty v tomto pořadí a jménem):

```
FUNCTION transition_matrix(bigrams)
  n ← LENGTH(alphabet)
  // Inicializace matice TM o rozměru n x n s jednotkovými hodnotami; řádky a sloupce pojmenujeme podle znaků v alphabet
  TM ← MATRIX(n x n) FILLED WITH 0 WITH ROW_NAMES = alphabet AND COLUMN_NAMES = alphabet
  FOR each bigram IN bigrams DO:
    c1 ← FIRST_CHARACTER(bigram)
    c2 ← SECOND_CHARACTER(bigram)
    i ← INDEX_OF(c1, alphabet)
    j ← INDEX_OF(c2, alphabet)
    TM[i][j] ← TM[i][j] + 1
```

```
  // Abychom se vyhnuli log(0) v dalších výpočtech, nuly nahradíme hodnotou 1:
  FOR i FROM 1 TO n DO:
    FOR j FROM 1 TO n DO:
      IF TM[i][j] == 0 THEN:
        TM[i][j] ← 1
  RETURN TM
```

Plausability / věrohodnost či likelihood se pak počítá jako (povinné argumenty v tomto pořadí a jménem):

```
FUNCTION plausibility(text, TM_ref)
  bigrams_obs ← get_bigrams(text)
  TM_obs ← transition_matrix(bigrams_obs, alphabet)
  likelihood ← 0
  FOR i FROM 1 TO LENGTH(alphabet) DO:
    FOR j FROM 1 TO LENGTH(alphabet) DO:
      likelihood ← likelihood + LOG( TM_ref[i][j] ) * TM_obs[i][j]
  RETURN likelihood
```

Pseudokod popisující šifrování a dešifrování + základní API (povinné argumenty v tomto pořadí a jménem):

```
FUNCTION substitute_encrypt(plaintext, key)
  // Vytvoříme mapování: každé písmeno z alphabet → odpovídající znak z key
  mapping ← DICTIONARY()
  FOR i FROM 1 TO LENGTH(alphabet) DO:
    mapping[alphabet[i]] ← key[i]
  encrypted_text ← empty string
  FOR each char IN plaintext DO:
    IF char IS IN mapping THEN:
      encrypted_text ← encrypted_text + mapping[char]
    ELSE:
      encrypted_text ← encrypted_text + char
  RETURN encrypted_text

FUNCTION substitute_decrypt(ciphertext, key)
  // Vytvoříme inverzní mapování: každé písmeno z key → odpovídající znak z alphabet
  reverse_mapping ← DICTIONARY()
  FOR i FROM 1 TO LENGTH(alphabet) DO:
    reverse_mapping[key[i]] ← alphabet[i]
  decrypted_text ← empty string
  FOR each char IN ciphertext DO:
    IF char IS IN reverse_mapping THEN:
      decrypted_text ← decrypted_text + reverse_mapping[char]
    ELSE:
      decrypted_text ← decrypted_text + char
  RETURN decrypted_text
```

Ukázka

text_1000_sample_1_ciphertext.txt

ABM_DEAOMARDHMAVA_VNAERDALD_UAOMAZDNYPAA_VZHBDSVANDAYVWAWIOPABCKVBMARDLMABSDBMAYDOPAXDAWMRDZACYVSANDAYUNDACMWPBSV
AHSVBMIAYDOPAXDAWMRDZAMYDBKDSAOBUKWVABPNWMZUSA_ABM_IAVACMNYVBUSANDACKDOAWMSVAXDOAKDWSAZHKVCYUBDACMXDODNACKDNDAER
DAIXDSVANABM_DEAOBVAWKMWPA_CDYACMXOAEINUEDAOVSAOMBD_IAYDAVNCMRALSU_AWAHKVRUZUEAWVEAZHZDNA_CVYWPANWKUCDSA_ILPA_CVYWPA
NAYDLMIANDAERMIARDRUAVRUAOMCKDOIAVRUA_CVYWPAZMCVWAEUARDKM_IEUNAEINUEAYMAIODSVYAVLPNABUODSAVLPALPSMAXUNYMA_DAXNDEAYDAE
DSVAKVOVAEPNSUNA_DALPZHAEMHSVAXDNYDAXDORMIANSPNDYAZMNAEUAKDWSA_CVYWPARDEI_DNALIOALPNAEINDSABPOVYAYMAZMARDZHZDNAVARDNEU
NARDLMALPAYDAMOBD_SUAVAXVAANCINYUSVAKIZDAOMAWSURVABUOUNAUARVAYMAXNDEAEPNSDSVA_DALPZHANSVANAYDLMIAOMCKDOIAOMBDOSVALPZH
AYMAOMBDOSVALPZHAYMAXUNYDAVSDAAYPAXNUAYVEARDWODA_VNRMILDRAXOUAWARUAHSDOARUWOPAEDARDRVCVOSMACYVYANDAYDARVAYMAWOP_AX
DAZSMBDWACKURZD_RVAEPNSUANUA_DAXDARVANBDYDANVEAEVNAXUAKVOACMHSDOSARVARUAIYKP_RDRPEVAMZUEVACKDZDAXDRARDOMBDOSA_VCKUYAA
YVWABUOUNABPODZHSVAYPARDIEUNAVRUASHVYAYPAEUSPAVSDACMZHMCAWOP_AXNDEANUAYMACVW

text_1000_sample_1_key.txt:

VLZODTQHUXWSERMCFKNYIBJGP_A

Soubor: text_1000_sample_1_plaintext.txt:

_VOZEM_DO_NEHO_A_ZAS_MNE_BEZI_DO_CESTY__ZACHVELA_SE_TAK_KUDY_VPRAVO_NEBO_VLEVO_TEDY_JE_KONEC_PTAL_SE_TISE_POKYVLA_HLAVOU_TEDY
_JE_KONEC_OTEVREL_DVIRKA_VYSKOCIL_Z_VOZU_A_POSTAVIL_SE_PRED_KOLA_JED_REKL_CHRAPTIVE_POJEDES_PRESE_MNE_UJELA_S_VOZEM_DVA_KROKY_ZPE

T_POJD_MUSIME_DAL_DOVEZU_TE_ASPON_BLIZ_K_HRANICIM_KAM_CHCES_ZPATKY_SKRIPEL_ZUBY_ZPATKY_S_TEBOU_SE_MNOU_NENI_ANI_DOPREDU_ANI_ZPATKY_COPAK_MI_NEROZUMIS_MUSIM_TO_UDELAT_ABYS_VIDEL_ABY_BYLO_JISTO_ZE_JSEM_TE_MELA_RADA_MYSLIS_ZE_BYCH_MOHLA_JESTE_JEDNOU_SLYSET_COS_MI_REKL_ZPATKY_NEMUZES_BUD_BYS_MUSEL_VYDAT_TO_CO_NECHCES_A_NESMIS_NEBO_BY_TE_ODVEZLI_A_JA__SPUSTILA_RUCE_DO_KLINA_VIDIS_I_NA_TO_JSEM_MYSLELA_ZE_BYCH_SLA_S_TEBOU_DOPREDU_DOVEDLA_BYCH_TO_DOVEDLA_BYCH_TO_JISTE_ALE__TY_JSI_TAM_NEKDE_ZASNOUBEN_JDI_K_NI_HLED_NIKDY_ME_NENAPADLO_PTAT_SE_TE_NA_TO_KDYZ_JE_CLOVEK_PRINCEZNA_MYSLI_SI_ZE_JE_NA_SVETE_SAM_MAS_JI_RAD_POHLEDL_NA_NI_UTRYZNENYMA_OCIMA_PRECE_JEN_NEDOVEDL_ZAPRIT__TAK_VIDIS_VYDECHLA_TY_NEUMIS_ANI_LHAT_TY_MILY_ALE_POCHOP_KDYZ_JSEM_SI_TO_PAK