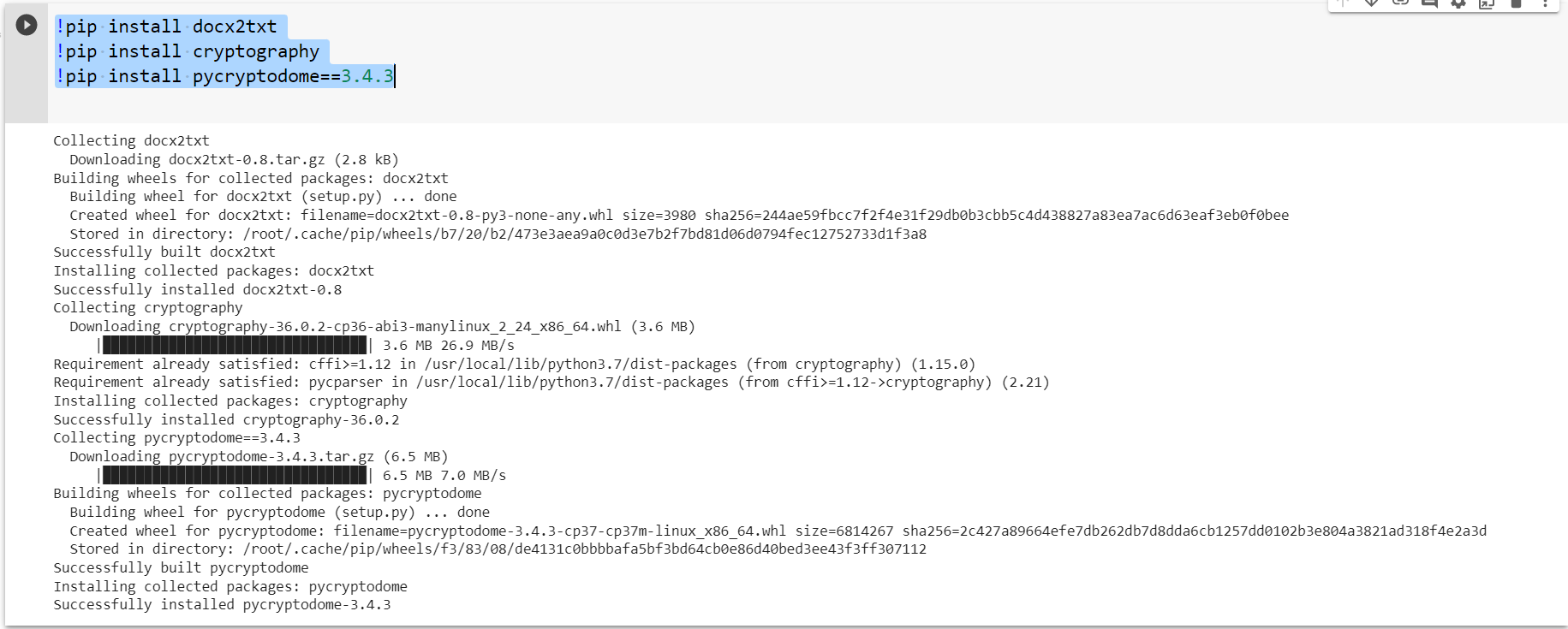
**Data Security using Hybrid Cryptography and Steganography.**

**Module-1**

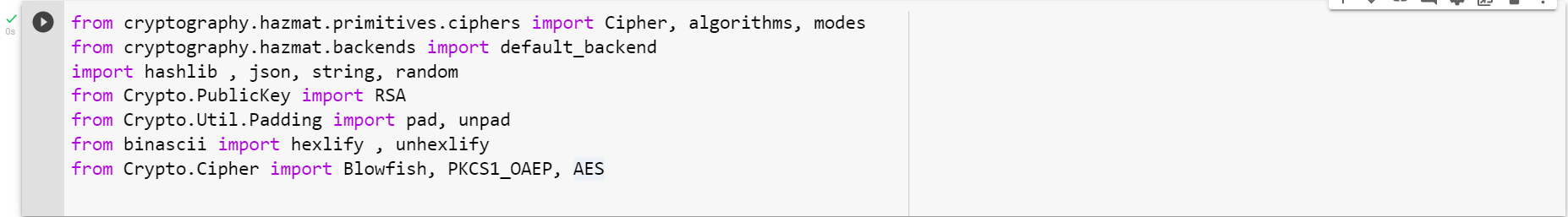
**Data Pre-processing and Key Generation**

**i)Importing Dependencies and packages.**

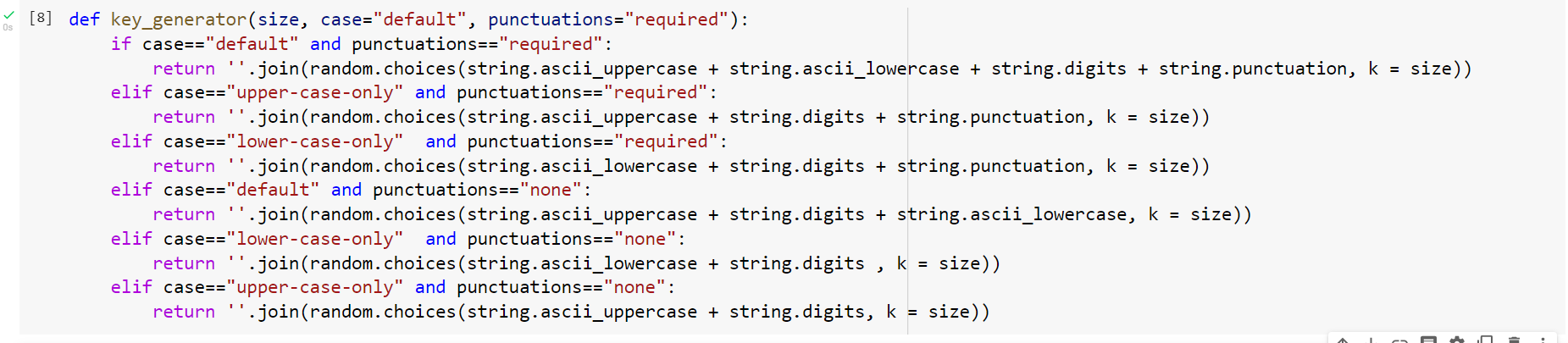
First several python packages that are required to implement cryptography algorithm are installed.



Import requires modules/functions from the packages to perform document pre-processing and random functions for generating keys for cryptography algorithm.

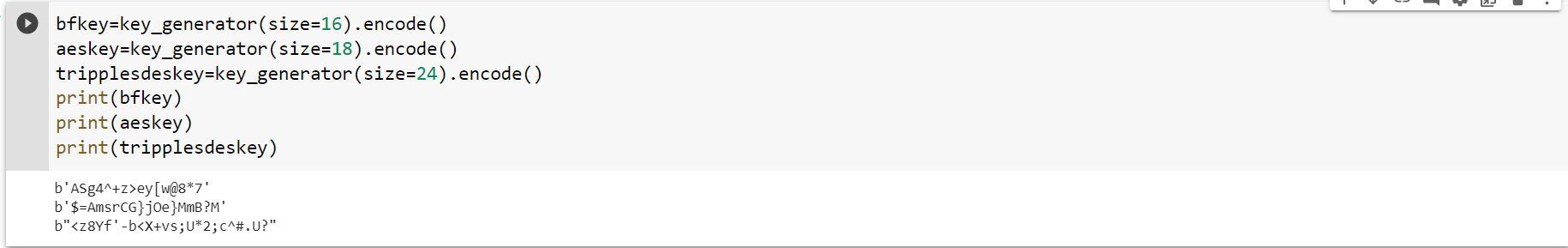


**ii)Key generation:**



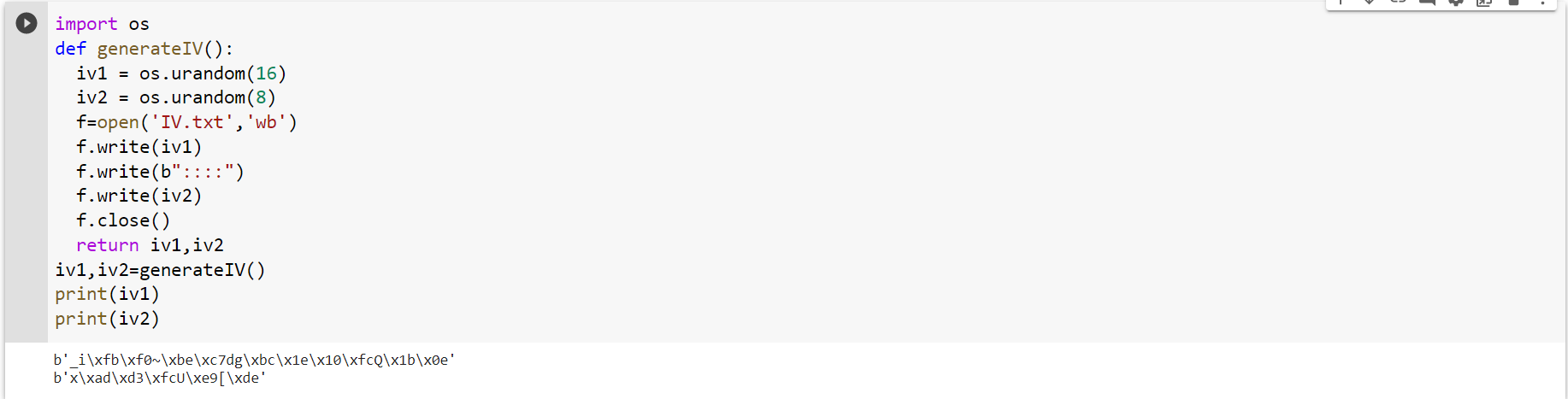
Random function is used to generate keys for blowfish, AES and TrippleDes cryptography algorithm. It generates different set of keys of required length everytime.

1st time:



**iii)Initialization vector generation:**

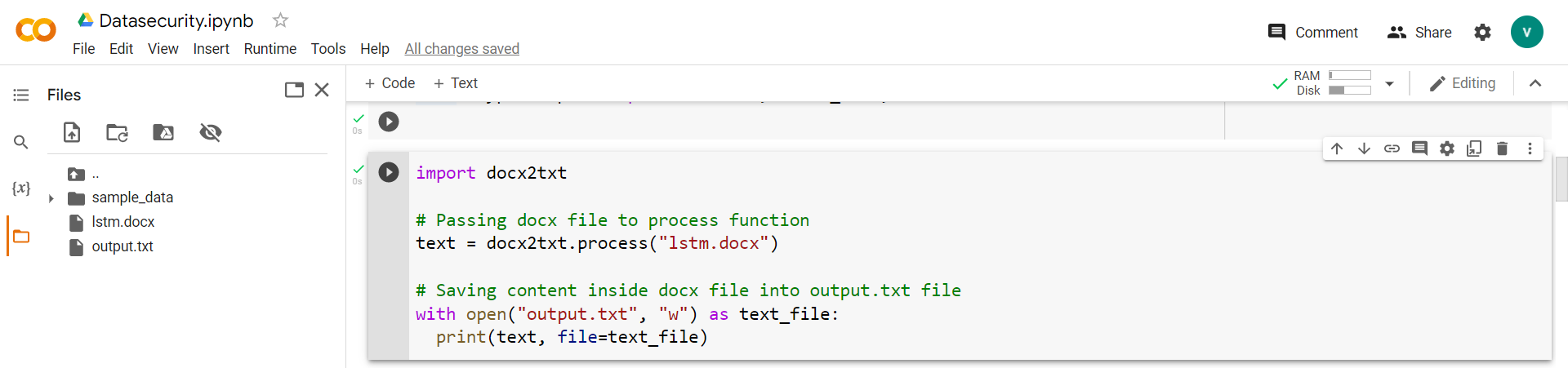
In cryptography, an initialization vector (IV) or starting variable (SV)is an input to a cryptographic primitive being used to provide the initial state. The IV is typically required to be random or pseudorandom, but sometimes an IV only needs to be unpredictable or unique. Randomization is crucial for some encryption schemes to achieve semantic security, a property whereby repeated usage of the scheme under the same key does not allow an attacker to infer relationships between (potentially similar) segments of the encrypted message. For block ciphers, the use of an IV is described by the modes of operation.



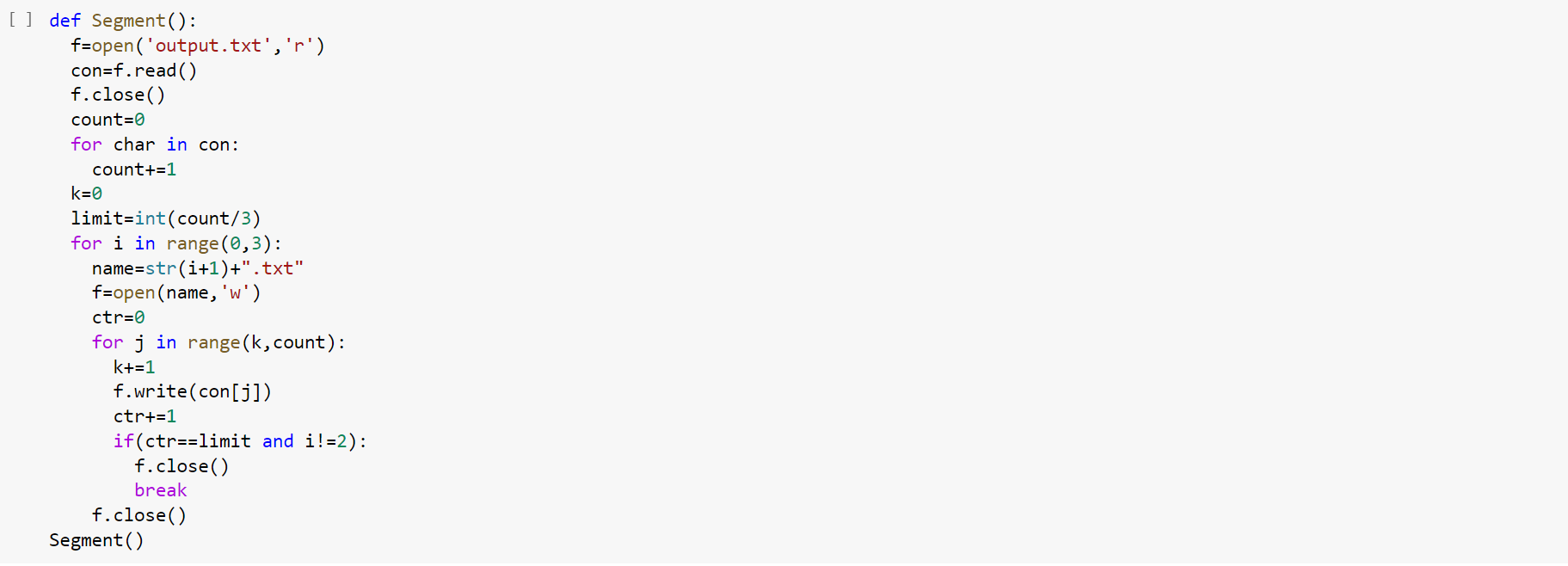
Two Initialization vectors are generated one of length 16 and other 8, it is stored in “IV.TXT” file for further reference.

**iv)Data pre-processing and separation.**

When a user submits a document for encryption it is converted into text file using doc2txt package. Since text format is easier and convenient to apply encryption algorithm. The pre-processing of document to text fil.

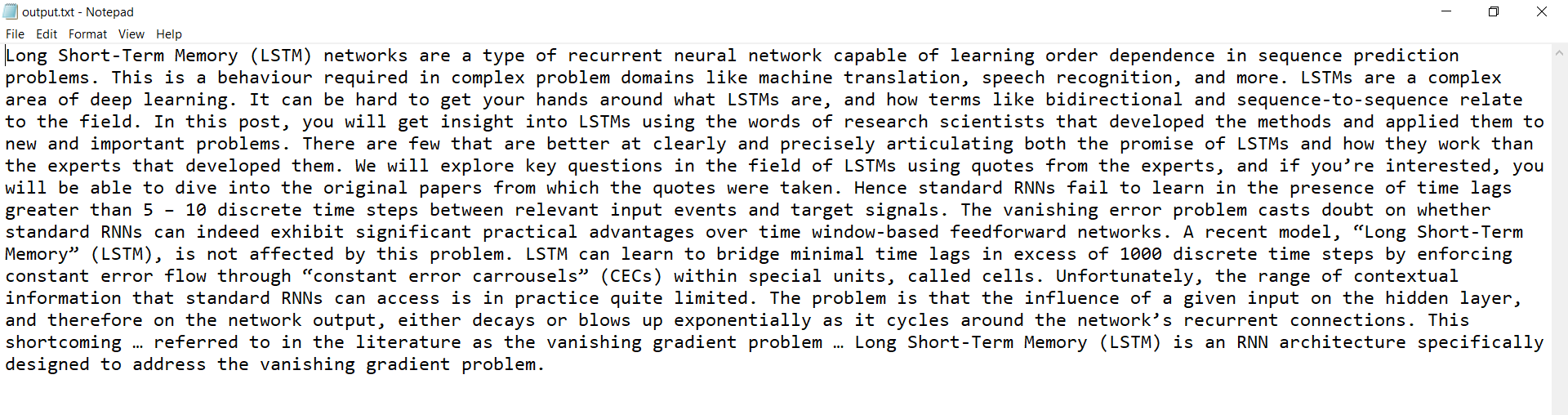


Here we upload a sample file named “ LSTM.DOCX” which is converted to “output.txt”.

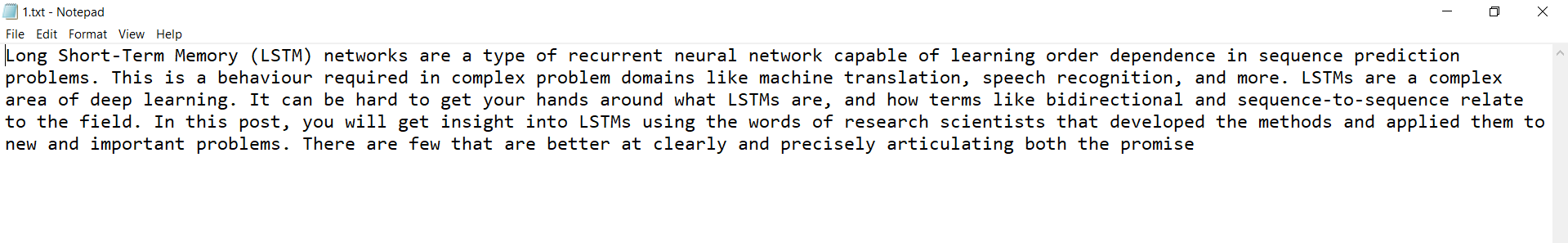


In order to perform hybrid cryptography (BLOWFISH, AES, TrippleDes ) we have split the document into 3 parts based on the character count. Below are the original document and the 3 parts namely “1.txt, 2.txt, 3.txt”.

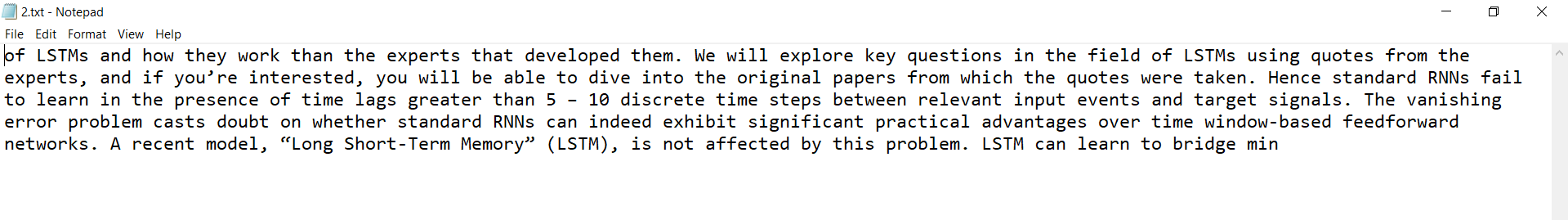
**Whole document:**



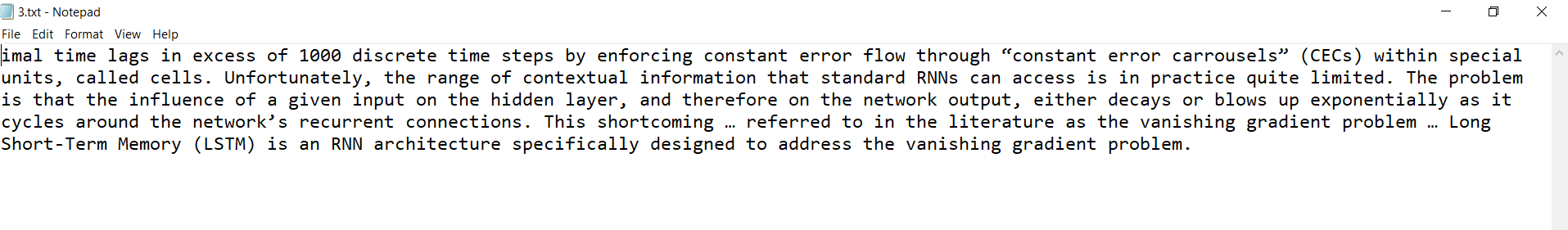
Part-1 **1.txt**



Part-2 **2.txt**



Part-3 **3.txt**



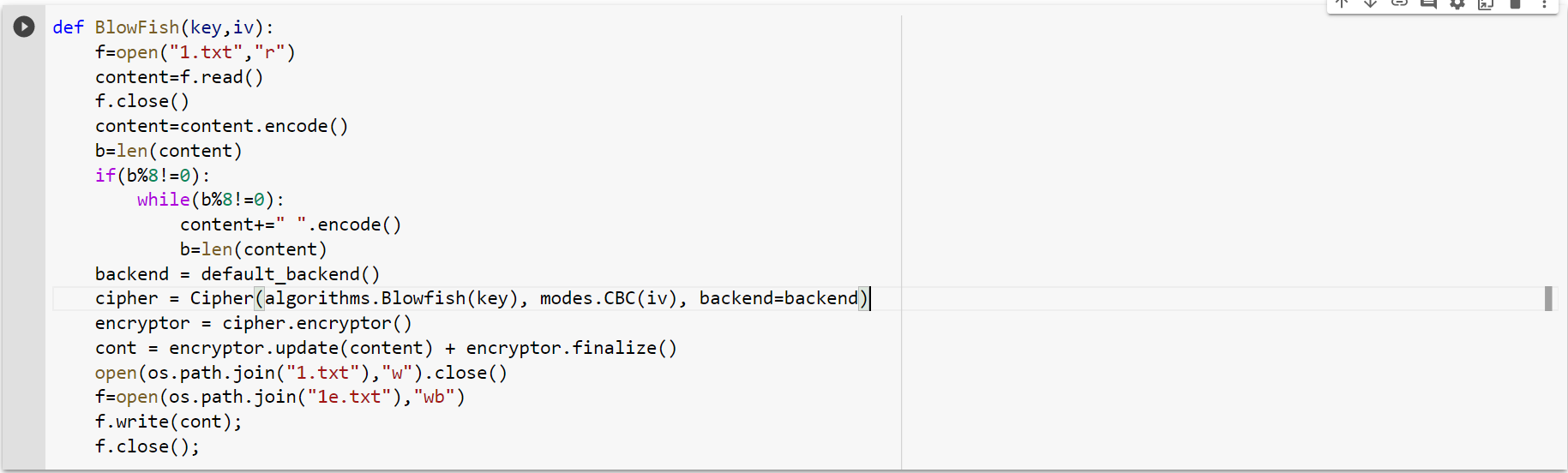
**Module-2**

**Cryptography algorithm-Encryption**

**i)Blowfish Encryption.**

Blowfish, a Symmetric-Key Block Cipher, was developed by B. Schneier in the year 1993. Blowfish algorithm has 64 Bits block size and variable key length of 32 to 448 Bits. It is particularly known for its features like complicated key schedules and key dependent s-boxes. Being a Feistel cipher it has 16 rounds. Each round, consists of four steps. In nth round, the left half of the block and the nth element in the subkey-array are XORed followed by passing it to the round function F. The return from the function F and the right half of the initial block are XORed and then swapped. The round function F divides the 32-bit input into four 8-bit blocks that are then fed to 4 different S-Boxes. The returns from the 1st and 2nd s-box are added and the return is XORed with the returns from the 3rd s-box and again added with the output of the 4th s-box. It is one of the fastest algorithms for encryption.

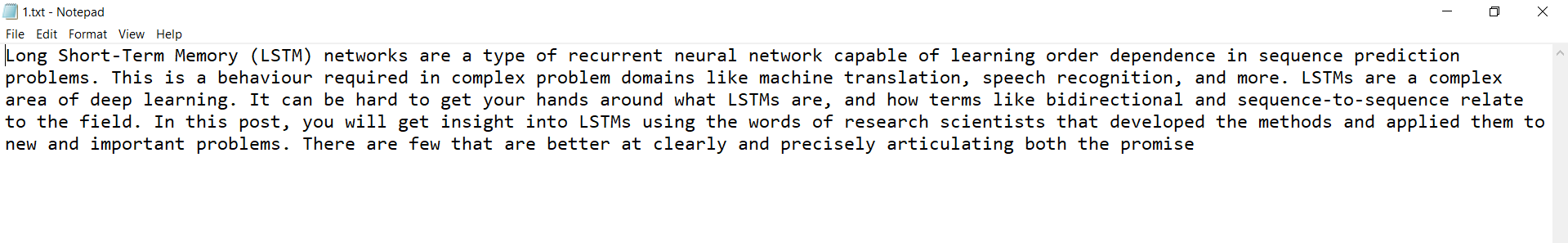
Previously the documents have been split into 3 parts the first part encrypted using Blowfish.





**1.txt** is encrypted to “1e.txt” file using blowfish. Keys are generated using key generator module.

**Original- document :**

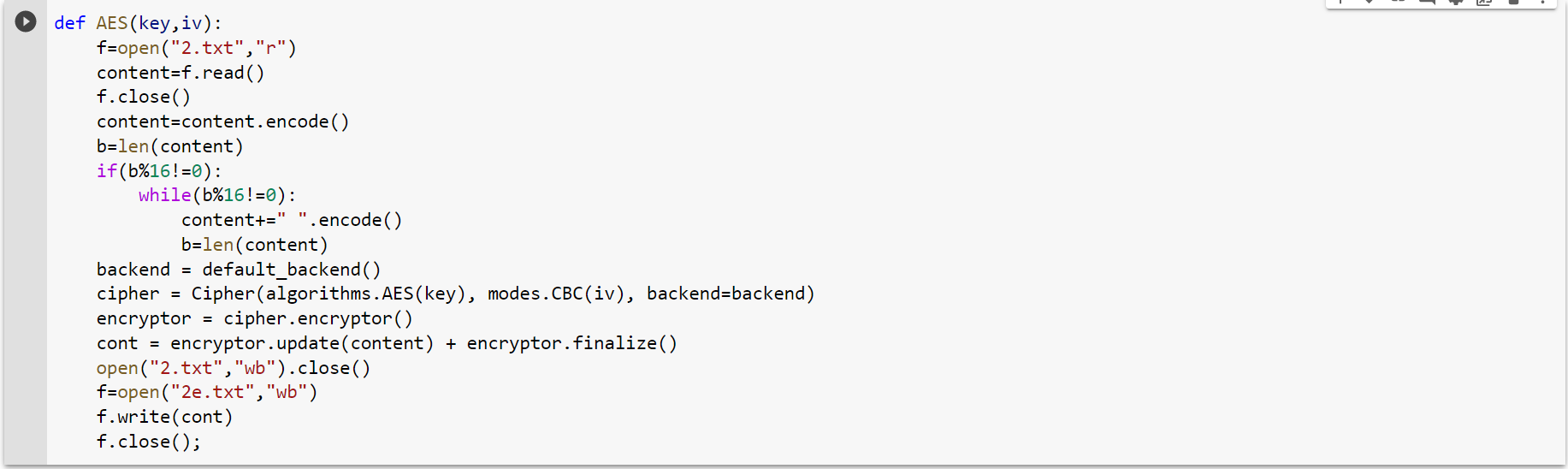


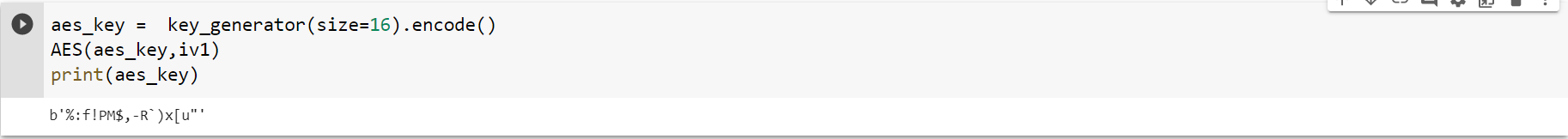
**Encrypted Cipher-Text:**



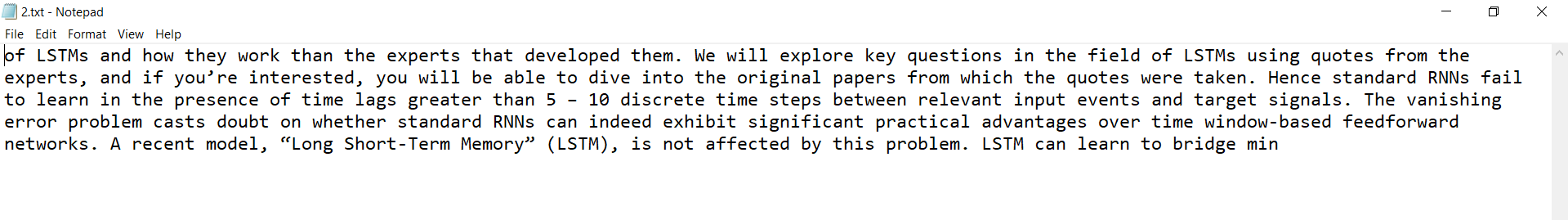
**ii)Advanced Encryption Standard Encryption.**

Rijndael, proposed by Belgian cryptographers, Vincent Rijmen and Joan Daemen, is Symetric-key Block-Cipher that has been established as the Advanced Encryption Standard by the National Institute of Standards and Technology (NIST) of The United States of America, in the year of 2001. The AES may have keys varying in size between 128, 192, 256 bits, & having 10, 12, 14 rounds respectively. The n-Bits key is expanded using AES Key-Scheduling into several subkeys depending on the number of rounds. In the beginning, the input block is XORed with an Initial Round-Key. Then, for the first N-1 rounds, 4 Round Functions are applied on each block. The first-round function is Substitute Bytes where every byte is substituted by another, from the lookup table. Followed by Shift-Rows, where the last three rows are cyclically shifted by certain number of steps. Shift-Rows is followed by Mix-Columns, where a linear mixing operation is executed on the columns, combining the 4-bytes of each column. Lastly, Add-Round Key function is executed on the current state, where each byte, and a byte of the round key are combined using bit-wise XOR operation. For the Nth round, i.e the last round all the above functions are applied except the Mixed Columns step. AES is one of the most extensively used and secure algorithms for data security. Even though it is slower than blowfish, it provides a higher level of data security.

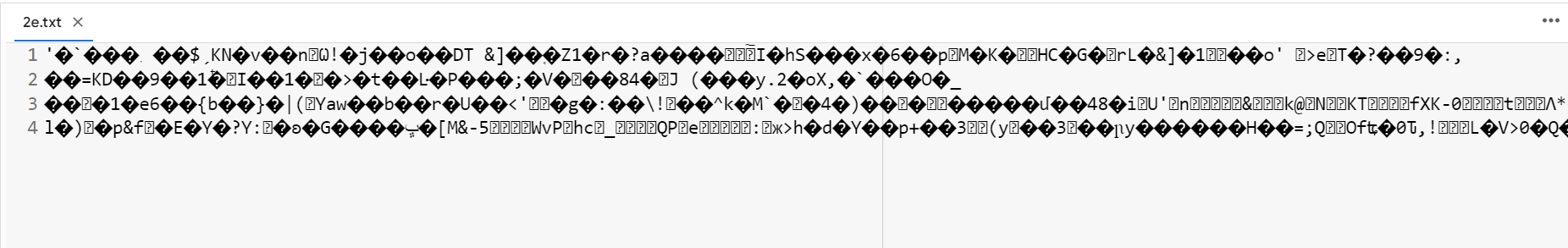




Second part “2.txt” is encrypted using Advanced encryption standard cryptography algorithm into “2e.txt”.

**Original Document**

**Encrypted Cipher-Text:**



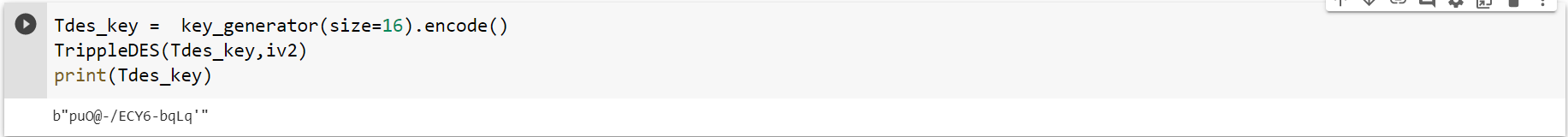
**iii)Triple-DES Encryption.**

The DES (Data Encryption Standard) In cryptography, Triple DES (3DES or TDES), officially the Triple Data Encryption Algorithm (TDEA or Triple DEA), is a symmetric-key block cipher, which applies the DES cipher algorithm three times to each data block. While the government and industry standards abbreviate the algorithm's name as TDES (Triple DES) and TDEA (Triple Data Encryption Algorithm), RFC 1851 referred to it as 3DES from the time it first promulgated the idea, and this namesake has since come into wide use by most vendors, users, and cryptographers. Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K1, K2 and K3. This means that the actual 3TDES key has length 3×56 = 168 bits.

* Encrypt the plaintext blocks using single DES with key K1.
* Now decrypt the output of step 1 using single DES with key K2.
* Finally, encrypt the output of step 2 using single DES with key K3.
* The output of step 3 is the ciphertext.

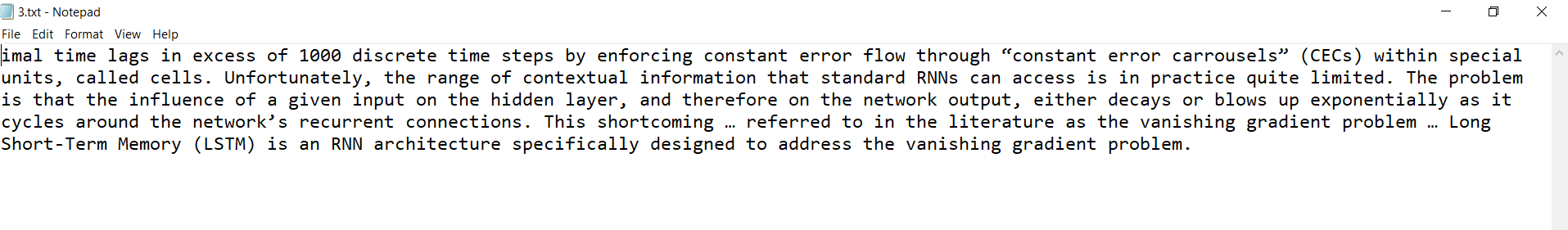
Decryption of a ciphertext is a reverse process. User first decrypt using K3, then encrypt with K2, and finally decrypt with K1.Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for single DES by setting K1, K2, and K3 to be the same value. This provides backwards compatibility with DES. Second variant of Triple DES (2TDES) is identical to 3TDES except that K3is replaced by K1. In other words, user encrypt plaintext blocks with key K1, then decrypt with key K2, and finally encrypt with K1 again. Therefore, 2TDES has a key length of 112 bits. Triple DES systems are significantly more secure than single DES, but these are clearly a much slower process than encryption using single DES.



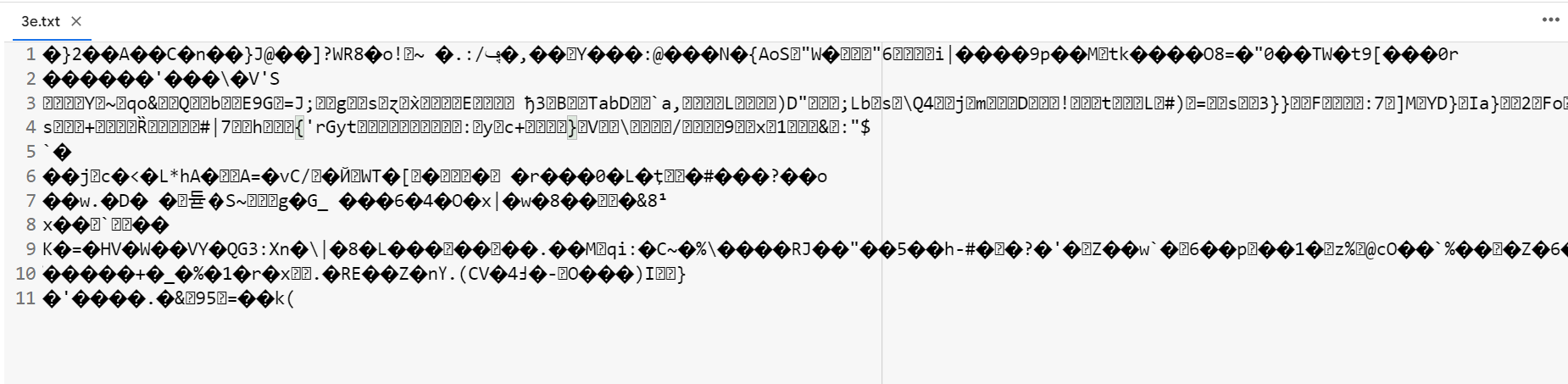


Third part “3.txt” is encrypted using Triple Data encryption Standard or Triple DES algorithm to get encrypted file “3e.txt”.

**Original document**:



**Encrypted Cipher-text:**



**Module-3**

**Steganography and Transfer**

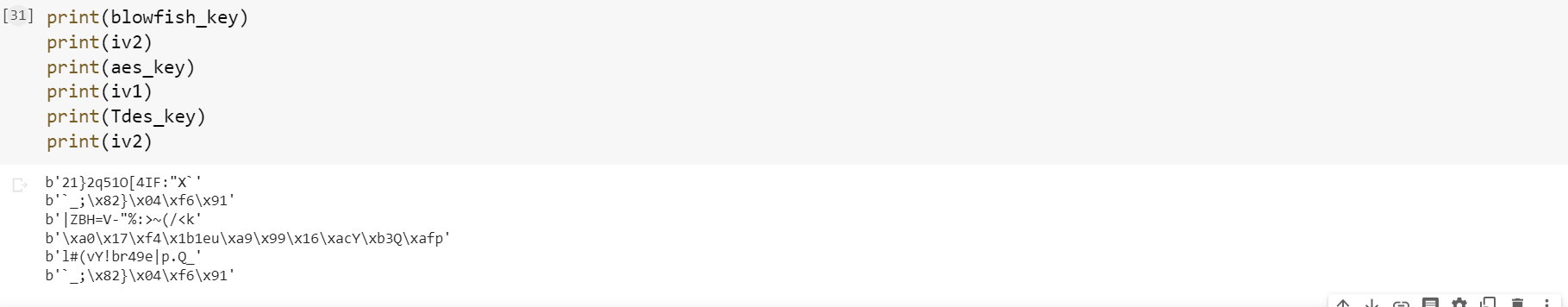
**i)AES to combine keys of 3 Algorithms.**

The Keys used for encryption at the various layers can be securely stored. The List of keys, L stores all the keys generated throughout the Data Encryption Process. Whenever the key for a particular Encryption Layer is generated, it is appended to the List of Keys, L.

In the system, the encryption layers are Blowfish, Triple-Des and AES, respectively, so the Keys used, are stored in the same order as: List of Keys,

L = [ KBlowf ish ,blowfish-iv, K-Tripledes, TripleDES-iv, KAES,aes\_iv ]

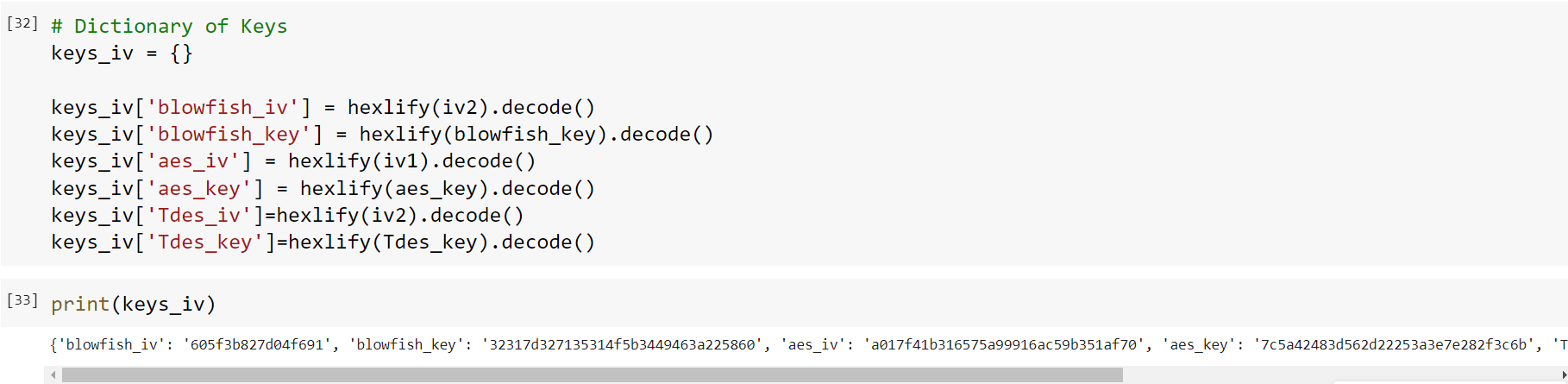
Keys, initialization vectors used for encryption are shown below:



**Step 1:**

This List, L is then passed into a function that converts the list into a single string of keys separated by separators ( x , \* , / )

LS = hexlify( L, separator = 0× 0 )



**Step 2:**

The String, LS is then encrypted using the AES Encryption Algorithm with a Key generated from user-input password. The user inputs a password, PW which is hashed using SHA1, & the first 16 Bits of the Hash is used as the key KP assword .

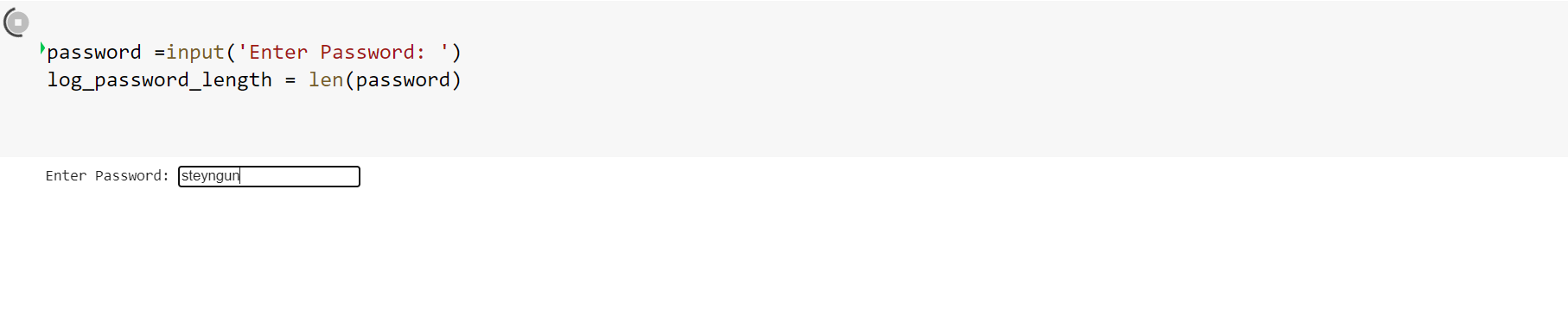
The Key, KP assword is used for the Encryption, generating the encrypted string LS−Encrypted.

HashedPassword, HP = SHA(PW ) Key,

KPassword = HP [0 : 16]

LS−Encrypted = AES(LS, KPassword)

It prompts to enter a password to encrypt the keys using AES.



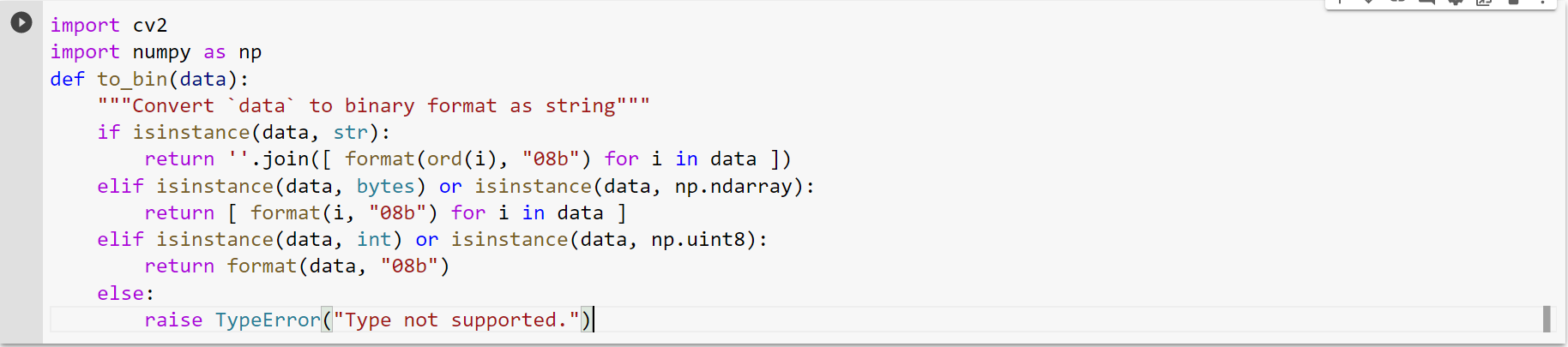


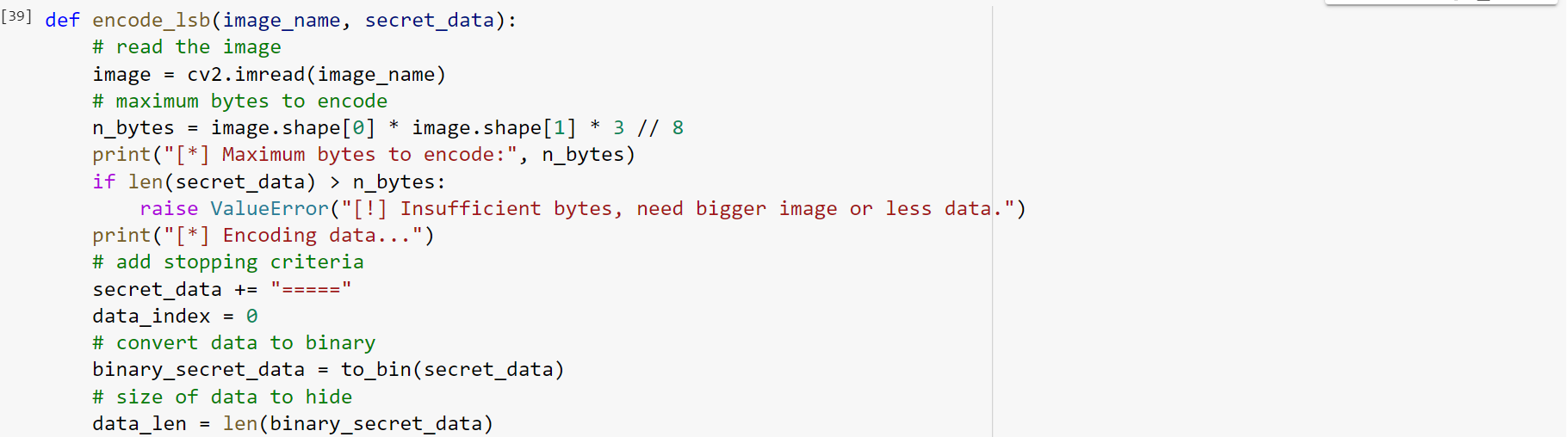
The keys are encrypted using the given password and AES algorithm.

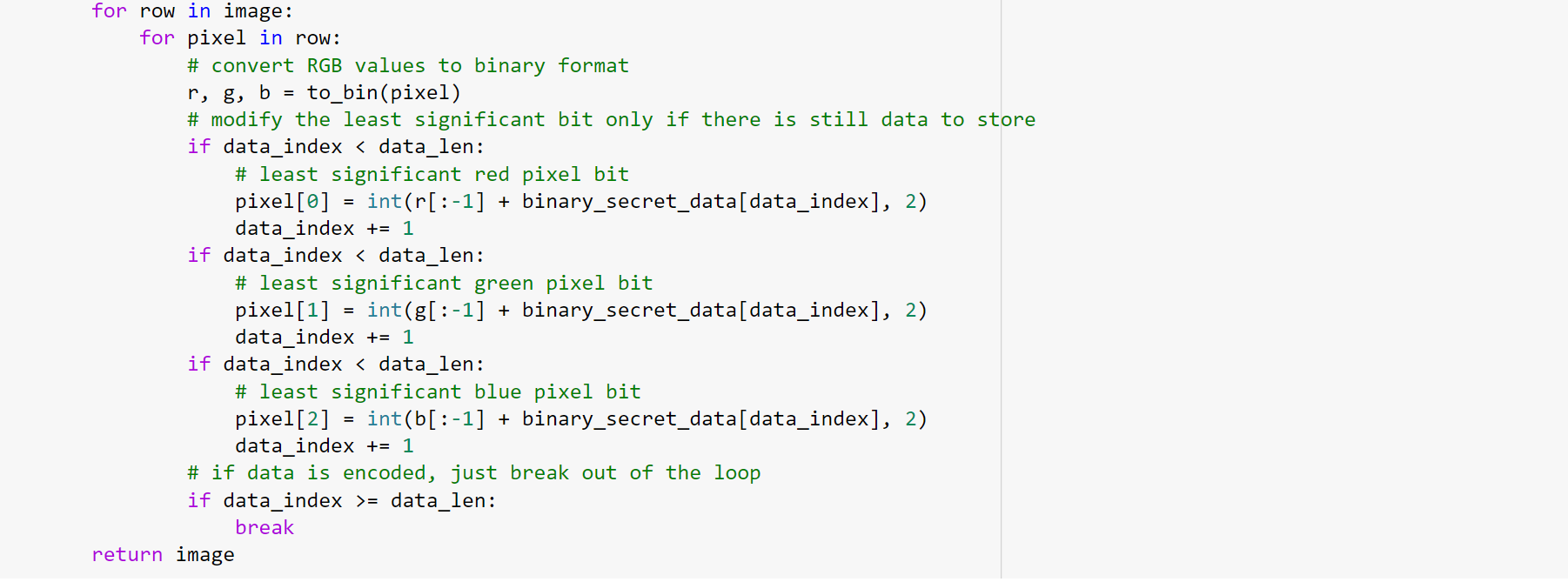
**ii) LSB steganography**

Least Significant Bit Steganography is a technique of hiding data within digital media, here, Image. Images are made up of pixels, and the value of each pixel usually refers to the color-code of that pixel. In a photo’s gray-scale mode, these pixel values range from 0-255. In LSB Image Steganography, the least-significant bit of a pixel is changed, but that doesn’t have much of a visible change in the image. A cover image is where the data is hidden. The cover image is converted to greyscale. The message is converted into binary. Each pixel of the image is traversed through, and for each pixel, initiate a temporary variable, temp. If the LSB of the Pixel Value and the message bit is the same, set temp as 0 and set temp as 1 otherwise. Update the output image pixel as image pixel value added with the temporary variable value, temp. This is done until the message is completely embedded.

The keys encrypted are converted to bits so that they can be placed in lsb of the image.



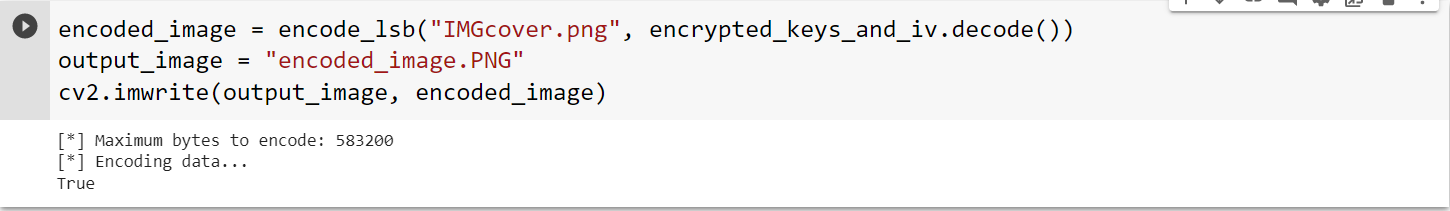




**Cv2**  module is used to convert the images into pixels and place our encrypted keys into it. So that it cannot be misused by a hacker.



The above IMGcover.png is chosen to contain our keys in the LSB bits.

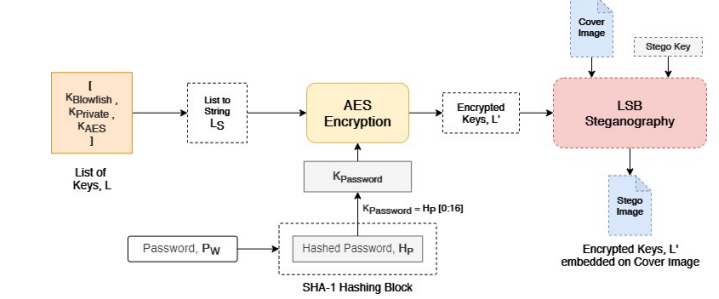


The encoded\_image.png is the image that contains the keys in it least significant bit.



We can notice that there is no change in the image and its resolution even though the LSB bits are modified and keys are placed by steganography.

**Overall block-diagram:**



**Module:4**

**Steganography and Transfer.**

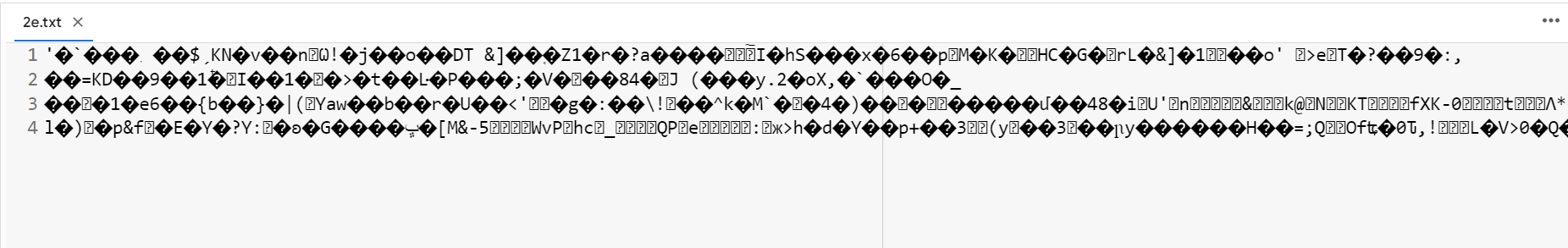
**iii)Encrypted data and Key Transfer**

Encrypted files of Blowfish, Advanced Encryption standard and TripleDES are 1e.txt,2e.txt,3e.txt.

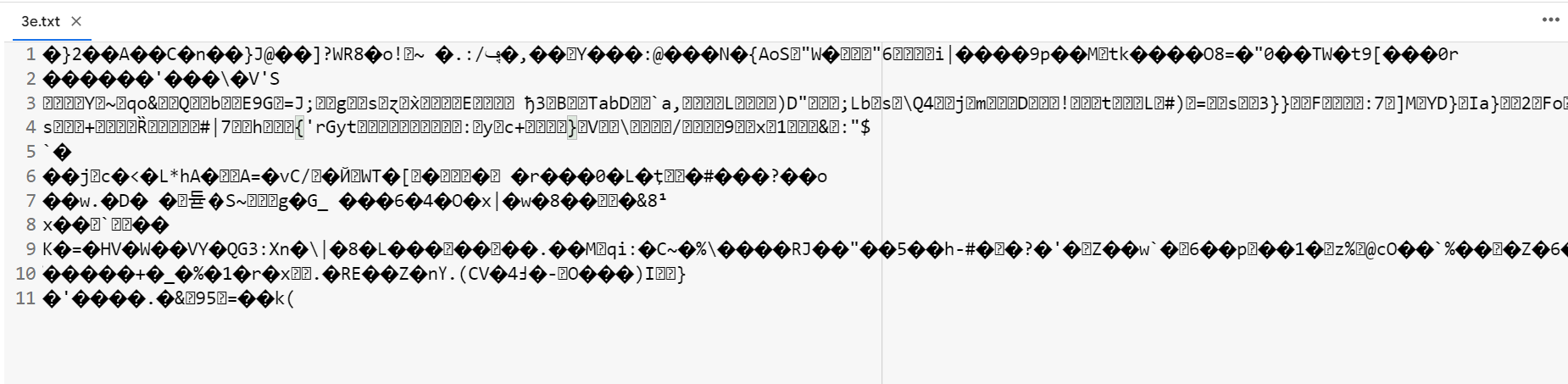
**1e.txt**



**2e.txt**



**3e.txt**



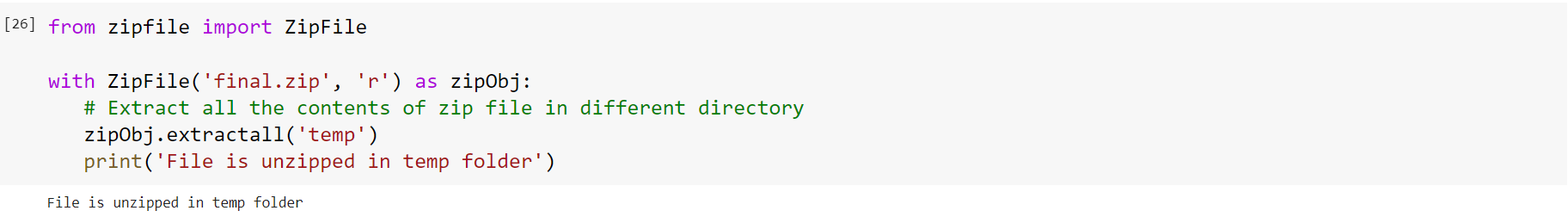
Keys used for encryption algorithms where encrypted using AES by a password and hidden in an image by LSB Steganography. The image used in steganography is cover.png and the outcome image containing hidden keys is encoded\_image.png.

**Encoded\_image.png:**

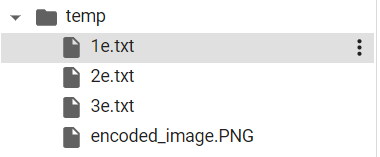


In order to transfer the encrypted files and steganography image ,the files are compressed into a zip folder. Named **Final.zip** file. Later the receiver unzips the compressed file into a folder called temp where he gets 1e.txt,2e.txt,3e.txt and the encoded image.



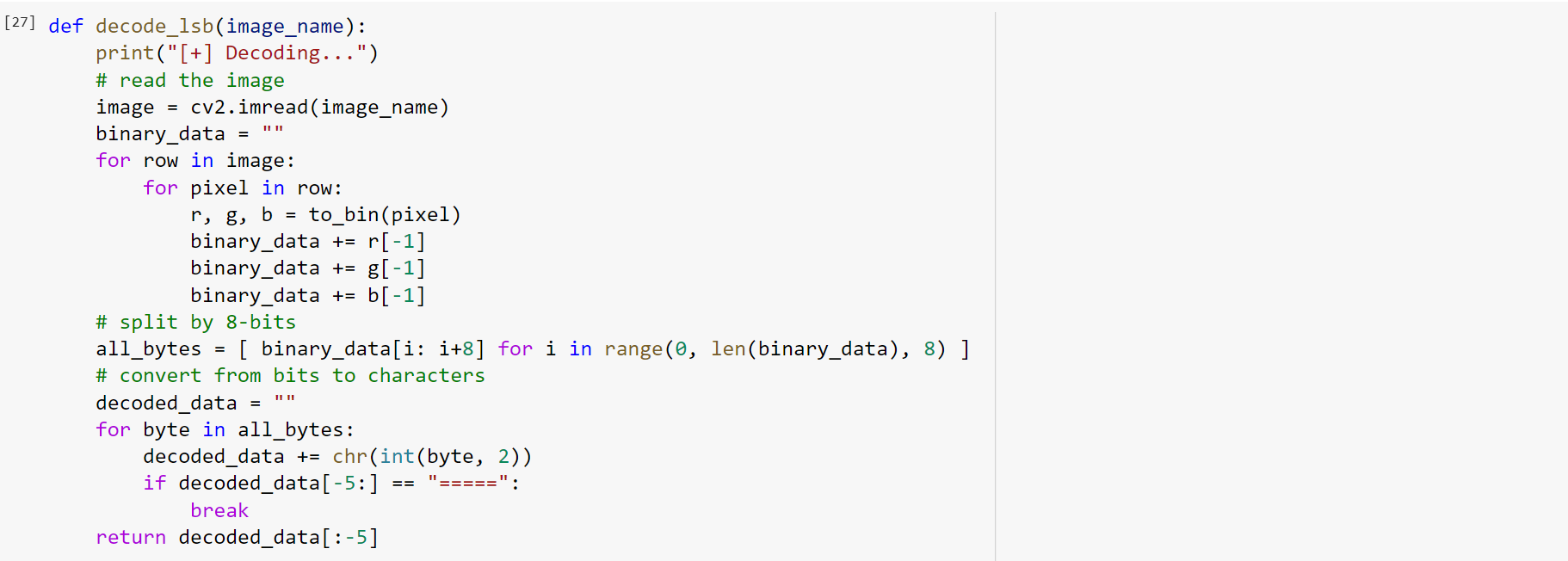


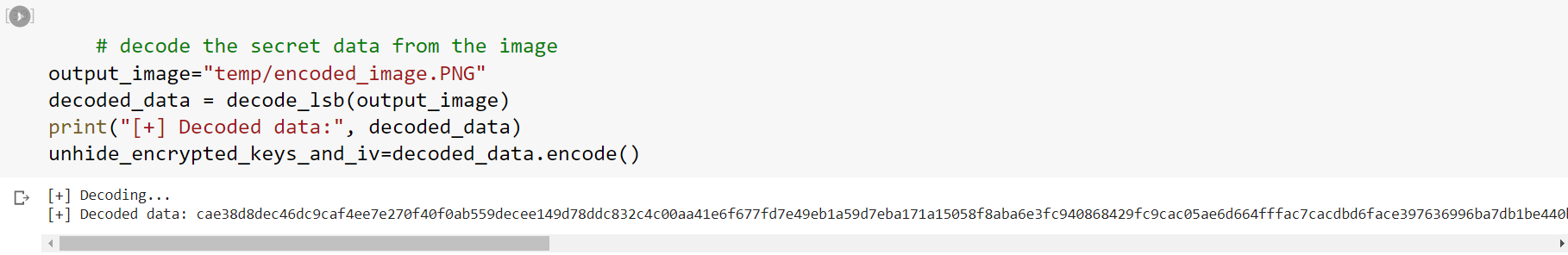
Temp folder after unzipping.



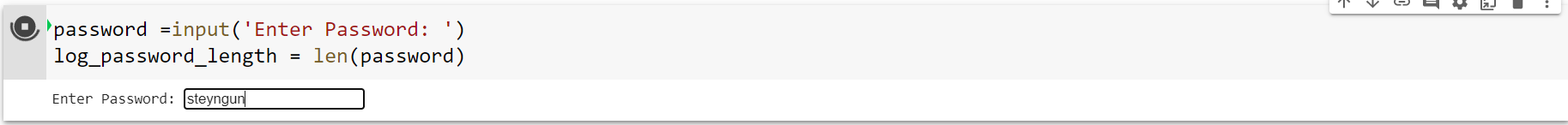
**Module-4:**

1. **key extraction from Stegano Image**





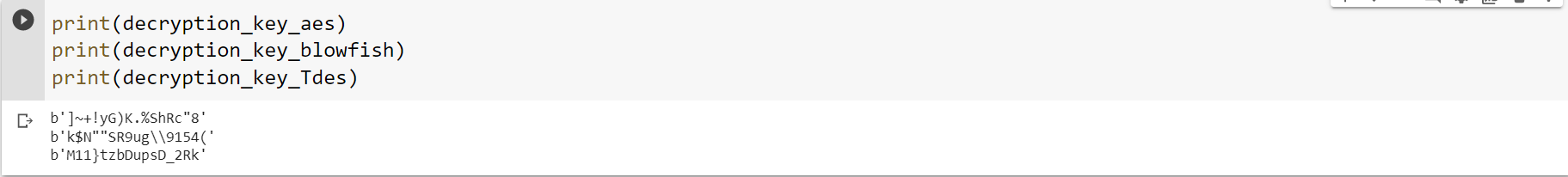
We have obtained the keys from the encoded image using reverse of LSB steganography. The have been encrypted by us using AES encryption using a password.(say steyngun here)



We perform AES Decryption and extract our keys used for decrypt our data files.



The keys used for Blowfish, AES, TripleDES are printed which are used to decrypt the cipher text to plain text.



The extracted keys and the encrypted files unzipped from final.zip which is placed folder temp is used to extract the original data.

**ii)Decryption of Data**

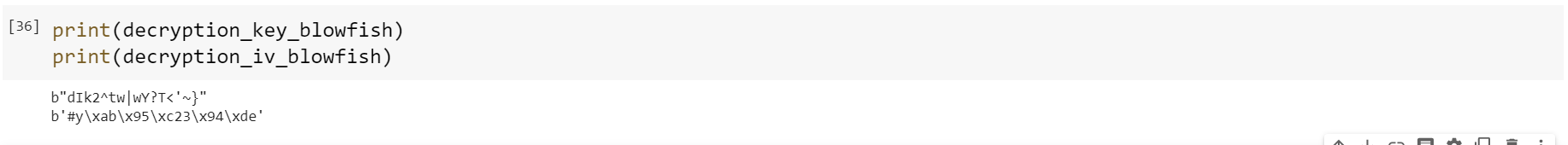
**i)Blowfish-Decryption:**

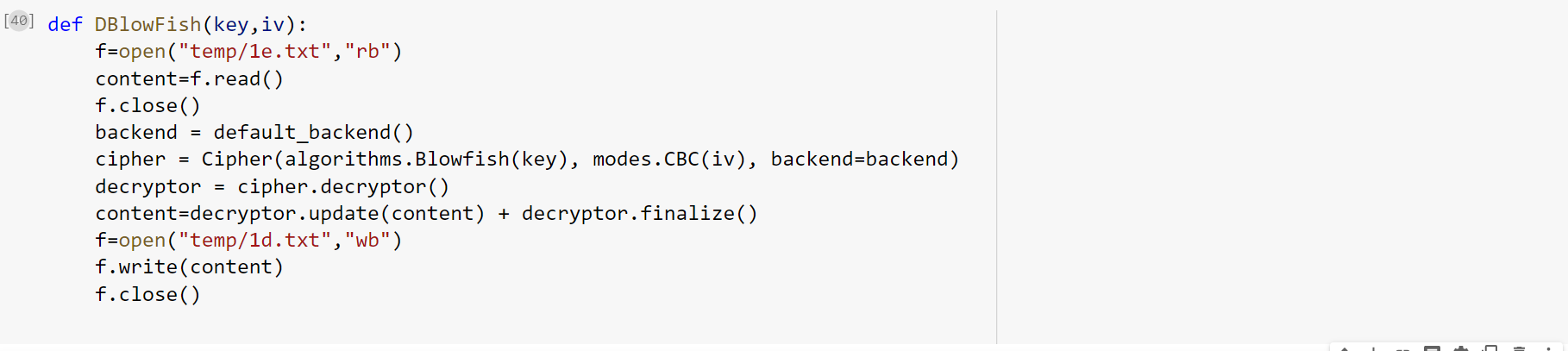
The first part the text file was encrypted by Blowfish algorithm in order convert it into a cipher text. That cipher text received from zipped file is now decrypted using the key that we extracted from reverse LSB steganography.

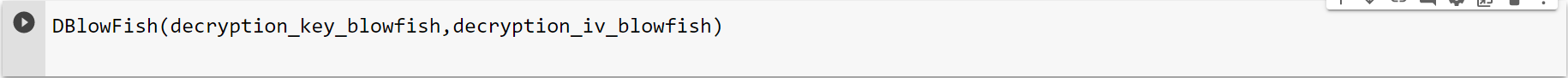
The plaintext P is first Encrypted using the Blowfish Algorithm with a 32 Bit / 64 Bit, KBlowfish . The Key, KBlowfish is generated by the Key Generator and is used for Blowfish Encryption. It is then appended to the List of Keys, L. The Plaintext, P is encrypted to generate Ciphertext C1.

C1 = Blowfish(Plaintext = P, Key = K Blowfish)

The below code prints the keys and initialization vector used for Blowfish encryption that where extracted by us.

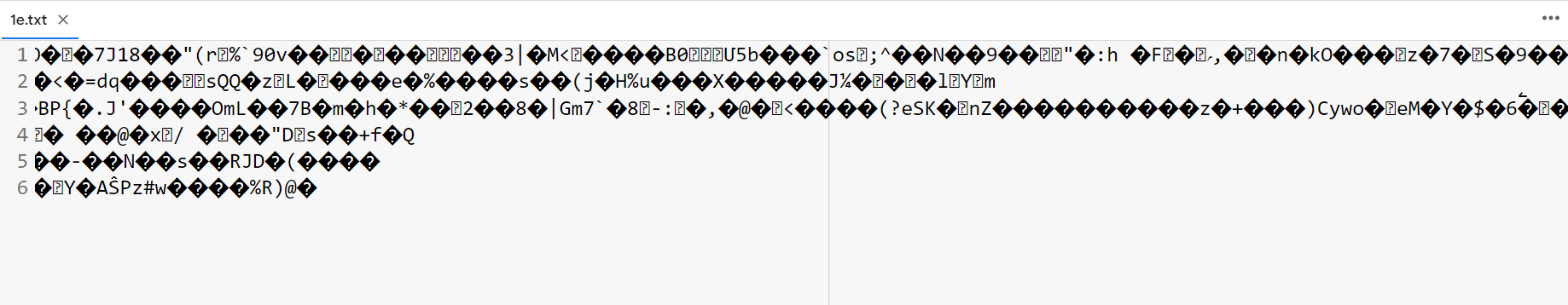






The keys and iv vector used for encryption is used here since Blowfish is a Symmetric key cryptography technology same keys are used for both encryption and decryption. The decrypted plain file obtained from Cipher text is stored as

**Cipher text 1e.txt:**



**1d.txt:**



**ii) Advanced Encryption Standard Decryption:**

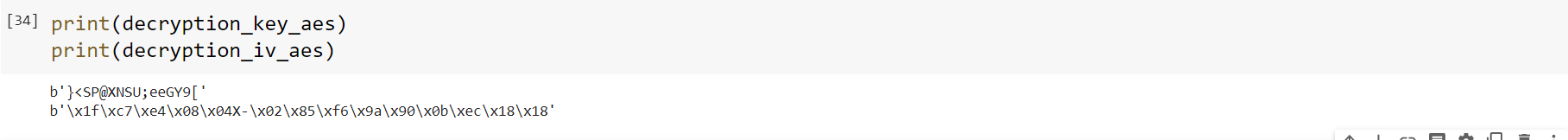
The Ciphertext, C2 is then encrypted using AES-128 Encryption with the 128 Bit, KAES generated by the Key Generator. The Key, KAES generated is appended to the List of Keys, L.

This Steps gives the final encrypted ciphertext C.

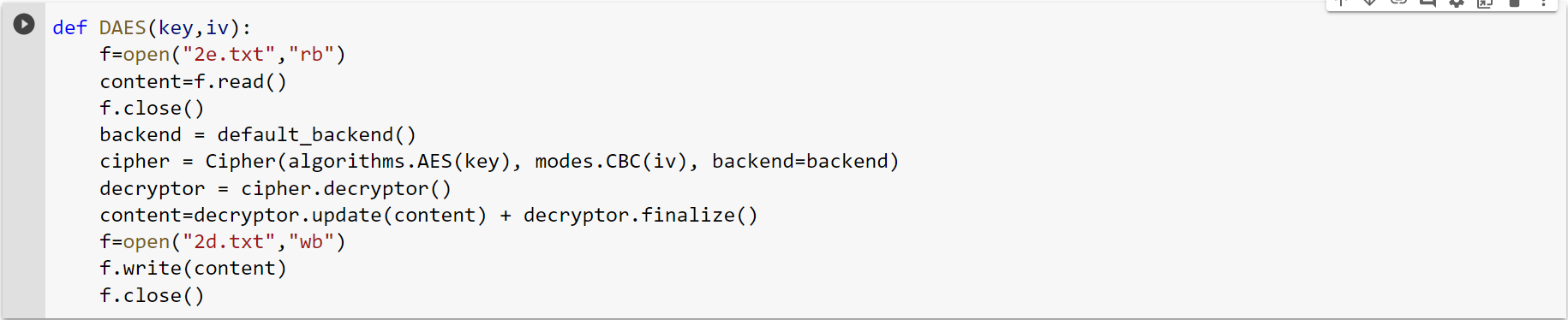
C = AES(Plaintext = C2,Key = K AES)

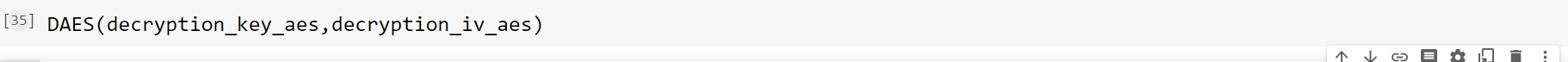
The Second part of the text file was encrypted by AES algorithm in order convert it into a cipher text. That cipher text received from zipped file is now decrypted using the key that we extracted from reverse LSB steganography.

The below code prints the keys and initialization vector used for AES encryption that where extracted by us.



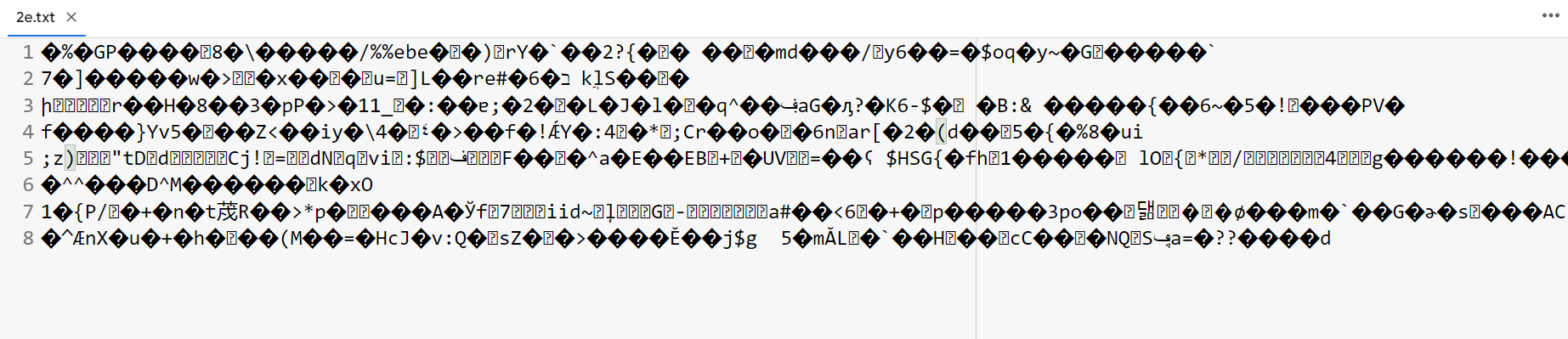
Decrypting AES cryptography algorithm, using extracted keys and iv vectors.





The keys and iv vector used for encryption is used here since Advanced Encryption standard is a Symmetric key cryptography technology same keys are used for both encryption and decryption. The decrypted plain file obtained from Cipher text is stored as 2d.txt

**Cipher text 2e.txt:**



**Plain Decrypted text 2d.txt:**



**iii)Triple Data Encryption standard decryption**

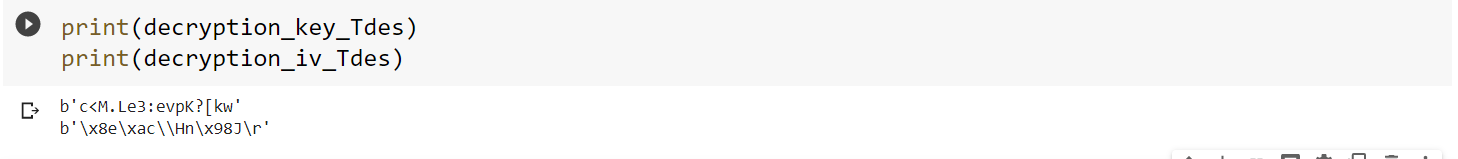
The Ciphertext, C3 is then encrypted using Triple DES Encryption with the 128 Bit, TripleDES generated by the Key Generator. The Key, TripleDES generated is appended to the List of Keys, L.

This Steps gives the final encrypted ciphertext C.

C = TripleDES(Plaintext = C3,Key = TripleDES)

The Third part of the text file was encrypted by TripleDES algorithm in order convert it into a cipher text. That cipher text received from zipped file is now decrypted using the key that we extracted from reverse LSB steganography.

The below code prints the keys and initialization vector used for TripleDES encryption that where extracted by us.



Cipher text 3 is decrypted using Triple Data Encryption Standard decryption cryptography algorithm using extracted keys and Initialization vectors.



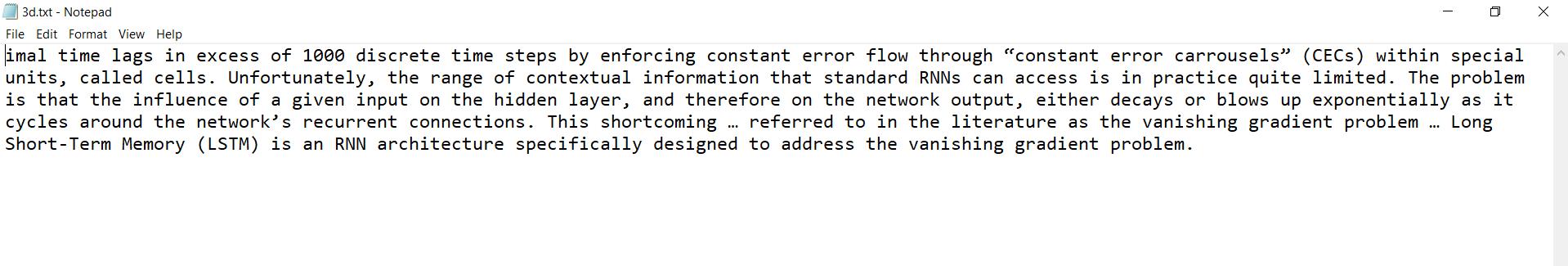


The keys and iv vector used for encryption is used here since Triple Data Encryption standard is a Symmetric key cryptography technology same keys are used for both encryption and decryption. The decrypted plain file obtained from Cipher text 3e.txt is stored as 3d.txt

**Cipher Text 3e.txt:**



**Plain Decrypted text 2d.txt:**

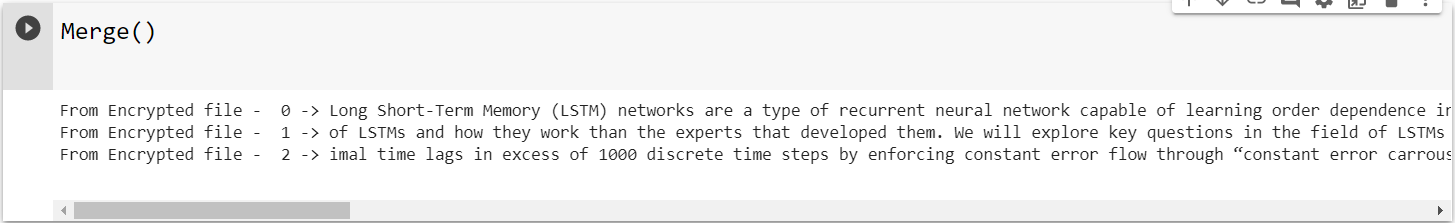


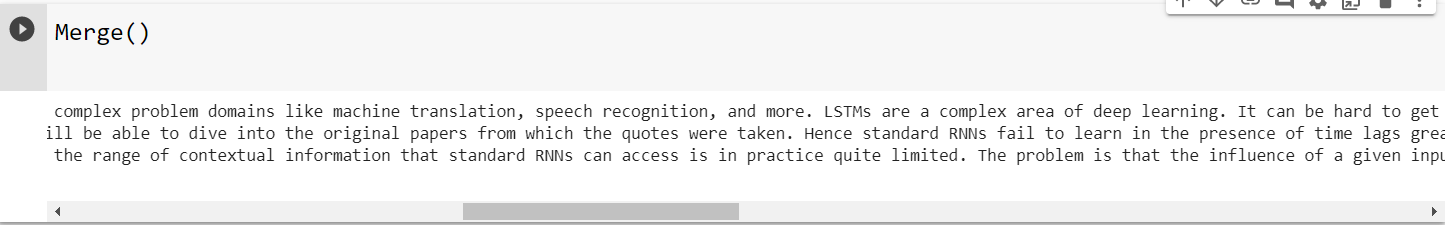
**iii)Output data**

Data decrypted from three cryptography algorithms such as Blowfish, Advanced Encryption standard and Triple Data encryption standard are decrypted into three parts separately since we have applied hybrid cryptography each part is encrypted by separate algorithms now we have to combine those decrypted files so that we can obtain the original text back



Merge function is used to merge all the three parts obtained such as 1d.txt,2d.txt,3d.txt.





**Merged** file is stored into a single text file called final.txt

**Final.txt:**



**Conclusion:**

The proposed cryptosystem uses symmetric cryptography to secure data. The system also introduces a sub-process to encrypt the keys used for encryption before embedding them in an image. The combination of Blowfish—AES-TripleDES has significantly improved the security and also ensured that the drawbacks of the standalone systems are addressed. The system also helps in improving security without the use of keys of larger lengths. We have also seen from the test results that the system is less susceptible to brute force attacks as the decryption time is significantly high. The manyfold expansion of plaintext into ciphertext also helps in ensuring a high level of security. While the system successfully does its intended work, it still required minor improvements for larger adoption**.**