UNIWERSYTET KOMISJI EDUKACJI NARODOWEJ
W KRAKOWIE

Instytut Bezpieczeństwa i Informatyki

KRYPTOGRAFIA
(ćwiczenia laboratoryjne)

Ćwiczenie numer:
1

Imię i nazwisko: Grzegorz Golonka
Numer grupy: L2
Studia stacjonarne

Czas realizacji zajęć: 90 min
Temat ćwiczenia: Szyfr Playfair

1. Wstęp

Celem ćwiczenia było zapoznanie się z algorytmem szyfrowania Playfair oraz jego implementacją w języku Python. Zadanie obejmowało stworzenie wersji algorytmu dla standardowego alfabetu angielskiego (z pominięciem litery 'J' lub traktowaniem jej jako 'I') oraz rozszerzenie go do obsługi pełnego alfabetu polskiego. Implementacja miała umożliwiać szyfrowanie tekstu zgodnie z zasadami klasycznego szyfru Playfair.

2. Wprowadzenie teoretyczne

Szyfr Playfair to klasyczny szyfr symetryczny, w którym szyfrowane są pary liter (digramy), co czyni go odporniejszym na podstawową analizę częstotliwości niż szyfry monoalfabetyczne. Podstawą działania algorytmu jest siatka liter o ustalonym rozmiarze – standardowo 5x5 dla alfabetu angielskiego (z łączeniem liter 'I' oraz 'J'), oraz 6x6 dla rozszerzonego alfabetu polskiego.

Generowanie siatki rozpoczyna się od wpisania unikalnych liter z klucza szyfrującego, a następnie uzupełnienia pozostałych pól literami alfabetu, pomijając duplikaty. W przypadku siatki 6x6, konieczne jest dodanie znaków uzupełniających (np. V, Q, X, 0), by osiągnąć pełne 36 pól.

Proces szyfrowania obejmuje :

- **Przygotowanie tekstu** – konwersję na wielkie litery, usunięcie niedozwolonych znaków, zamianę 'J' na 'I' (w wersji angielskiej), oraz podział na pary. W przypadku pary zawierającej identyczne litery – lub tekstu o nieparzystej długości – stosuje się znak dopełniający (np. 'X').

- **Szyfrowanie par** – na podstawie pozycji liter w siatce:

    - **Ten sam wiersz** – każdą literę zastępuje kolejna po prawej (z zawijaniem).
    - **Ta sama kolumna** – każdą literę zastępuje kolejna poniżej (z zawijaniem).
    - **Różne wiersze i kolumny** – litery zamieniane są na odpowiadające im w przeciwległych rogach prostokąta.

3. Opis implementacji

- **3. Konfiguracja globalna**

Na początku skryptu zdefiniowano osobne zestawy konfiguracyjne dla języka polskiego i angielskiego – zawierające bazowe alfabety, znaki dopełniające i rozmiary siatek (5x5 i 6x6). Uwzględniono również walidację, zapewniającą poprawność zestawu znaków.

```
import string
import sys
sys.stdout.reconfigure(encoding='utf-8')

POLISH_ALPHABET_BASE = "AĄBCĆDEĘFGHIJKLŁMNŃOÓPRSŚTUWYZŹŻ"
POLISH_GRID_FILLERS = "VQX0"
POLISH_FULL_SET = POLISH_ALPHABET_BASE + POLISH_GRID_FILLERS
POLISH_GRID_SIZE = 6
PAD_CHAR_PL = "X"

ENGLISH_ALPHABET_BASE = string.ascii_uppercase.replace("J", "")
ENGLISH_FULL_SET = ENGLISH_ALPHABET_BASE
ENGLISH_GRID_SIZE = 5
PAD_CHAR_EN = "X"
```

- **Tworzenie tabeli Playfair (create_table)**

Funkcja ta generuje siatkę, umieszczając najpierw litery klucza (bez duplikatów), a następnie pozostałe znaki alfabetu. Dla alfabetu polskiego zapewniono pełne 36 pól poprzez dodanie znaków uzupełniających.

```python
def create_table(self):
    seen = set()
    processed_key = ""
    key = self.key.replace("J", "I") if self.handle_j_like_i else self.key
    for c in key:
        if c in self.char_pool and c not in seen:
            seen.add(c)
            processed_key += c
    table_content = list(processed_key) + [c for c in self.char_pool if c not in seen]
    return [table_content[i:i + self.grid_size] for i in range(0, len(table_content), self.grid_size)]
```

- **Przygotowanie tekstu do szyfrowania (prepare_text):**

Funkcja `prepare_text` realizuje wszystkie etapy przygotowania tekstu: konwersję na wielkie litery, oczyszczenie z niedozwolonych znaków, zamianę 'J' na 'I' (dla EN), podział na pary liter, uzupełnianie znakami dopełniającymi. Rezultatem jest lista par gotowych do zaszyfrowania.

```python
def prepare_text(self, text):
    text = text.upper().replace("J", "I") if self.handle_j_like_i else text.upper()
    text = "".join(c for c in text if c in self.char_pool)
    if not text:
        return []
    pairs = []
    i = 0
    while i < len(text):
        a = text[i]
        b = text[i + 1] if i + 1 < len(text) else self.pad_char
        if a == b:
            pairs.append(a + self.pad_char)
            i += 1
        else:
            pairs.append(a + b)
            i += 2
    if len(pairs[-1]) == 1:
        pairs[-1] += self.pad_char
    return pairs
```

- **Znajdowanie pozycji znaku (find_position)**

Funkcja pomocnicza `find_position` zwraca współrzędne litery w siatce. Wykorzystywana jest przy szyfrowaniu każdej pary.

```
def find_position(self, letter):
    for row_idx, row in enumerate(self.table):
        if letter in row:
            return row_idx, row.index(letter)
    return None
```

- **Transformacja Playfair (Szyfrowanie pary) (playfair_transform):**

Właściwa logika szyfrowania:
- Jeśli litery są w tym samym wierszu – przesuwane są w prawo.
- Jeśli w tej samej kolumnie – przesuwane są w dół.
- Jeśli tworzą prostokąt – zamieniane są kolumnami.

```
def transform_pair(self, pair, encrypt=True):
    a, b = pair
    r1, c1 = self.find_position(a)
    r2, c2 = self.find_position(b)

    if r1 == r2:
        shift = 1 if encrypt else -1
        return self.table[r1][(c1 + shift) % self.grid_size] + self.table[r2][(c2 + shift) % self.grid_size]
    elif c1 == c2:
        shift = 1 if encrypt else -1
        return self.table[(r1 + shift) % self.grid_size][c1] + self.table[(r2 + shift) % self.grid_size][c2]
    else:
        return self.table[r1][c2] + self.table[r2][c1]
```

- **Proces szyfrowania i deszyfrowania:**

Zaimplementowano pełny proces szyfrowania i deszyfrowania poprzez funkcję `process`. W oparciu o przygotowaną siatkę i tekst wejściowy, program generuje szyfrogram lub tekst jawny.

```python
def process(self, text, encrypt=True):
    pairs = self.prepare_text(text)
    return ''.join(self.transform_pair(pair, encrypt) for pair in pairs)

def encrypt(self, text):
    return self.process(text, encrypt=True)

def decrypt(self, text):
    return self.process(text, encrypt=False)
```

- **Funkcje dla języków (`encrypt_pl`, `encrypt_en`, `decrypt_pl`, `decrypt_en`)**

Dla ułatwienia użytkowania powstały funkcje opakowujące z predefiniowanymi parametrami dla języka polskiego i angielskiego.

```python
# --- Przykład użycia ---

cipher_pl = PlayfairCipher("Gęś", lang='PL')
text_pl = "grzegorz golonka"
print("Tabela PL:")
cipher_pl.print_table()
encrypted_pl = cipher_pl.encrypt(text_pl)
decrypted_pl = cipher_pl.decrypt(encrypted_pl)

print(f"PL: '{text_pl}' -> {encrypted_pl} -> {decrypted_pl}")

cipher_en = PlayfairCipher("START", lang='EN')
text_en = "Programm"
print("\nTabela EN:")
cipher_en.print_table()
encrypted_en = cipher_en.encrypt(text_en)
decrypted_en = cipher_en.decrypt(encrypted_en)

print(f"EN: '{text_en}' -> {encrypted_en} -> {decrypted_en}")
```

Testowanie i wyniki

Dla języka polskiego:

- Klucz: `Gęś`
- Tekst jawny: `Skóra`
- Wynik: `BNWHŚNZÓŚNKÓÓIAQ`

Dla języka angielskiego:

- Klucz: `START`
- Tekst jawny: `Programm`
- Wynik: `QAUDBRKZKZ`

```
Tabela PL:
G | Ę | Ś | A | Ą | B
C | Ć | D | E | F | H
I | J | K | L | Ł | M
N | Ń | O | Ó | P | R
S | T | U | W | Y | Z
Ź | Ż | V | Q | X | 0
----------------------
PL: 'grzegorz golonka' -> BNWHŚNZ0ŚNKÓOIĄQ -> GRZEGORZGOLONKAX

Tabela EN:
S | T | A | R | B
C | D | E | F | G
H | I | K | L | M
N | O | P | Q | U
V | W | X | Y | Z
------------------
EN: 'Programm' -> QAUDBRKZKZ -> PROGRAMXMX
```

4. Omówienie wyników

Zaimplementowany algorytm działa poprawnie w obu wersjach językowych. Dla języka polskiego zastosowano siatkę 6x6 z dodatkowymi znakami i obsługą znaków diakrytycznych.

Dla języka angielskiego wykorzystano klasyczne 5x5 z traktowaniem `J` jako `I`. Wyniki szyfrowania potwierdzają zgodność działania z zasadami algorytmu.

5. Wnioski

Zadanie zostało zrealizowane zgodnie z założeniami. Stworzono w pełni działającą implementację szyfru Playfair w języku Python, obejmującą zarówno klasyczny wariant angielski, jak i rozszerzoną wersję polską. Zrozumienie i implementacja obu wersji pozwoliły na poznanie wyzwań adaptacji kryptografii klasycznej do różnych systemów językowych.