Fundament Concepts of Cryptography: Assignment One

1.0 - Affine Cipher

1.1 - Eligible Keys

As per the assignment specification, I implemented the Affine Cipher using mod 27 rather than mod 26.

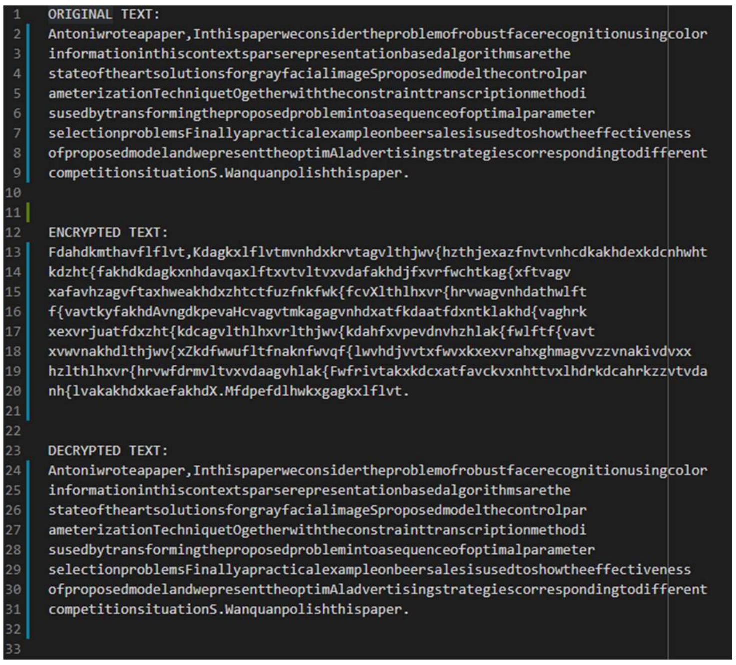
This resulted in an increase in the number of possible eligible keys to 486. This is achieved by there being a total of 18 possible values less than 27, that are coprime with 27.

Possible values for key a: 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26.

Therefore, 18 * 27 = 486. Please see the validKeys.txt file output from the program to see a full list of eligible keys.

1.2 - Recovered plaintext after decryption

Using the keys (A = 4, B =5), the recovered plaintext after decryption is as follows:



```
ORIGINAL TEXT:
Antoniwroteapaper,Inthispaperweconsidertheproblemofrobustfacerecognitionusingcolor
informationinthiscontextsparserepresentationbasedalgorithmsarethe
stateoftheartsolutionsforgrayfacialimageSproposedmodelthecontrolpar
ameterizationTechniquetOgetherwiththeconstrainttranscriptionmethodi
susedbytransformingtheproposedproblemintoasequenceofoptimalparameter
selectionproblemsFinallyapracticalexampleonbeersalesisusedtoshowtheeffectiveness
ofproposedmodelandwepresenttheoptimAladvertisingstrategiescorrespondingtodifferent
competitionsituationS.Wanquanpolishthispaper.


ENCRYPTED TEXT:
Fdahdkmthavflflvt,Kdagkxlflvtmvnhdxkrvtagvlthjwv{hzthjexazfnvtvnhcdkakhdexkdcnhwht
kdzht{fakhdkdagkxnhdavqaxlftxvtvltvxvdafakhdjfxvrfwchtkag{xftvagv
xafavhzagvftaxhweakhdxzhtctfuzfnkfwk{fcvXlthlhxvr{hrvwagvnhdathwlft
f{vavtkyfakhdAvngdkpevaHcvagvtmkagagvnhdxatfkdaatfdxntklakhd{vaghrk
xexvrjuatfdxzht{kdcagvlthlhxvrlthjwv{kdahfxvpevdnvhzhlak{fwlftf{vavt
xvwvnakhdlthjwv{xZkdfwwufltfnaknfwvqf{lwvhdjvvtxfwvxkxexvrahxghmagvvzzvnakivdvxx
hzlthlhxvr{hrvwfdrmvltvxvdaagvhlak{Fwfrivtakxkdcxatfavckvxnhttvxlhdrkdcahrkzzvtvda
nh{lvakakhdxkaefakhdX.Mfdpefdlhwkxgagkxlflvt.


DECRYPTED TEXT:
Antoniwroteapaper,Inthispaperweconsidertheproblemofrobustfacerecognitionusingcolor
informationinthiscontextsparserepresentationbasedalgorithmsarethe
stateoftheartsolutionsforgrayfacialimageSproposedmodelthecontrolpar
ameterizationTechniquetOgetherwiththeconstrainttranscriptionmethodi
susedbytransformingtheproposedproblemintoasequenceofoptimalparameter
selectionproblemsFinallyapracticalexampleonbeersalesisusedtoshowtheeffectiveness
ofproposedmodelandwepresenttheoptimAladvertisingstrategiescorrespondingtodifferent
competitionsituationS.Wanquanpolishthispaper.
```

1.3 - Mathematic Proof

The encryption formula is:

$$\mathrm{E}(x) = (ax + b) \bmod m,$$

The decryption formula is:

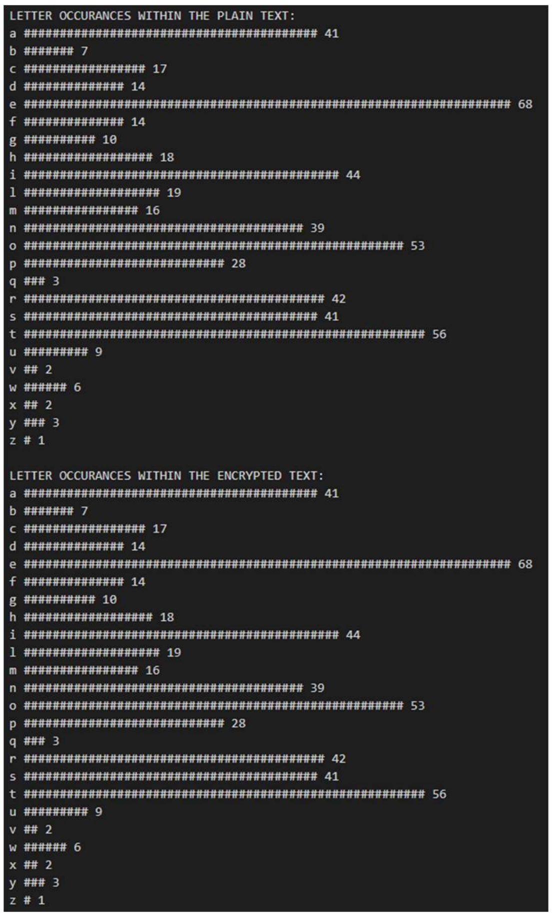
$$\mathrm{D}(x) = a^{-1}(x - b) \bmod m,$$

To prove that the resulting decrypted message equals the original message, we can use these two formulas to show that decryption is the inverse of encryption:

$$\begin{aligned}\mathrm{D}(\mathrm{E}(x)) &= a^{-1}(\mathrm{E}(x) - b) \bmod m \\ &= a^{-1}(((ax + b) \bmod m) - b) \bmod m \\ &= a^{-1}(ax + b - b) \bmod m \\ &= a^{-1}ax \bmod m \\ &= x \bmod m.\end{aligned}$$

The above example indicates that multiplicative inverse of ‘a’ only exists if ‘a’ and ‘m’ are coprime.

1.4 - Letter Distribution of test file



```
LETTER OCCURANCES WITHIN THE PLAIN TEXT:
a ######################################### 41
b ####### 7
c ################# 17
d ############## 14
e #################################################################### 68
f ############## 14
g ########## 10
h ################## 18
i ############################################ 44
l ################### 19
m ################ 16
n ####################################### 39
o ##################################################### 53
p ############################ 28
q ### 3
r ########################################## 42
s ######################################### 41
t ######################################################## 56
u ######### 9
v ## 2
w ###### 6
x ## 2
y ### 3
z # 1

LETTER OCCURANCES WITHIN THE ENCRYPTED TEXT:
a ######################################### 41
b ####### 7
c ################# 17
d ############## 14
e #################################################################### 68
f ############## 14
g ########## 10
h ################## 18
i ############################################ 44
l ################### 19
m ################ 16
n ####################################### 39
o ##################################################### 53
p ############################ 28
q ### 3
r ########################################## 42
s ######################################### 41
t ######################################################## 56
u ######### 9
v ## 2
w ###### 6
x ## 2
y ### 3
z # 1
```

1.5 - How to skip/ignore non-letter symbols

My code checks each character of the input plaintext to identify the following before encryption:

1. Characters between ‘a’ and ‘z’
2. Characters between ‘A’ and ‘Z’
3. All other characters

If the character being checked isn’t an alphabet letter, then it is not encrypted and remains unchanged in the encrypted text.

2.0 - DES Encryption

2.1 - Mathematic Proof

DES decryption is able to retrieve the original plaintext by using the key schedule in reverse order as it steps through the same algorithmic processes as encryption.

During decryption, the reverse permutation is cancelled out by the initial permutation.

Consider the first iteration of the left and right halves in the decryption process:

$$L_{16} = R_{15}$$

$$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$$

R16 is also equal to: $L_{15} \oplus f(R_{15}, K_{16}) \oplus f(R_{15}, K_{16}) = L_{15}$.

Therefore, L16 = R15 and R16 = L15. Proving that decryption is the inverse of encryption to retrieve the plaintext from encrypted text.

2.2 - Pseudocode

2.2.1 - Key generation

```
String Array keygen(String key)
    key = padOrChop(key)
    binaryKey = convert to binary(key)
    permKey = permutePC1(binaryKey)
    leftKey = splitKey(permKey)
    rightKey = splitKey(permKey)
    shiftedKeys = shiftKeys(leftKey, rightKey)
    for i = 0;  i < length of shiftedKeys array; increment i
        shiftedKeys[i] = permutePC2(shiftedKeys[i])
    export shiftedKeys
```

2.2.2 - Switch function

```
String switchFunc(String left, String right)

    switch = right + left <-- concatenate into one string

    Export switch
```

2.2.3 - F function

```
String fFunction(String right, String key)

    String expansion = permuteE(right) <-- expand to 48bits

    String XORbits = startXOR(expansion, key) <-- XOR function

    sCount  = 1

    for(i = 6; I <= XORbits length; i = i + 6)

        String address = XORBits substring between i-6 and i

        Integer array sBox = get relevant sBox

        String sBits = permuteSBox(address, sBox)

        Increment sCount


    String fBits = permute(sBits)

    Export fBits
```

2.3 - Main difficulties

My program appears to encrypt and decrypt accurately. However, when I perform a:

```
diff testfile-DES.txt Decrypted.txt
```

The result indicates that the two files are different. It appears to be a new line instance at the end of the first line.

When I run the diff with the “-w” flag to ignore whitespaces, the result indicates there are no differences between the two files.

2.4 - Encryption and Decryption with key of all 0’s

My program was still able to encrypt and decrypt accurately.

3.0 - Additional Questions

3.1 - Describe what kind of possible threats you can overcome with DES and justify your reply

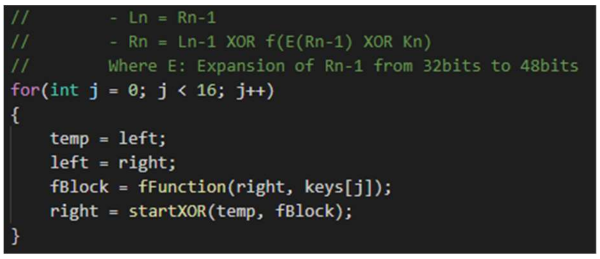
In terms of data transmission, the main threat is the interception of the data by malicious threat actors as it passes from sender to receiver. Such an event compromises the confidentiality and integrity of the data.

DES addresses this threat by using keys to encrypt the data before transmission. The data then, can only be decrypted by the receiver using the same key. This adds a layer of security in that, the key must be acquired in order to decrypt the data.

3.2 - Explain what you have done for source coding in Question 2 for DES

The plaintext is input in the algorithm from a text file line by line. Each line, from the source file, is then passed to the encryption function where it undergoes the following operations:

1. The line is padded with 0's, if necessary, to ensure the complete line can be broken up into blocks of 8 characters.
2. The line is then split up into equal blocks of 8 characters.
3. Each block is then converted into a 64bit binary String version of itself.
4. Each block is then permuted against the Initial Permutation table
5. Each block is then split into equal halves
6. Both halves then, go through 16 rounds of iterations:



```
//          - Ln = Rn-1
//          - Rn = Ln-1 XOR f(E(Rn-1) XOR Kn)
//          Where E: Expansion of Rn-1 from 32bits to 48bits
for(int j = 0; j < 16; j++)
{
    temp = left;
    left = right;
    fBlock = fFunction(right, keys[j]);
    right = startXOR(temp, fBlock);
}
```

7. After the final round of iterations, the two subsequent halves are then switched.
8. A final reverse permutation is then performed.
9. The resulting blocks are then converted back to text Strings, which becomes the encrypted text.

This process is repeated for each line of the source file.

3.3 - If you want to achieve the highest probability of error-correction in information transmission, tell us what you should do when you design a channel encoder.

You should implement Error Correction Code (ECC), by encoding with additional redundant information.

An example of redundant information is to transmit each data bit 3 times. Then, whichever bit is in majority is correct, and the other 2 bits are considered errors and discarded. See table below for illustrated example:

Triplet received	Interpreted as
000	0 (considered error free)
001	0
010	0
100	0
111	1 (considered error free)
110	1
101	1
011	1

4.0 - References

- Affine cipher. https://en.wikipedia.org/wiki/Affine_cipher. Accessed 18th April 2020.
- Error correction code. https://en.wikipedia.org/wiki/Error_correction_code. Accessed 18th April 2020.
- Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. “Handbook of Applied Cryptography” – Chapter 7: Block Ciphers. Accessed 18th April 2020.
- Dr Wanquan Liu. Fundamental Concepts of Cryptography – Lecture 2: “Principles of Information Security”. Accessed 18th April 2020.
- Dr Wanquan Liu. Fundamental Concepts of Cryptography – Lecture 3: “Coding”. Accessed 18th April 2020.
- Dr Wanquan Liu. Fundamental Concepts of Cryptography – Lecture 4: “Conventional Encryption Technique, Stream Ciphers and DES”. Accessed 18th April 2020.