Cryptography module, Exercises 1 (unassessed) - Answers

All the code contained in this exercise solution is hosted at: https://github.com/amar-laksh/UNI/assignments/CRYPTO/code

Answer - 1:

Listing 1 contains python code to brute-force the given cipher-text "*AVUEVLETSEISBNACBOOLEOBTILBDLCOBOOE*".

Listing 1: ex1.1.py

```
from math import ceil

def encode(msg, key):
    cipher = ""
    for rails in range(0, key):
        for char in range(rails, len(msg), key):
            cipher += msg[char]
    return cipher

def decode(cipher, key):
    msg = ""
    count = ceil(len(cipher)/key)
    for rails in range(0, count):
        for c in range(rails, len(cipher), count):
            msg += cipher[c]
    return msg

def crack_cipher(cipher):
    for key in range(1, len(cipher)):
        print(decode(cipher, key))

cipher = "AVUEVLETSEISBNACBOOLEOBTILBDLCOBOOE"

crack_cipher(cipher)
```

Listing 2 contains the output of the python code in **Listing 1** containing the message, "*ALICELOVESBOBBUTBOBDOESNOTLOVEALICE*" shown in the bold.

Listing 2: output of ex1.1.py

```
AVUEVLETSEISBNACBOOLEOBTILBDLCOBOOE
AOVLUEEOVBLTEITLSBEDILSCBONBAOCOBEO
ABIVNLUABECDVBLLOCEOOTLBSEOEOOIBEST
AEODVILLUSECEBOOVNBBLATOECIOTBLESOB
ATAOLVSCBCUEBTOEIOIBVSOLOLBLBOENEDE
AEBOIOVTNLLBUSAEBOEECODOVIBBLELSOTC
ALICELOVESBOBBUTBOBDOESNOTLOVEALICE
ALICELOVESBOBBUTBOBDOESNOTLOVEALICE
AVSBBEILOVLENOOLCOUEIAOBBOEETSCLTDB
AVSBBEILOVLENOOLCOUEIAOBBOEETSCLTDB
AVSBBEILOVLENOOLCOUEIAOBBOEETSCLTDB
AEEEBCOOIDOOVVTINBLBLLBEULSSAOETBCO
AEEEBCOOIDOOVVTINBLBLLBEULSSAOETBCO
AEEEBCOOIDOOVVTINBLBLLBEULSSAOETBCO
```

```
AEEEBCOOIDOOVVTINBLBLLBEULSSAOETBCO
AEEEBCOOIDOOVVTINBLBLLBEULSSAOETBCO
AEEEBCOOIDOOVVTINBLBLLBEULSSAOETBCO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
AUVESIBABOEBIBLOOEVELTESNCOLOTLDCBO
```

Answer - 2:

Listing 3 contains the python code to perform encryption and decryption using the block cipher scheme mentioned in the question.

Listing 3: ex1.2.py

```python
ROUNDS = 2


def key_function(K, i):
    return K + 75 * (i % 256)


def F(Ki, Pi):
    return 127 * Ki + (Pi % 256)


def encrypt(msg, key):
    Li = msg[0]
    Ri = msg[1]
    temp = 0
    for i in range(0, ROUNDS):
        Ki = key_function(key, i)
        temp = Li ^ F(Ki, Ri)
        Li = Ri
        Ri = temp
    return [Ri, Li]


def decrypt(cipher, key):
    Li = cipher[1]
    Ri = cipher[0]
    temp = 0
    for i in range(ROUNDS, 0, -1):
        Ki = key_function(key, (i - 1))
        temp = Ri ^ F(Ki, Li)
        Ri = Li
        Li = temp
```

```
    return [Li, Ri]

print(encrypt([86, 83], 89))
```

Listing 4 contains the output of the python code in **Listing 3** containing the encrypted cipher-text "*[20955, 11308]*".

Listing 4: output of ex1.2.py

```
[20955, 11308]
```

Answer - 3:

Listing 5 contains the python code to perform encryption using a modified version of the DES implementation in Python.

The modified fork of the DES python library can be found at: https://github.com/amar-laksh/des

Listing 5: ex1.3.py

```
import sys
sys.path.insert(1, 'des')
from des import DesKey
key = DesKey(b"00000000", rounds=1)
print(key.encrypt(b"00000000").hex())
```

Listing 6 contains the output of the python code in **Listing 5** containing the encrypted cipher-text "*3574206561312435*".

Listing 6: output of ex1.3.py

```
3574206561312435
```

This cipher-text is obtained even when the encryption key and plain-text is all zeroes.