SWE3002 Information and Systems Security

Winter 2020-2021

B2 slot

EPJ- FINAL REVIEW

SUBMITTED BY

18MIS0082-VAISHALI S

TOPIC- Image Encryption Using AES and DNA Algorithm

ABSTRACT:

Due to increase in use of images in various fields, it is very important to protect the confidential image data from unauthorized access and availability to strangers. The design uses the iterative approach with block size of 128 bit and key size of 256 bit. As secret key increases the security as well as complexity of the cryptography algorithms. Here algorithm in which the image is an input to AES Encryption to get the encrypted image and the encrypted image is the input to AES Decryption to get the original image. It has various advantages where it can be used in applications like Military communication, Forensics, Intelligent systems etc. In order to make it even more secure the power of DNA computing is used which will strengthen the existing security systems by opening up a new possibility of a hybrid cryptographic system. This makes it even more suitable in DNA computer maintaining same robustness and security. DNA Strands are long polymers of many numbers of connected nucleotides. These nucleotides contain of one of four nitrogen bases, a five-carbon sugar and a phosphate gathering. The nucleotides that make these polymers are named after the nitrogen base like ACGT Adenine, Cytosine, Guanine, and Thymine. Main aim of using this is because of its speed, less storage, minimal power requirements. The bits are converted into characters and encoding unit takes this input data and generates a triplet code which will include a combo of three bases of DNA into an image and later the reverse is done for decryption.

LITERATURE REVIEW:

[1] Simulation of Image Encryption using AES Algorithm- this paper explained about the aes algorithm and all the sub process (128 bit encoder)and the design steps- state array, key expansion, add round key and also experimental analysis and the image encryption. Xilinx ISE9.2i software is employed for synthesis, Timing simulation is performed to verify the functionality of the designed circuit.

[2] This book by Bruce Schneier explains about the protocols cryptographic techniques and special algorithms and also about the real world entities the mathematical theory behind the complexity and number theory is explained.

[3] With the quick progression of knowledge exchange in electronic approach, data security is turning into additional vital in knowledge storage and transmission. This paper presents the planning of a 128 bit encoder exploitation AES Rijndael algorithmic rule for image secret writing. The AES algorithmic rule outlined by the National Institute of ordinary and Technology (NIST) of us has been wide accepted. Optimized and Synthesizable VHDL code is developed for the implementation of 128- bit encoding and method. Xilinx ISE9.2i package is employed for synthesis. Timing simulation is performed to verify the practicality of the designed circuit.

[4] In this paper a compact FPGA design for the AES rule with 128-bitkey targeted for low-cost embedded applications is conferred. Encryption, coding and key schedule area unit all enforced victimization little resources of solely 222 Slices and three Block RAMs. This implementation simply fits in an exceedingly low-cost Xilinx Spartan II XC2S30 FPGA. This implementation will encipher and rewrite knowledge streams of one hundred fifty Mbps, that satisfies the wants of most embedded applications, as well as wireless communication. and a brand new method of implementing Mix Columns and Inverse Mix Columns transformations victimization shared logic resources is conferred.

[5] This paper presents an application of AES operations in image encryption and decryption. The encrypted cipher images displays uniformly distributed RGB pixels. This paper explains the basic aes algorithm of converting plain plain text into cipher text and strengths of the aes and about crypt analysis using aes which offers the flexibility of allowing different key sizes 128 bit, 192 bit and 256-bit key and therefore the security is predicated on the varied random key selections, different S-box and powerful transformations

[6] This paper is discussed about the algorithm which groups DNA cryptography, advanced encryption standard and diffie- Hellman key exchange for the purpose of fast processing and secured algorithm. MD_AES algorithm is developed by modifying AES and DNA cryptography. The encryption process is same as the AES algorithm but uses dynamic S-boxes to provide higher security from the attacks like differential and linear crypt analysis attacks. The discussed algorithm can be applied for swapping data securely.

[7] propose a double section coding and decryption algorithms that's supported shuffling the image pixels victimization affine remodel and that they encrypting the ensuing image victimization XOR operation. They redistribute the element values to totally different location victimization affine remodel technique with four 8-bit keys. The transformed image is then divided into a pair of pixels 'x' pixels blocks and every block is then encrypted with XOR operation by four 8-bit keys. the whole key size employed in algorithm is 64 bit. Their results well-tried that when the affine remodel the correlation between element values was considerably attenuated

[8] This paper presents the look of a128 bit encoder exploitation AES Rijndael rule for image encryption. The AES rule outlined by the National Institute of ordinary and Technology(NIST) of us has been wide accepted. Optimized and Synthesizable VHDL code is developed for the implementation of 128- bit information encryption and method. Xilinx ISE9.2i computer code is employed for synthesis. Timing simulation is performed to verify the practicality of the designed circuit.

[9] The paper describes on how it chooses to use DNA encoding with its merits and demerits. Modern cryptography image encryption techniques are considered to create the entire 2D image data as a 1D textual bitstream and then to apply any conventional cipher that has been validated in modern cryptography like, DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm), and AES (Advanced Encryption Standard) to encrypt. The main idea of the transform domain-based image encryption is that the digital image is operated in the transform domain. When using DNA encoding, process is complicated, and biological operation error is bigger, and the experimental cost is expensive. It also explains about various DNA sequence operation like XOR and multiplication.

[10] In this paper they proposed a unique algorithm for image secret writing supported SHA-512 hash operate. The algorithmic rule consists of 2 main sections: the primary will pre processing operation to shuffle one half image. The second uses hash function to come up with a random range mask. The mask is then XORed with the opposite a part of the image that is going to be encrypted.

[11] The main purpose of this criterion is to investigate the prevailing S-boxes and study their strengths and weaknesses so as to work out their suitableness in image encoding applications. The projected criterion uses the results from correlation analysis, entropy analysis, distinction analysis, homogeneity analysis, energy analysis, and mean of absolute deviation

analysis. These analyses area unit applied to advanced encoding commonplace (AES), affine-power-affine (APA), gray Lui J, residue prime, S8 AES, SKIPJACK, and Xyi Sboxes.

[12] In this paper they explained the projected image encoding theme, associate external secret key of 104 bit and 2 chaotic logistical maps square measure used to confuse the link between the cipher image and therefore the plain image. Further, to create the cipher a lot of sturdy against any attack, the key key's changed when encrypting of every component of the plain image.
The hardness of the projected system is additional bolstered by a feedback mechanism, that makes the encoding of every plain component depend on the key.

[13] This paper analysis the Advance Encryption Standard (AES) algorithm and presents a modification to the Advanced Encryption Standard (MAES) with a high-level security and image encryption. The result is so that after modification image security is high. It also compares their algorithm with original AES encryption algorithm

[14] It has proposed a cryptography and decryption algorithms that's supported shuffling the image pixels exploitation affine remodel and that they encrypting the ensuing image exploitation XOR operation. They redistribute the pel values to totally different location exploitation affine remodel technique with four 8-bit keys. The transformed image then divided into two pixels x two pixels blocks and every block is encrypted exploitation XOR operation by four 8-bit keys. the entire key size employed in algorithm is sixty four-- bit.

[15] In this paper it focuses on DNA computing where it is utilized as a data-conveyor. DNA cryptography is an upcoming innovation, in which it combines bio-logical data and information security. De-oxy Ribo nucleic corrosive atom has four bases, Adenine (A), Thiamine (T), Guanine (G) and Cytosine (C). Fundamental phrases used to comprehend DNA are Codons, Genes, Chromosome, Genome. It mainly helps in Hamiltonian path and NP-complete problems. Hence when this is used it helps to have the plain text or image even more secure. It also brings a mixer of substitution, permutation and diffusion.

[16] The paper discusses about how AES and DNA are implemented in one another. The main idea is on converting each step in AES to its complementary DNA form instead of binary basis including inputs, outputs, the operations and keys. The DNA- based AES have the same strength properties and high security as by AES. It is also showed how the complicated binary based operations can be implemented on DNA basis.

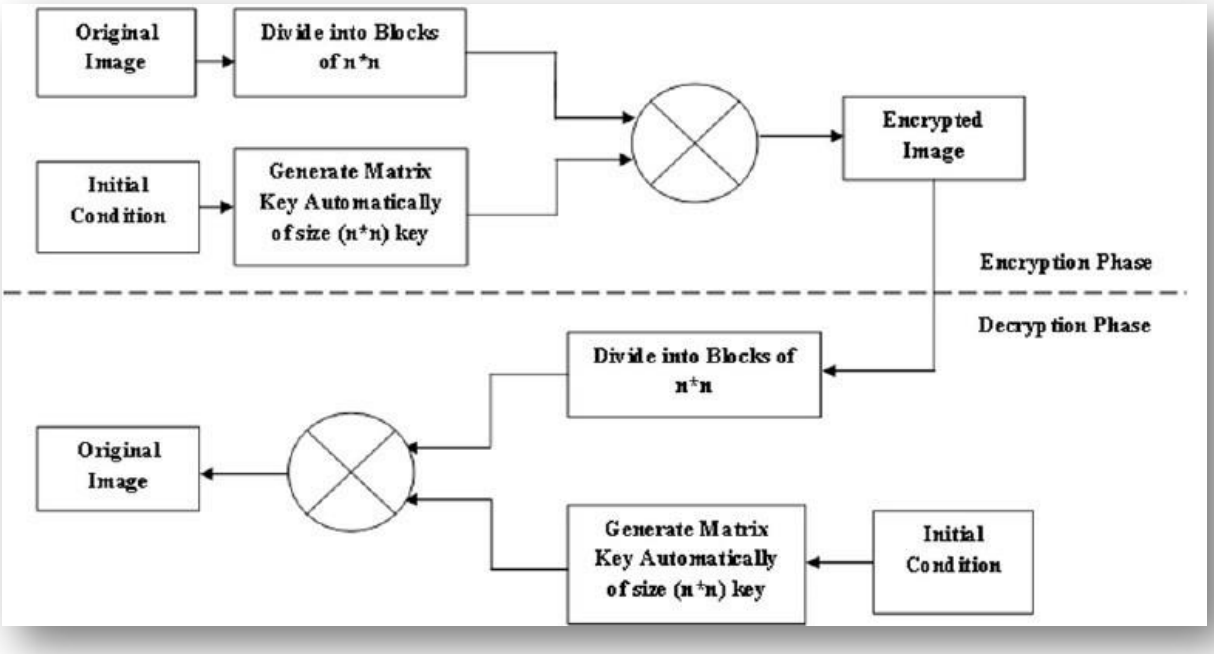
[17] DNA cryptography uses the huge parallelism and storage capacity of DNA molecules with the traditional methods of cryptography. At present, Microsoft is taking initiative to commercialize DNA computers in near future. In upcoming ten to thirty years the virtually non-hackable DNA cryptography techniques will be in the position as effective alternative to classical cryptosystem. Modern biological science is becoming more digitalized which is beneficial in Cyber-biosecurity. Storage of genome information in electronic database is the keystone of modern digitized biotechnology.

[18] The paper describes on how DNA based cryptography benefits from techniques such as One Time Pad (OTP), DNA fragmentation and DNA amplification through Polymerase Chain Reaction (PCR). Histograms and variances are shown for the plain image and encrypted image proving low variance gives high uniformity. Methods like, plain image can be encrypted by fragmenting it into non-overlapping blocks for adding watermarks that are followed by DNA addition and complementation using a Logistic Map. Also, various security analysis is done to find the efficiency.

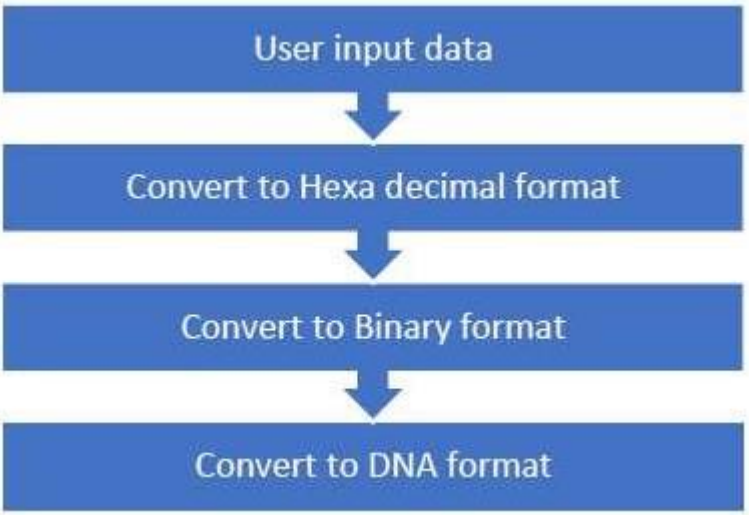
Reference no	Author/ year	Title	Methods ,algorithms and S/W used
1	Karthigai kumar, P., Rasheed, S. (2011)	Simulation of image encryption using AES algorithm	AES, Xilinx ISE9.2i software
2	Zeghid, M, Machhout, M., Khriji, Baganne, A., & Tourki, R. (2007)	A modified AES based algorithm for image encryption	advanced usage of AES encryption
3	Zhang, X., & Parhi, K. K. (2004)	High-speed VLSI architectures for the AES algorithm.	128 bit encoder exploitation AES Rijndael algorithmic
4	Gaj, K., & Chodowiec, P. (2009).	FPGA and ASIC implementations of AES	FPGA design for the AES rule with 128-bitkey, Spartan II FPGAs sanctionative compact logic
5	Radhadevi, P., & Kalpana, P. (2012)	Secure image encryption using AES.	AES with 128 bit, 192 bit and 256-bit key
6	Bhavani, Y., Puppala, S. S., Krishna, B. J., & Madarapu, S. (2019, December).	Modified AES using Dynamic S-Box and DNA Cryptography.	DNA ,MAES
7	Nag, A., Singh, J. P., Khan, S., Ghosh, S., Biswas, S., Sarkar, D., & Sarkar, P. P. (2011, July).	Image encryption using affine transform and XOR operation	encrypting the ensuing image victimization XOR operation
8	Sklavos, N., & Koufopavlou, O. (2002).	Architectures and VLSI implementations of the AES-proposal Rijndael	AES Rijndael rule , Xilinx ISE9.2i
9	Zhou, S., Wang, B., Zheng, X., & Zhou, C. (2016).	An image encryption scheme based on DNA computing and cellular automata.	DNA and AES
10	Norouzi, B., Seyedzadeh, S. M., Mirzakuchaki, S., & Mosavi, M. R. (2014).	A novel image encryption based on hash function with only two-round diffusion process.	SHA-512 hash operation
11	Shah, T., Hussain, I., Gondal, M. A., & Mahmood, H. (2011).	Statistical analysis of S-box in image encryption applications based on majority logic criterion	AES, affine-power-affine (APA), gray Lui J, residue prime, S8 AES, SKIPJACK, and Xyi Sboxes.
12	Ismail, I. A., Amin, M., & Diab, H. (2010).	A digital image encryption algorithm based a composition of two chaotic logistic maps	AES,external secret key of 104 bit and 2 chaotic logistical maps

13	indrakanti, S. P., & Avadhani, P. S. (2011)..	Permutation based image encryption technique	AES and MAES
14	Enayatifar, R., & Abdullah, A. H. (2011).	Image security via genetic algorithm	affine remodel technique and also XOR operations
15	Bhoi, G., Bhavsar, R., Prajapati, P., & Shah, P. (2020, December).	A Review of Recent Trends on DNA Based Cryptography	DNA
16	Sabry, M., Hashem, M., Nazmy, T., & Khalifa, M. E. (2015)	Design of DNA-based advanced encryption standard (AES)	DNA and AES
17	Mondal, M., & Ray, K. S. (2019).	Review on DNA cryptography	DNA
18	Samiullah, M., Aslam, W., Nazir, H., Lali, M. I., Shahzad, B., Mufti, M. R., & Afzal, H. (2020).	An image encryption scheme based on DNA computing and multiple chaotic systems	DNA AND LOGISTIC MAPS

Architectural block diagram:



Coding DNA nucleotide	Decimal	Binary
A	0	00
C	1	01
G	2	10
T	3	11



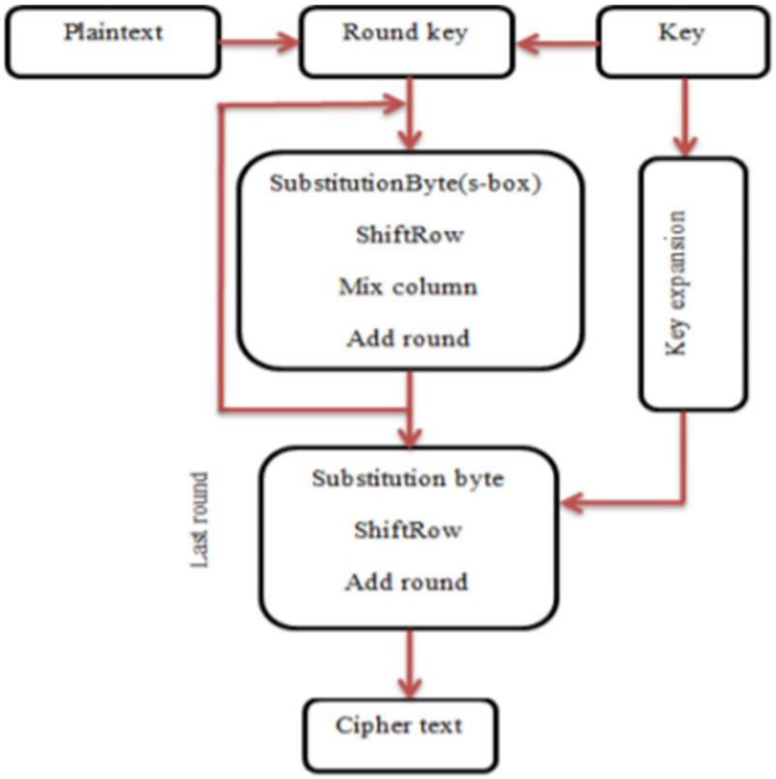
Converting S-Box into hex format and to binary format an then to DNA format.

$$\text{S-Box(hex form)}= \begin{matrix} 3 & E & 7 & 0 \\ 9 & 8 & 1 & 2 \\ C & 4 & 5 & D \\ 6 & F & B & A \end{matrix}$$

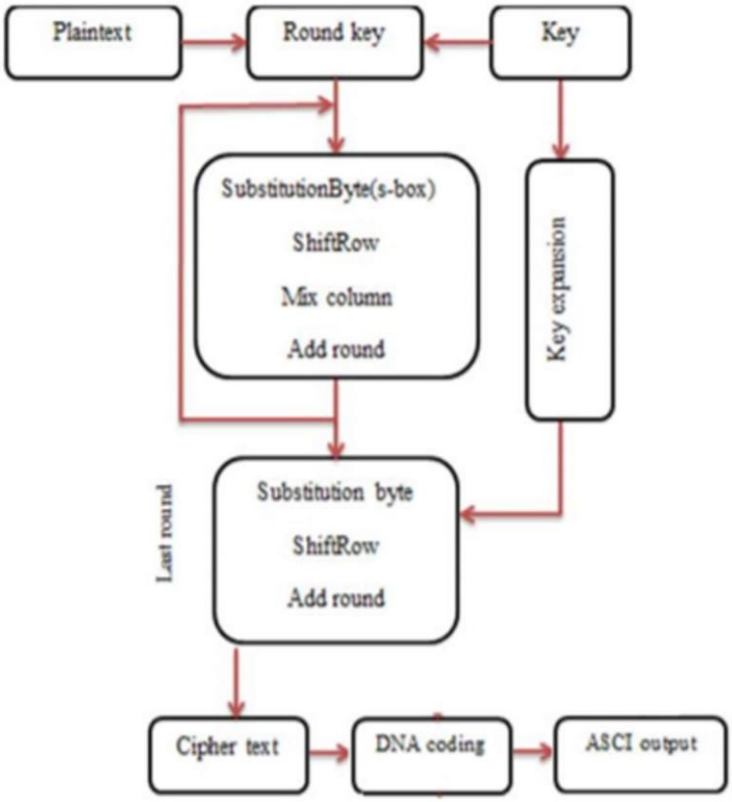
$$\text{S-Box(binary format)} = \begin{matrix} 0011 & 1110 & 0111 & 0000 \\ 1001 & 1000 & 0001 & 0010 \\ 1100 & 0100 & 0101 & 1101 \\ 0110 & 1111 & 1011 & 1010 \end{matrix}$$

$$\text{S-Box(DNA format)} = \begin{matrix} AT & TG & CT & AA \\ GC & GA & AC & AG \\ TA & CA & CC & TC \\ CG & TT & GT & GG \end{matrix}$$

BLOCK DIAGRAM OF AES ALGORITHIM:



PROPOSED ALGORITHIM WITH DNA CRYPTOGRAPHY:



Advantages

•The image can only be viewed by the receiver as the image is encrypted using AES and the key is only known to the sender and receiver.

•Since the image is encrypted using AES, it is more secure than the DES and triple DES.

•Since the key size is 192 bits, it makes the encryption and decryption more secure.

Disadvantages

•The file size to be transmitted becomes large since it contains encrypted data.

•Since the file size is huge it can be suspected to contain some critical information.

Analysis of DNA over AES:

DNA cryptography is a new promising direction in cryptography research that emerged with the progress in DNA computing field. Traditional cryptographic systems have long legacy and are built on a strong mathematical and theoretical basis. So, an important perception needs to be developed that the DNA cryptography is not to negate the tradition, but to create a bridge between existing and new technology. The power of DNA computing will strengthen the existing security systems by opening up a new possibility of a hybrid cryptographic system. In our work, we are presenting the DNA-based design and implementation to “Advanced Encryption Standard” [AES]

The DNA-based cryptography technique is a new paradigm in the cryptography field that is used to protect data during transmission. In this paper we introduce the asymmetric DNA cryptography technique for encrypting and decrypting plain-texts. This technique is based on the concept of data dependency, dynamic encoding and asymmetric cryptosystem. The asymmetric cryptosystem is used solely to initiate the encryption and decryption processes that are completely conducted using DNA computing.

Conclusion

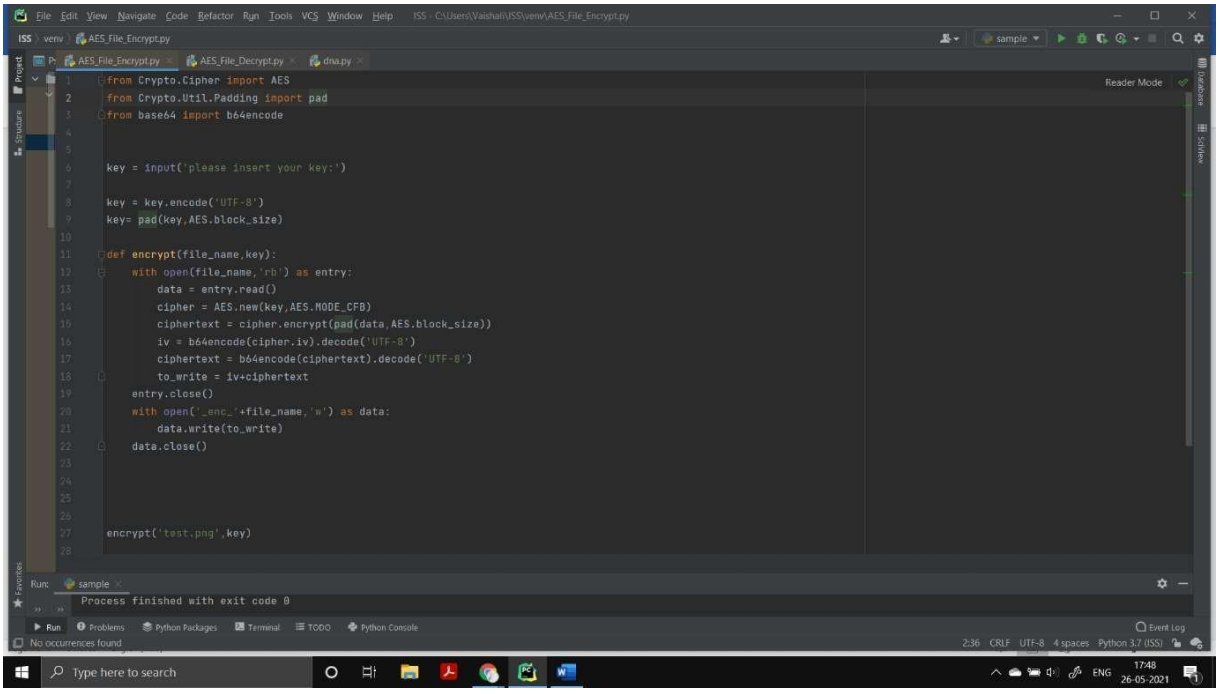
Since image steganography is done using AES, this system provides security from intrusion attacks and the usage of AES technique allows the encryption and decryption process to be more secure and faster. Thus, this system provides security in storage and transmission of digital images

CODES:

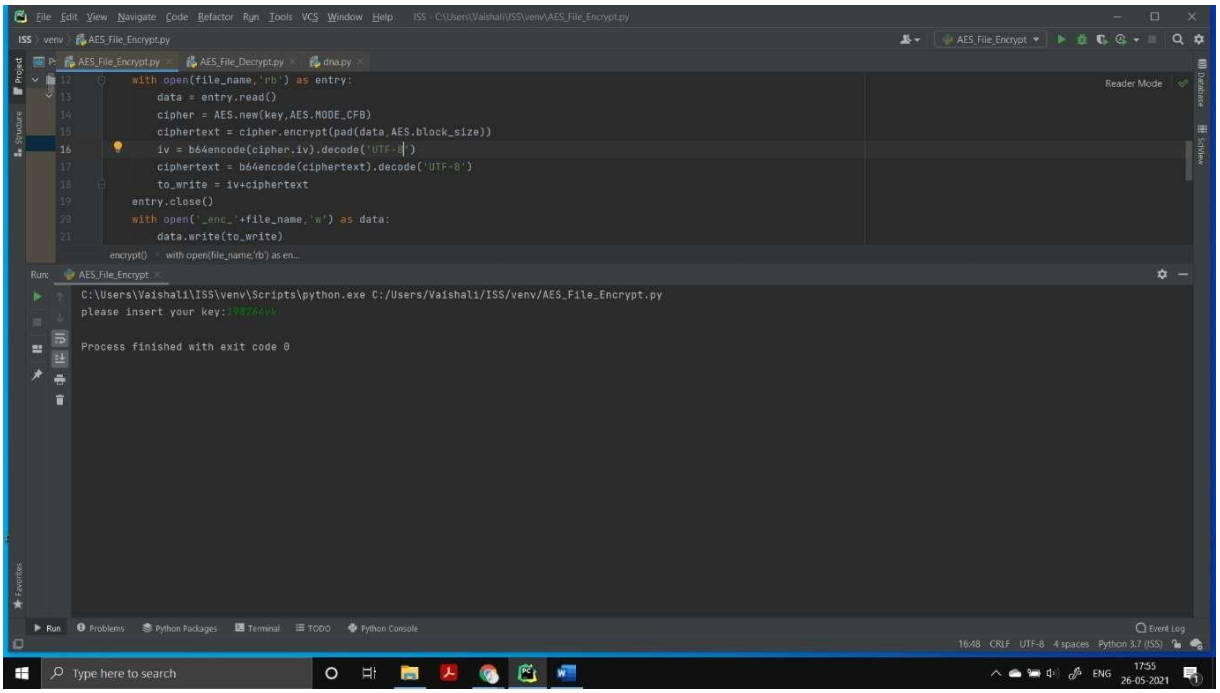
Original image – 8KB

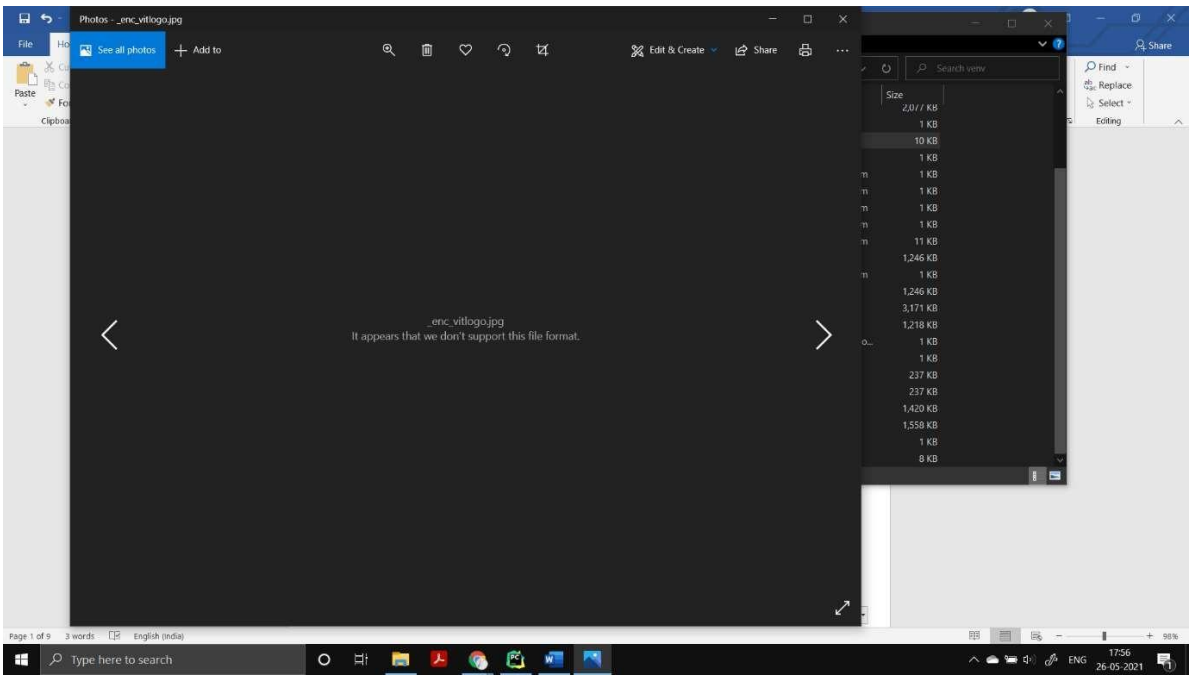
Encrypted - 10KB

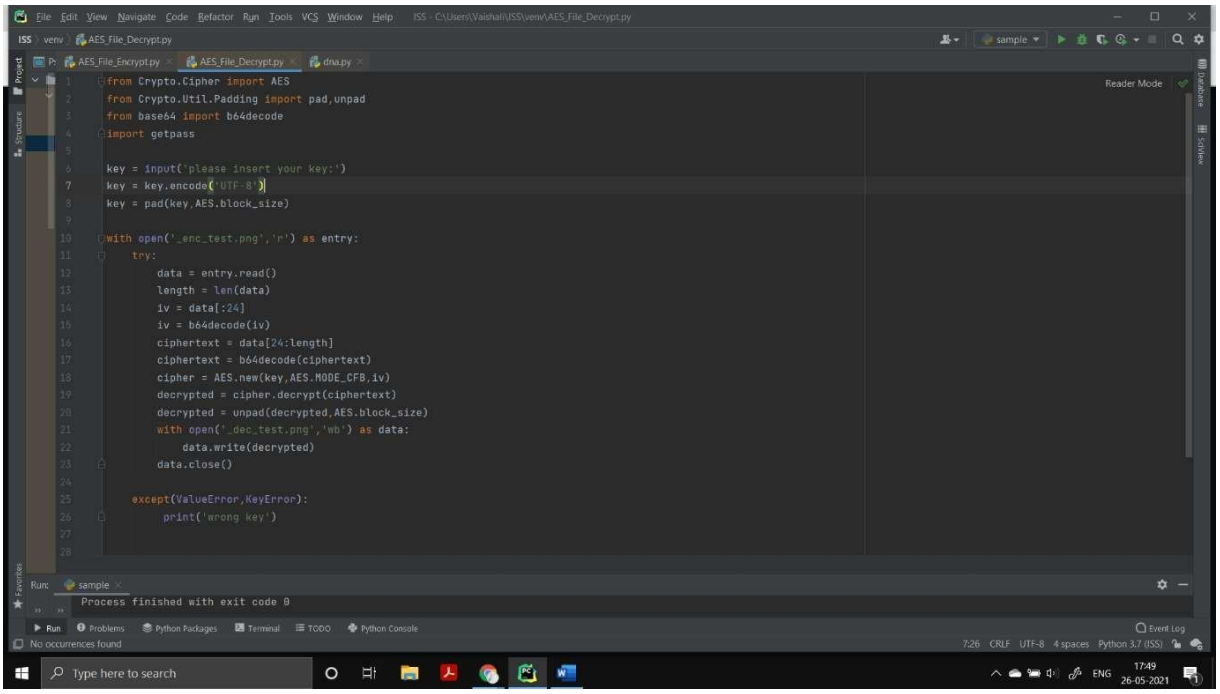
Decrypted - 8KB



Output


File Edit View Navigate Code Refactor Run Tools VCS Window Help
AES_File_Encrypt.py
AES_File_Decrypt.py
dna.py
Reader Mode
with open(file_name,'rb') as entry:
data = entry.read()
cipher = AES.new(key,AES.MODE_CFB)
ciphertext = cipher.encrypt(pad(data,AES.block_size))
iv = b64encode(cipher.iv).decode('UTF-8')
ciphertext = b64encode(ciphertext),decode('UTF-8')
to_write = iv+ciphertext
entry.close()
with open('_enc_'+file_name,'w') as data:
data.write(to_write)
Run: AES_File_Encrypt
C:\Users\Vaishali\ISS\venv\Scripts\python.exe C:/Users/Vaishali/ISS/venv/AES_File_Encrypt.py
please insert your key:198764vk
Process finished with exit code 0
Run
Problems
Python Packages
Terminal
TODO
Python Console
Event Log
16:48 CRLF UTF-8 4 spaces Python 3.7 (ISS)
Type here to search
ENG
17:55
26-05-2021

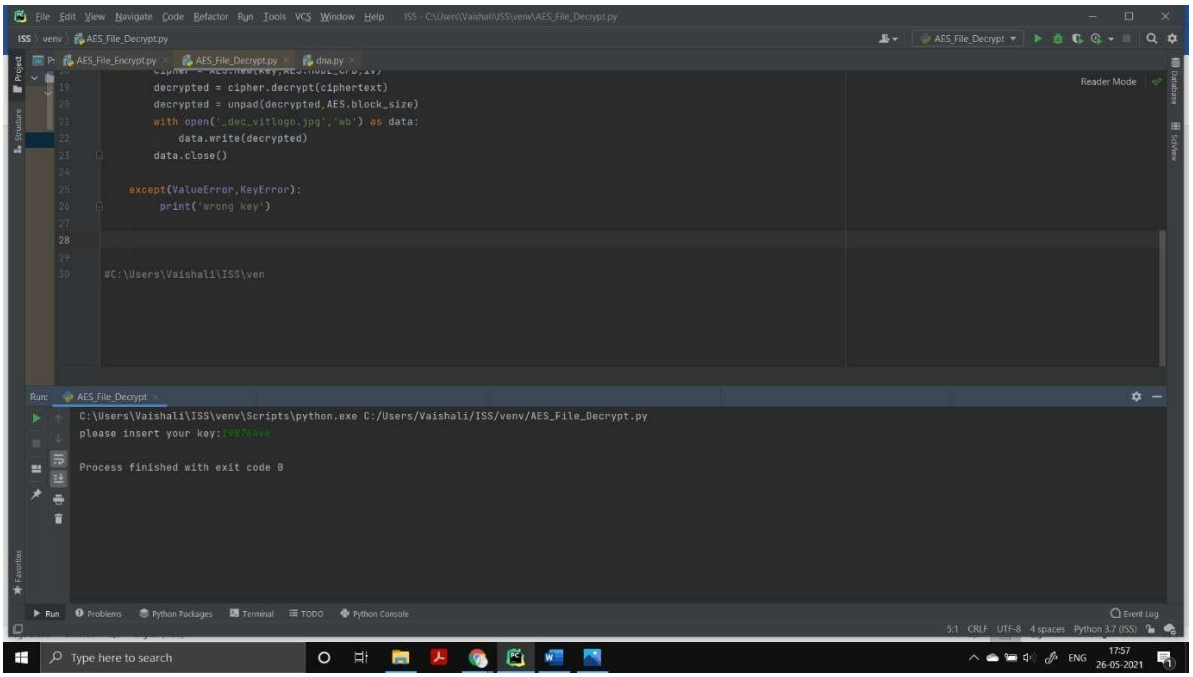

Photos - _enc_vitlogo.jpg
See all photos
Add to
Edit & Create
Share
_enc_vitlogo.jpg
It appears that we don't support this file format.
Paste
Find
Replace
Select
Editing
Size
Page 1 of 9
3 words
English (India)
Type here to search
ENG
17:56
26-05-2021



```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad,unpad
from base64 import b64decode
import getpass

key = input('please insert your key:')
key = key.encode('UTF-8')
key = pad(key,AES.block_size)

with open('_enc_test.png','r') as entry:
    try:
        data = entry.read()
        length = len(data)
        iv = data[:24]
        iv = b64decode(iv)
        ciphertext = data[24:length]
        ciphertext = b64decode(ciphertext)
        cipher = AES.new(key,AES.MODE_CFB,iv)
        decrypted = cipher.decrypt(ciphertext)
        decrypted = unpad(decrypted,AES.block_size)
        with open('_dec_test.png','wb') as data:
            data.write(decrypted)
        data.close()

    except(ValueError,KeyError):
        print('wrong key')
```

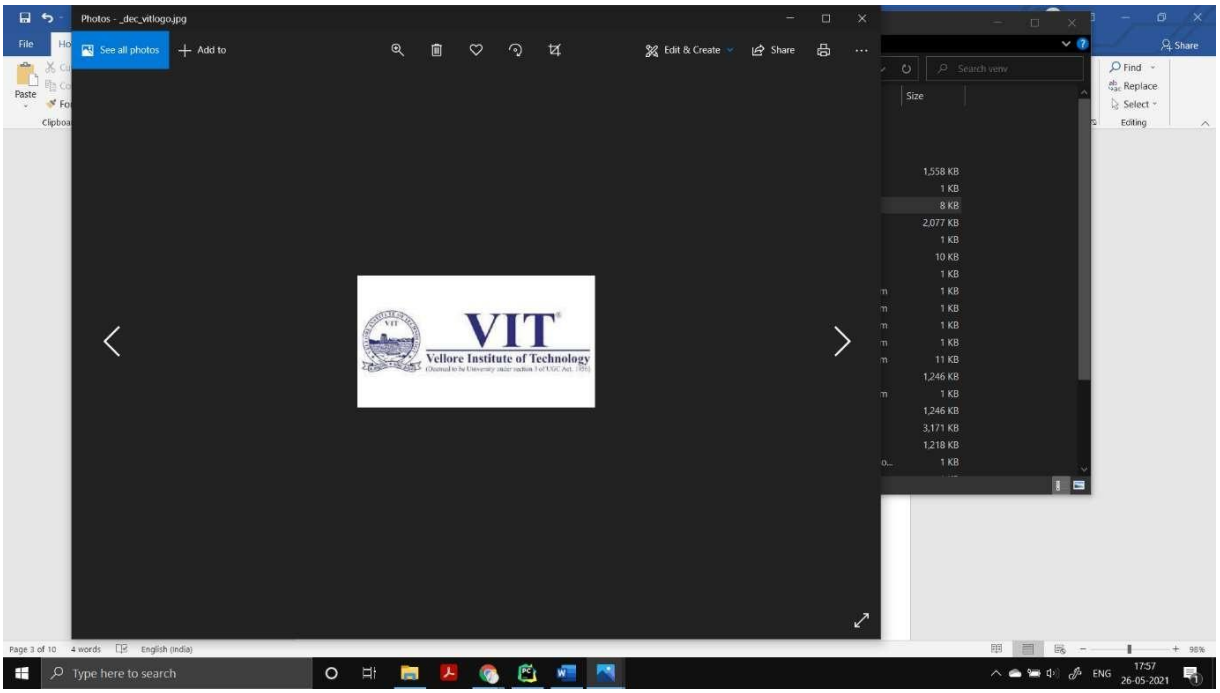
```
Process finished with exit code 0
```

output



```
        decrypted = cipher.decrypt(ciphertext)
        decrypted = unpad(decrypted,AES.block_size)
        with open('_dec_vitlogo.jpg','wb') as data:
            data.write(decrypted)
        data.close()

    except(ValueError,KeyError):
        print('wrong key')


#C:\Users\Vaishali\ISS\ven
```

```
C:\Users\Vaishali\ISS\venv\Scripts\python.exe C:/Users/Vaishali/ISS/venv/AES_File_Decrypt.py
please insert your key:198764vk

Process finished with exit code 0
```

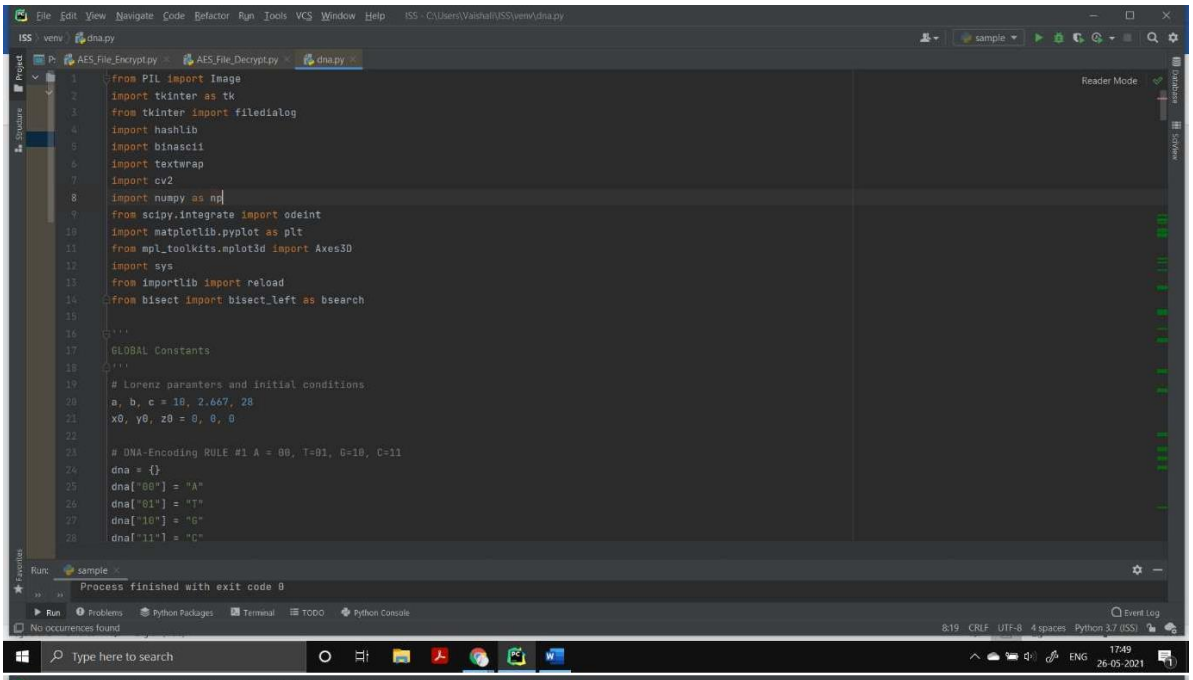


DNA

Original image – 8KB

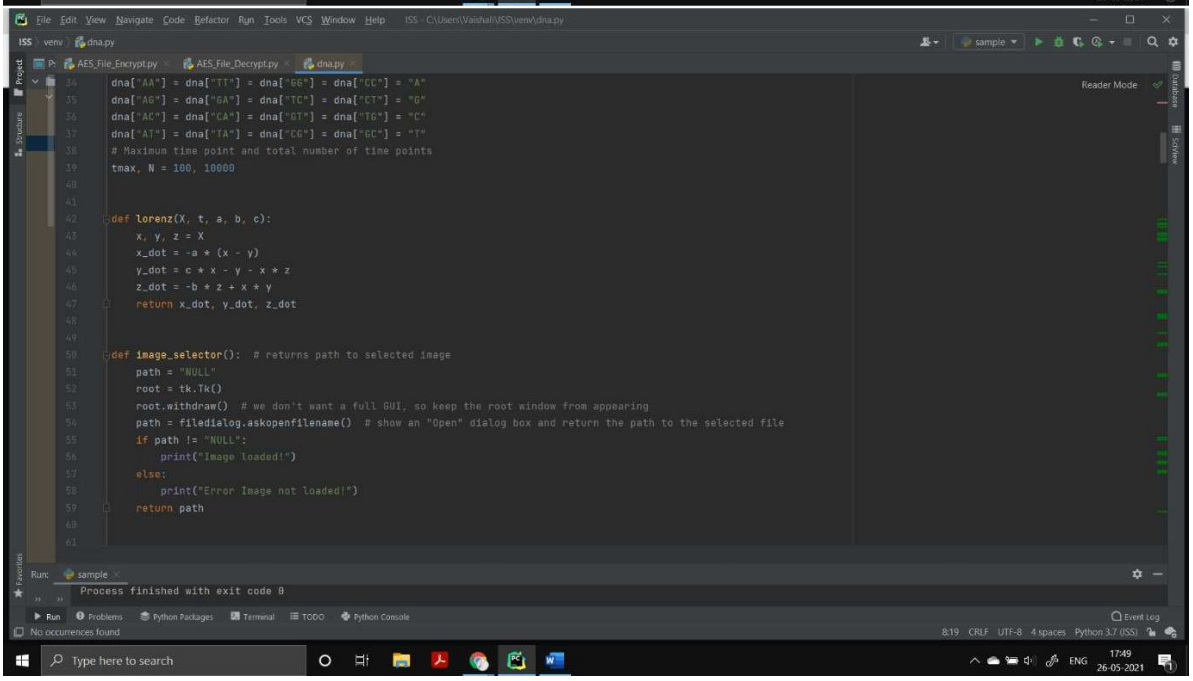
Encrypt – 60KB

Decrypt – 15KB



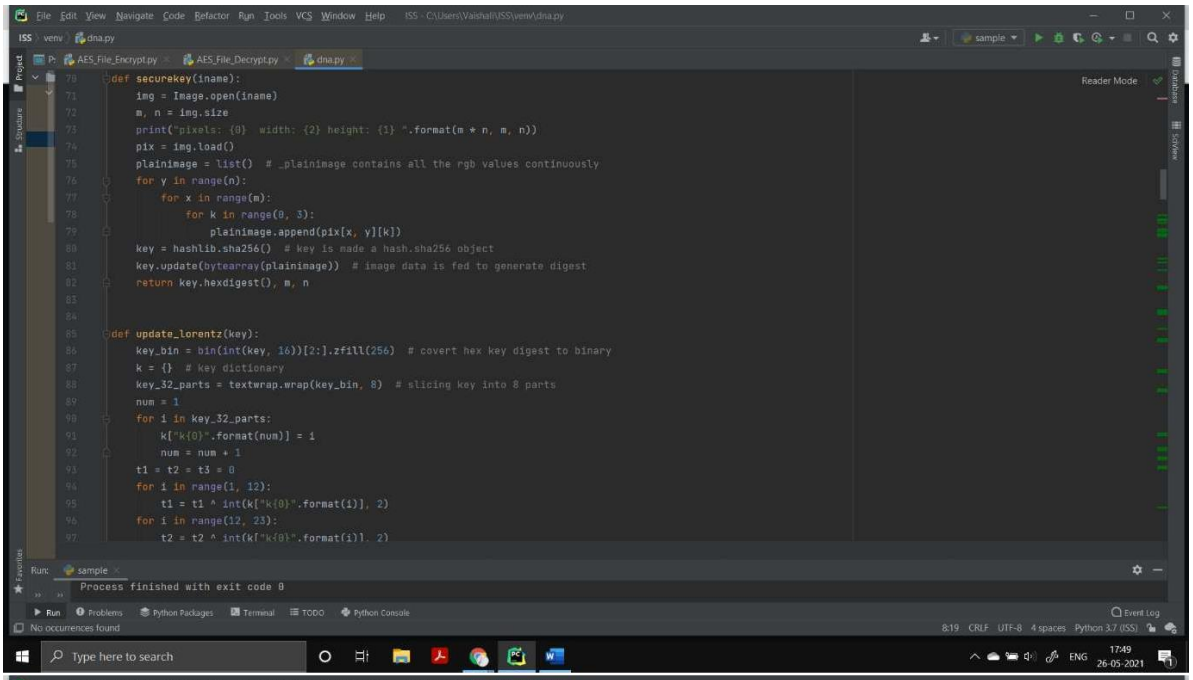
```
from PIL import Image
import tkinter as tk
from tkinter import filedialog
import hashlib
import binascii
import textwrap
import cv2
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import sys
from importlib import reload
from bisect import bisect_left as bsearch

'''
GLOBAL Constants
'''
# Lorenz paramters and initial conditions
a, b, c = 10, 2.667, 28
x0, y0, z0 = 0, 0, 0

# DNA-Encoding RULE #1 A = 00, T=01, G=10, C=11
dna = {}
dna["00"] = "A"
dna["01"] = "T"
dna["10"] = "G"
dna["11"] = "C"
```

```
Process finished with exit code 0
```



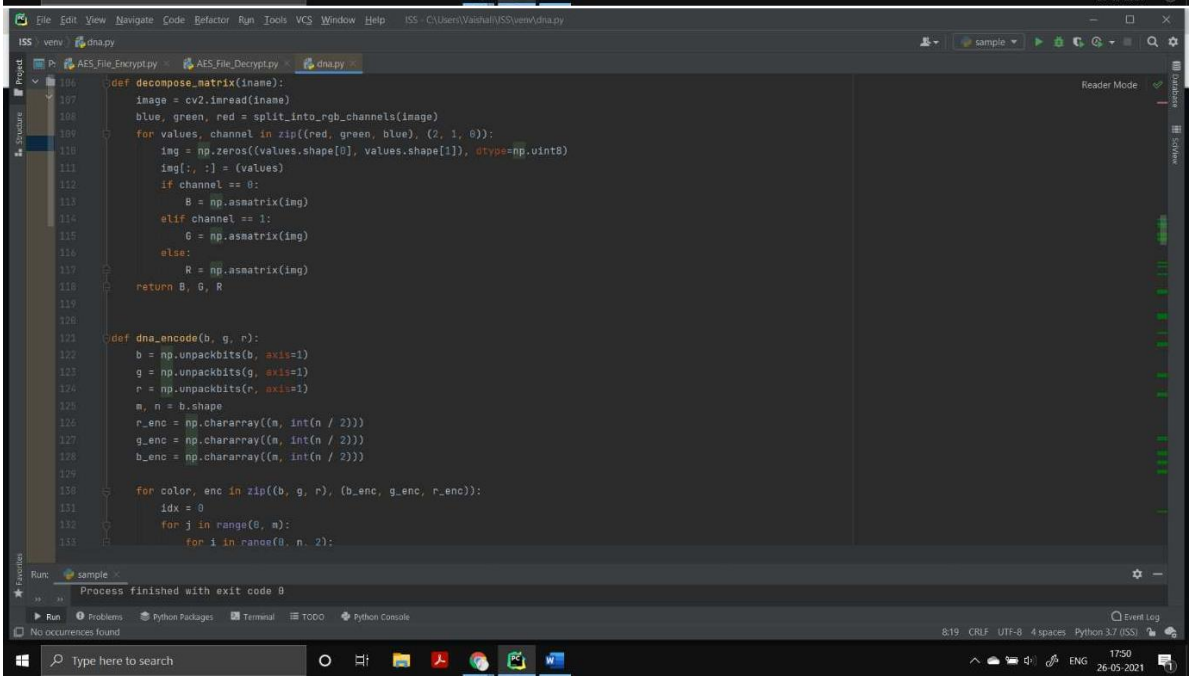
```
dna["AA"] = dna["TT"] = dna["GG"] = dna["CC"] = "A"
dna["A6"] = dna["GA"] = dna["TC"] = dna["CT"] = "G"
dna["AC"] = dna["CA"] = dna["GT"] = dna["TG"] = "C"
dna["AT"] = dna["TA"] = dna["CG"] = dna["GC"] = "T"
# Maximum time point and total number of time points
tmax, N = 100, 10000


def lorenz(X, t, a, b, c):
    x, y, z = X
    x_dot = -a * (x - y)
    y_dot = c * x - y - x * z
    z_dot = -b * z + x * y
    return x_dot, y_dot, z_dot


def image_selector():  # returns path to selected image
    path = "NULL"
    root = tk.Tk()
    root.withdraw()  # we don't want a full GUI, so keep the root window from appearing
    path = filedialog.askopenfilename()  # show an "Open" dialog box and return the path to the selected file
    if path != "NULL":
        print("Image loaded!")
    else:
        print("Error Image not loaded!")
    return path
```

```
Process finished with exit code 0
```



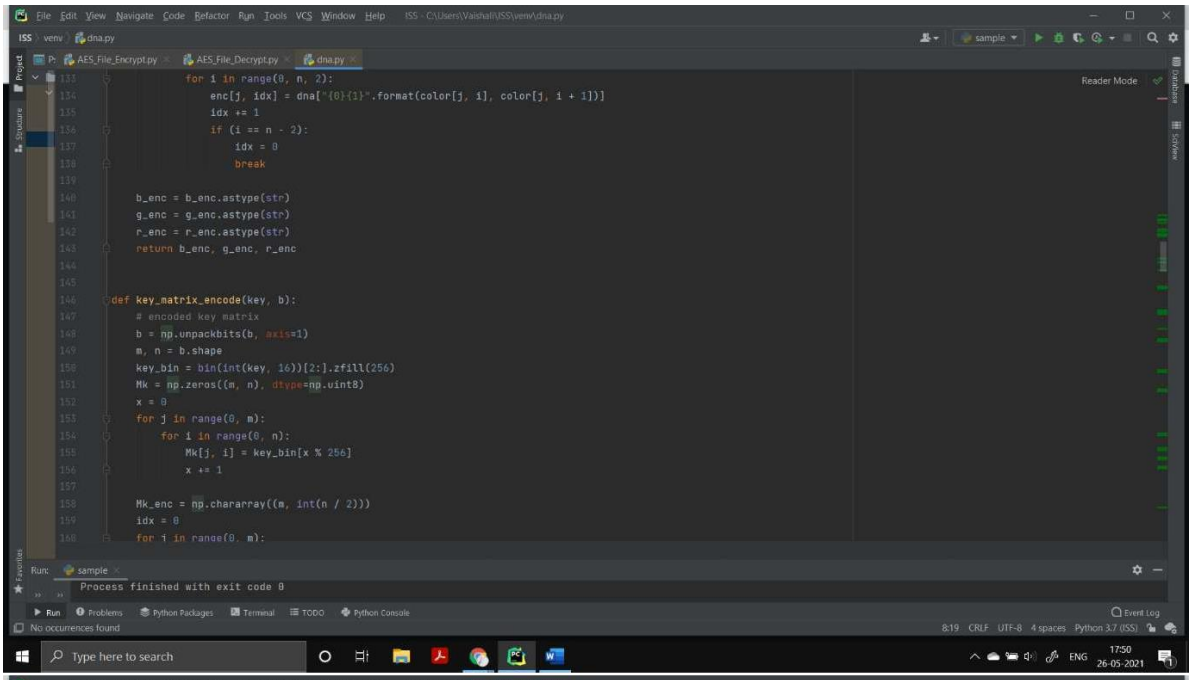
```
def securekey(iname):
    img = Image.open(iname)
    m, n = img.size
    print("pixels: {0}  width: {2} height: {1} ".format(m * n, m, n))
    pix = img.load()
    plainimage = list()  # _plainimage contains all the rgb values continuously
    for y in range(n):
        for x in range(m):
            for k in range(0, 3):
                plainimage.append(pix[x, y][k])
    key = hashlib.sha256()  # key is made a hash.sha256 object
    key.update(bytearray(plainimage))  # image data is fed to generate digest
    return key.hexdigest(), m, n


def update_lorentz(key):
    key_bin = bin(int(key, 16))[2:].zfill(256)  # covert hex key digest to binary
    k = {}  # key dictionary
    key_32_parts = textwrap.wrap(key_bin, 8)  # slicing key into 8 parts
    num = 1
    for i in key_32_parts:
        k["k{0}".format(num)] = i
        num = num + 1
    t1 = t2 = t3 = 0
    for i in range(1, 12):
        t1 = t1 ^ int(k["k{0}".format(i)], 2)
    for i in range(12, 23):
        t2 = t2 ^ int(k["k{0}".format(i)], 2)
```

Process finished with exit code 0



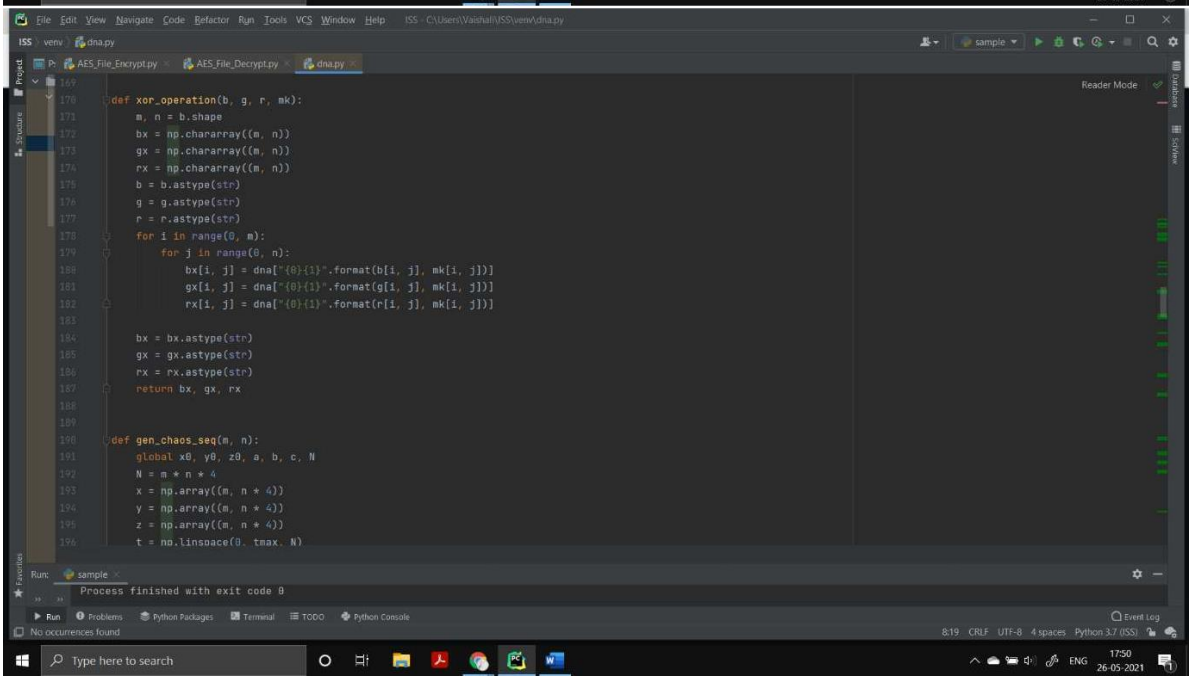
```
def decompose_matrix(iname):
    image = cv2.imread(iname)
    blue, green, red = split_into_rgb_channels(image)
    for values, channel in zip((red, green, blue), (2, 1, 0)):
        img = np.zeros((values.shape[0], values.shape[1]), dtype=np.uint8)
        img[:, :] = (values)
        if channel == 0:
            B = np.asmatrix(img)
        elif channel == 1:
            G = np.asmatrix(img)
        else:
            R = np.asmatrix(img)
    return B, G, R


def dna_encode(b, g, r):
    b = np.unpackbits(b, axis=1)
    g = np.unpackbits(g, axis=1)
    r = np.unpackbits(r, axis=1)
    m, n = b.shape
    r_enc = np.chararray((m, int(n / 2)))
    g_enc = np.chararray((m, int(n / 2)))
    b_enc = np.chararray((m, int(n / 2)))

    for color, enc in zip((b, g, r), (b_enc, g_enc, r_enc)):
        idx = 0
        for j in range(0, m):
            for i in range(0, n, 2):
```

Process finished with exit code 0



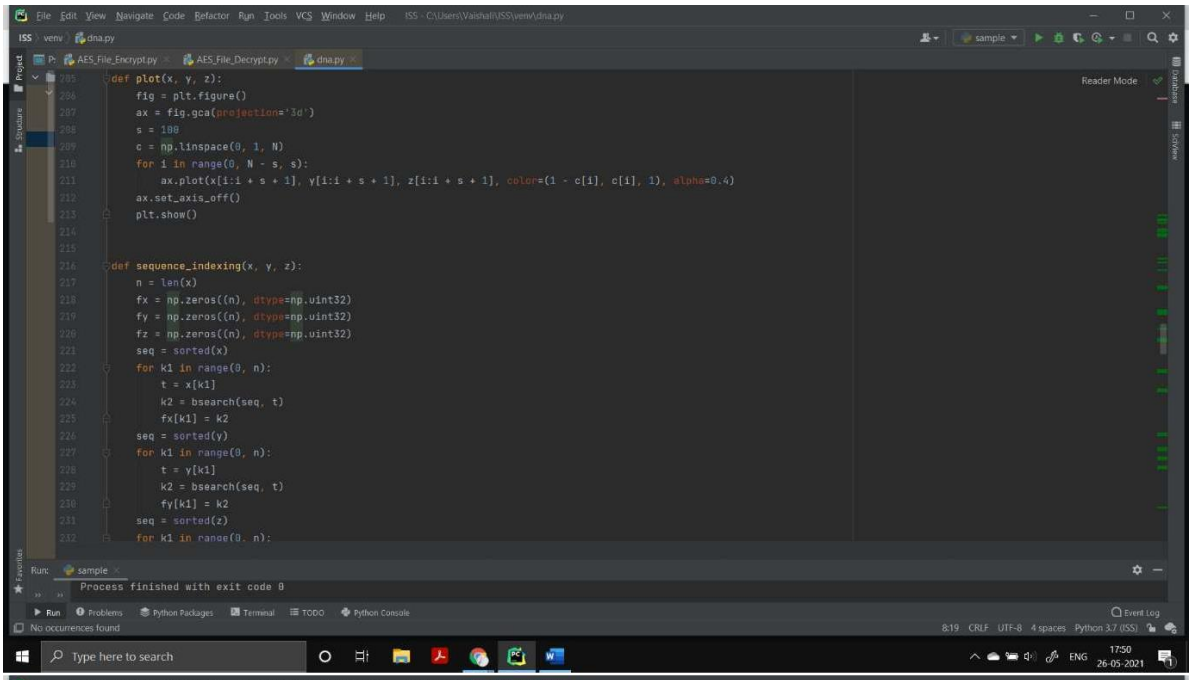
```
                for i in range(0, n, 2):
                    enc[j, idx] = dna["{0}{1}".format(color[j, i], color[j, i + 1])]
                    idx += 1
                    if (i == n - 2):
                        idx = 0
                        break

        b_enc = b_enc.astype(str)
        g_enc = g_enc.astype(str)
        r_enc = r_enc.astype(str)
        return b_enc, g_enc, r_enc


    def key_matrix_encode(key, b):
        # encoded key matrix
        b = np.unpackbits(b, axis=1)
        m, n = b.shape
        key_bin = bin(int(key, 16))[2:].zfill(256)
        Mk = np.zeros((m, n), dtype=np.uint8)
        x = 0
        for j in range(0, m):
            for i in range(0, n):
                Mk[j, i] = key_bin[x % 256]
                x += 1

        Mk_enc = np.chararray((m, int(n / 2)))
        idx = 0
        for i in range(0, m):
```

Process finished with exit code 0



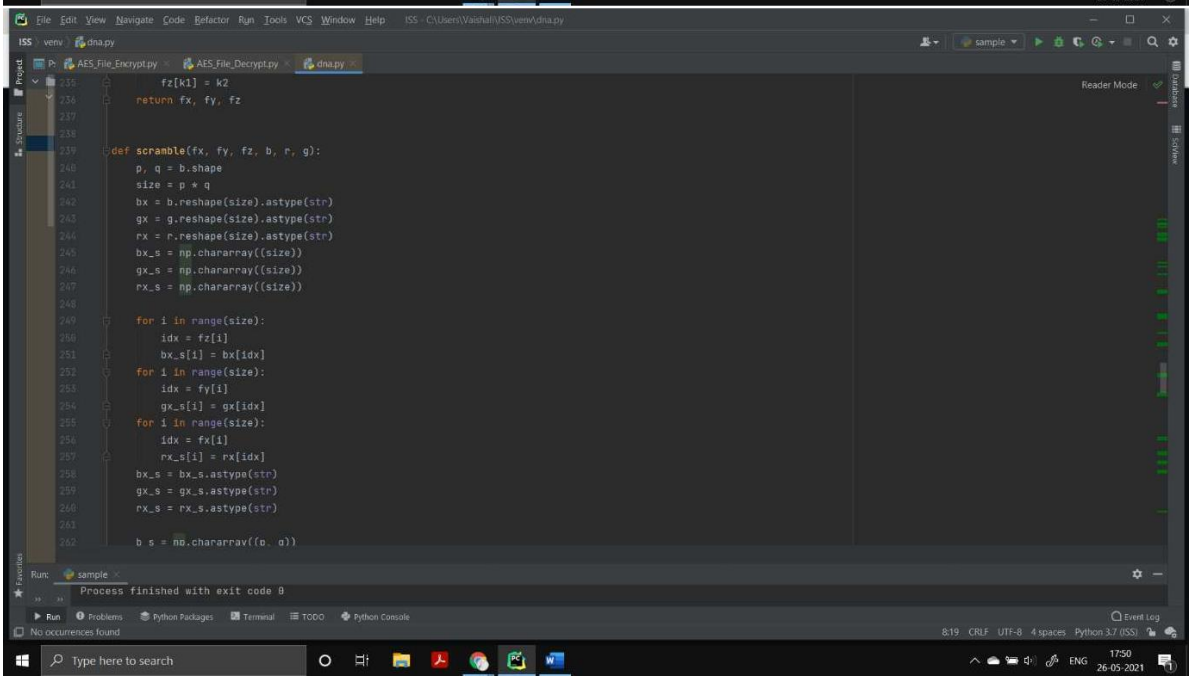
```
    def xor_operation(b, g, r, mk):
        m, n = b.shape
        bx = np.chararray((m, n))
        gx = np.chararray((m, n))
        rx = np.chararray((m, n))
        b = b.astype(str)
        g = g.astype(str)
        r = r.astype(str)
        for i in range(0, m):
            for j in range(0, n):
                bx[i, j] = dna["{0}{1}".format(b[i, j], mk[i, j])]
                gx[i, j] = dna["{0}{1}".format(g[i, j], mk[i, j])]
                rx[i, j] = dna["{0}{1}".format(r[i, j], mk[i, j])]

        bx = bx.astype(str)
        gx = gx.astype(str)
        rx = rx.astype(str)
        return bx, gx, rx


    def gen_chaos_seq(m, n):
        global x0, y0, z0, a, b, c, N
        N = m * n * 4
        x = np.array((m, n * 4))
        y = np.array((m, n * 4))
        z = np.array((m, n * 4))
        t = np.linspace(0, tmax, N)
```

Process finished with exit code 0



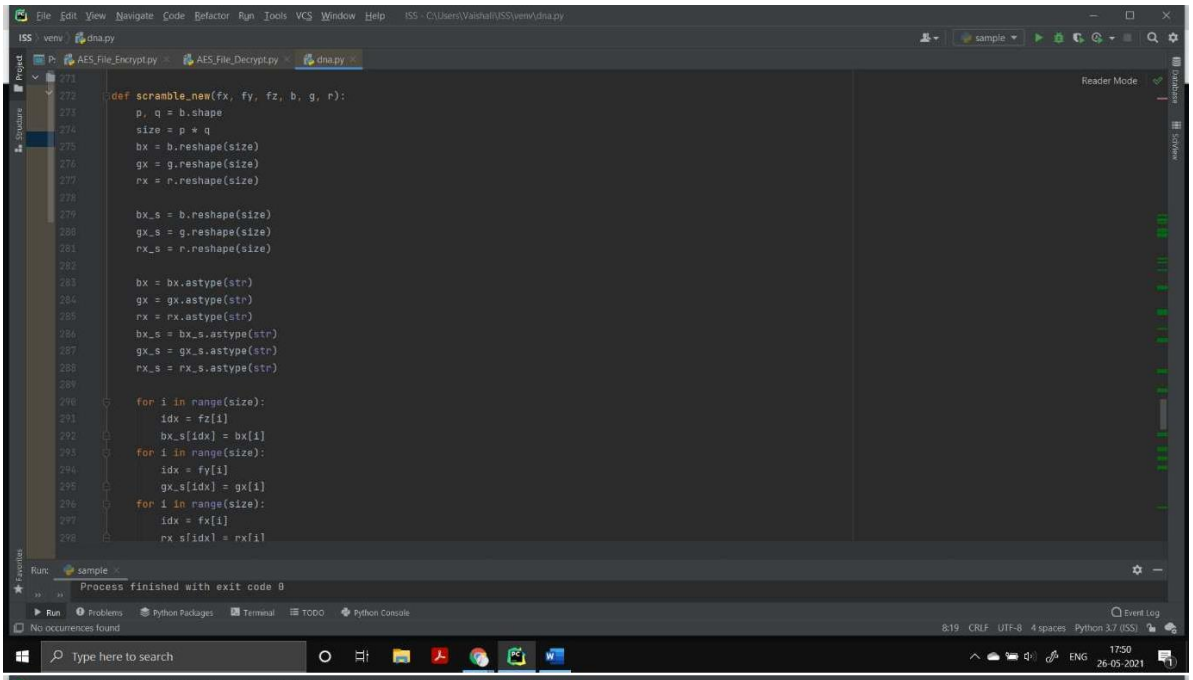
```
def plot(x, y, z):
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    s = 100
    c = np.linspace(0, 1, N)
    for i in range(0, N - s, s):
        ax.plot(x[i:i + s + 1], y[i:i + s + 1], z[i:i + s + 1], color=(1 - c[i], c[i], 1), alpha=0.4)
    ax.set_axis_off()
    plt.show()


def sequence_indexing(x, y, z):
    n = len(x)
    fx = np.zeros((n), dtype=np.uint32)
    fy = np.zeros((n), dtype=np.uint32)
    fz = np.zeros((n), dtype=np.uint32)
    seq = sorted(x)
    for k1 in range(0, n):
        t = x[k1]
        k2 = bsearch(seq, t)
        fx[k1] = k2
    seq = sorted(y)
    for k1 in range(0, n):
        t = y[k1]
        k2 = bsearch(seq, t)
        fy[k1] = k2
    seq = sorted(z)
    for k1 in range(0, n):
```

```
Process finished with exit code 0
```



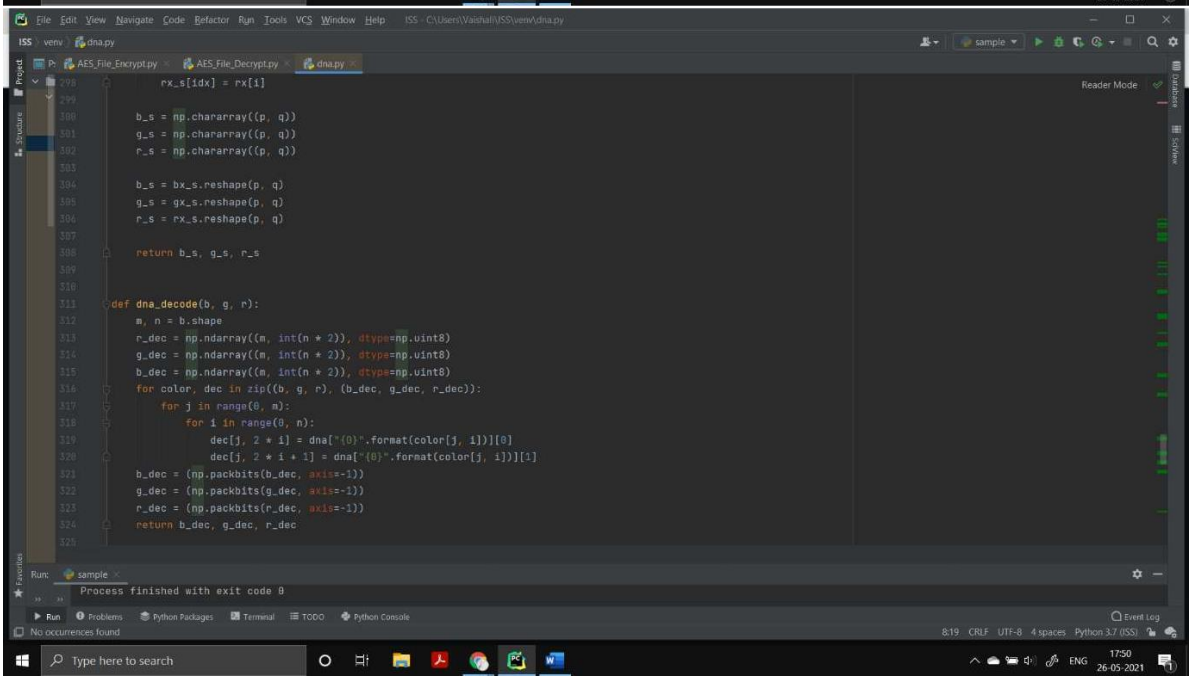
```
        fz[k1] = k2
    return fx, fy, fz


def scramble(fx, fy, fz, b, r, g):
    p, q = b.shape
    size = p * q
    bx = b.reshape(size).astype(str)
    gx = g.reshape(size).astype(str)
    rx = r.reshape(size).astype(str)
    bx_s = np.chararray((size))
    gx_s = np.chararray((size))
    rx_s = np.chararray((size))

    for i in range(size):
        idx = fz[i]
        bx_s[i] = bx[idx]
    for i in range(size):
        idx = fy[i]
        gx_s[i] = gx[idx]
    for i in range(size):
        idx = fx[i]
        rx_s[i] = rx[idx]
    bx_s = bx_s.astype(str)
    gx_s = gx_s.astype(str)
    rx_s = rx_s.astype(str)

    b_s = np.chararray((p, q))
```

```
Process finished with exit code 0
```



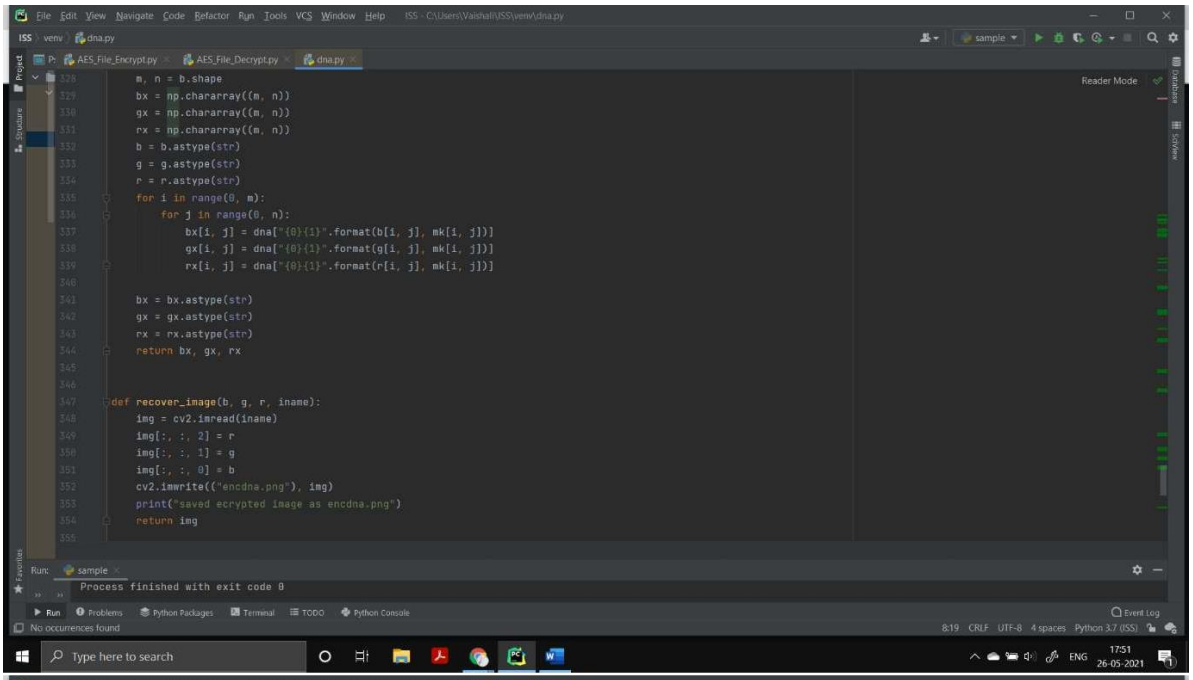
```
def scramble_new(fx, fy, fz, b, g, r):
    p, q = b.shape
    size = p * q
    bx = b.reshape(size)
    gx = g.reshape(size)
    rx = r.reshape(size)

    bx_s = b.reshape(size)
    gx_s = g.reshape(size)
    rx_s = r.reshape(size)

    bx = bx.astype(str)
    gx = gx.astype(str)
    rx = rx.astype(str)
    bx_s = bx_s.astype(str)
    gx_s = gx_s.astype(str)
    rx_s = rx_s.astype(str)

    for i in range(size):
        idx = fz[i]
        bx_s[idx] = bx[i]
    for i in range(size):
        idx = fy[i]
        gx_s[idx] = gx[i]
    for i in range(size):
        idx = fx[i]
        rx_s[idx] = rx[i]
```

```
Process finished with exit code 0
```



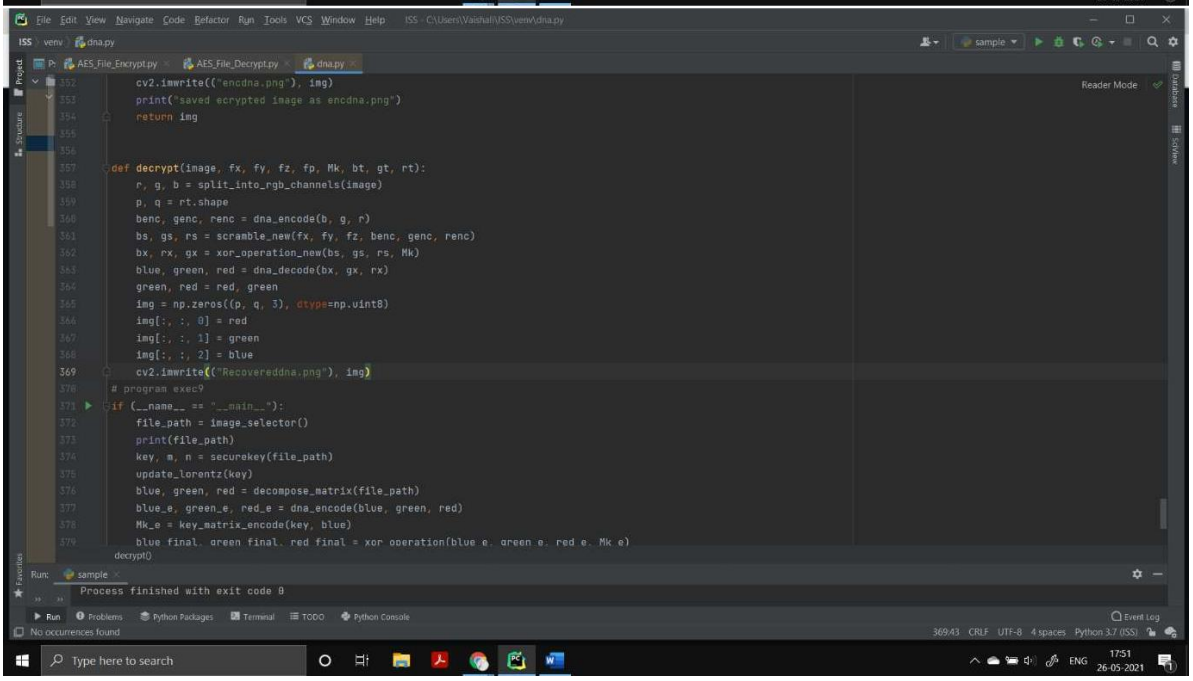
```
        rx_s[idx] = rx[i]

    b_s = np.chararray((p, q))
    g_s = np.chararray((p, q))
    r_s = np.chararray((p, q))

    b_s = bx_s.reshape(p, q)
    g_s = gx_s.reshape(p, q)
    r_s = rx_s.reshape(p, q)

    return b_s, g_s, r_s


def dna_decode(b, g, r):
    m, n = b.shape
    r_dec = np.ndarray((m, int(n * 2)), dtype=np.uint8)
    g_dec = np.ndarray((m, int(n * 2)), dtype=np.uint8)
    b_dec = np.ndarray((m, int(n * 2)), dtype=np.uint8)
    for color, dec in zip((b, g, r), (b_dec, g_dec, r_dec)):
        for j in range(0, m):
            for i in range(0, n):
                dec[j, 2 * i] = dna["{0}".format(color[j, i])][0]
                dec[j, 2 * i + 1] = dna["{0}".format(color[j, i])][1]
    b_dec = (np.packbits(b_dec, axis=-1))
    g_dec = (np.packbits(g_dec, axis=-1))
    r_dec = (np.packbits(r_dec, axis=-1))
    return b_dec, g_dec, r_dec
```

```
Process finished with exit code 0
```



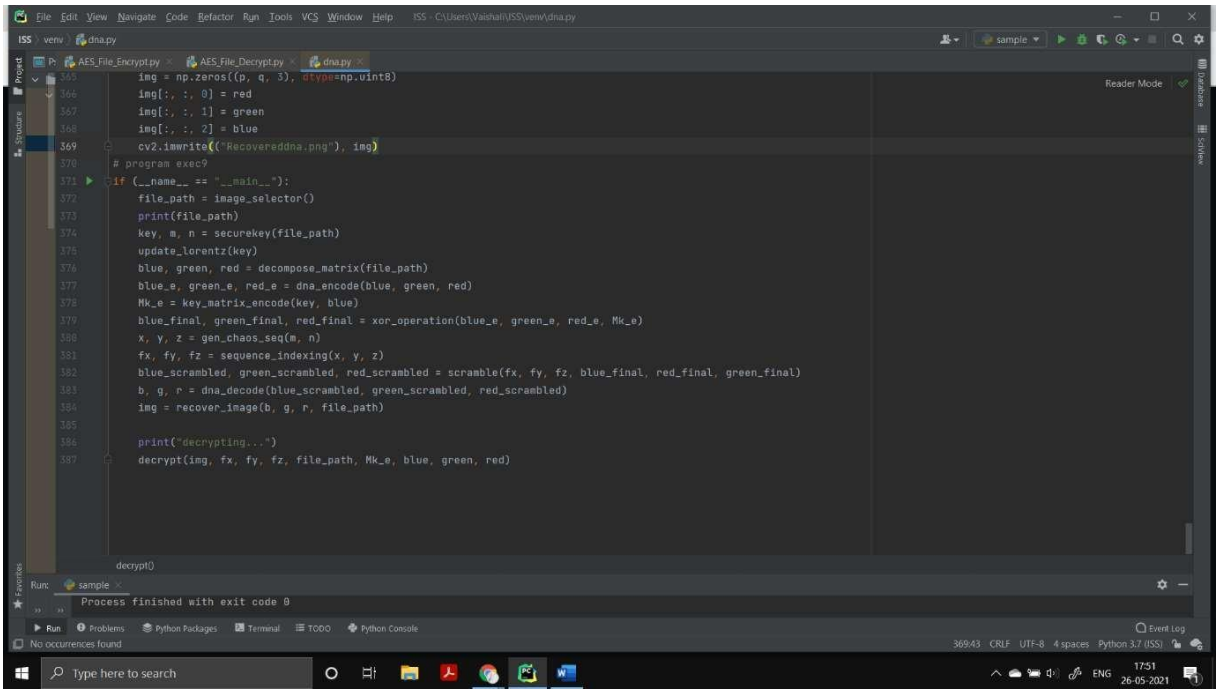
```
        m, n = b.shape
        bx = np.chararray((m, n))
        gx = np.chararray((m, n))
        rx = np.chararray((m, n))
        b = b.astype(str)
        g = g.astype(str)
        r = r.astype(str)
        for i in range(0, m):
            for j in range(0, n):
                bx[i, j] = dna["{0}{1}".format(b[i, j], mk[i, j])]
                gx[i, j] = dna["{0}{1}".format(g[i, j], mk[i, j])]
                rx[i, j] = dna["{0}{1}".format(r[i, j], mk[i, j])]

        bx = bx.astype(str)
        gx = gx.astype(str)
        rx = rx.astype(str)
        return bx, gx, rx


def recover_image(b, g, r, iname):
    img = cv2.imread(iname)
    img[:, :, 2] = r
    img[:, :, 1] = g
    img[:, :, 0] = b
    cv2.imwrite(("encdna.png"), img)
    print("saved ecrypted image as encdna.png")
    return img
```

```
Process finished with exit code 0
```

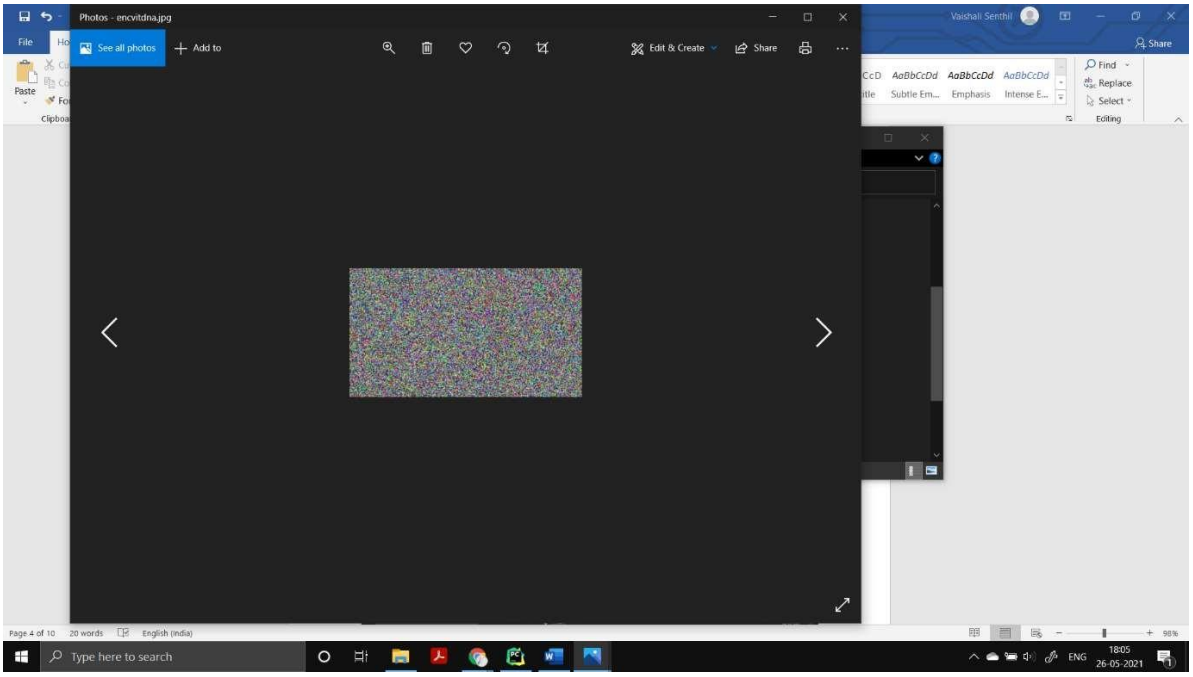


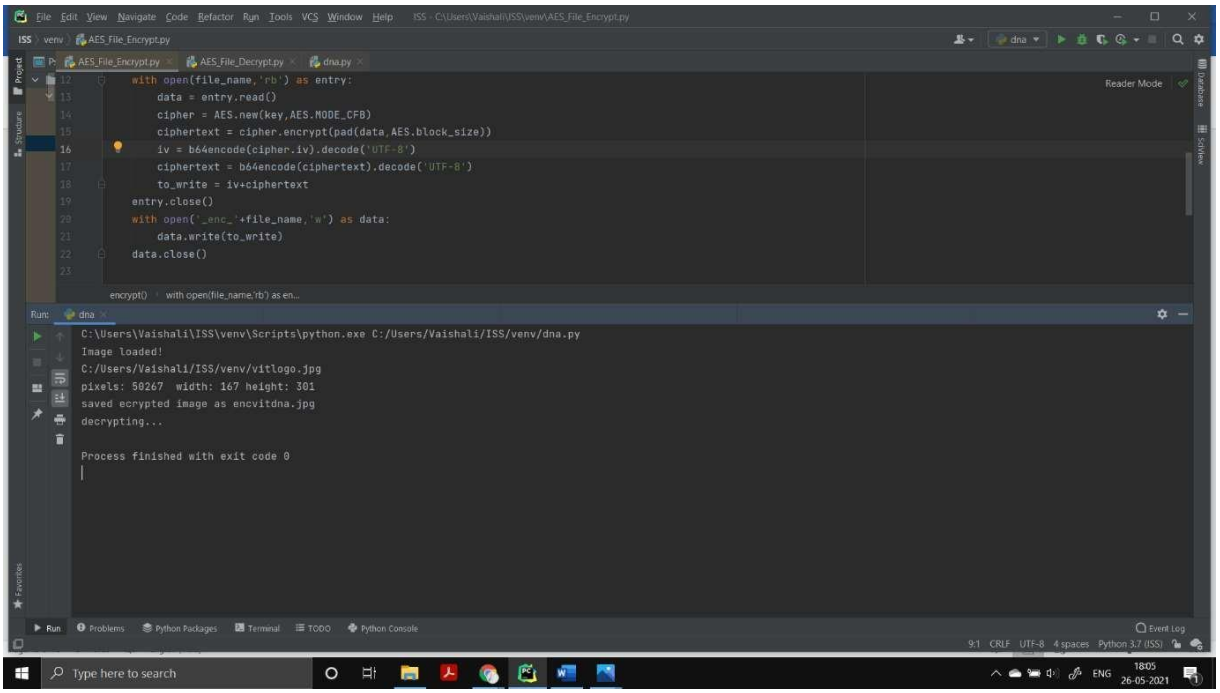
```
        cv2.imwrite(("encdna.png"), img)
        print("saved ecrypted image as encdna.png")
        return img


def decrypt(image, fx, fy, fz, fp, Mk, bt, gt, rt):
    r, g, b = split_into_rgb_channels(image)
    p, q = rt.shape
    benc, genc, renc = dna_encode(b, g, r)
    bs, gs, rs = scramble_new(fx, fy, fz, benc, genc, renc)
    bx, rx, gx = xor_operation_new(bs, gs, rs, Mk)
    blue, green, red = dna_decode(bx, gx, rx)
    green, red = red, green
    img = np.zeros((p, q, 3), dtype=np.uint8)
    img[:, :, 0] = red
    img[:, :, 1] = green
    img[:, :, 2] = blue
    cv2.imwrite(("Recovereddna.png"), img)
# program exec9
if (__name__ == "__main__"):
    file_path = image_selector()
    print(file_path)
    key, m, n = securekey(file_path)
    update_lorentz(key)
    blue, green, red = decompose_matrix(file_path)
    blue_e, green_e, red_e = dna_encode(blue, green, red)
    Mk_e = key_matrix_encode(key, blue)
    blue_final, green_final, red_final = xor_operation(blue_e, green_e, red_e, Mk_e)
decrypt()
```

```
Process finished with exit code 0
```


img = np.zeros((p, q, 3), dtype=np.uint8)
img[:, :, 0] = red
img[:, :, 1] = green
img[:, :, 2] = blue
cv2.imwrite(("Recovereddna.png"), img)
# program exec9
if (__name__ == "__main__"):
file_path = image_selector()
print(file_path)
key, m, n = securekey(file_path)
update_lorentz(key)
blue, green, red = decompose_matrix(file_path)
blue_e, green_e, red_e = dna_encode(blue, green, red)
Mk_e = key_matrix_encode(key, blue)
blue_final, green_final, red_final = xor_operation(blue_e, green_e, red_e, Mk_e)
x, y, z = gen_chaos_seq(m, n)
fx, fy, fz = sequence_indexing(x, y, z)
blue_scrambled, green_scrambled, red_scrambled = scramble(fx, fy, fz, blue_final, red_final, green_final)
b, g, r = dna_decode(blue_scrambled, green_scrambled, red_scrambled)
img = recover_image(b, g, r, file_path)
print("decrypting...")
decrypt(img, fx, fy, fz, file_path, Mk_e, blue, green, red)
Process finished with exit code 0

Output


Photos - encvitdna.jpg
See all photos
Add to
Edit & Create
Share



REFERENCES:

[1] Karthigaikumar, P., & Rasheed, S. (2011). Simulation of image encryption using AES algorithm. *IJCA special issue on "computational science-new dimensions & perspectives" NCCSE*, 166-172.

[2] Zeghid, M., Machhout, M., Khriji, L., Baganne, A., & Tourki, R. (2007). A modified AES based algorithm for image encryption. *International Journal of Computer Science and Engineering*, *1*(1), 70-75.

[3] Zhang, X., & Parhi, K. K. (2004). High-speed VLSI architectures for the AES algorithm. *IEEE transactions on very large scale integration (VLSI) systems*, *12*(9), 957-967.

[4] Gaj, K., & Chodowiec, P. (2009). FPGA and ASIC implementations of AES. In *Cryptographic engineering* (pp. 235-294). Springer, Boston, MA.

[5] Radhadevi, P., & Kalpana, P. (2012). Secure image encryption using AES. *International Journal of Research in Engineering and Technology*, *1*(2), 15-117.

[6] Bhavani, Y., Puppala, S. S., Krishna, B. J., & Madarapu, S. (2019, December). Modified AES using Dynamic S-Box and DNA Cryptography. In 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 164-168). IEEE.

[7] Nag, A., Singh, J. P., Khan, S., Ghosh, S., Biswas, S., Sarkar, D., & Sarkar, P. P. (2011, July). Image encryption using affine transform and XOR operation. In *2011 International conference on signal processing, communication, computing and networking technologies* (pp. 309-312). IEEE.

[8] Sklavos, N., & Koufopavlou, O. (2002). Architectures and VLSI implementations of the AES-proposal Rijndael. *IEEE Transactions on computers*, *51*(12), 1454-1459.

[9] Zhou, S., Wang, B., Zheng, X., & Zhou, C. (2016). An image encryption scheme based on DNA computing and cellular automata. Discrete Dynamics in Nature and Society, 2016.

[10] Norouzi, B., Seyedzadeh, S. M., Mirzakuchaki, S., & Mosavi, M. R. (2014). A novel image encryption based on hash function with only two-round diffusion process. *Multimedia systems*, *20*(1), 45-64.

[11] Shah, T., Hussain, I., Gondal, M. A., & Mahmood, H. (2011). Statistical analysis of S-box in image encryption applications based on majority logic criterion. *International Journal of Physical Sciences*, *6*(16), 4110-4127.

[12] Ismail, I. A., Amin, M., & Diab, H. (2010). A digital image encryption algorithm based a composition of two chaotic logistic maps. *IJ Network Security*, *11*(1), 1-10.

[13] Indrakanti, S. P., & Avadhani, P. S. (2011). Permutation based image encryption technique. *International Journal of Computer Applications*, *28*(8), 45-47.

[14] Enayatifar, R., & Abdullah, A. H. (2011). Image security via genetic algorithm. In International Conference on Computer and Software Modeling (Vol. 14, pp. 198-203).

[15] Bhoi, G., Bhavsar, R., Prajapati, P., & Shah, P. (2020, December). A Review of Recent Trends on DNA Based Cryptography. In 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS) (pp. 815-822). IEEE.

[16] Sabry, M., Hashem, M., Nazmy, T., & Khalifa, M. E. (2015, December). Design of DNA-based advanced encryption standard (AES). In 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS) (pp. 390-397). IEEE.

[17] Mondal, M., & Ray, K. S. (2019). Review on DNA cryptography. arXiv preprint arXiv:1904.05528.

[18] Samiullah, M., Aslam, W., Nazir, H., Lali, M. I., Shahzad, B., Mufti, M. R., & Afzal, H. (2020). An image encryption scheme based on DNA computing and multiple chaotic systems. IEEE Access, 8, 25650-25663.