HW 1
James Foti
Red ID: 820124143
CS 574
Professor Song
2/24/2020

1) Decrypt the following ciphertext which was encrypted with Caesar Cipher. The key shift isgiven in each case. Show the substitution key mapping in each case.

a. Cipher Text: "dtzw ynrj nx qnrnyji xt itsy bxfyj ny sn qnansl xtrjtsj jqxjx qnaj", Key Shift: 5

Key shift (5) mapping for the entire alphabet:
a $\Longrightarrow$ v
b $\Longrightarrow$ w
c $\Longrightarrow$ x
d $\Longrightarrow$ y
e $\Longrightarrow$ z
f $\Longrightarrow$ a
g $\Longrightarrow$ b
h $\Longrightarrow$ c
i $\Longrightarrow$ d
j $\Longrightarrow$ e
k $\Longrightarrow$ f
l $\Longrightarrow$ g
m $\Longrightarrow$ h
n $\Longrightarrow$ i
o $\Longrightarrow$ j
p $\Longrightarrow$ k
q $\Longrightarrow$ l
r $\Longrightarrow$ m
s $\Longrightarrow$ n
t $\Longrightarrow$ o
u $\Longrightarrow$ p
v $\Longrightarrow$ q
w $\Longrightarrow$ r
x $\Longrightarrow$ s
y $\Longrightarrow$ t
z $\Longrightarrow$ u

decrypted ciphertext: your time is limited so dont wsate it ni living someone elses live

b. Cipher Text: "xli gsso aew e kssh gsso ew gssow sk erh ew gssow ks wil airx", Key Shift: 4

Key shift (4) mapping for the entire alphabet:
a $\Longrightarrow$ w
b $\Longrightarrow$ x
c $\Longrightarrow$ y
d $\Longrightarrow$ z
e $\Longrightarrow$ a
f $\Longrightarrow$ b
g $\Longrightarrow$ c
h $\Longrightarrow$ d
i $\Longrightarrow$ e
j $\Longrightarrow$ f
k $\Longrightarrow$ g
l $\Longrightarrow$ h
m $\Longrightarrow$ i
n $\Longrightarrow$ j

o $\Longrightarrow$ k
p $\Longrightarrow$ l
q $\Longrightarrow$ m
r $\Longrightarrow$ n
s $\Longrightarrow$ o
t $\Longrightarrow$ p
u $\Longrightarrow$ q
v $\Longrightarrow$ r
w $\Longrightarrow$ s
x $\Longrightarrow$ t
y $\Longrightarrow$ u
z $\Longrightarrow$ v

decrypted ciphertext: "the cook was a good cook as cooks og and as cooks go seh went"

2) For any block cipher, the fact that it is a nonlinear function is crucial to its security. To see this, suppose that we have a linear block cipher EL that encrypts 128-bit blocks of plaintext into 128-bit blocks of ciphertext. Let EL(k, m) denote the encryption of a 128-bit message m under a key k (the actual bit length of k is irrelevant). Thus

EL(k, [m1 **XOR** m2]) = EL (k, m1) **XOR** EL (k, m1) for all 128-bit patterns m1, m2

Describe how, with 128 chosen ciphertexts, an adversary can decrypt any ciphertext without knowledge of the secret key k. (A "chosen ciphertext" means that an adversary has the ability to choose a ciphertext and then obtain its decryption. Here, you have 128 plaintext/ciphertext pairs to work with and you can choose the value of the ciphertexts.)

$$c_i \epsilon \{0,1\}^{128}$$
$$m_i = i^{th} \text{ plaintext}$$
$$c_i = i^{th} \text{ cipher text for } m_i$$
$$i = \text{position of the 128-bit message}$$

An adversary will try to take the cipher text of 128 bits that corresponds to the plain text of $m_1, m_2, \ldots m_{128}$.
The adversary also knows that the cipher text dows not contain all zeros and that there is no empy subset of
The adversary will assume the subset if $I(c) \subseteq \{1, 2, 3, \ldots, 128\}$

Then, $c = \oplus_{i \epsilon I(c)} E\Big(\oplus_{i \epsilon I(c)} m_i \Big)$
$c = 0$ when $c_i = 0$ means that $m = 0$ so the adversary will try for every $c_i \epsilon \{0,1\}^{128}$

3) This problem uses a real-world example of a symmetric cipher, from an old U.S. Special Forces manual (public domain). See the attached document on page 3.

a. Using the two keys (memory words) “cryptographic” and “network security”, encrypt the following message:

“Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful bring two friends.”

Make reasonable assumptions about how to treat redundant letters and excess letters in the memory words and how to treat spaces and punctuation. Indicate what your assumptions are.

b. Decrypt the ciphertext. Show your work.

a. ENCRYPTING THE MESSAGE

For cryptographic (1st key):

The first key is 'cryptographic' which becomes 'cryptogahi' because we removed redundant letters. Then we write the message in a 10x10 matrix because the size of the key word is now 10.

The alphabetical order of 'cryptogahi' is 'a, c, g, h, i, o, p, r, t, y'.

a $\Longrightarrow$ 8th position $\Longrightarrow$ 8th column read first
c $\Longrightarrow$ 1st position $\Longrightarrow$ 1st column read first
g $\Longrightarrow$ 7th position $\Longrightarrow$ 7th column read first
h $\Longrightarrow$ 9th position $\Longrightarrow$ 9th column read first
i $\Longrightarrow$ 10th position $\Longrightarrow$ 10th column read first
o $\Longrightarrow$ 6th position $\Longrightarrow$ 6th column read first
p $\Longrightarrow$ 4th position $\Longrightarrow$ 4th column read first
r $\Longrightarrow$ 2nd position $\Longrightarrow$ 2nd column read first
t $\Longrightarrow$ 5th position $\Longrightarrow$ 5th column read first
y $\Longrightarrow$ 3rd position $\Longrightarrow$ 3rd column read first

The encrypted message becomes: *"trhe hftin brouy rtust eaeth gisre hftea tyrnd irolt aougs hllet inibi tihiu oveuf edmtc esatw tledm nedlr aptse terfo"

For network security (2nd key):

We follow the same steups as before and the 2nd key becomes 'networkscu' and the alphabetical order is 'c, e, k, n, o , r, s, t, u, w'

We follow the same steps as before and the encrypted message becomes: "isrng butlf rrafr lidlp fiyonvsee tbehi hteta eyhat tucme hrgta ioent tusru ieadr foeto lhmet nteds ifwro hutel eitds"

b. DECRYPTING THE MESSAGE

For cryptographic (1st key):

We are going to work backwards by starting with the second key and the second encrypted message (displayed in the table below).

The order second key is '4, 2, 8, 10, 5, 6, 7, 1, 9'
4th row $\Longrightarrow$ read 1st (left to right) $\Longrightarrow$ becomes 1st column
2nd row $\Longrightarrow$ read 2nd $\Longrightarrow$ becomes 2nd column
8th row $\Longrightarrow$ read 3rd $\Longrightarrow$ becomes 3rd column
.
.
.
9th row $\Longrightarrow$ read 10th $\Longrightarrow$ becomes 10th column

We repeat the process for the order of the first key as well: '2, 7, 10, 7, 9, 6, 3, 1, 4, 5.'

Thus, the decrypted message is: "Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful bring two friends."

4) Assume that we are planning to decrypt the cyphertext which was encrypted with DES encryption. We are using an ordinary household computer with 2GHz processor. Estimate the amount of time necessary to crack DES by testing all 56-bit possible keys. Also estimate the similar time for AES encryption with 128-bit key. (Assume that machine takes 100 cycles per brute force against a single key) Note: For this question, the exact answer is not as important as approach to calculate the answer.

$$2GHz = \frac{2 * 10^9 cycles}{seconds}$$

$$\text{56-bit possible keys for DES: } 2^{56} keys$$

$$\text{128-bit possible keys for AES: } 2^{128} keys$$

$$\text{Time to crack DES: } 2^{56} * \frac{100cycles}{key} * \frac{seconds}{2 * 10^9 cycles} * \frac{1min}{60seconds} * \frac{1hour}{60min} * \frac{1day}{24hours} * \frac{1year}{365days} = 114.254$$

$$\text{Time to crack AES: } 2^{128} * \frac{100cycles}{key} * \frac{seconds}{2 * 10^9 cycles} * \frac{1min}{60seconds} * \frac{1hour}{60min} * \frac{1day}{24hours} * \frac{1year}{365days} \approx 5.40 *$$

5) List three applications in which a stream cipher would be desirable. Be sure to explain your answer.

1 - Stream ciphers can be used on devices where their hardware resources are limited because stream ciphers can operate very fast on limited hardware resources.

2 - They can be used where plaintext is inputed with unkown lengths since the length of the plaintext is not a factor where block ciphers cannot encrypt plaintext in which the length of the plaintext is smaller than their block size. Since stream ciphers encrypt each individual bit/byte one at a time, this is not an issue.

3 - Rotor machines found in enigma machineds during WWII because these machines encrypted and decrypted plaintext one letter/number at a time.

6) Write a program to implement DES encryption.

In [1]:
```
from Crypto.Cipher import DES
```

In [2]:
```
class DESMod:
    encrypted_data = b""
    decrypted_data = b""

    def pad(self, text):
        while len(text) % 8 != 0:
            text += b' '
        return text

    def encrypt_file(self, file_reference, key):
        cipher = DES.new(key)

        with open(file_reference, 'rb+') as fw:
            file_contents = fw.read()

            if len(file_contents) % 8 != 0:
                file_contents = self.pad(file_contents)

            self.encrypted_data = cipher.encrypt(file_contents) # Encrypt file
            fw.seek(0) # fw.seek(0) and fw.truncate() clear the file for new data to be
 written
            fw.truncate()
            fw.write(self.encrypted_data)

        # The return statement is only used for displaying the encryption w/ out opening
txt file
        return self.encrypted_data

    def decrypt_file(self, file_reference, key):
        cipher = DES.new(key)

        with open(file_reference, 'rb+') as fw:
            file_contents = fw.read()
            self.decrypted_data = cipher.decrypt(file_contents) # Decrypt file
            fw.seek(0) # fw.seek(0) and fw.truncate() clear the file for new data to be
 written
            fw.truncate()
            fw.write(self.decrypted_data)

         # The return statement is only used for displaying the decryption w/ out openin
g txt file
        return self.decrypted_data
```

In [3]:
```
# Test message to place in the text file: "This is a test message for the DES encryptio
n/decryption!"
des = DESMod()
file_path = "test_file.txt"
key = "8bytekey" # key must be a size of 8
```

In [4]:
```
cipher_text = des.encrypt_file("test_file.txt", key)
print("Contents of file: ", cipher_text)
```

```
Contents of file:  b'\xbd\x19\xae\xbeSo\x0e\xab\xe1l\xce0\x95\x85\x85\xaa\xe5(@\xea\x16
\x99\xc9\xd8s\x9c\'a\xf6\xc8nr=\x86\xe0x\xebl-w\xc6!%\xc3"H9\xac\xce.\xf4\x89\xf8\xc8\xf
4\x9e\x07\x16\x0f\xb7\x1co\xf3\xaa'
```

In [5]:
```python
# Note: the key that was used for encryption MUST be the same for decryption if you want
to see original message!
plain_text = des.decrypt_file("test_file.txt", key)
print("Contents of file: ", plain_text)
```

```
Contents of file:  b'This is a test message for the DES encryption/decryption!       '
```