

Sprawozdanie z pracowni specjalistycznej

Sztuczna inteligencja

Ćwiczenie numer: 2-3

Temat: Implementacja podstawowych modułów kryptograficznych

Wykonujący ćwiczenie: **Michał Mitrosz**

Studia dzienne

Kierunek: Informatyka

Semestr: VI

Grupa zajęciowa: PS 4

Prowadzący ćwiczenie: mgr inż. Dariusz Jankowski

Data wykonania ćwiczenia:
21.05.2021 r.

1. Realizacja zadań

1. Zaimplementuj algorytm kodujący i dekodujący z wykorzystaniem szyfru prostego przestawiania „rail fence” dla $k = n$.

M = CRYPTOGRAPHY, n=3

C										
	R		P		T		O		A	
		Y				G			P	H

C = CTARPORPYYGH

Rys. 1 - Przykład

```
def railfence_encode(M, n = 3):  
    arrs = [[] for i in range(n)]  
  
    current = 0  
    dir = 1  
  
    for i in M:  
        arrs[current].append(i)  
        current += dir  
        if dir == 1 and current == n-1:  
            dir = -1  
        if dir == -1 and current == 0:  
            dir = 1  
  
    return "".join(["".join(i) for i in arrs])
```

Rys. 2 - Kod realizujący kodowanie rail fence

```

def railfence_decode(M, n = 3):
    arrs = [[] for i in range(n)]

    current = 0
    dir = 1

    for i in range(len(M)):
        for j in range(n):
            if j == current:
                arrs[j].append(True)
            else:
                arrs[j].append("")
        current += dir
        if dir == 1 and current == n-1:
            dir = -1
        if dir == -1 and current == 0:
            dir = 1

    index = 0

    for i in range(n):
        for j in range(len(M)):
            if arrs[i][j]:
                arrs[i][j] = M[index]
                index += 1

    ret = ""
    current = 0
    dir = 1

    for i in range(len(M)):
        for j in range(n):
            if arrs[j][i]:
                ret += arrs[j][i]

    return ret

```

Rys. 3 - Kod realizujący dekodowanie rail fence

```

def test_1(self):
    self.assertEqual(railfence_encode("CRYPTOGRAPHY", 3), "CTARPORPYYGH")
    self.assertEqual(railfence_decode("CTARPORPYYGH", 3), "CRYPTOGRAPHY")
    self.assertEqual(railfence_encode("ALAMAKOTAMMM", 3), "AAALMKTMMMAOM")
    self.assertEqual(railfence_decode("AAALMKTMMMAOM", 3), "ALAMAKOTAMMM")

```

Rys. 4 - Testy dla kodowania rail fence

2. Zaimplementuj kryptosystem przestawieniowy bazujący na przykładzie 2a dla $d = 5$ oraz klucza $\text{key} = 3-4-1-5-2$

$M = \text{CRYPTOGRAPHYOSA}$, $\text{key} = 3-1-4-2$

1	2	3	4
C	R	Y	P
T	O	G	R
A	P	H	Y
O	S	A	

$C = \text{YCPRGTROHAYPAOS}$

Rys. 5 - Przykład 2a

```
def encode_2a(M, n):  
    key = list(map(lambda a: int(a) - 1, n.split("-")))  
  
    ret = ""  
    for i in range(len(M) + len(key)):  
        index = ((i//len(key)) * len(key)) + key[i%len(key)]  
        if index < len(M):  
            ret += M[index]  
  
    return ret
```

Rys. 6 - Kod realizujący kodowanie systemem przestawieniowym

```

def decode_2a(M, n):
    key = list(map(lambda a: int(a) - 1, n.split("-")))
    ret = []

    for i in range(len(M) // len(key) + 1):
        ret.append(['' for i in range(len(key))])
        for j in range(len(key)):
            try:
                ret[len(ret)-1][key[j]] = M[i*len(key) + j]
            except:
                pass

    ret = list(map(lambda a: "".join(a), ret))
    ret = "".join(ret)

    if len(M) % len(key) > len(key) // 2:
        ret = ret[:-2] + ret[-1] + ret[-2]

    return ret

```

Rys. 7 - Kod realizujący dekodowanie systemem przestawieniowym

```

def test_2a(self):
    self.assertEqual(encode_2a("CRYPTOGRAPHY", "3-4-1-5-2"), "YPCTRRAOPGHY")
    self.assertEqual(decode_2a("YPCTRRAOPGHY", "3-4-1-5-2"), "CRYPTOGRAPHY")
    self.assertEqual(encode_2a("CRYPTOGRAPHYOSA", "3-1-4-2"), "YCPRGTROHAYPAOS")
    self.assertEqual(decode_2a("YCPRGTROHAYPAOS", "3-1-4-2"), "CRYPTOGRAPHYOSA")
    self.assertEqual(encode_2a("ALAMAKOTAMMM", "3-1-4-2"), "AAMLOATKMAMM")
    self.assertEqual(decode_2a("AAMLOATKMAMM", "3-1-4-2"), "ALAMAKOTAMMM")

```

Rys. 8 - Testy dla kodowania przestawieniowego

3. Zaimplementuj kryptosystem przedstawieniowy bazujący na przykładzie 2b oraz 2c dla dowolnego klucza.

M=HERE IS A SECRET MESSAGE ENCIPHERED BY TRANSPOSITION²

Key=CONVENIENCE										
C	O	N	V	E	N	I	E	N	C	E
1	10	7	11	3	8	6	4	9	2	5
H	E	R	E	I	S	A	S	E	C	R
E	T	M	E	S	S	A	G	E	E	N
C	I	P	H	E	R	E	D	B	Y	T
R	A	N	S	P	O	S	I	T	I	O
N										

C=HECRN CEYI ISEP SGDI RNT0 AAES RMPN SSRO EEBT ETIA EEHS³

Rys. 9 - Przykład 2b

```
def encode_2b(M, key):
    if len(key) > len(M):
        key = key[:len(M)]

    indices = get_indices(key)

    ret = ""

    for x in range(len(key)):
        for y in range(int(len(M) / len(key)) + 1):
            index = (y*len(key)) + indices.index(x+1)
            if index < len(M):
                ret += M[index]

    return ret
```

Rys. 10 - Kod realizujący kodowanie do przykładu 2b

```

def decode_2b(M, key):
    if len(key) > len(M):
        key = key[:len(M)]

    indices = get_indices(key)

    longer_blocks = len(M) % len(indices)
    shorter_length = len(M) // len(indices)

    blocks = []
    block_start = 0

    for i in range(len(indices)):
        letters_in_block = shorter_length+1 if longer_blocks else shorter_length
        blocks.append(M[block_start: block_start+letters_in_block])
        block_start += letters_in_block
        if longer_blocks:
            longer_blocks -= 1

    output = ""

    for i in range(shorter_length+1): # position in block
        for j in indices:             # block number
            if i < len(blocks[j-1]):
                output += blocks[j-1][i]

    return output

```

Rys. 11 - Kod realizujący dekodowanie do przykładu 2b

```

def test_2b(self):
    self.assertEqual(encode_2b("HEREISASECRETMESSAGEENCIPHEREDBYTRANSPOSITION", "CONVENIENCE"), "HECRNCEYIIEPSGDIRNTOAAESRMPNSSROEBTETIAEEHS")
    self.assertEqual(decode_2b("HECRNCEYIIEPSGDIRNTOAAESRMPNSSROEBTETIAEEHS", "CONVENIENCE"), "HEREISASECRETMESSAGEENCIPHEREDBYTRANSPOSITION")
    self.assertEqual(encode_2b("ALAMAKOTAMMM", "CONVENIENCEMM"), "AMATMOMAKALM")
    self.assertEqual(decode_2b("AMATMOMAKALM", "CONVENIENCEMM"), "ALAMAKOTAMMM")

```

Rys. 12 - Testy do przykładu 2b

C	O	N	V	E	N	I	E	N	C	E
1	10	7	11	3	8	6	4	9	2	5
<hr/>										
H										
E	R	E	I	S	A	S	E	C	R	
E	T	M	E	S						
S	A	G	E	E	N	C	I			
P	H	E	R	E	D	B	Y	T	R	A
N	S	P	O	S	I	T				
I	O	N								

C=HEESPNI RR SSEES EIY A SCBT EMGEPN ANDI CT RTAHSO IEERO

Rys. 13 - Przykład 2c

```
def encode_2c(M, key):
    if len(key) > len(M):
        key = key[:len(M)]

    indices = get_indices(key)

    blocks = []
    M_index = 0

    for i in range(len(indices)):
        blocks.append(M[M_index:M_index + indices.index(i+1) + 1])
        M_index = M_index + indices.index(i+1) + 1

        if M_index >= len(M):
            break

    output = ""

    for index in range(len(indices)):
        for block in blocks:
            if indices.index(index+1) < len(block):
                output += block[indices.index(index+1)]

    return output
```

Rys. 14 - Kod realizujący kodowanie do przykładu 2c


```

def decode_2c(M, key):
    if len(key) > len(M):
        key = key[:len(M)]

    indices = get_indices(key)

    M_left = len(M)
    blocks = []

    for i in range(len(indices)):
        string = (indices.index(i+1)+1 if indices.index(i+1)+1 < M_left else M_left) * "."
        blocks.append(string)
        M_left -= len(string)
        if not M_left:
            break

    M_index = 0

    for i in range(len(indices)):
        for j in range(len(blocks)):
            if M_index == len(M):
                break
            if indices.index(i+1) < len(blocks[j]):
                blocks[j] = blocks[j][:indices.index(i+1)] + M[M_index] + blocks[j][indices.index(i+1)+1:]
                M_index += 1

    return "".join(blocks)

```

Rys. 15 - Kod realizujący dekodowanie do przykładu 2c

```

def test_2c(self):
    self.assertEqual(encode_2c("HEREISASECRETMESSAGEENCIPHEREDBYTRANSPOSITION", "CONVENIENCE"), "HEESPNIIRSSEESIYASCBTEMGEPNANDICTRTAHSOIEERO")
    self.assertEqual(decode_2c("HEESPNIIRSSEESIYASCBTEMGEPNANDICTRTAHSOIEERO", "CONVENIENCE"), "HEREISASECRETMESSAGEENCIPHEREDBYTRANSPOSITION")
    self.assertEqual(encode_2c("ALAMAKOTAMMM", "CONVENIENCEMMM"), "ALMMKATMOMAA")
    self.assertEqual(decode_2c("ALMMKATMOMAA", "CONVENIENCEMMM"), "ALAMAKOTAMMM")

```

Rys. 16 - Testy do przykładu 2c

4. Zaimplementuj szyfr cezara bazując na przykładzie 3b.

szyfrowanie: $c = (a * k_1 + k_0) \bmod n$
deszyfrowanie: $a = [c + (n - k_0)] k_1^{\varphi(n)-1} \bmod n$

dla $n=21$ $\varphi(n)=12$

k_1, k_0 muszą być pierwsze względem n .

Rys. 17 - Przykład 3b

```
def caesar_encode(M, k0, k1, n=26):  
    ret = ""  
    for i in M:  
        ret += chr(((ord(i) - ord("A")) * k1 + k0) % n + ord("A"))  
    return ret
```

Rys. 18 - Kod realizujący kodowanie z przykładu 3b

```
from sympy.ntheory.factor_ import totient
```

```
def caesar_decode(M, k0, k1, n=26):  
    ret = ""  
    for i in M:  
        ret += chr((((ord(i) - ord("A")) + (n - k0)) * (k1 ** (totient(n)-1))) % n + ord("A"))  
    return ret
```

Rys. 19 - Kod realizujący dekodowanie z przykładu 3b

```
def test_3b(self):  
    self.assertEqual(caesar_encode("CRYPTOGRAPHY", 5, 7), "TURGIZVUFGCR")  
    self.assertEqual(caesar_decode("TURGIZVUFGCR", 5, 7), "CRYPTOGRAPHY")  
    self.assertEqual(caesar_encode("ALAMAKOTAMMM", 5, 7), "FEFLFXZIFLLL")  
    self.assertEqual(caesar_decode("FEFLFXZIFLLL", 5, 7), "ALAMAKOTAMMM")
```

Rys. 20 - Testy do przykładu 3b

5. Zaimplementuj kryptosystem bazujący na tablicy Vigenere'a

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

Rys. 21 - Tablica Vigenere'a

M = CRYPTOGRAPHY
K = BREAKBREAKBR
EK(M) = DICPDPXVAZIP

Rys. 22 - Przykład

```
def vigenere_encode(M, key):
    ret = ""
    for index, item in enumerate(M):
        a = ord(item) - ord("A")
        b = ord(key[index % len(key)]) - ord("A")
        ret += chr((a + b) % 26 + ord("A"))
    return ret
```

Rys. 23 - Kod realizujący kodowanie

```
def vigenere_decode(M, key):
    ret = ""
    for index, item in enumerate(M):
        a = ord(item) - ord("A")
        b = ord(key[index % len(key)]) - ord("A")
        ret += chr((a - b) % 26 + ord("A"))
    return ret
```

Rys. 24 - Kod realizujący dekodowanie

```
def test_4(self):
    self.assertEqual(vigenere_encode("CRYPTOGRAPHY", "BREAK"), "DICPDPXVAZIP")
    self.assertEqual(vigenere_decode("DICPDPXVAZIP", "BREAK"), "CRYPTOGRAPHY")
    self.assertEqual(vigenere_encode("ALAMAKOTAMMM", "BREAK"), "BCEMKLFXAWND")
    self.assertEqual(vigenere_decode("BCEMKLFXAWND", "BREAK"), "ALAMAKOTAMMM")
```

Rys. 25 - Testy do zadania

2. Wnioski

Wszystkie zadania udało się zrealizować bez większych problemów, wszystkie testy zaprezentowane na zrzutach zwróciły oczekiwany rezultat.