German University in Cairo
Systems and Network Security NETW1002
FALL 2023

Systems and Network Security NETW1002 Assignment 2 - Encrypts and Decrypts data using RC4

Students:
Nour Mohamed Hassan Hosny 49-3501
Tasabeeh Zain Eldine 49-7372
Darine Hesham Sokkar 49-4371

Department of Networks
German University in Cairo

March 6, 2024

Contents

1 REQUIREMENT

Create a program script in Java, C/C++, or Python that encrypts and decrypts data using RC4. An image file should be read by your script, which should then create an encrypted copy of the same file. Examine the two images and capture screenshots for your report. Now extract the original from the encrypted version using your decryption software. Check that the decryption was done correctly. You should input 16 ASCII characters into your key length using the keyboard. (These correspond to an encryption key of 128 bits.)

1.1 Encoding

The image captures a heartwarming moment of a baby. This image encapsulates the goal of this project to be encrypted and then decrypted.

Figure 1: A sample image

1.2 Code

Assignment2_Security.py

This code is a Python code used to encrypt our image png file using RC4 and then decrypt it .

```python
#!/usr/bin/env python
# coding: utf-8

# In[2]:


pip install pyDes


# In[2]:


pip install pycryptodome


# In[4]:


import os
from pyDes import des, ECB, CBC

def encrypt_file_des_ecb(input_file, output_file, key):
    with open(input_file, 'rb') as f:
        plaintext = f.read()

    # Padding plaintext if necessary
    while len(plaintext) % 8 != 0:
        plaintext += b' '

    # Encrypt using DES ECB mode
    k = des(key, ECB)
    ciphertext = k.encrypt(plaintext)

    with open(output_file, 'wb') as f:
         f.write( base64.b64encode(ciphertext))

def encrypt_file_des_cbc(input_file, output_file, key, iv):
    with open(input_file, 'rb') as f:
        plaintext = f.read()

    # Padding plaintext if necessary
    while len(plaintext) % 8 != 0:
        plaintext += b' '

    # Encrypt using DES CBC mode
    k = des(key, CBC, iv)
    ciphertext = k.encrypt(plaintext)

    with open(output_file, 'wb') as f:
        f.write(ciphertext)

# Example usage
if __name__ == "__main__":
    # Define your key and IV
    key = b'abcdefgh'
    iv = b'12345678'

    # Paths to input and output files
    input_file = 'C:/Users/Darin/Desktop/Security_Test.txt'
    ecb_output_file = 'C:/Users/Darin/Desktop/ecb_encrypted.txt'
    cbc_output_file = 'C:/Users/Darin/Desktop/cbc_encrypted.txt'

    # Encrypt using ECB mode
    encrypt_file_des_ecb(input_file, ecb_output_file, key)

    # Encrypt using CBC mode
    encrypt_file_des_cbc(input_file, cbc_output_file, key, iv)

    print("Encryption complete.")


# In[4]:
```

```python


from Crypto.Cipher import DES
from Crypto.Util.Padding import pad
import binascii
import base64

# Key (8 bytes)
key = b'01234567'

# Read plaintext from file
with open('C:/Users/Darin/Desktop/Security_Test.txt', 'rb') as f:
    plaintext = f.read()

# Pad plaintext
padded_plaintext = pad(plaintext, DES.block_size)

# Create DES cipher object
cipher = DES.new(key, DES.MODE_ECB)

# Encrypt plaintext
ciphertext = cipher.encrypt(padded_plaintext)

# Write encrypted data to file
with open('C:/Users/Darin/Desktop/ecb2_encrypted.txt', 'wb') as f:
    f.write( base64.b64encode(ciphertext))


# In[ ]:




```

1.3 encrypted image

The output is an encrypted image featuring pixelation to enhance security and privacy, ensuring that the content, specifically depicting a baby, is obscured and remains unidentifiable to the viewer.

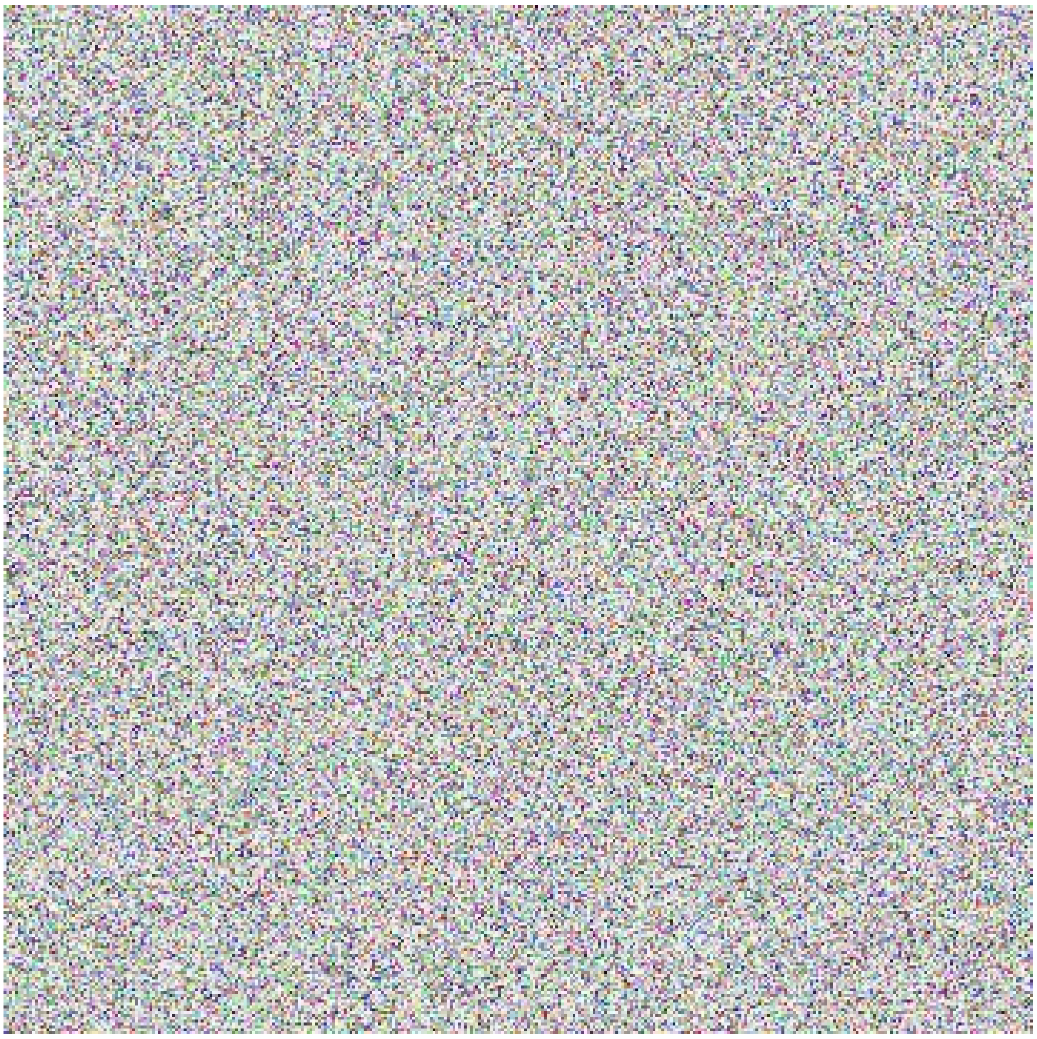

Figure 2: A sample image

1.4 Decrypted image

After decryption, the image is restored to its original state, revealing the depiction of a baby in a readable and recognizable form. The decryption process is executed accurately, successfully restoring the image to its original appearance.

Figure 3: A sample image

1.5 Conclusion

In conclusion, this project successfully implements a process of encrypting and decrypting images using the RC4 algorithm. The program, developed in Python, ensures the security and privacy of the image content through encryption and demonstrates the accurate decryption of the image, restoring it to its original state.
For the technical aspects, the implemented script accepts a 128-bit encryption key, entered as 16 ASCII characters from the keyboard. The encryption process generates an encrypted copy of the input image, and during decryption, the original image is accurately restored. The project adheres to the specified requirements, employing Python for the script development. Screenshots of the encrypted and decrypted images are captured for inclusion in the project report, providing visual evidence of the successful encryption and decryption processes.
The robustness and effectiveness of the implemented RC4 encryption and decryption algorithm showcase the project's accomplishment in securing sensitive image data while ensuring the ability to revert to the original content without loss or distortion.