NIPES

Journal of Science and Technology Research

Journal homepage: www.nipesjournals.org.ng

# Text Encryption Using Advanced Encryption Standard (AES) Algorithm

***[1]Akwukwuma, V.V.N., [1]Chete F.O., [1]Oshioluamhe, M.N. and [2]Okpako A.E.***
[1]Department of Computer Science, University of Benin, Benin City, Nigeria
[2] Department of Cybersecurity, University of Delta, Agbor, Delta State
Corresponding author: odichet@yahoo.com

**Article Info**

***Keywords:*** *Advanced Encryption Standard, encryption, decryption, ciphertext, algorithm*

*Received 4 March 2024*
*Revised 17 May 2024*
*Accepted 22 May 2024*
*Available online 26 June 2024*

https://doi.org/ 10.5281/zenodo.12558923

ISSN-2682-5821/© 2024 NIPES Pub. All rights reserved.

**Abstract**

*With the increasing importance of data security in the digital age, the need for robust encryption techniques has become paramount. The Advanced Encryption Standard (AES) is regarded as one of the safest encryption techniques currently in use. The AES encryption algorithm is a symmetric block cipher algorithm with a block/chunk size of 128 bits. These distinct blocks are converted using keys that are 128, 192, and 256 bits long. The ciphertext is created by joining these blocks together once they have been encrypted. The decryption is done in the reverse order. This study discusses the implementation of text encryption using the AES algorithm. The Graphical User Interface of the text encryption system was built using python library Tkinter while the encryption and decryption algorithm were programmed using python and its libraries. The tools used for the implementation were the text editor and python Integrated Development Environment (IDE).*

## 1. Introduction

Text encryption is a fundamental concept in the field of cryptography, which focuses on protecting information by changing it into an unintelligible form. It is essential in preserving the confidentiality, integrity, and authentication of sensitive data. Encryption is done to reduce the chances of unauthorized individual(s) to decipher encrypted data without possessing the required key [1]. The basis of encryption is encryption algorithms. They choose the precise procedures and actions needed to convert plaintext into ciphertext and vice versa. These algorithms frequently use confusion, mixing, permutation, and substitution operations. Advanced Encryption Standard (AES), introduced by the National Institute of Standards and Technology (NIST) in 2001, is a symmetric block cipher which overcomes the key size weakness of Data Encryption Standard (DES) [2]. The selection of Rijndael as the AES, after a rigorous evaluation process by NIST, was based on its robustness, efficiency, and broad applicability [3], [6]. The algorithm comes with variable key sizes i.e. 128-bit key, 192-bit key and a 256-bit key [2]. The AES encryption algorithm is a symmetric block cipher algorithm with a block/chunk size of 128 bits. These distinct blocks are converted using keys that are 128, 192, and 256 bits long. Once it encrypts these blocks, it joins them together to form the ciphertext [3].

The need for AES arose as an improvement for existing Data Encryption Standard (DES), which exhibited vulnerabilities to emerging computing power and cryptanalysis [4]. Cryptanalysis is the process of decrypting and analysing codes, encrypted material or ciphers [5]. The field covers deciphering encrypted messages without having access to the key that was used during encryption [5]. The current industry standard for secret key encryption is called Advanced Encryption Standard (AES) [5].

The strength of AES is its resistance to various cryptographic attacks, such as differential and linear cryptanalysis, which were major concerns for DES [4]. The cryptographic community has closely examined AES, but no real weaknesses have been found yet. It is thought to be computationally safe against brute-force attacks and satisfies the security requirements for secrecy and integrity [6], [4]. AES is renowned for its efficient implementation and computational speed. It is well-suited for a wide range of computing platforms, including hardware and software-based systems.

With the growing volume of digital data and the need to protect sensitive information, encryption is critical to data security. Passwords, personal information, financial records, and secret communications all require strong encryption technologies. Implementing AES encryption in Python can address these security concerns, which are motivated by the requirement for safe textual data storage, transfer, and processing.

Text encryption using AES algorithm is a highly secure algorithm, which provides data confidentiality and protection. Python is a well-known and versatile programming language with an extensive ecosystem of libraries and frameworks. AES encryption libraries for Python are widely accessible, making it simple to include AES encryption features into existing Python programs. Python's widespread support for AES simplifies development and supports the adoption of secure data encryption procedures. This study implements text encryption using AES algorithm in python to provide secure and comprehensive data protection in the face of growing risks and weaknesses, supporting confidentiality, and integrity in digital conversations and information transmission

[7] implemented AES in Python using PyCryptodone. By using the pycryptodone library in python, encrypting a password took three lines of code, while decryption was just as simple. However, depending on the cypher used, different elements like tags, nonce, initialization vectors, MAC, etc. would need to be stored.

[8] implemented AES encryption and decryption in Java .To implement input string encryption, a secret key was generated, thereafter, an instance from the Cipher class was created by using the getInstance() method. Additionally, a cipher instance was configured using the init() method with a secret key, and encryption mode. Finally, the input string was encrypted by invoking the doFinal() method. This method gets bytes of input and returns ciphertext in bytes. For decrypting an input string, the cipher was initialized using the DECRYPT_MODE to decrypt.

[9] implemented AES encryption and decryption using the Java Cryptography Architecture (JCA) within the JDK. The study discussed how to encrypt and decrypt input data like strings, files, objects, and password-based data using the AES algorithm in Java. For the password based data, the AES encryption and decryption was done using the secret key derived from a given password. For generating a secret key, the *getKeyFromPassword()* method was used. Thereafter, the encryption was performed by using the instantiated cipher and the provided secret key. To encrypt a Java object, the *SealedObject* class was used. The encrypted object was later decrypted using the correct cipher.

For files, encryption was done by encrypting a buffer at a time. To decrypt a file, similar steps were used and the cypher was initialized using *DECRYPT_MODE.* Again for encrypting and decrypting a text file, the *baeldung.txt* file was read from the test resource directory, then encrypted into a file called *baeldung.encrypted,* and thereafter decrypted the file into a new file.

[10] used the symmetric key to encrypt and decrypt data in C#. The same key was used for both encryption and decryption in the system security cryptographic namespace by having a predefined AES class. An initialization vector IV) which is 16 bytes was also used, while the algorithm's block size. IV is optional. Implementation codes were then written in the Main method inside the Program.cs file.

[11] used two scripts in Python to encrypt/decrypt using the 128 bits AES algorithm, ECB mode with hex "00" as padding for each character. For the encryption, an ASCII plaintext file was taken as the input, and then an encrypted hex file was outputted. For the decryption, a ciphertext hex file was taken as the input, and then a decrypted ASCII file was outputted.

[12] presented an Image Encryption and Decryption algorithm using Advance Encryption Standard (AES) which accepted an image as input to both the AES encryption and decryption module. The design used the iterative approach with block size of 128 bit and key size of 256 bit. The numbers of round for key size of 256 bits is 14. The study underscore the unmet potential of using image in data encryption and key generation for diverse areas of information security where sensitive and confidential data needs to be transmitted along with the image.

## 2. Methodology

The implementation of the text encryption using AES algorithm in python was done with python, its libraries and tools such as the tkinter (“TK Interface”), which is the standard Graphic User interface library for python. Python was used for the implementation because of its simplicity and vast collection of libraries to make complex nature of the encryption algorithm looks easy by masking the complexity with a user interface. So, everything from the UML to the user interface design was done to make it intuitive and user friendly. A 128 bit AES algorithm was implemented using Python to provide a user friendly interface which was tested and well validated. The encryption module made use of 10 rounds for the processing of the 128 bit keys while the block cipher mode of operation was implemented using the Electronic Code Book(ECB) mode (which is one of the easiest and fastest algorithm to implement) ; that breaks the plaintext into blocks and uses the key to encrypt them individually. The study employed AES to encrypt text-based data such as strings or plaintext files. This involves encrypting critical messages, passwords, and other textual data. The implementation focused on encrypting the input text using the AES algorithm. This involved applying AES-specific operations, such as substitution, permutation, and mixing, to transform the plaintext into ciphertext. Thus, the study provided a user-friendly interface for AES encryption in python, then tested and validated the AES implementation. The study also made use of open source codes from Github for implementation.

### 2.1 Systems Analysis and Design

#### 2.1.1 The Present System

The AES is a symmetric key-block cypher which uses the same key for encryption and decryption and operates on a fixed data blocks of 128 bits and supports key sizes of 128, 192 and 256 bits, each with increasing complexity [4]. The larger the key sizes, the harder it is to break that key through brute –force attacks [4]. The encryption process is repeated multiple times, with the number of repetitions depending on the key size. AES uses a substitution-permutation network to shuffle the data, making it harder to crack [13]. In addition to performing multiple rounds of processing during encryption and decryption, the algorithm is resistance to modern attacks such as linear and differential cryptanalysis [4]. AES is fast, efficient, has a flexible key size, resistant to known attacks and widely used and supported by most modern cryptographic libraries [13]. The algorithms versatility and efficiency and robust security makes it an essential tool for protecting sensitive data across various industries, businesses and military communications [4]. The security of this algorithm, however, does depend on some factors such as the implementation, key management and the overall security of the system in which it is used [4]. Although, AES, as a symmetric encryption method does have significant performance benefits over asymmetric encryption, it has the drawback that it relies on a single key that must remain secret [4]. In addition, the encryption and decryption process is resource-intensive and can be slow on low-end devices [13]. Furthermore, AES can be vulnerable to side-channel attacks, where an attacker can observe the physical characteristics of the device performing the encryption or decryption process to gain information about the key [13].

### 2.1.2 The Proposed System

In the proposed system, the AES encryption system is implemented on a GUI, making it easier to use compared to encryption on terminals. No technical knowhow is needed as it was made to be as intuitive as possible. All the user needs to do is just enter the message and a passphrase, and then the system does the rest. The system uses the advantage of GUI over command line such as user friendliness, visual interaction, etc. Another advantage is that it is offline and internet access not required. Our proposed system has the following advantages over existing systems:

a) The system uses UI design to mask the complexities (Which makes it easy to use).
b) The system doesn’t need the passphrase to be exactly 16 bits; it can be 16 bits or lower thus allowing flexibility of choice of passphrase
c) It can be used offline (without internet), which makes it different from the website that can do the AES encryption
d) It saves the stress of creating files when running on terminal.
e) It is easier to copy the ciphertext/plaintext than on terminal
f) It is simple to use, as the UI design is user friendly and a page on how to use it has been created.
g) It is easily accessible as it can be gotten from GitHub.

The Graphical User Interface (GUI) implementation of Text encryption system using AES algorithm in python contains the following features:

i. Main page: This feature contains buttons that lead you to other pages.
ii. How to use page: This feature contains a simple instruction on how to efficiently use the GUI.
iii. Encryption page: This feature allows the user enter the plaintext and the key to produce the ciphertext.
iv. Decryption page: This feature allows the user enter the ciphertext and the key to produce the plaintext.

## 2.2 System Requirements

The system requirements are divided into two categories, the functional requirements and the non-functional requirements.

### 2.2.1 Functional Requirements

A functional requirement is an explanation of the service that the program needs to provide. It explains a system or an aspect of it. The functional requirement for this system is designed in a way that the interface meets the needs of the users.
The requirements include:

i. Input plaintext and key
ii. Copying of the ciphertext.
iii. Encryption of the plaintext
iv. Decryption of the ciphertext

### 2.2.2Non-Functional Requirements

The non-functional requirement is a specification that outlines the operational capabilities of the system and the constraints that improve its functionality. The non-functional requirements for this system include:

a. Performance and scalability requirements: This illustrates how quickly the system can provide results and how much a higher workload will change performance. It includes how the system performs given a large number of texts.
b. Usability requirements: This relates to how user-friendly the system is for the client. They include:
   i. The user interface (UI) is easy to understand
   ii. In order to prevent users from straining their eyes, the user interface was designed with good colours.
c. Compatibility requirements: This deals on how the hardware, operating systems and their versions of this GUI run. . They include:
   i. Python is cross-platform and will work on all Windows, MAC and Linux.
   ii. The program was built by efficiently using libraries, which reduces load time.

## 2.3 System Design

Here, we describe the architecture of the proposed GUI implementation of Text encryption system using AES algorithm in python throughout the system design phase.

### 2.3.1 Architecture of proposed system

We outline the system's components and libraries as we show the architecture of the proposed GUI implementation of Text encryption system using AES algorithm in python.

i. **User Interface (GUI):** Tkinter was used to build the GUI. Tkinter is a built-in option and is easy to use for beginners.
ii. **Encryption and Decryption Module:** Implementation of the AES encryption and decryption logic using Python libraries like ‘math’ and an external library called ‘bit vector’ was used.
iii. **Text Input and Output Widgets:** these are widgets for users to input the text they want to encrypt and display the encrypted text after encryption.
iv. **Key Management:** describes the methods for generating, storing, and managing encryption keys securely. The method used in this system was for the user to enter a 16 bit or less key.

## 2.4 Design Model

For the GUI implementation of Text encryption system using AES algorithm in python, this was done with the use of UML (Unified Modeling Language) and user interface designs. Unified Modeling Language (UML) is a common modeling language used in software engineering that allows software systems, their parts, and their interactions to be visually represented and documented.

The following UML was used:

i. Sequence diagram
ii. Activity diagram.
iii. Use case diagram.

### 2.4.1 Sequence diagram

In a sequence diagram an object's interactions are arranged chronologically. It depicts the participants in the scenario as well as the flow of messages between them that is necessary for the scenario to work. Figure 1 shows the design of the sequence diagram for the user.

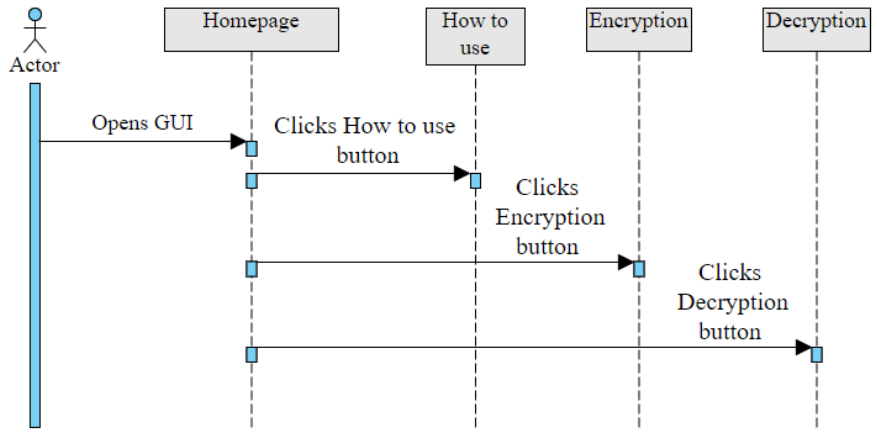


**Figure 1 Sequence diagram for the user**

### 2.4.2 Activity diagram

An activity diagram in UML (Unified Modeling Language) is a form of behavioral diagram that depicts the movement of activities and actions inside a system, process, or use case. It is employed to depict a system's dynamic components and is especially helpful for illustrating workflow, business processes, and the order of operations within a software program.
Figure 2 shows the design of the Activity diagram for the user to navigate the GUI .

### 2.4.3 Use case diagram

A use case diagram represents potential user interactions with a system graphically. The use cases for this system are the administrators and the user. Figure 3 shows the design of the use case diagram.

## 2.5 User Interface Designs

This is the method used by designers to create user interfaces for software and electronic devices while putting an emphasis on aesthetics.
This section shows the design for the interfaces for the users when interacting with the system.

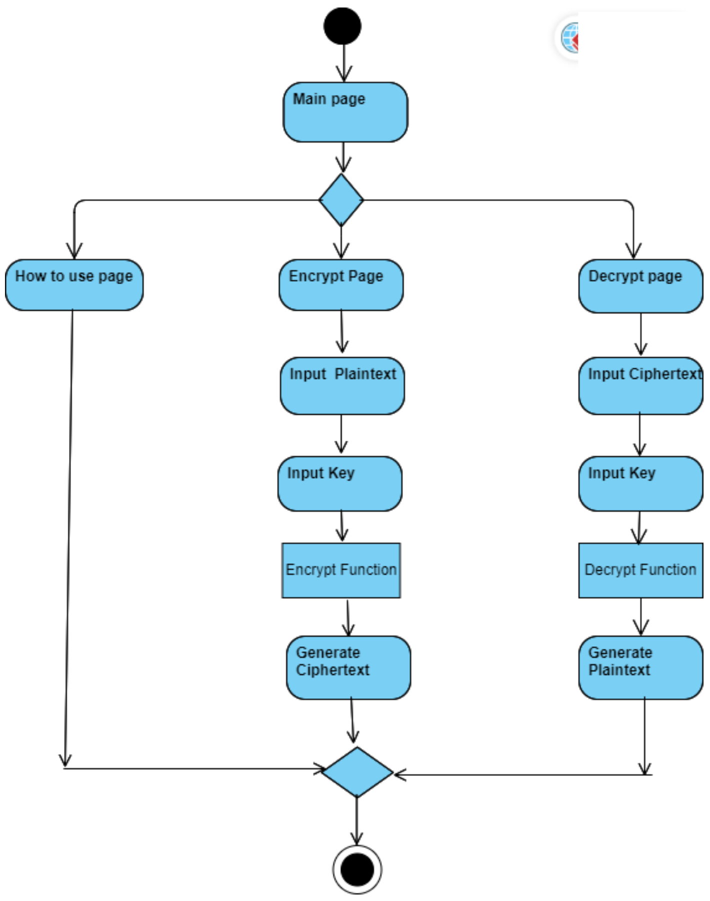
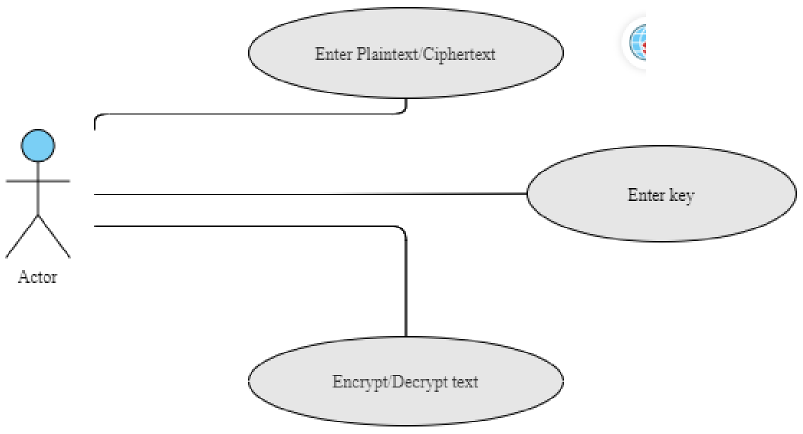


**Figure 2: Activity diagram for the user to navigate the GUI**



Use case diagram for the user

**Figure 3: Use case diagram for the users**

### 2.5.1 Home page design

This is the first page the user visit to open the program. The page contains buttons that lead to the “How To Use”, “Encryption” and “Decryption” pages respectively. The following codes were used for the design. These codes, depicted in Figure 1, show the design for the home page, including how the page is linked to the other pages.

```
from tkinter import *
import os
root= Tk()
root.title("AES ENCRYPTION AND DECRYPTION GUI")
root.geometry("600x400")
def run1():
os.system("Encrypttk.py")
def run2():
os.system("Decrypttk.py")
def run3():
os.system("HowToUse.py")
button1= Button(root,text="HOW TO USE",command=run3, height=5, width= 30)
button1.pack(pady = 20)
button2= Button(root,text="Encryption",command=run1,height=5, width= 30)
button2.pack(pady = 20)
button3= Button(root,text="Decryption",command=run2, height=5, width= 30)
button3.pack(pady = 20)
root.mainloop()
```

Figure 4 shows the design of the home page.

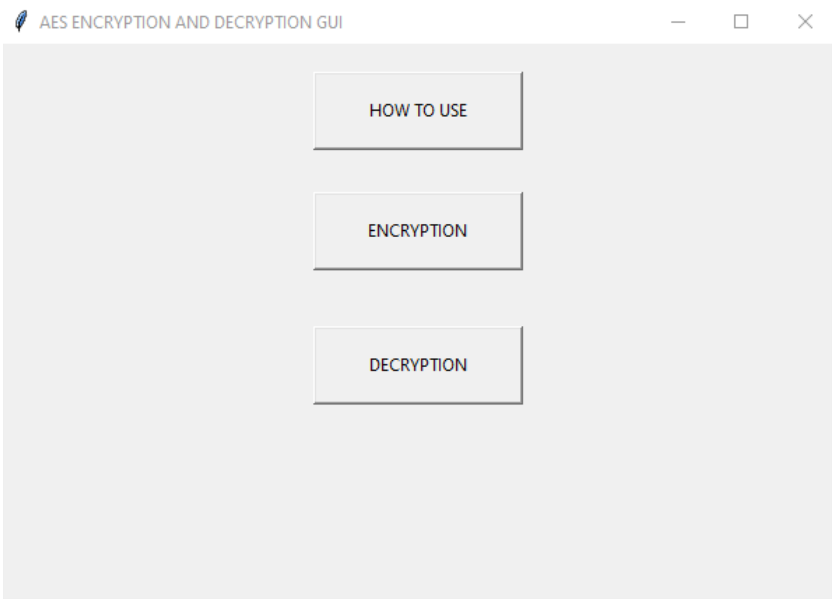


**Figure 4: Interface design of Home page**

### 2.5.2 Encryption page design

Here, the user encrypts the text. The user fills in the plaintext and key details and produces the ciphertext as the result.
The Encrypt Function: The encrypt function depicts where the encryption occurs. This part of the code shown shows the specification for the passphrase, which should not be more than 16 characters, if less it would be filled until it’s 16 characters. Reason for 16 characters is because 16 characters * 8 bits = 128 bits strength. The codes are depicted in Figure 2.

```
def aesEncrypt(PassPhrase, message):
   while(len(PassPhrase)!=16):
      if(len(PassPhrase)<16):#check if less than 16 characters, if so add one space character until 16 chars
         while(len(PassPhrase)!=16):
            PassPhrase=PassPhrase+"\00"
      if(len(PassPhrase)>16):#check if bigger than 16 characters, if so then truncate it to be only 16 chars from [0:16]
         print("Your passphrase was larger than 16, truncating passphrase.")
         PassPhrase=PassPhrase[0:16]
```

Fig2: codes for the encryption page

Fig 5 shows the design of the encryption page.

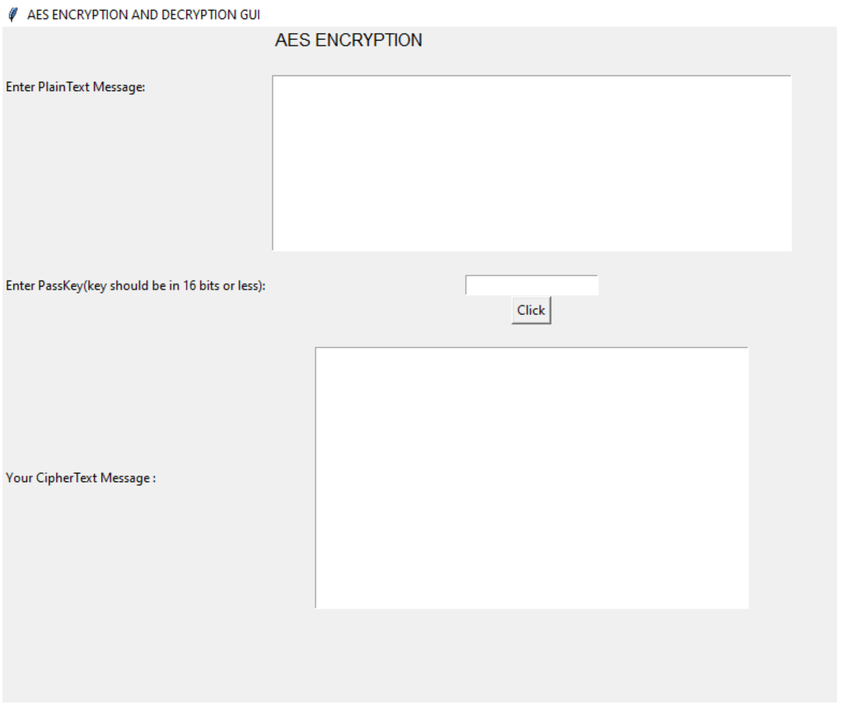

AES ENCRYPTION AND DECRYPTION GUI
AES ENCRYPTION
Enter PlainText Message:
Enter PassKey(key should be in 16 bits or less):
Click
Your CipherText Message :

**Figure 5: Interface design of Encryption page**

### 2.6.3 Decryption page design

Here,  the user decrypts the text. The user fills in the ciphertext and key details and produces the plaintext as the result.
The Decrypt Function: The decrypt function is where the decryption occurs. This part of the code depicts the checking of the passphrase to make sure it is 16 characters. If it is less than 16 characters, \00 is added. The codes are depicted in Figure3.

```
def decrypt(PassPhrase,message):
   while(len(PassPhrase)!=16):
      if(len(PassPhrase)<16):#check if less than 16 characters, if so add one space character until 16 chars
         while(len(PassPhrase)!=16):
            PassPhrase=PassPhrase+"\00"
      if(len(PassPhrase)>16):#check if bigger than 16 characters, if so then truncate it to be only 16 chars from [0:16]
```

```
        print("Your passphrase was larger than 16, truncating passphrase.")
        PassPhrase=PassPhrase[0:16]
```

**Fig3: Codes for the decryption**

Figure 6 shows the design of the decryption page

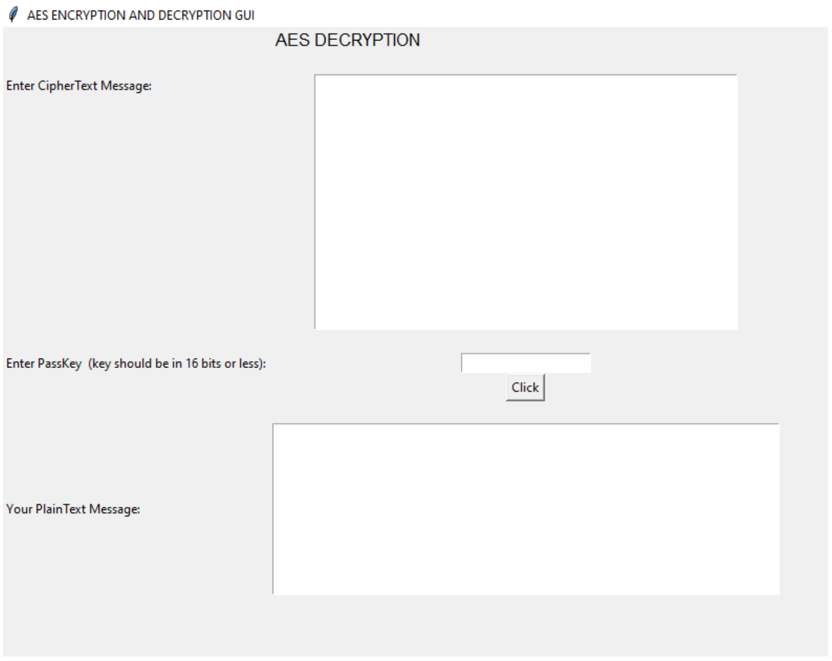
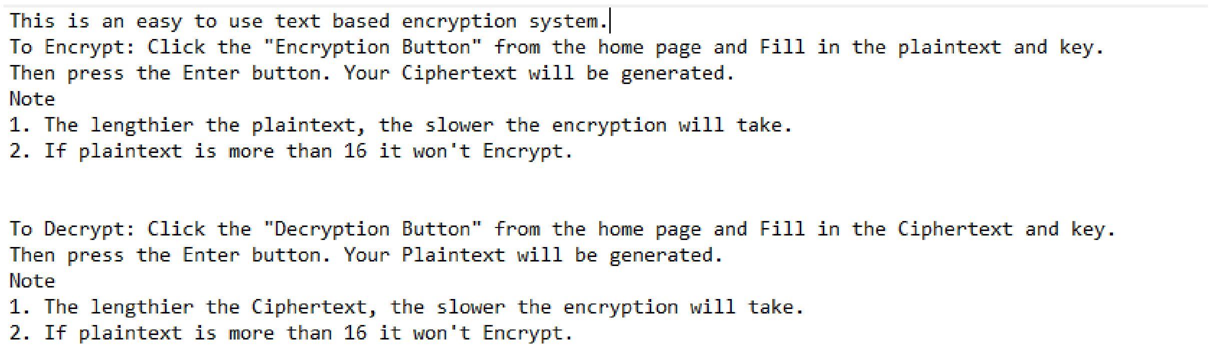

AES ENCRYPTION AND DECRYPTION GUI
AES DECRYPTION
Enter CipherText Message:
Enter PassKey (key should be in 16 bits or less):
Click
Your PlainText Message:

**Figure 6: Interface design of Decryption page**

### 2.6.4 How to use page design

This page provides instruction on how to use the GUI
Fig 7 shows the design of the 'how to use page'.



```
This is an easy to use text based encryption system.
To Encrypt: Click the "Encryption Button" from the home page and Fill in the plaintext and key.
Then press the Enter button. Your Ciphertext will be generated.
Note
1. The lengthier the plaintext, the slower the encryption will take.
2. If plaintext is more than 16 it won't Encrypt.


To Decrypt: Click the "Decryption Button" from the home page and Fill in the Ciphertext and key.
Then press the Enter button. Your Plaintext will be generated.
Note
1. The lengthier the Ciphertext, the slower the encryption will take.
2. If plaintext is more than 16 it won't Encrypt.
```

**Figure 7: Interface design of How to use page**

## 3. Implementation and Testing

### 3.1 Implementation Tools

The Graphical User Interface was built using Python and its libraries such as Tkinter, and Math.

a. **Python:** Python is a high-level, versatile, and dynamically-typed programming language known for its simplicity and readability. Python is a popular high-level, multipurpose programming language that is easy to learn and comprehend. Python has been incredibly popular in a number of fields, such as artificial intelligence, web development, and other areas.

b. **Math Library (math module):** Python's math library, represented by the `math` module, is part of the Python Standard Library. It provides a comprehensive set of mathematical functions and constants for various mathematical operations. These operations include elementary arithmetic, trigonometry, logarithmic functions, and more.

c. **Tkinter Library:** A common Python package for designing graphical user interfaces (GUIs) is called Tkinter. It offers a collection of widgets and tools for creating windows, dialog boxes, buttons, menus, and other graphical user interface elements. The Tk GUI toolkit, which is frequently used to create cross-platform GUI applications, is the foundation of Tkinter.

**d.** **Bitvector Library**: This is an external library created by Avinash Kak. The main function of this Library is to manipulate Binary, Hexadecimal and Decimal values to give you your desired result. For example. get_bivector_in_hex() returns the result in Hexadecimal.

e. **Visual Studio Code:** The Visual Studio Code also known as VS Code is a popular and versatile code editor developed by Microsoft. Because of its lightweight design, extensible nature and support for a wide range of programming languages, it has becoming increasingly popular among developers.

**e. Python IDE (Integrated Development Environment):** A Python is a software application that provides developers and programmers with a comprehensive and integrated environment for writing, testing, debugging, and managing Python code. It offers a range of tools and features to streamline the software development process and enhance productivity.

### 3.2 System Implementation

#### 3.2.1 Home page

This is the first page the user visit when he opens the program. The page contains buttons that lead to the “How to Use”, “Encryption” and “Decryption” pages respectively.
Figure 8: shows a snapshot of the home page.

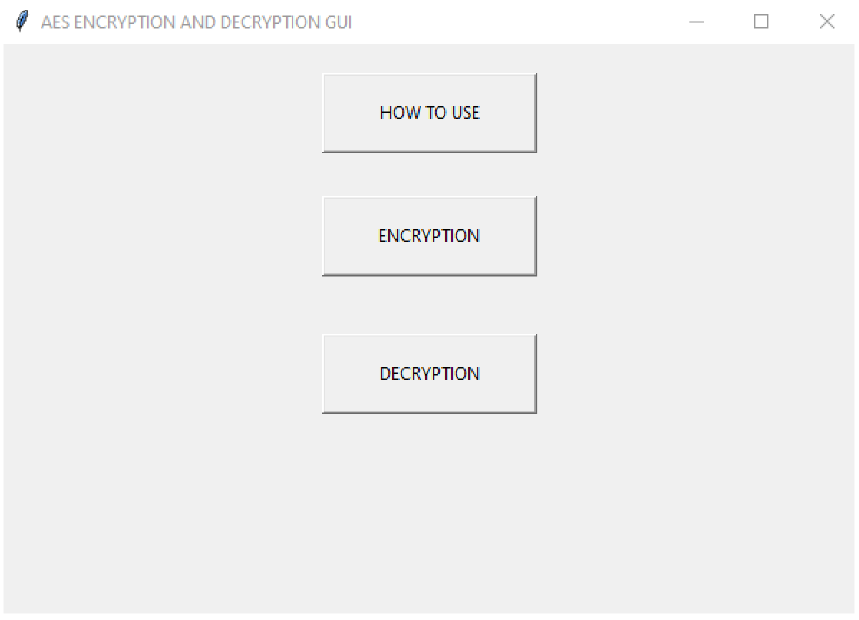


**Figure 8: Home page**

### 3.2.2 Encryption page

This is where the user encrypts the text. The user fills in the plaintext and key details and produces the ciphertext as the result.
Figure 9 shows a snapshot of the encryption page.

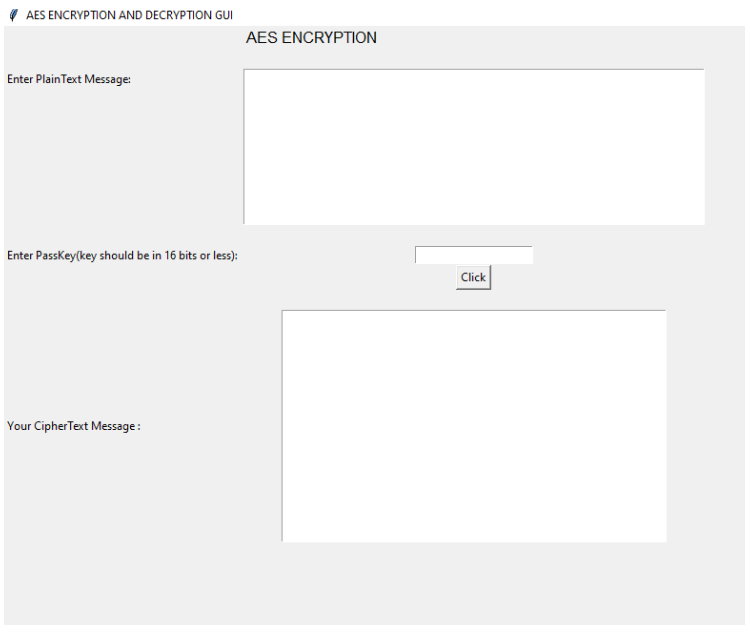

AES ENCRYPTION AND DECRYPTION GUI
AES ENCRYPTION
Enter PlainText Message:
Enter PassKey(key should be in 16 bits or less):
Click
Your CipherText Message :

**Figure 9: Encryption page**

### 3.2.3 Decryption page

This page is where the user decrypts the text. The user fills in the ciphertext and key details and produces the plaintext as the result.
Figure 10 shows a snapshot of the decryption page.

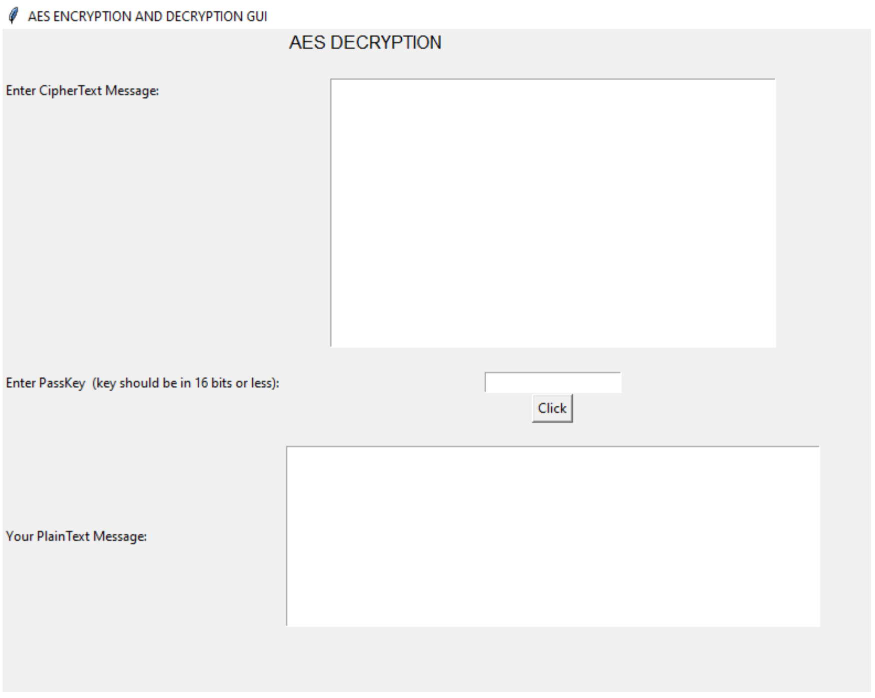
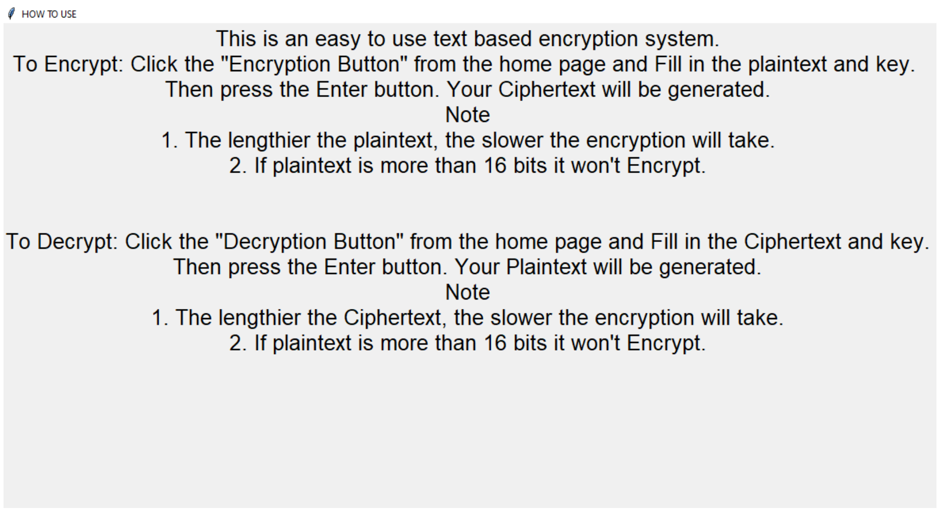

AES ENCRYPTION AND DECRYPTION GUI
AES DECRYPTION
Enter CipherText Message:
Enter PassKey (key should be in 16 bits or less):
Click
Your PlainText Message:

**Figure 10: Decryption page**

### 3.2.4 How to use page

This page is where the user learns how to use the GUI (Graphical User Interface). It provides instruction on how to use the GUI (Graphical User Interface).
Figure 11 shows a snapshot of the how to use page.



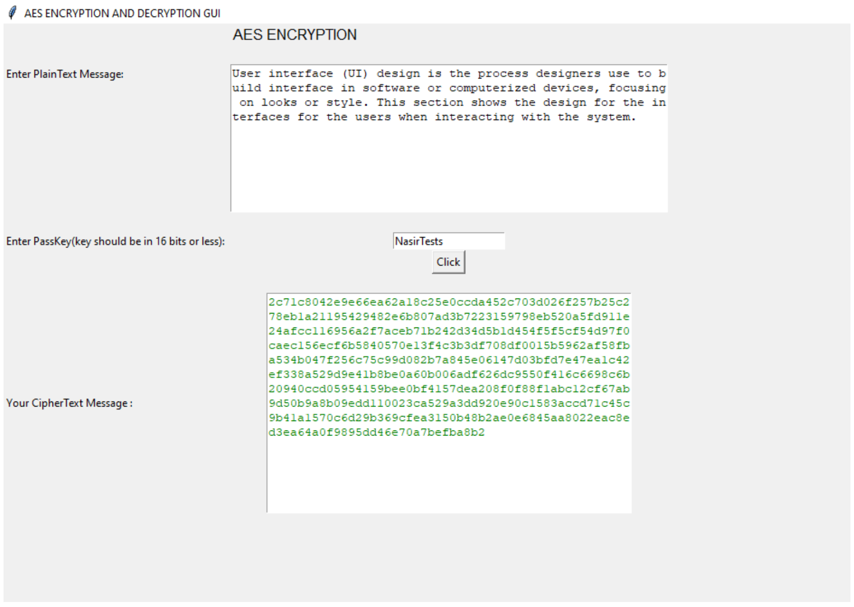
This is an easy to use text based encryption system.
To Encrypt: Click the "Encryption Button" from the home page and Fill in the plaintext and key.
Then press the Enter button. Your Ciphertext will be generated.
Note
1. The lengthier the plaintext, the slower the encryption will take.
2. If plaintext is more than 16 bits it won't Encrypt.

To Decrypt: Click the "Decryption Button" from the home page and Fill in the Ciphertext and key.
Then press the Enter button. Your Plaintext will be generated.
Note
1. The lengthier the Ciphertext, the slower the encryption will take.
2. If plaintext is more than 16 bits it won't Encrypt.

**Figure 11: How to use page**

## 3.3. Testing

The testing of the Graphical User Interface was carried out during programming and implementation. Each page of this web application was tested in isolation before it was integrated to form the system. The system is initialized with a test case, the expected outcome was achieved, and errors debugged. Figure 12 and Figure 13 shows the testing of the encryption system and the decryption system.



AES ENCRYPTION AND DECRYPTION GUI

AES ENCRYPTION

Enter PlainText Message:

```
User interface (UI) design is the process designers use to b
uild interface in software or computerized devices, focusing
 on looks or style. This section shows the design for the in
terfaces for the users when interacting with the system.
```

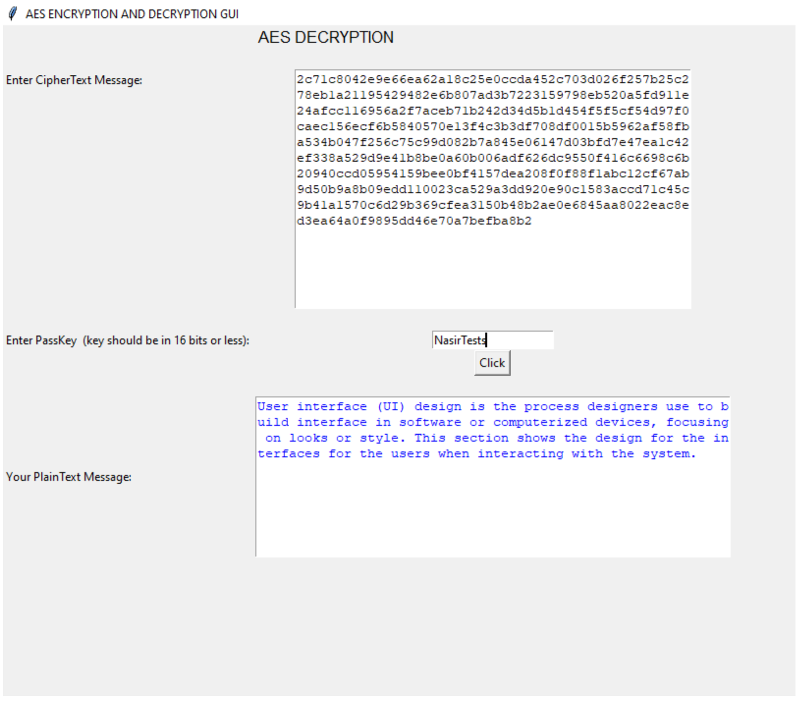
Enter PassKey(key should be in 16 bits or less): NasirTests

Click

Your CipherText Message :

```
2c71c8042e9e66ea62a18c25e0ccda452c703d026f257b25c2
78eb1a21195429482e6b807ad3b7223159798eb520a5fd911e
24afcc116956a2f7aceb71b242d34d5b1d454f5f5cf54d97f0
caec156ecf6b5840570e13f4c3b3df708df0015b5962af58fb
a534b047f256c75c99d082b7a845e06147d03bfd7e47ea1c42
ef338a529d9e41b8be0a60b006adf626dc9550f416c6698c6b
20940ccd05954159bee0bf4157dea208f0f88f1abc12cf67ab
9d50b9a8b09edd110023ca529a3dd920e90c1583accd71c45c
9b41a1570c6d29b369cfea3150b48b2ae0e6845aa8022eac8e
d3ea64a0f9895dd46e70a7befba8b2
```

**Figure 12: Testing the encryption system**



AES ENCRYPTION AND DECRYPTION GUI

AES DECRYPTION

Enter CipherText Message:

```
2c71c8042e9e66ea62a18c25e0ccda452c703d026f257b25c2
78eb1a21195429482e6b807ad3b7223159798eb520a5fd911e
24afcc116956a2f7aceb71b242d34d5b1d454f5f5cf54d97f0
caec156ecf6b5840570e13f4c3b3df708df0015b5962af58fb
a534b047f256c75c99d082b7a845e06147d03bfd7e47ea1c42
ef338a529d9e41b8be0a60b006adf626dc9550f416c6698c6b
20940ccd05954159bee0bf4157dea208f0f88f1abc12cf67ab
9d50b9a8b09edd110023ca529a3dd920e90c1583accd71c45c
9b41a1570c6d29b369cfea3150b48b2ae0e6845aa8022eac8e
d3ea64a0f9895dd46e70a7befba8b2
```

Enter PassKey (key should be in 16 bits or less): NasirTests

Click

Your PlainText Message:

```
User interface (UI) design is the process designers use to b
uild interface in software or computerized devices, focusing
 on looks or style. This section shows the design for the in
terfaces for the users when interacting with the system.
```

**Figure 13: Testing the decryption system**

## 4. Conclusion

Data security and cryptography have received wide publicity owing to its diverse application in securing confidential information/data**.** A key development in the field of data security and cryptography is the effective application of the Text Encryption System utilizing the AES algorithm. This method tends to prove to be an invaluable resource for people and businesses looking to safeguard their confidential text data easily without using online encryptors/decryptors.The complexities of its implementation make it not easily adopted and used by several users. Simplifying and masking these complexities from the users through design of good user friendly interface will be a panacea in encouraging the adoption and use of the algorithms by non-technical users**.**

This research work presents an implementation of a Text Encryption using AES algorithm in python that provides users with a convenient secured method for encrypting and decrypting text. The requirements for the design and development of the implemeted system were explored and identified during the course of this research. No software is flawless or error-free, even though the bulk of the requirements for implementation of a Text Encryption using the AES algorithm were satisfied. Future work will delve into increasing the length of the key from 128 to 192 or 256 to give user enough choice when using the algorithm. Also, other alternatives to Electronic Code Book (ECB) will be explored for a more secure block cipher mode of operation. Any improvement will be straightforward to incorporate because the program is designed with agile framework. To enhance Text Encryption using the AES algorithm, new modules or functions may be added, existing modules altered, or both.

## References

[1] D. R. Stinson (2005). Cryptography: Theory and Practice. CRC Press.

[2] Binary Terms (2024). Advanced Encryption Standard (AES). Retrieved March 10, 2024 from http://binaryterms.com

[3] J. Daemen and V. Rijmen (2001). AES Proposal: Rijndael. Retrieved September10, 2023 from https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf

[4] J. Sha (2023). What is AES Encryption? The complete guide. Retrieved March 12, 2024 from https://www.1kosmos.com/authentication/aes-encryption/

[5] K Amrita, N. Gupta and R. Mishra (2018). An Overview of Cryptanalysis on AES. International Journal of Advance Research in Science and Engineering. Vol7, special issue 01, April 2018. Retrieved April 10, 2024. http://www.ijarse.com/images/fullpdf/1522563469_BIT836ijarse.pdf

[6] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti and E. Roback (2001). Report on the Development of the Advanced Encryption Standard (AES). J Res Natl Inst Stand Technol. 2001 May-Jun; 106(3): 511–577, doi: 10.6028/jres.106.023

[7] Basile (2022).AES Encryption and Decyption in Python: Implementation, Modes and Key Management. Retrieved April 10, 2024 from http://onboard base.com

[8] D. Sirohi (2022) Java AES Encryption and Decryption Retrieved May 10, 2024 from https://medium.com/@deepak.sirohi9188/java-aes-encryption-and-decryption-1b30c9a5d900

[9] H. R. Sharifi and J. Cook (2024) Java AES Encryption and Decryption Retrieved May 12, 2024 from https://www.baeldung.com/java-aes-encryption-decryption

[10] V. Kumar (2023). Encryption and Decryption Using a Symmetric Key in C#. Retrieved May 20, 2024 from https://www.c-sharpcorner.com/article/encryption-and-decryption-using-a-symmetric-key-in-c-sharp/

[11] GitHub (2024) . AES Encryption Python. Retrieved May 20, 2024 from https://github.com/topics/aes-encryption-python

[12] S. Vashistha, K. Soni and V. Jethani (2019). Implementation of Advanced Encryption Standard (AES) Algorithm for Image Encryption. Retrieved April 15, 2024 from https://www.researchgate.net/publication/380178538

[13] K. Nagaraj (2023) Advanced Encryption Standard (AES): A Secure and Efficient Symmetric Encryption Algorithm : Understanding AES Encryption, Key Generation, and Applications. Retrieved May 22, 2024 from https://infosecwriteups.com/advanced-encryption-standard-aes-a-secure-and-efficient-symmetric-encryption-algorithm-319eedb49905?gi=636fd374d24f