

NAVAL SCIENCE & TECHNOLOGICAL LABRTORY
DRDO, MINISTRY OF DEFENCE, GOVERNMENT OF INDIA,
VIGYAN NAGAR, VISAKHAPATNAM-27

A project report for the period
(15th May 2023 to 15th June 2023)

On

CUSTOMIZED ENCRYPTION ALGORTIHM

Submitted by
PETLA GIRISH

Under the supervision
TANGUTURI RAMU, Sc-'D', GH (IT & Cyber Security)

INDEX

1. INTRODUCTION	3
2. BACKGROUND	5
3. LITERATURE SURVEY	7
3.1 EXISTING METHODS	7
3.1.1 Simple AES Cipher package	7
3.1.2 devglan website:	9
3.1.3 JavalnUse encryption:	10
3.2. DIFFERENT ENCRYPTION ALGORITHMS :	11
1. Rivest-Shamir-Adleman (RSA)	11
2. Triple Data Encryption Standard (DES)	11
3. Blowfish.....	12
4. Two-fish	12
5. Format-Preserving Encryption (FPE)	12
3.3. CRYPTANALYSIS:	13
3.3.1 Cipher text-Only Attack:	13
3.3.2 Known Plaintext Attack:	13
3.3.3 Man-in-the-Middle Attack:	13
3.3.4 Side-Channel Attack:	14
3.3.5 Dictionary Attack.....	14
4. Working of AES algorithm	14
4.1 Add Round Key :	16
4.2 Sub-bytes :	16
4.3 Shift-rows :	16
4.4 Shift columns :	17
5. AES MODES:	18
5.1 ECB Mode	18
5.2 CBC mode	19
5.3 CFB mode	19
5.4 OFB mode.....	19
5.5 CTR mode.....	19
6. PROPOSED WORK :	20

7. PRE-REQUISITES:	21
7.1 Hardware requirements:	21
7.2 Software requirements:	21
7.2.1 PYTHON INSTALLATION :	22
7.2.2 PYCRYPTODOME PACKAGE :	24
7.2.3 VISUAL STUDIO CODE	25
7.2.4 python extension :	27
8. IMPLEMENTATION	28
9. EVALUATION AND RESULT:.....	40
9.1 Errors :	45
1.UTF-8 codec	45
2. Padding is incorrect:	45
10. CONCLUSION	46

1. INTRODUCTION

Today, **cryptography** is an important study of secure data-transfer and communications between people or machines over a network. It provides security by preventing the involvement of any third parties or hackers. Thus, the objective of cryptography is to hide the information so that only an intended recipient can read it. **Encryption** and **decryption** are the techniques that are used in cryptography.

Encryption refers to hiding the original plain text (Message) by transform them into unreadable text called as cipher. In order to unlock the message, both the sender and the recipient have to use a “secret” encryption key. Encryption ensures that information stays private and confidential, whether it's being stored or in transit. Any unauthorized access to the data will only see a scrambled text. The key length, functionality, and features of the encryption system in use determine the effectiveness of the encryption.

Decryption is the process of converting an unreadable cipher text to readable information. This is extracted by the authorized access only. Decryption enables data protection from cyber-attacks. With the secret encryption key, one can decode the cipher text to obtain the original message from sender. The Encryption classified as two types, Symmetric encryption and Asymmetric encryption.

1. Symmetric encryption: This encryption method uses a single secret key for encryption of plaintext to produce the cipher and this same secret key is also used for decryption of cipher. In this type of encryption, the secret key should not expose to public, and it is only known to the respective sender and the recipient.

2. Asymmetric encryption: This encryption method uses two different keys for encryption and decryption known as public key and private key. The public key is shared among users and a private key is not shared and kept secret. By using the public key, the sender can encrypt the data and share to the recipient, with the use of private key the recipient will decrypt the data.

AES (Advanced Encryption Standard) is one of the most widely used algorithms for block encryption. It has been standardized by the NIST (National Institute of Standards and Technology) in 2001, in order to replace DES and 3DES which were used for encryption in that period. The size of an AES block is 128 bits, whereas the size of the encryption key can be 128, 192 or 256 bits. In each of the stages of encryption, four functions are applied: substitution of bytes, permutation, arithmetic operations over finite fields and an XOR operation with the encryption key. The size of the AES block provides efficiency, but also sufficient security. This encryption is the strongest and most robust encryption standard, which makes it impossible to crack.

2. BACKGROUND

The Data Encryption Standard (DES) is a symmetric key block cipher developed by IBM in 1977. DES is used to encrypt digital data before the development of AES. Due its key length of 56 bits, makes it too weak to protect most applications and it is cracked by brute-force using modern tools within less time. In order to overcome the flaws of DES algorithm, the “**Advanced Encryption algorithm**” was introduced. The AES encryption algorithm is a symmetric block cipher created by Joan Daemen and Vincent Rijmen. Since its implementation as a standard, AES has become one of the world's most popular encryption algorithms that use symmetric keys (same key) for encryption and

decryption. The suggested architecture is capable of handling all possible combinations of bit lengths (128,192,256) of data and key.

The earliest forms of data encryption were primitive; some involved scrambling letters in the sentence. This rendered the whole sentence absolutely unreadable and required a lot of time to figure out what the original characters meant. With time and updated technology, people discovered how to crack codes, and the encryption techniques. Therefore it required more protection to ensure that the message was kept private. Today, the data encryption algorithms find extensive application in file transfer Protocol (FTP) transfers and computer systems to offer protected transfers and communications. Even there are number of algorithms that provide secure encryption standards, the **AES-256** (256 bits) is the strongest and complex to being cracked till now, due to its 2^{256} unique combinations which can take millions of years to crack the code. It is most common security technique used for wide variety of applications such as wireless communication, bank transactions, e-commerce, encrypted data storage etc. Some of the drawbacks of AES are it uses too simple algebraic structure and it is hard to implement in software whereas the hardware implementation of AES by FPGA (field programmable gate array) has the advantage of increased better security by pipelined architecture of the AES. It is used in order to increase the throughput of the algorithm. General attacks of the usual encryption methods are Brute force, MITM (man-in-the-middle) etc.

Samsung and other manufacturers of storage devices, which are known as Solid Storage Devices (SSD), use the AES algorithm of 256-bit for saving the data and process. The data that we store on Google drive is an example of the usage of the AES algorithm. The cloud on which the user data is stored and visible on Google uses AES encryption algorithm. This software can be used for public and private, commercial and non-

commercial implementations. Today, AES is a trusted system with widespread adoption. AES libraries have been developed for programming languages including C, C++, Java, JavaScript, and Python making it easy to develop the applications using the AES algorithm.

3. LITERATURE SURVEY

The Advanced Encryption Standard (AES) specifies a FIPS-approved [12.5] cryptographic algorithm that can be used to protect electronic data. This standard may be used by Federal departments and agencies when they determine that sensitive information requires cryptographic protection. In addition, this standard may be adopted and used by non-Federal Government organizations.

Such use is encouraged when it provides the desired security for commercial and private organizations. The algorithm specified in this standard may be implemented in software, firmware, hardware, or any combination. The specific implementation may depend on several factors such as the application, the environment, the technology used, etc. Since, cryptographic security depends on many factors besides the correct implementation.

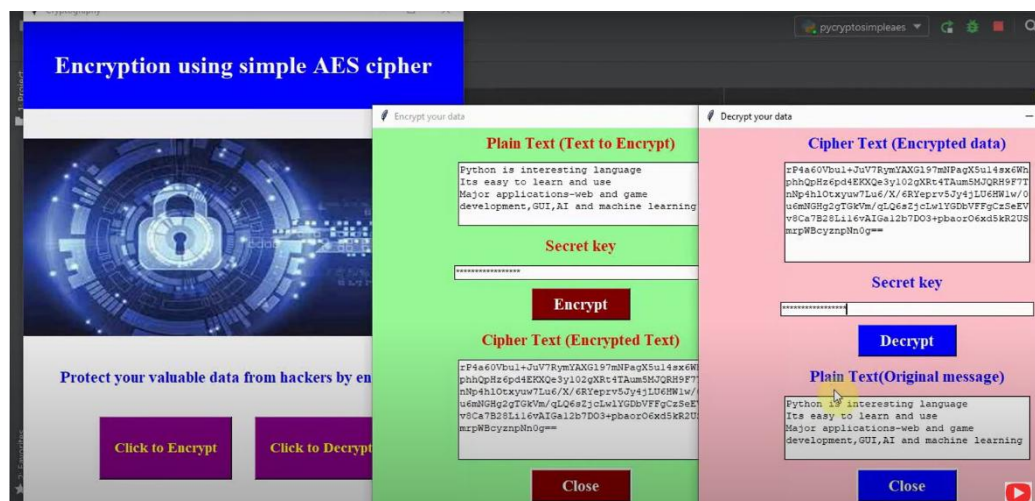
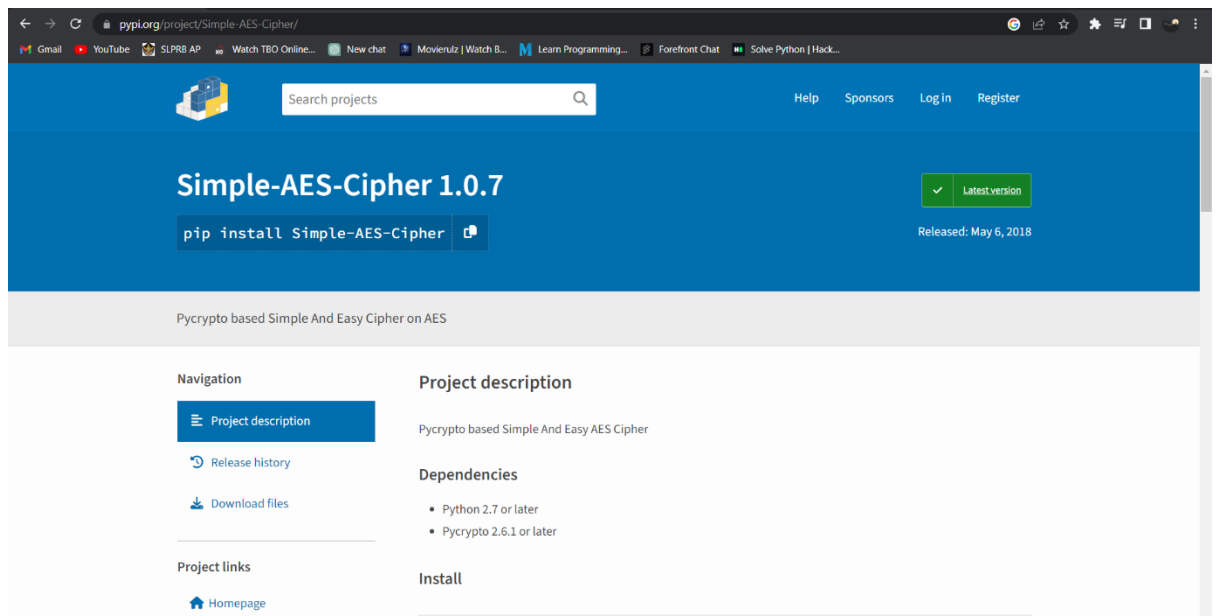
3.1 EXISTING METHODS

There are some existing AES interface techniques and web interfaces which provide the encryption and decryption operations to the user.

3.1.1 Simple AES Cipher package

This package is used in python program to install the cipher module and perform the AES encryption and decryption by generating the interface with the user, for encrypt and decrypt windows, where user can enter the plaintext and get cipher. So, when they

want to decode the cipher, they select the decryption in the interface and paste the cipher in the field and process the plain text. In this method, the user cannot obtain the specific mode of operations like variation in key size and modes like, ECB, CTR, OFB, CFB and they only get the CBC mode of encryption. Here are the visuals of this method.



Advantages:

- Interface is included in this package
- User-friendly
- Python IDE is sufficient to run this application

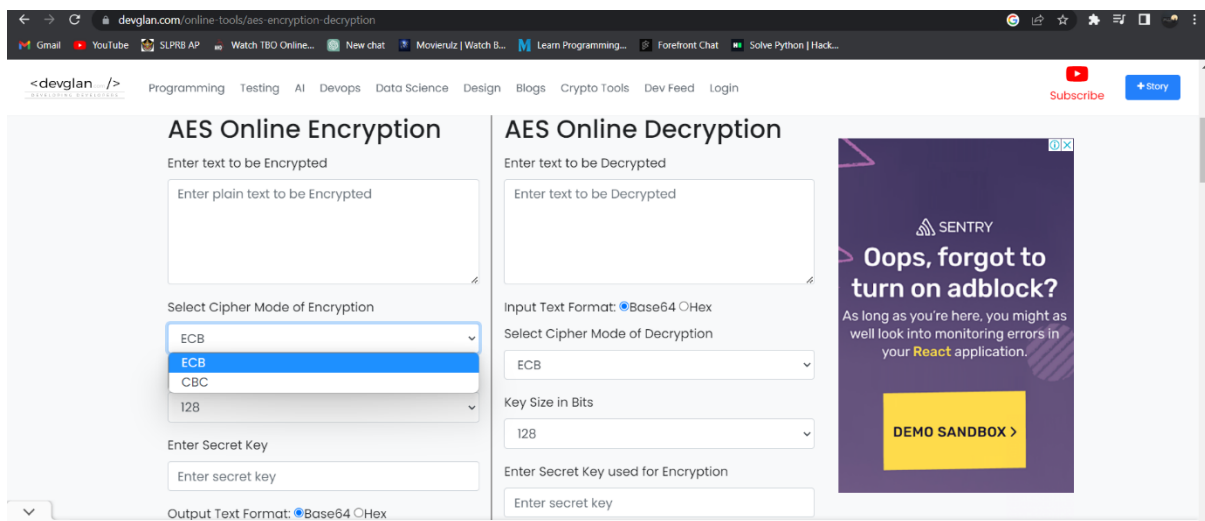
Disadvantages:

- This application doesn't support windows 7
- This application only performs AES ECB mode encryption
- Key-size cannot be modified by the user

3.1.2 devglan website:

<https://www.devglan.com/online-tools/aes-encryption-decryption>

This website works as the tool to perform encryption and decryption operations on the data by the user, there are no logins needed, once they entered this site, they enter the plain text and select the mode ECB or CBC, key size and secret key based on the key size length, if they select the CBC, they have to enter the IV value. Based on the same parameters they can decrypt the data in the decryption box.



The screenshot shows the web interface of the devglan.com AES Online Encryption and Decryption tool. The page is divided into two main sections: 'AES Online Encryption' on the left and 'AES Online Decryption' on the right. The encryption section includes a text input for 'Enter text to be Encrypted', a dropdown for 'Select Cipher Mode of Encryption' (with ECB selected), a dropdown for 'Key Size in Bits' (with 128 selected), and a text input for 'Enter Secret Key'. The decryption section includes a text input for 'Enter text to be Decrypted', a dropdown for 'Select Cipher Mode of Decryption' (with ECB selected), a dropdown for 'Key Size in Bits' (with 128 selected), and a text input for 'Enter Secret Key used for Encryption'. Both sections have radio buttons for 'Input Text Format' (Base64 selected) and 'Output Text Format' (Base64 selected). A SENTRY advertisement is visible on the right side of the page.

In this web interface, the user can only generate the CBC and ECB modes and cannot perform CTR, OFB and CFB modes. There is no padding feature in this interface, so the

user should enter the specific key-size. This website has some minor bugs as it needs to be refreshed after every operation.

3.1.3 JavaInUse encryption:

<https://www.javainuse.com/aesgenerator> (Online)

The screenshot shows the 'AES Encryption' section of the website. It features a navigation bar at the top with links to various technologies. The main interface includes a text input field labeled 'Enter Plain Text to Encrypt' containing the text 'hello, good morning'. Below this is a 'Select Mode' dropdown menu set to 'CBC' and a 'Key Size In Bits' dropdown menu set to '128'. To the right, there is explanatory text about AES modes: CBC (Cipher Block Chaining) and ECB (Electronic Code Book). A note specifies that the input can be 128, 192, or 256 bits, and provides an example of a valid 128-bit key.

The screenshot shows the 'AES Decryption' section of the website. It features a text input field labeled 'Enter Encrypted Text to Decrypt' containing a long hexadecimal string. Below this is an 'Input Text Format' dropdown menu set to 'Hex', a 'Select Mode' dropdown menu set to 'CBC', and a 'Key Size In Bits' dropdown menu set to '128'. To the right, there is explanatory text about AES modes: CBC (Cipher Block Chaining) and ECB (Electronic Code Book). A note specifies that the input can be 128, 192, or 256 bits.

This web interface is used to encrypt and decrypt the data in text file and media files. It takes the plain text and the user has to select only the modes ECB and CBC, with IV value then it generates the cipher. In the same way they can decrypt the cipher with the selected parameters in encryption

From the existing methods of encryption, this project “**Customized Encryption Algorithm**” interface enables the pleasant look and feel to the user and takes the particular values from the users with no extent by providing the five modes of AES namely ECB, CBC, CTR, OFB and CFB, and also with the three key variations, included with the IV values for the CBC, OFB and CFB modes of encryption. In the same way, they can decode the cipher with their certain particulars in order to obtain the plain text.

3.2. DIFFERENT ENCRYPTION ALGORITHMS :

Apart from the AES encryption, other algorithms are developed in the cryptography. Here are some of the encryption algorithms that you can use to safeguard sensitive data for your small business and as well as wide purposes.

1. Rivest-Shamir-Adleman (RSA)

Rivest-Shamir-Adleman is an asymmetric encryption algorithm based on the factorization of the product of two large prime numbers. Only someone with the knowledge of these numbers can decode the message successfully. RSA is often used to secure data transmission between two communication points. However, its efficiency decreases when encrypting large data volumes. Hence, this encryption method is very reliable in transmitting confidential data due to its properties and complexity.

2. Triple Data Encryption Standard (DES)

Triple DES is a symmetric encryption technique and a more advanced form of the Data Encryption Standard (DES) method that encrypts data blocks using a 56-bit key. Triple DES applies the DES cipher algorithm three times to each data block. You can use Triple

DES to encrypt ATM PINs and UNIX passwords. Popular applications such as Microsoft Office and Mozilla Firefox also use Triple DES.

3. Blowfish

This is originally designed to replace the DES, Blowfish is a symmetric algorithm process that divides messages into 64-bit segments and encrypts them individually. Blowfish is known for being fast, flexible, and unbreakable. It's in the public domain, so it's free to use, which makes it even more appealing. You can use Blowfish to safeguard transactions in your e-Commerce platforms. It can also be effective in securing your business's email encryption tools, password management systems, and backup software.

4. Two-fish

Two fish is a symmetric, license-free encryption method that ciphers data blocks of 128 bits. It's a more versatile successor to the Blowfish and Three fish encryption methods. Two fish always encrypts data in 16 rounds regardless of the encryption key size. Though it's slower than AES encryption, you can use Two fish to secure your file and folder security.

5. Format-Preserving Encryption (FPE)

Format-Preserving Encryption is a symmetric algorithm that retains the format as well as length of your data while encoding it. For example, if a customer's phone number is 813-204-9012, FPE will change it to a different one; say 386-192-4019. This way, the format and length remain the same, but the characters are changed to safeguard the

original data. Amazon Web Services (AWS) and Google Cloud, some of the most trusted cloud platforms, use this method for cloud encryption.

6. Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography is a newer type of public-key cryptography that is stronger than RSA encryption. It uses shorter keys, which makes it faster. It's asymmetric, means you can use it in SSL/TLS protocols to strengthen your web communications security.

3.3. CRYPTANALYSIS:

Attacks that occur on these encryption standards are analyzed by the "Cryptanalysis". There are many different ways of attacking the digital data. Some of the approaches:

3.3.1 Cipher text-Only Attack:

The attacker only has access to at least one encrypted message but does not know the plaintext data, any cryptographic key data used, or the encryption algorithm being employed. Intelligence agencies often face this challenge when they've intercepted encrypted communications from a target.

3.3.2 Known Plaintext Attack:

This attack is easier to implement, compared to the cipher text-only attack. With a known plaintext attack, the analyst most likely has access to some or all the cipher text's plaintext. The cryptanalyst's goal is to discover the key the target uses to encrypt the message and use the key to decrypt the message. Once the key is discovered, the attacker can decrypt every message encrypted with that specific key. Known plaintext attacks rely on the attacker finding or guessing all or part of an encrypted message, or alternately, even the original plaintext's format.

3.3.3 Man-in-the-Middle Attack:

This attack occurs when the intruder finds a way to insert themselves into an otherwise secure channel between two parties that want to exchange keys. The cryptanalyst conducts the key exchange with each end-user, with the latter believing that they're conducting the exchange with each other. Thus, the involved parties are none the wiser and are now using a key that the attacker knows.

3.3.4 Side-Channel Attack:

Side-channel attacks rely on information obtained from the physical system employed in the encryption/decryption process. This attack uses data related to the target system's response time to queries or power usage rather than the plaintext that's slated to be encrypted or the cipher text that comes from the encryption process.

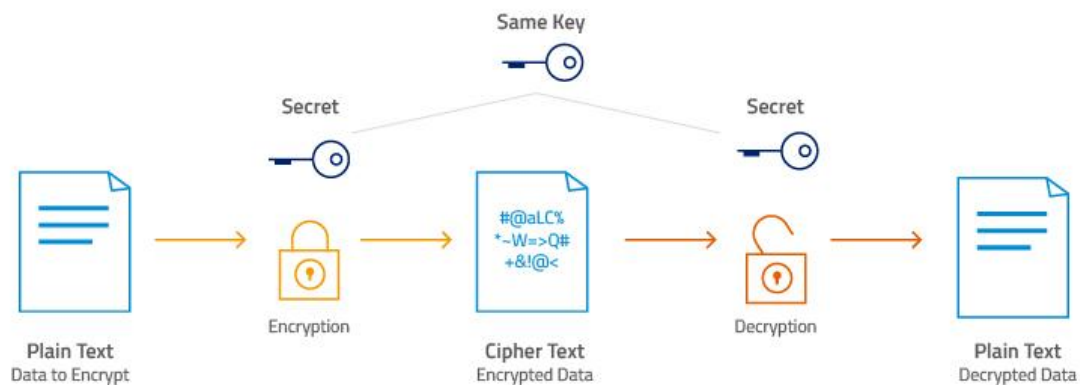
3.3.5 Dictionary Attack

Many people typically use passwords consisting either of easily guessed alphanumeric sequences or actual words. Analysts exploit this habit by encrypting all the words in a dictionary and checking if a resulting hash matches the encrypted password residing in a file format or another password file.

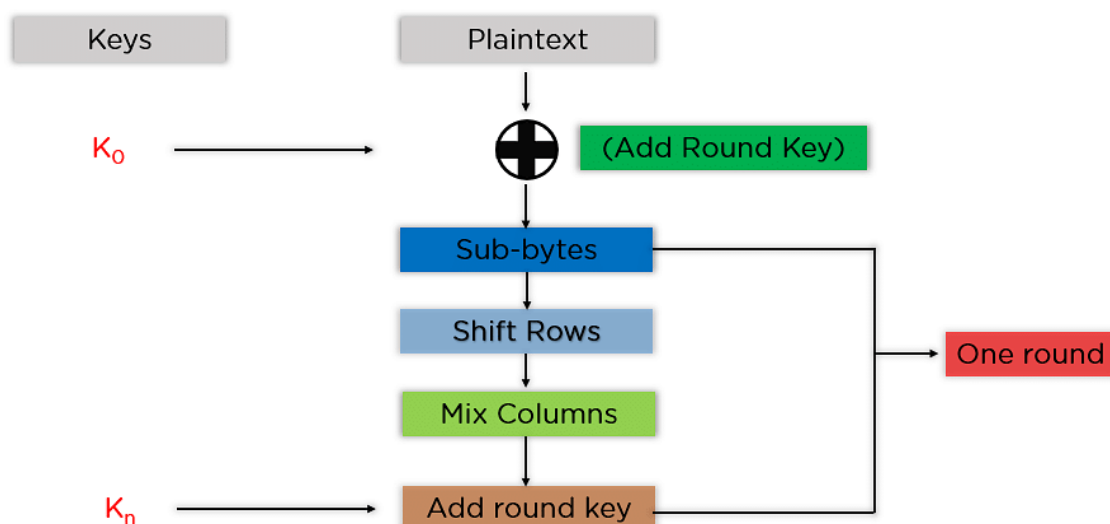
4. Working of AES algorithm

AES is a symmetric key cipher. This means the same secret key is used for both encryption and decryption, and both the sender and receiver of the data need a copy of the key. This key should be confidential and should not expose to public, without the key they cannot decrypt the message. AES uses a 128-bit block size, and input key is also 128-bit. This algorithm performs operations on byte data instead of bit data. So, it treats the 128-bit block size as 16 bytes during the encryption process. The AES algorithm converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the cipher text. AES is based on a substitution-permutation network, also known as an SP network. It consists of a series of linked operations, including replacing inputs with specific outputs (substitutions) and others involving bit shuffling (permutations). All these operations are called as rounds.

The number of rounds to be performed is classified by the key length. The 128-bit key size has ten rounds, the 192-bit key size has 12 rounds, and 256-bit key size has 14.

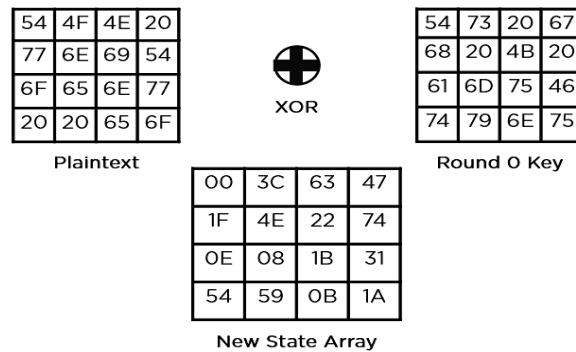


In each round, the following operations are:

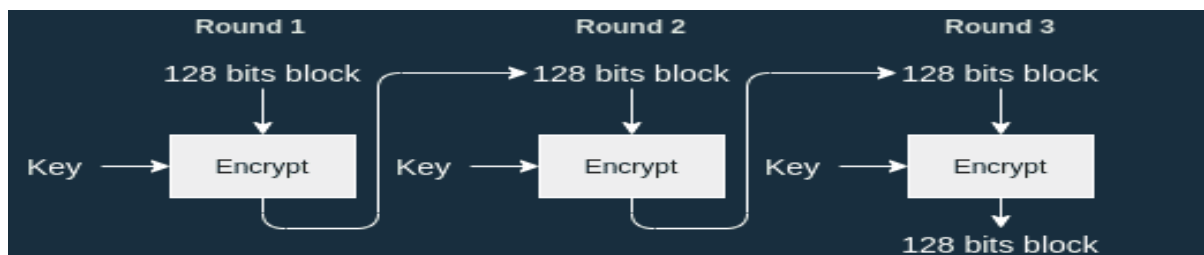


Initially, the input of 16 bytes is substituted by looking up a fixed table (S-box). It is replaced by their values of hex decimal. The data is in a matrix of four rows and four columns (4*4 matrix). There will be a unique sub key per round, (plus one more which will run at the end).

4.1 Add Round Key :



Key Expansion refers to the generating the new key for each round with only one key that may be previous key (represented as K_0, \dots, K_n). The AES key expansion takes the input plain text as four word key for 128bit, six word for 192 bit and eight word for 256bit. This produces a linear array of 44 words. Thus, the key expanded for each round with that previous round key of multiple rounds.



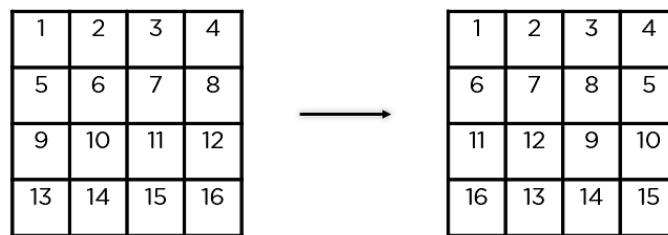
4.2 Sub-bytes:

The transformation technique is a nonlinear byte substitution and also a simple table lookup technique. The implementation of this transformation is complex. Sub Bytes transformation is an S-Box which consist of 16×16 matrix in which the S-Box is used for both forward and inverse transformation. It first runs each byte through a different function to create a new value. Then it uses a table called the S-box to convert the byte into another value using its hexadecimal code.

4.3 Shift-rows :

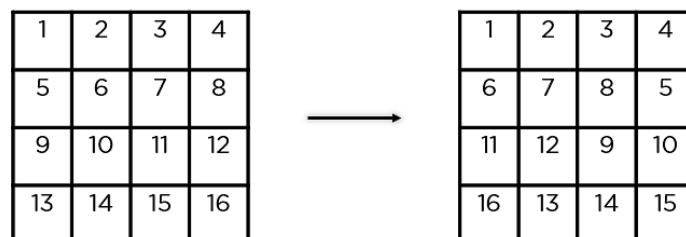
In this operation, the rows in the matrix are shifts with the elements. It swaps the row elements among each other. It skips the first row. It shifts the elements in the second

row to one position to the left. It shifts the elements from the third row to two consecutive positions to the left, and it shifts the last row three positions to the left.



4.4 Shift columns :

This performs operations with the columns of the matrix. In this Mix column operation, each column of the state is multiplied by the known matrix. It's a process which takes in 32 bits of Data and outputs 32 bits of data. Each column is treated as a vector of bytes and is multiplied by a fixed matrix to get the column for the modified State. In the last round, this mix column operation is never performed in order to make the cipher and its inverse more similar in structure.



The state array obtained after the mix column. This state array is now the final cipher text for this particular round. This becomes the input for the next round depending on the key length; the above steps are repeated, until the last round after which you receive the final cipher text.

Final State Array after Round 10

29	57	40	1A
C3	14	22	02
50	20	99	D7
5F	F6	B3	3A

AES Final Output

29 C3 50 5F 57 14 20 F6 40 22 99 B3 1A 02 D7 3A



Ciphertext

This is the entire operations in each rounds of AES algorithm. Thus, the cipher text is produced by encrypting the plain text.

5. AES MODES:

The different modes of AES ^[12.1] commonly used are:

- ECB mode: Electronic Code Book mode
- CBC mode: Cipher Block Chaining mode
- CFB mode: Cipher Feedback mode
- OFB mode: Output Feedback mode
- CTR mode: Counter mode

5.1 ECB Mode

The ECB (Electronic Code Book) mode is the simplest of all. Due to its weaknesses, it is generally not recommended, the plaintext is divided into blocks as the length of the block of AES, 128. So this mode needs to pad data until it is same as the length of the block. Then every block will be encrypted with the same key and same algorithm. So if we encrypt the same plaintext, we will get the same cipher text. There is a high risk in this mode and the plaintext and cipher text blocks are a one-to-one correspondence. In the database encryption, we can use ECB to encrypt the tables, index, temp files, and system catalogs. But with the issues of security, we don't suggest to use ECB.

5.2 CBC mode

The CBC (Cipher Block Chaining) mode provides encryption by using an initialization vector – IV. The IV has the same size as the block that is encrypted. The IV is a random number, not a nonce. The plaintext is divided into blocks and needs to add padding data. First, we will use the plaintext block xor with the IV. Then CBC will encrypt the result to the cipher text block. In the next block, we will use the encryption result to xor with plaintext block until the last block. In this mode, even if we encrypt the same plaintext block, we will get a different cipher text block. If a plaintext or cipher text block is broken, it will affect all following block.

5.3 CFB mode

The CFB (Cipher Feedback) mode of operation allows the block encryptor to be used as a stream cipher. It also needs an IV. First, CFB will encrypt the IV, and then it will xor with plaintext block to get cipher text. Then we will encrypt the encryption result to xor the plaintext. Because this mode will not encrypt plaintext directly, it just uses the cipher text to xor with the plaintext to get the cipher text. So in this mode, it doesn't need to pad input. Similar to the CBC, so if there is a broken block, it will affect all following block..

5.4 OFB mode

The OFB (Output Feedback) mode of operation also enables a block encryptor to be used as a stream encryptor. It also doesn't need padding. In this mode, it will encrypt the IV first with the key. Then it will use the encrypted result to xor with the plaintext to get cipher text. It is different from CFB, it always encrypts the IV. It will not be affected by the broken block.

5.5 CTR mode

The CTR (Counter) mode of operation, an input block to the encryptor (Encrypt), i.e. as an IV, the value of a counter (Counter, Counter + 1... Counter + N - 1) is used. It also is a stream encryptor. The counter has the same size as the used block. The XOR operation with the block of plain text is performed on the output block from the encryptor. All

encryption blocks use the same encryption key. In this mode, it will not be affected by the broken block. It is very like OFB. But CTR will use the counter to be encrypted every time instead of the IV. So if you get counter value, you can encrypt/decrypt data

Applications of AES:

- Wireless Security: Wireless networks are secured using the Advanced Encryption Standard to authenticate routers and clients. Wi-Fi networks have firm and complete security systems based on this algorithm and are now in everyday use.
- Web Browsing: AES plays a huge role in securing website server authentication from both client and server end. This algorithm also helps in SSL/TLS encryption protocols to always browse with the utmost security and privacy.
- General File Encryption: Not only in corporate necessities, AES is also used to transfer files between associates in an encrypted format. The encrypted information can extend to communications like text messages, family pictures, legal docs, etc.

The drawbacks of AES algorithm are:

- AES need more time to encrypt large size files
- ECB mode of encryption in AES can be attacked by brute force
- AES in counter mode is complex to implement in software taking both performance and security into considerations.

6. PROPOSED WORK :

In this project, for the encryption of text data and sending the secret codes between any entities. In this scenario, AES algorithm plays a key role for protected transfer of messages as we see in apps like Facebook, WhatsApp, etc. To demonstrate how the data is secured in AES where by converting the data into scrambled text known as cipher and that cipher is then decoded to get the original data, this project is designed using python language to perform the AES (advanced encryption standard) algorithm for encrypting and decrypting the text messages that are given by the user. With the improvement of

the programming languages, we can develop the crypto operations by some in-built packages, by using those packages we can get the working of the algorithm. Here in this project, the pycryptodome and pycrypto packages and other modules to prepare the effective functionalities.

The TKinter GUI (graphical user-interface) toolkit is used to create the interface with the user by creating the window with particular element. TKinter is the module of python which is used for GUI. The agenda of this project is to prevent the eavesdropping, cyber-attacks and involvement of any third-parties. This encryption standard can provide security in all aspects and solves the issues. The cracking of this algorithm is impossible as it needs years of time and super systems to obtain the all-possible digits to uncover the cipher text to achieve the data. Most commonly AES 256-bit mode is used, but in this project, we can also alter the different modes for different purpose.

In this project, we can get the desired and customized encryption mode with different key sizes and IV values by the user and encrypts the message. After the program executed, it displays the window with the mode encrypt or decrypt, input text, IV value and special key fields, with the selection of key sizes and different modes, hereby after selecting all the required fields, they have to click the process button, then it displays the cipher-text in the text area. When they want to decode a cipher text, they have to opt the decrypt mode and enter the cipher text as input in input field and make sure the key and IV, key size and modes have to be same when they are encrypted, if any values changes, then it displays the errors in the output area and specifies the padding or incorrect key size, etc. to the user.

7. PRE-REQUISITES:

7.1 Hardware requirements:

- Minimum 32 bit OS
- RAM 4GB
- Processor INTEL i3

7.2 Software requirements:

- Python 3.2

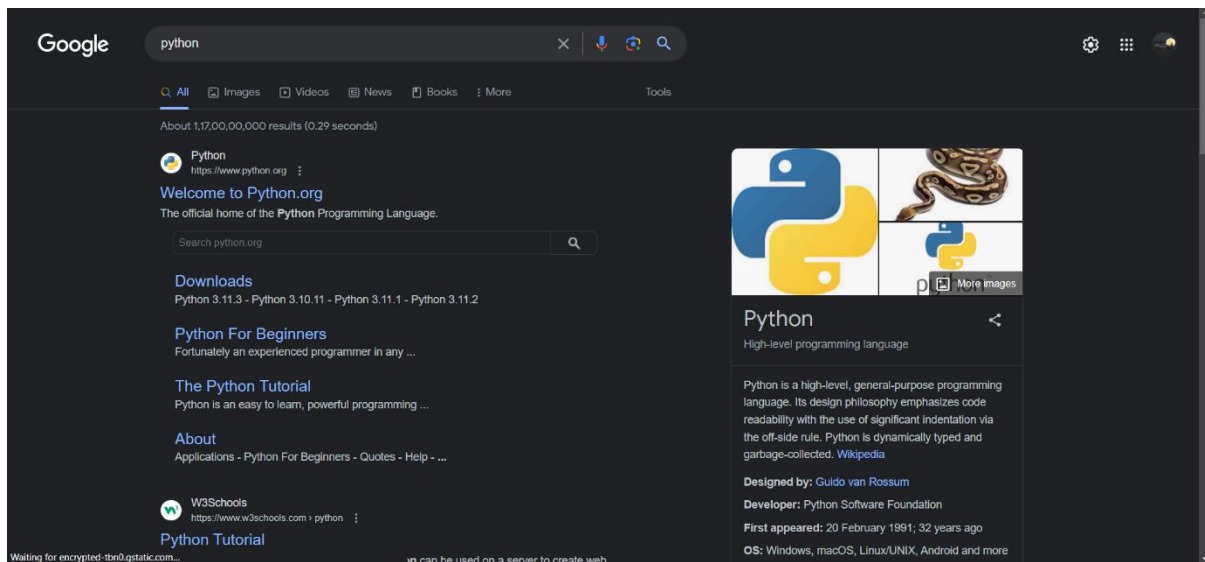
- Pycryptodome package
- Visual studio code

To execute this project in PC, you need to install the python interpreter in your machine.

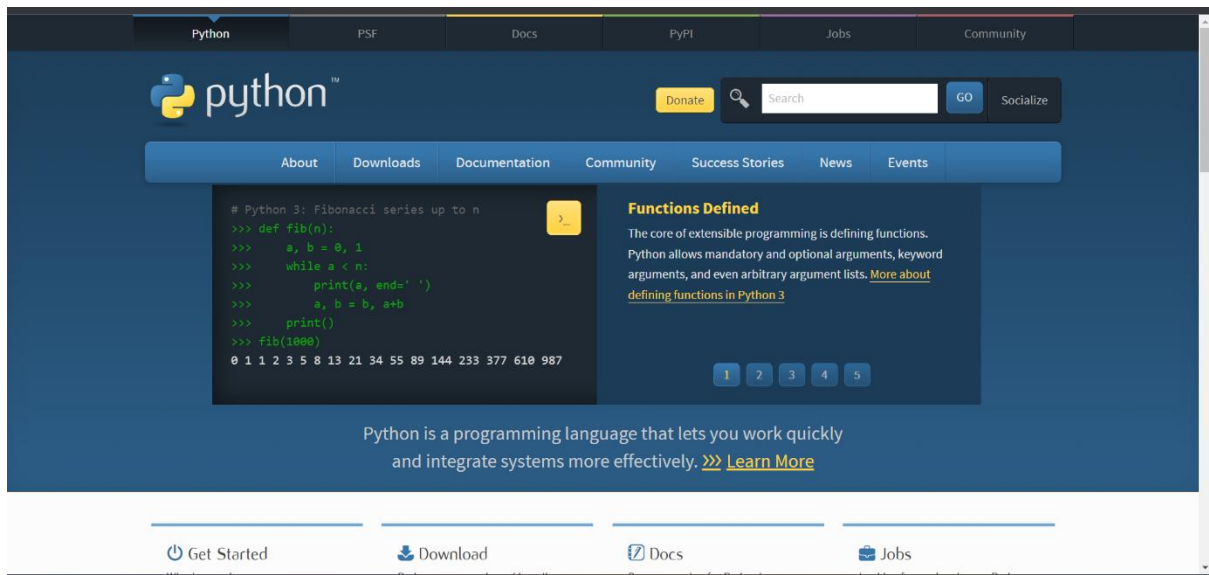
Python language is well known for the development of apps, games, and data analytics.

7.2.1 PYTHON INSTALLATION :

For the installation of python, go to website: <https://www.python.org/>



This is the official website of the python language. Here you can get the detailed documentation, history and other information about the language. After entering to the website, click downloads and it will take you to the list of versions and its release dates.



3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537

Looking for a specific release?

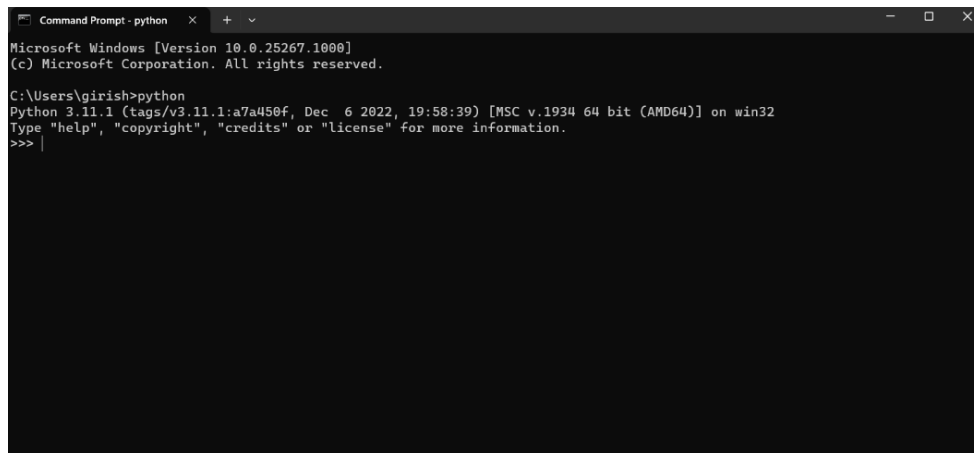
Python releases by version number:

Release version	Release date	Click for more	
Python 3.10.12	June 6, 2023	Download	Release Notes
Python 3.11.4	June 6, 2023	Download	Release Notes
Python 3.7.17	June 6, 2023	Download	Release Notes
Python 3.8.17	June 6, 2023	Download	Release Notes
Python 3.9.17	June 6, 2023	Download	Release Notes
Python 3.10.11	April 5, 2023	Download	Release Notes
Python 3.11.3	April 5, 2023	Download	Release Notes

[View older releases](#)

Depending on the specifications of your system, download the specific release and version of the python. After the download, install the python by clicking the setup.exe file and setting the directory of the path.

To check whether the python is installed or not, go to the command prompt and type the following “python “. If it is successfully downloaded, then it displays the version of the python installed in your system.



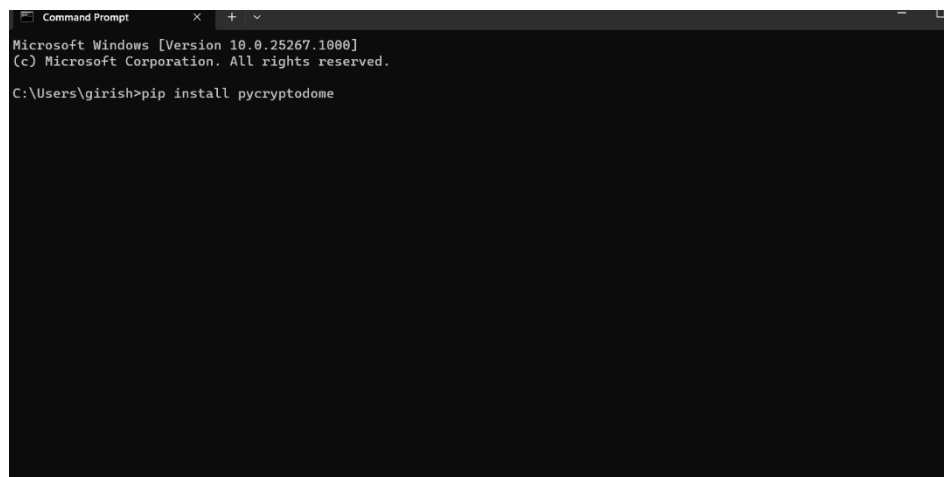
```
Command Prompt - python
Microsoft Windows [Version 10.0.25267.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\girish>python
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

Once the installation is successful, then you can run mini python codes or programs in your machine, by saving the files as filename.py.

7.2.2 PYCRYPTODOME PACKAGE :

Now, we have to download the other package known as pycryptodome^[12.2] in our pc. Pycryptodome is a python package, which includes the cryptography methods AES, block cipher and many other key words. We can install the package by typing the following command in the command prompt “pip install pycryptodome “



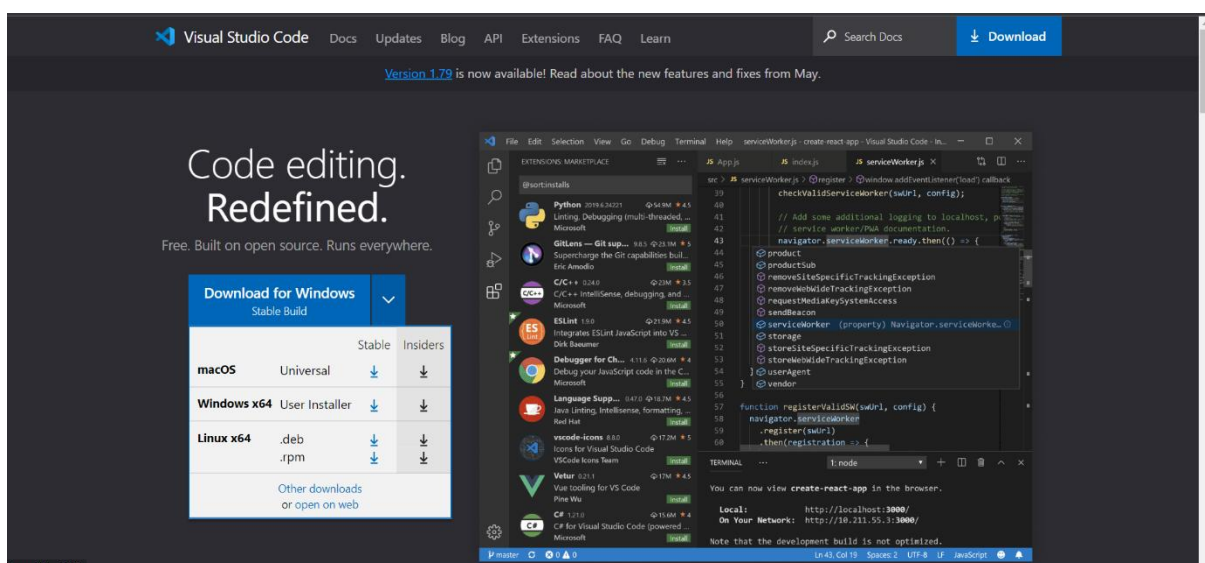
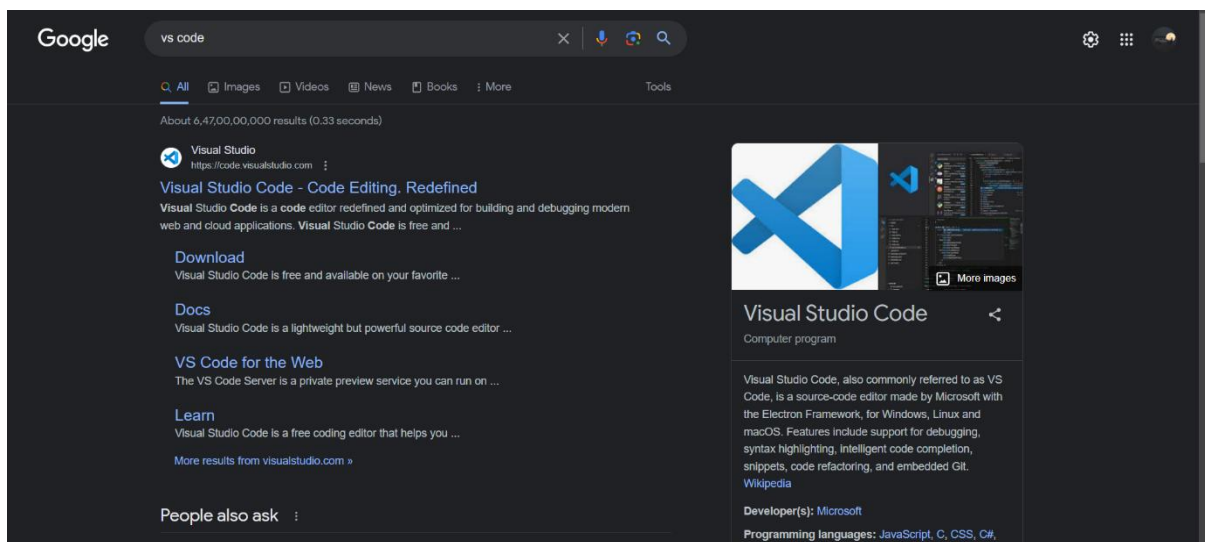
```
Command Prompt
Microsoft Windows [Version 10.0.25267.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\girish>pip install pycryptodome
```

Note that this command doesn't work when the system requirements doesn't match with the package requirements which results to error in installing. By this command, the pycryptodome package is installed in your machine, which is used to import to the program while execution. This command works with the windows OS only.

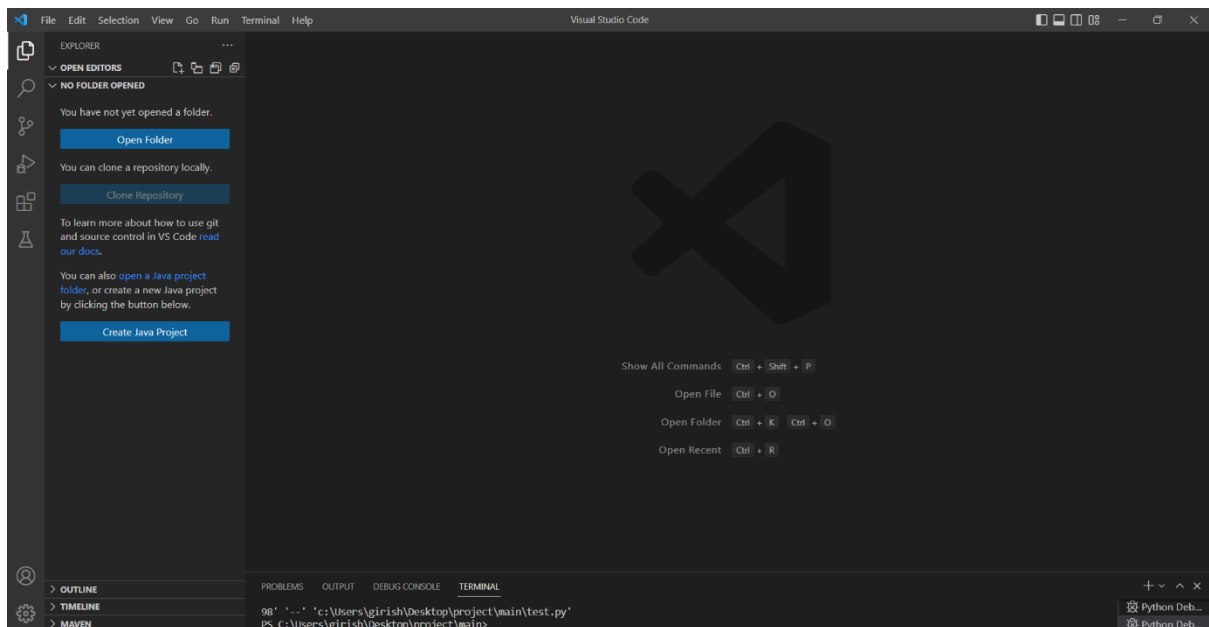
7.2.3 VISUAL STUDIO CODE

The final step is that you need to install the python platform in your system. In internet, there are many IDEs that follows python program like PyCharm, notebook ++, Jupiter, etc. For this project, the VS code (visual studio) software should be installed in your pc. Visual studio is an integrated development environment from Microsoft. It is used to develop computer programs including websites, web apps, web services and mobile apps. It is easier to manipulate the program. So, we must install the VS code software from the browser by visiting the link <https://code.visualstudio.com/>



Based on the specifications of your computer or laptop, you can select the version and type of installation of the software. After you click the download button, the software starts to download in your system. Please wait until the download is completed successfully

After the download is completed, open the downloaded file location and find the setup.exe file and double click it. It then installs the VS code app in your system. After the installation, open the VS code app in your pc.



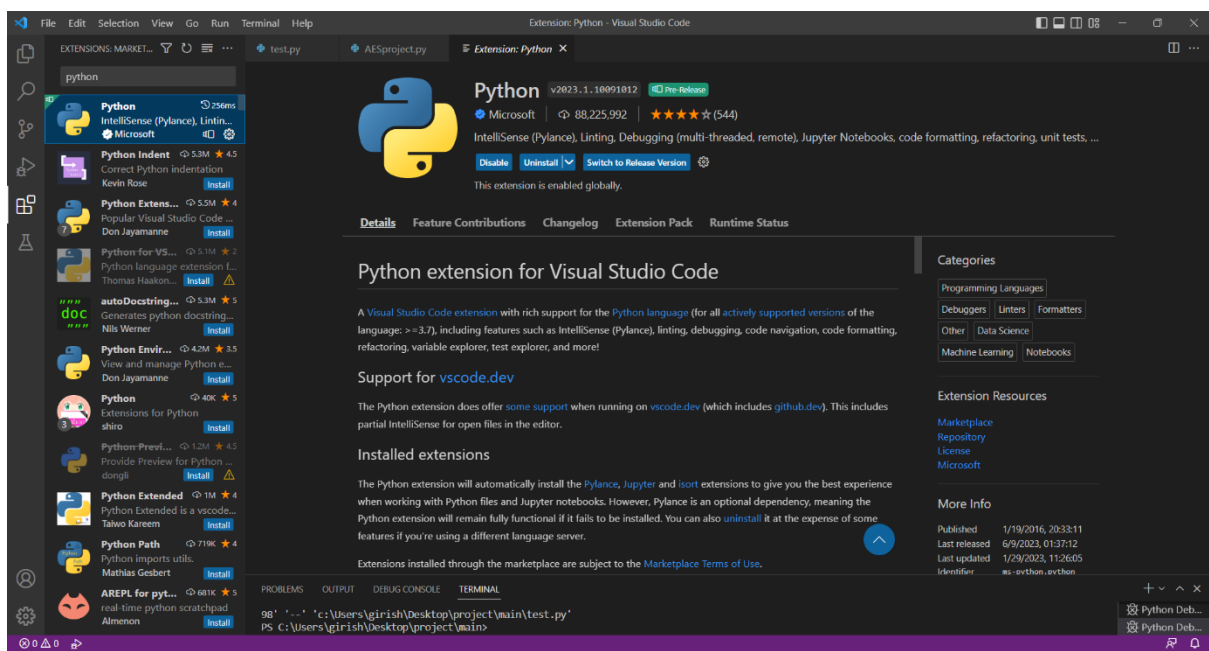
Then you need to open the file of where the program is placed in the directory, by clicking the open folder option.

We can execute many other programming languages in this platform, by downloading the extensions from the extension side bar of the application. For this project, we need to install the python extension to get the required packages and modules. Hence, we have already installed the python interpreter separately, this extension utilizes the python functionalities.

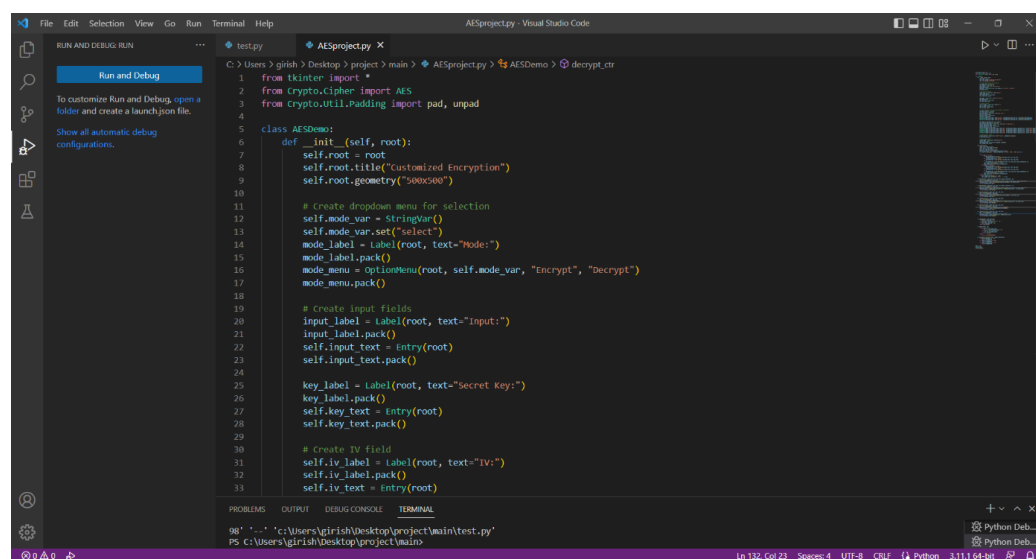
Open the program file from the open folder option, and save the file by clicking ctrl + s and select the suitable location in the system, after the file is saved, then it is ready to executed.

Click the F5 key and it will run the program and raise the error if any. If there are no errors then it is ready to execute and runs the code.

7.2.4 python extension :



CODE ON VISUAL STUDIO :



8. IMPLEMENTATION

```
from tkinter import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

class AESDemo:
    def __init__(self, root):
        self.root = root
        self.root.title("AES Encryption/Decryption")
        self.root.geometry("400x400")

        # dropdown for selection
        self.mode_var = StringVar()
        self.mode_var.set("Encrypt")
        mode_label = Label(root, text="Mode:")
        mode_label.pack()
        mode_menu = OptionMenu(root, self.mode_var, "Encrypt", "Decrypt")
        mode_menu.pack()

        #input fields
        input_label = Label(root, text="Input:")
        input_label.pack()
        self.input_text = Entry(root)
        self.input_text.pack()

        key_label = Label(root, text="Secret Key:")
        key_label.pack()
        self.key_text = Entry(root)
        self.key_text.pack()

        #IV field
        self.iv_label = Label(root, text="IV:")
        self.iv_label.pack()
```

```

self.iv_text = Entry(root)
self.iv_text.pack()

# checkboxes for key size and mode of operation
self.key_size_var = IntVar()
self.key_size_var.set(128)
key_size_label = Label (root, text="Key Size:")
key_size_label.pack()
key_size_frame = Frame(root)
key_size_frame.pack()
Radiobutton(key_size_frame, text="128-bit", variable=self.key_size_var,
value=128).pack(side=LEFT)
Radiobutton(key_size_frame, text="192-bit", variable=self.key_size_var,
value=192).pack(side=LEFT)
Radiobutton(key_size_frame, text="256-bit", variable=self.key_size_var,
value=256).pack(side=LEFT)

self.mode_of_operation_var = StringVar()
self.mode_of_operation_var.set("ECB")
mode_of_operation_label = Label(root, text="Mode of Operation:")
mode_of_operation_label.pack()
mode_of_operation_frame = Frame(root)
mode_of_operation_frame.pack()
Radiobutton(mode_of_operation_frame, text="CBC",
variable=self.mode_of_operation_var, value="CBC").pack(side=LEFT)
Radiobutton(mode_of_operation_frame, text="OFB",
variable=self.mode_of_operation_var, value="OFB").pack(side=LEFT)
Radiobutton(mode_of_operation_frame, text="CTR",
variable=self.mode_of_operation_var, value="CTR").pack(side=LEFT)
Radiobutton(mode_of_operation_frame, text="ECB",
variable=self.mode_of_operation_var, value="ECB").pack(side=LEFT)
Radiobutton(mode_of_operation_frame, text="CFB",
variable=self.mode_of_operation_var, value="CFB").pack(side=LEFT)

```

```

# process button
process_button = Button(root, text="Process", command=self.process)
process_button.pack()

# output area
output_label = Label(root, text="Output:")
output_label.pack()
self.output_text = Text(root, height=5, width=30)
self.output_text.pack()

def process(self):
    mode = self.mode_var.get()
    input_text = self.input_text.get()
    key = self.key_text.get()
    key_size = self.key_size_var.get()
    mode_of_operation = self.mode_of_operation_var.get()
    iv = self.iv_text.get() if mode_of_operation in ["CBC", "OFB", "CFB"] else None

    try:
        if mode == "Encrypt":
            if mode_of_operation == "ECB":
                encrypted_text = self.encrypt_ecb(input_text, key, key_size)
            elif mode_of_operation == "CTR":
                encrypted_text = self.encrypt_ctr(input_text, key, key_size)
            else:
                encrypted_text = self.encrypt(input_text, key, key_size, mode_of_operation, iv)
            self.output_text.delete("1.0", END)
            self.output_text.insert(END, encrypted_text)
        elif mode == "Decrypt":
            if mode_of_operation == "ECB":
                decrypted_text = self.decrypt_ecb(input_text, key, key_size)
            elif mode_of_operation == "CTR":

```

```

        decrypted_text = self.decrypt_ctr(input_text, key, key_size)
    else:
        decrypted_text = self.decrypt(input_text, key, key_size, mode_of_operation, iv)
    self.output_text.delete("1.0", END)
    self.output_text.insert(END, decrypted_text)

except Exception as e:
    self.output_text.delete("1.0", END)
    self.output_text.insert(END, "Error: " + str(e))

def encrypt(self, input_text, key, key_size, mode_of_operation, iv):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, self.get_mode_of_operation(mode_of_operation),
IV=self.pad_iv(iv))
    encrypted_bytes = cipher.encrypt(pad(input_text.encode(), AES.block_size))
    return encrypted_bytes.hex()

def decrypt(self, input_text, key, key_size, mode_of_operation, iv):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, self.get_mode_of_operation(mode_of_operation),
IV=self.pad_iv(iv))
    decrypted_bytes = unpad(cipher.decrypt(bytes.fromhex(input_text)),
AES.block_size)
    return decrypted_bytes.decode()

def encrypt_ecb(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_ECB)
    encrypted_bytes = cipher.encrypt(pad(input_text.encode(), AES.block_size))
    return encrypted_bytes.hex()

def decrypt_ecb(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_ECB)

```

```

        decrypted_bytes = unpad(cipher.decrypt(bytes.fromhex(input_text)),
AES.block_size)

    return decrypted_bytes.decode()

def encrypt_ctr(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_CTR, nonce=b'00000000')
    encrypted_bytes = cipher.encrypt(input_text.encode())
    return encrypted_bytes.hex()

def decrypt_ctr(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_CTR, nonce=b'00000000')
    decrypted_bytes = cipher.decrypt(bytes.fromhex(input_text))
    return decrypted_bytes.decode()

def pad_key(self, key, key_size):
    if len(key) <key_size // 8:
        key = key.ljust(key_size // 8, '\0')
    elif len(key) >key_size // 8:
        key = key[:key_size // 8]
    return key.encode()

def pad_iv(self, iv):
    if iv is not None:
        if len(iv) <AES.block_size:
            iv = iv.ljust(AES.block_size, '\0')
        elif len(iv) >AES.block_size:
            iv = iv[:AES.block_size]
        return iv.encode()
    else:
        return b'0000000000000000'

```



```

def get_mode_of_operation(self, mode_of_operation):
    if mode_of_operation == "CBC":
        return AES.MODE_CBC
    elif mode_of_operation == "OFB":
        return AES.MODE_OFB
    elif mode_of_operation == "CFB":
        return AES.MODE_CFB

root = Tk()
AESDemo(root)
root.mainloop()

```

The above program executes the AES algorithm using the Tkinter library in Python. It allows users to perform AES encryption and decryption using different modes of operation and key sizes. The program takes user input and the mode of operation, input text, secret key, and, when applicable, the initialization vector (IV). It then performs the encryption or decryption operation based on the user's selection and displays the output in the GUI.

The packages, methods, and functions used in this program are :

```

from tkinter import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

```

In order to perform some tasks, we have to import the packages Tkinter, Crypto.cipher and Crypto.util. Padding.

Tkinter: This line imports the Tkinter library ^[12.3], which provides the necessary classes and methods for creating the GUI application. This includes the text areas and fields,

checkboxes radio buttons, etc, and those are used to create the interface with the user, and they can be modified by the programmer to achieve their desired results.

Crypto. Cipher: This line imports the AES module from the Crypto. Cipher package, which is part of the PyCryptodome library. It allows us to perform AES encryption and decryption operations.

Pycryptodome package should be installed in the system to run this program, one can install this by entering the command in the cmd prompt: [pip install pycryptodome].Pycryptodome is a self-contained Python package of low-levelcryptographic primitives and it supports the python version 2.7 ,3.5 and newer. All modules are installed under the Crypto package.

Crypto.Util. Padding import pad, unpad: This line imports the pad and unpad functions from this module. These functions are used to add padding to the input text before encryption and remove padding from the decrypted output. Hence it helps to take input as the multiple of block size by adding the values to input.

```
class AESDemo:
    def __init__(self, root):
        self.root = root
        self.root.title("AES Encryption/Decryption")
```

The `AESDemo` class: This is the main class that represents the GUI application. It has the `__init__ (self, root)` the constructor method initializes the GUI window with a title, size, and different GUI elements such as labels, dropdown menus, checkboxes, input fields, and buttons.

In this function the whole elements and widgets are created and placed in the interface, tkinter library has the features to create the elements that are modified within the program.

The drop box for the modes like Encrypt and Decrypt are created to select by the user, and also the input fields for the plain text, secret key and initialization vector (IV) from the user.

The three different types of key size like 128, 192 and 256 bytes are specified by using radio button fields that are opted by the user below the input text areas.

The 5 different modes of AES encryption that are ECB (Electronic code block), CBC (cipher block chaining), OFB (output feedback block), CTR (Counter mode), CFB (cipher feedback block). Where each mode has different types of encryption patterns using IV values. The operations of each mode are described in the working of AES encryption algorithm.

The selection of modes is displayed to the user in checkboxes and they have to select for the mode of encryption in the interface.

```
# process button
process_button = Button(root, text="Process", command=self.process)
process_button.pack()
```

To perform the encryption, “process” button is included in order to click by the user to obtain encrypted text in the output

```
# output area
output_label = Label(root, text="Output:")
output_label.pack()
```

```
self.output_text = Text(root, height=5, width=30)
self.output_text.pack()
```

For the output, the above code is written and labelled, which is used to display the cipher text to the user.

```
def encrypt(self, input_text, key, key_size, mode_of_operation, iv):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, self.get_mode_of_operation(mode_of_operation),
IV=self.pad_iv(iv))
    encrypted_bytes = cipher.encrypt(pad(input_text.encode(), AES.block_size))
    return encrypted_bytes.hex()
```

For encryption mode, it takes the input text, key, key size, mode of operation and IV as parameters and perform the function encrypt to generate the cipher text with the mode of AES. Here only CBC, CFB and OFB have to be called in the “get mode of operation function” which is defined at the end of the program. And only those 3 modes are performed with the IV values given by the user in the input field.

```
def decrypt(self, input_text, key, key_size, mode_of_operation, iv):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, self.get_mode_of_operation(mode_of_operation),
IV=self.pad_iv(iv))
    decrypted_bytes = unpad(cipher.decrypt(bytes.fromhex(input_text)),
AES.block_size)
    return decrypted_bytes.decode()
```

This code is for decrypt operation, where the cipher text is decoded by the decode () method and unpadded with the IV and key, if both are wrong, then it returns the errors.

Here encrypt and decrypt functions are only for the modes CBC, OFB and CFB. For the CTR and ECB mode other specified functions are defined. In ECB mode, there is no need

for IV value and padding, so, the separate function for ECB encrypt and decrypt are defined , and called when the user selects the ECB mode.

```
def encrypt_ecb(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_ECB)
    encrypted_bytes = cipher.encrypt(pad(input_text.encode(), AES.block_size))
    return encrypted_bytes.hex()

def decrypt_ecb(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_ECB)
    decrypted_bytes = unpad(cipher.decrypt(bytes.fromhex(input_text)),
AES.block_size)
    return decrypted_bytes.decode()
```

For the CTR, the separate function is defined because the CTR mode don't use the IV value for encryption, instead of this, it uses a nonce value (which is initiated only once) with the key, and the same nonce is used for decryption in CTR mode

```
def encrypt_ctr(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_CTR, nonce=b'00000000')
    encrypted_bytes = cipher.encrypt(input_text.encode())
    return encrypted_bytes.hex()

def decrypt_ctr(self, input_text, key, key_size):
    key = self.pad_key(key, key_size)
    cipher = AES.new(key, AES.MODE_CTR, nonce=b'00000000')
    decrypted_bytes = cipher.decrypt(bytes.fromhex(input_text))
    return decrypted_bytes.decode()
```

For the padding for the input, the function is defined for the estimation of the length of the input and act accordingly. 'ljust' keyword is used for the padding.

```
def pad_key(self, key, key_size):
    if len(key) <key_size // 8:
        key = key.ljust(key_size // 8, '\0')
    elif len(key) >key_size // 8:
        key = key[:key_size // 8]
    return key.encode()
```

`pad_key (self, key, key size)`: This method pads the key with null bytes or truncates it to the required key size.

`pad_iv (self, iv)`: This method pads the IV with null bytes or truncates it to 16 bytes.

For the padding of the IV value, pad_iv function is created and it checks the length of the IV whether it is equals to the AES block size or not. If not then it returns the "0000000000000000" as the default IV value (only if the input of the IV is none).

```
def pad_iv(self, iv):
    if iv is not None:
        if len(iv) <AES.block_size:
            iv = iv.ljust(AES.block_size, '\0')
        elif len(iv) >AES.block_size:
            iv = iv[:AES.block_size]
        return iv.encode()
    else:
        return b'0000000000000000'

def get_mode_of_operation(self, mode_of_operation):
    if mode_of_operation == "CBC":
        return AES.MODE_CBC
```

```
elifmode_of_operation == "OFB":  
    return AES.MODE_OFB  
elifmode_of_operation == "CFB":  
    return AES.MODE_CFB
```

Here the function ``get_mode_of_operation (self, mode_of_operation) ``: This method returns the corresponding AES mode of operation based on the user's selection.

`Get_mode_of_operation` is defined for the selection of modes by the user. If the value satisfies with the conditions CBC, OFB and CFB, then the `AES.mode` is replaced by these modes and performs the encryption operation.

```
root = Tk()  
AESDemo(root)  
root.mainloop()
```

Tkinter GUI elements: The program uses various Tkinter elements such as labels, option menus, entry fields, checkboxes, buttons, and text areas to create the user interface. These elements are placed using the ``pack () `` method to arrange them in the GUI window.

The program follows an object-oriented design approach, with the `AESDemo` class encapsulating the functionality and the Tkinter library providing the tools for creating the GUI. The AES encryption and decryption operations are performed using the `AES` module from the `PyCryptodome` library, while the padding functions from the `Crypto.Util.Padding` module ensure that the input text is properly padded before encryption and unpadded after decryption.

Overall, the program provides a user-friendly interface for performing AES encryption and decryption operations with different modes of operation and key sizes, making use of the Tkinter library and the PyCryptodome library for cryptographic operations.

With the installation of the VS code platform from internet and pycryptodome in cmd prompt by command `pip install`, we can successfully open the file and execute by pressing the F5 button on the keyboard, then it will execute the program and displays the window to the user, so they can perform the operations of their necessity.

9. EVALUATION AND RESULT:

Here are some screenshots of the project interface below:

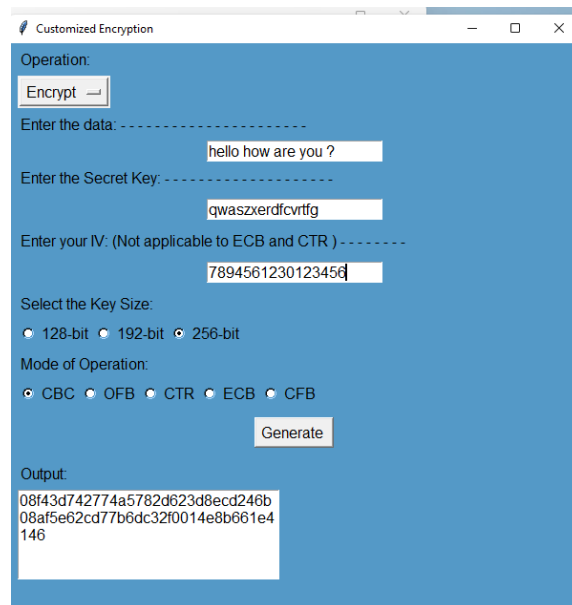
This is the initial view of the interface, after the execution of the code,

The operations, plain text, secret key, IV fields are specified in the interface, along with the key size and mode of operation selections.



Fig 9.1

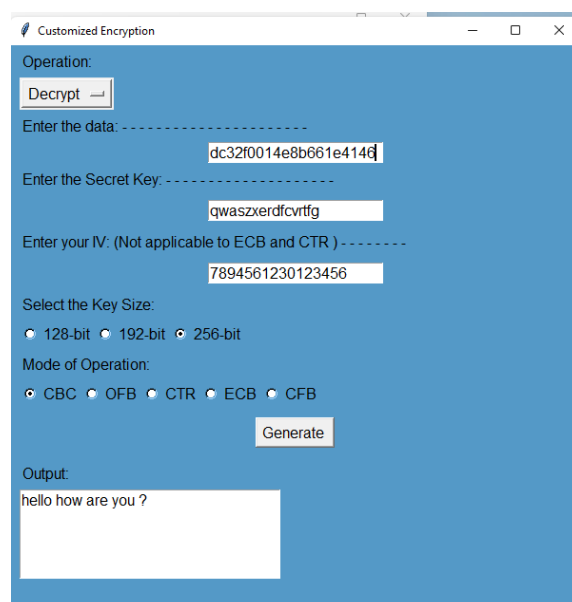
When the user enters the data, secret-key and IV after selecting the encrypt operation, they configure the key size and mode of operation of their necessity and click the generate button. By processing the parameters, it will produce the cipher text.



The screenshot shows a web application titled "Customized Encryption". The "Operation:" dropdown menu is set to "Encrypt". The "Enter the data:" field contains the text "hello how are you ?". The "Enter the Secret Key:" field contains "qwazxerdfcvrtfg". The "Enter your IV: (Not applicable to ECB and CTR)" field contains "7894561230123456". Under "Select the Key Size:", the "256-bit" radio button is selected. Under "Mode of Operation:", the "CBC" radio button is selected. A "Generate" button is visible. The "Output:" section displays a long hexadecimal string: "08f43d742774a5782d623d8ecd246b08af5e62cd77b6dc32f0014e8b661e4146".

Fig 9.2

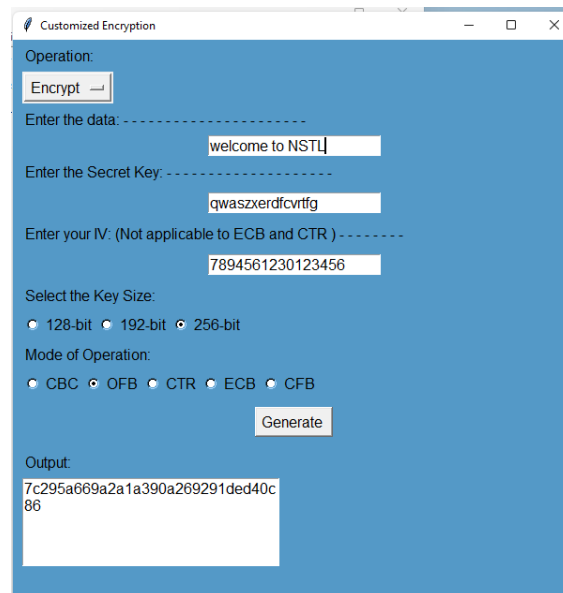
For decryption of the cipher text, the user selects the decrypt operation and enters the cipher in data field, secret key, IV value and select their respective mode and key-size. Hence, it will produce the original plain text in the output.



The screenshot shows the same "Customized Encryption" application. The "Operation:" dropdown menu is now set to "Decrypt". The "Enter the data:" field contains the hexadecimal string "dc32f0014e8b661e4146". The "Enter the Secret Key:" field still contains "qwazxerdfcvrtfg". The "Enter your IV: (Not applicable to ECB and CTR)" field still contains "7894561230123456". The "256-bit" radio button is still selected under "Select the Key Size:", and the "CBC" radio button is still selected under "Mode of Operation:". The "Generate" button is visible. The "Output:" section now displays the original plain text: "hello how are you ?".

Fig 9.3

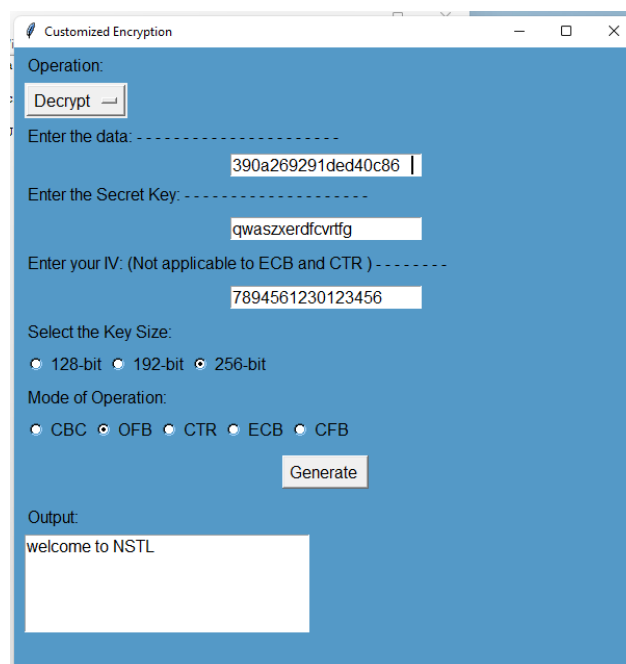
In the same way, the user can enter the parameters in the fields and select the different modes of encryption and key size and generate the cipher based on the inputs. (fig 9.4 refers the OFB encryption mode)



The screenshot shows a window titled "Customized Encryption". Under the "Operation:" label, the "Encrypt" button is selected. The "Enter the data:" field contains "welcome to NSTL". The "Enter the Secret Key:" field contains "qwazxerdfcvrtfg". The "Enter your IV: (Not applicable to ECB and CTR)" field contains "7894561230123456". Under "Select the Key Size:", the "256-bit" radio button is selected. Under "Mode of Operation:", the "OFB" radio button is selected. A "Generate" button is located below the mode selection. The "Output:" field displays the hexadecimal cipher text: "7c295a669a2a1a390a269291ded40c86".

Fig 9.4

To decrypt the cipher text, the users has to select the OFB mode in order to obtain the plain text in the output. Otherwise, it displays the errors to the user.

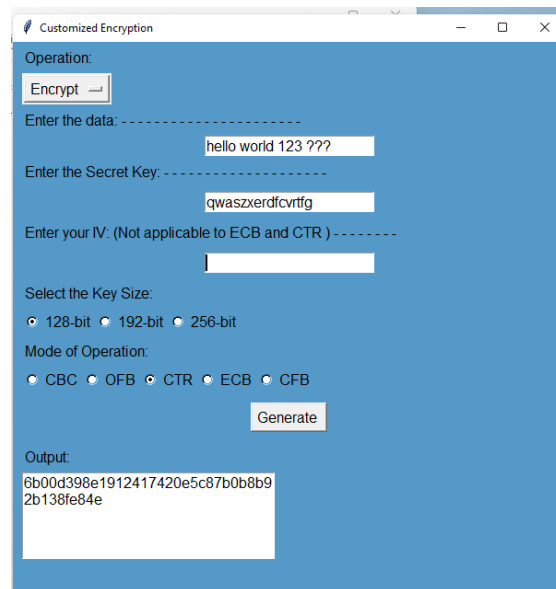


The screenshot shows the same "Customized Encryption" window, but now the "Decrypt" button is selected under "Operation:". The "Enter the data:" field contains the hexadecimal cipher text "390a269291ded40c86". The "Enter the Secret Key:" field still contains "qwazxerdfcvrtfg". The "Enter your IV: (Not applicable to ECB and CTR)" field still contains "7894561230123456". The "256-bit" radio button is still selected under "Select the Key Size:", and the "OFB" radio button is still selected under "Mode of Operation:". The "Generate" button is present. The "Output:" field now displays the plain text: "welcome to NSTL".

Fig 9.5

For the ECB [5.1.1] and CTR [5.1.5] modes in the interface, the user has to ignore the IV field and only enter the data and the secret key. Thus, the CTR mode uses a counter value, which is given in the code, by using that counter it decrypts the data from the user.

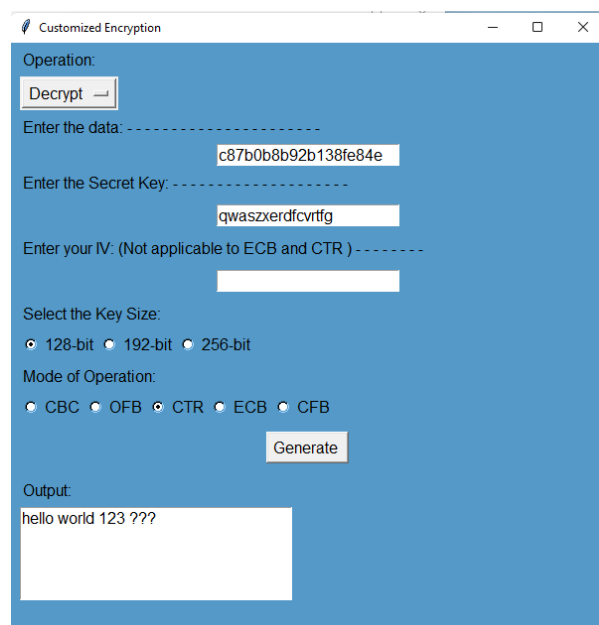
Fig 9.10 represents the CTR encryption.



The screenshot shows a web application titled "Customized Encryption". The "Operation:" dropdown is set to "Encrypt". The "Enter the data:" field contains "hello world 123 ???". The "Enter the Secret Key:" field contains "qwazxerdfcvrtfg". The "Enter your IV: (Not applicable to ECB and CTR)" field is empty. Under "Select the Key Size:", the "128-bit" radio button is selected. Under "Mode of Operation:", the "CTR" radio button is selected. A "Generate" button is visible. The "Output:" field displays the hexadecimal ciphertext: "6b00d398e1912417420e5c87b0b8b92b138fe84e".

Fig 9.10

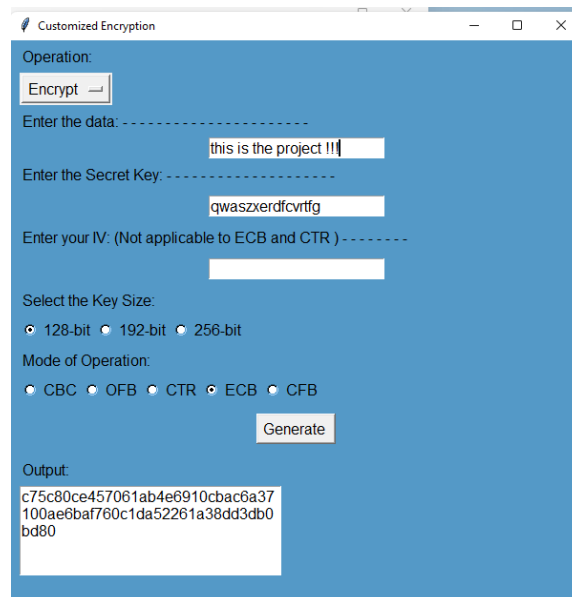
The decryption in CTR mode requires the cipher and a secret-key to produce the plain text in the output.



The screenshot shows the same "Customized Encryption" web application. The "Operation:" dropdown is now set to "Decrypt". The "Enter the data:" field contains the hexadecimal ciphertext "c87b0b8b92b138fe84e". The "Enter the Secret Key:" field still contains "qwazxerdfcvrtfg". The "Enter your IV: (Not applicable to ECB and CTR)" field is empty. Under "Select the Key Size:", the "128-bit" radio button is selected. Under "Mode of Operation:", the "CTR" radio button is selected. A "Generate" button is visible. The "Output:" field displays the decrypted plaintext: "hello world 123 ???".

Fig 9.11

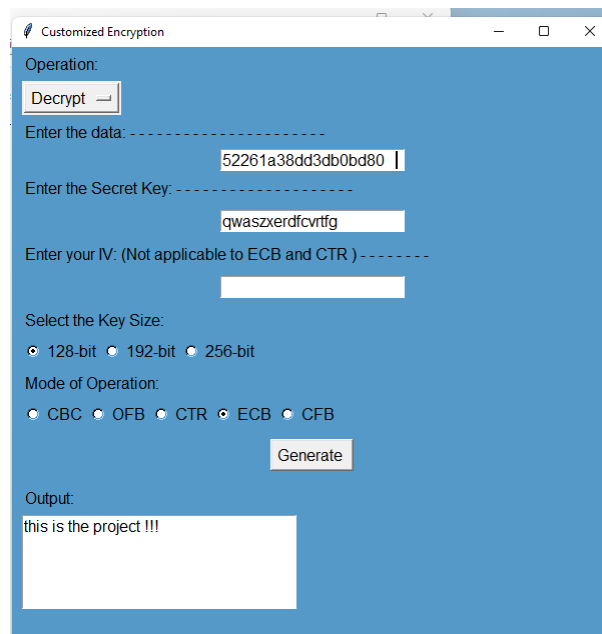
In the ECB mode, there is no IV value utilization for the data encryption. It only uses plain text and secret key to produce cipher to the user. Hence , the IV field can be ignored for this mode of encryption.



The screenshot shows a web application titled "Customized Encryption". The "Operation:" dropdown is set to "Encrypt". The "Enter the data:" field contains the text "this is the project !!!". The "Enter the Secret Key:" field contains the text "qwazxerdfcvrtfg". The "Enter your IV: (Not applicable to ECB and CTR)" field is empty. Under "Select the Key Size:", the "128-bit" radio button is selected. Under "Mode of Operation:", the "ECB" radio button is selected. A "Generate" button is visible. The "Output:" section displays the encrypted ciphertext: "c75c80ce457061ab4e6910cbac6a37100ae6baf760c1da52261a38dd3db0bd80".

Fig 9.12

For decryption in ECB, user has to select the decrypt operation and enter the data with secretkey to obtain the plain text.



The screenshot shows the same "Customized Encryption" application. The "Operation:" dropdown is now set to "Decrypt". The "Enter the data:" field contains the ciphertext "52261a38dd3db0bd80". The "Enter the Secret Key:" field still contains "qwazxerdfcvrtfg". The "Enter your IV:" field is empty. The "128-bit" radio button is still selected under "Select the Key Size:", and the "ECB" radio button is still selected under "Mode of Operation:". The "Generate" button is visible. The "Output:" section now displays the decrypted plaintext: "this is the project !!!".

Fig 9.13

9.1 Errors :

The Errors in the interface are recognized by the try and catch blocks in the code and they are displayed to the users in the output.

1. UTF-8 codec

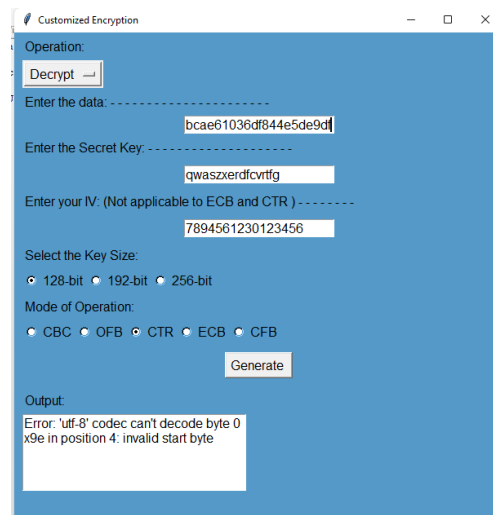
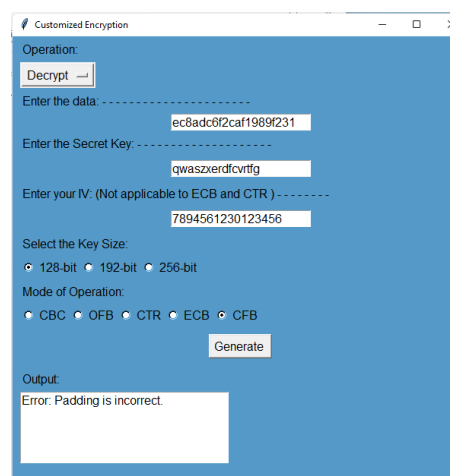


Fig 9.1.1

This error occurs when the user selects the different mode while encrypting, where they selected the ECB or CTR mode, after the encryption is done in CBC, OFB or CFB.

Therefore, to avoid this error, the user have to make sure that the specified parameters given in the encryption operation.

2. Padding is incorrect:



Here, **Fig 9.1.2** refers the error "padding is incorrect "

This error denotes in correct padding of the cipher, where the user have selected different is displayed in the output, which refers to the padding is different from encrypt mode. Hence, the user has selected the different key size in decrypt operation, which raised the error to the user.

10. CONCLUSION

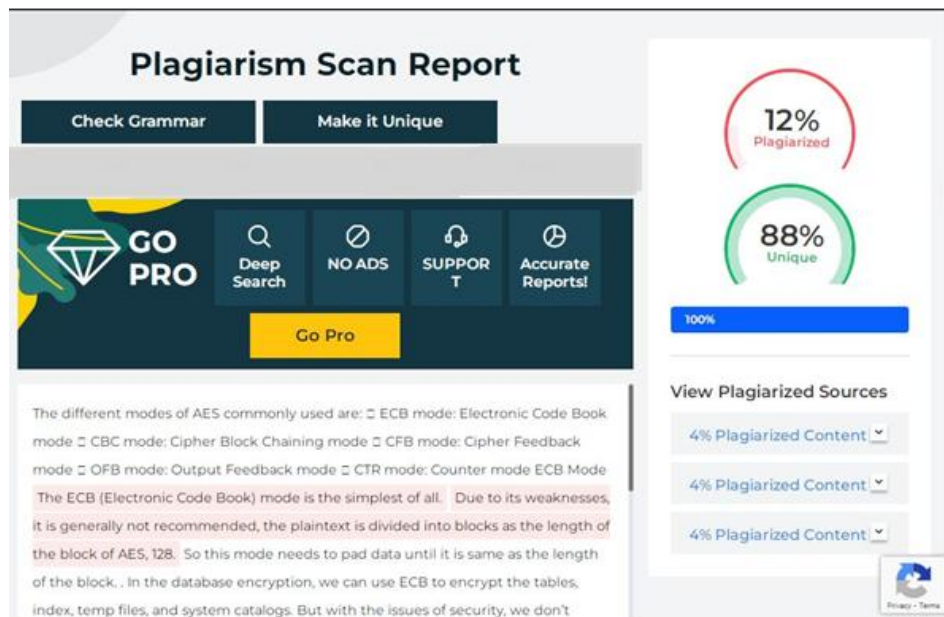
The Customized encryption algorithm project is designed in such a way that it can be used for secure communications between different entities to provide the communication security. The interface is designed and developed as an offline utility which can run without internet connection. Further, it is also easy to use to configure the required encryption parameters to perform encryption or decryption operations.

Many researches had found that the AES algorithm is the secured algorithm for encryption of data over the communication. In this view, the AES algorithm is utilized for my project to design and develop the customized encryption algorithm interface using python.

When compared to the existing AES encryption interfaces, this Customized encryption algorithm, this enables user to configure the different modes and key-sizes at one place through this interface. The result of this project is evaluated with the other existing methods and improved for functioning of the user requirements.

However, this Customized Encryption Algorithm will provide the privacy of the data and secured communications.

11. PLAGIARISM SCAN:



12. REFERNECES:

12.1 For the modes of encryption: xperti.io/blogs/java-aes-encryption-and-decryption/

12.2 Pycryptodome packages: pypi.org/project/pycryptodome

12.3 TKinter library in python: tutorialspoint.com/python/python_gui_programming.html

12.4 AES working concept: geeksforgeeks.org/advanced-encryption-standard-aes/amp/

12.5 FIPS : wikipedia/wiki/federal-information-processing-standards

