Performance Analysis of Python AES Module Using 2^k Factorial Design

Project By:

Shuvo Saha Roy- SAU/CS(M)/2020/22

Mahen Mondol-SAU/CS(M)/2020/09

Submitted To:

Shachi Sharma

Assistant Professor

South Asian University, New Delhi

Submission Date:

12-07-2021

ABSTRACT

Nowadays, information security has become an essential issue for exchanging information in data communication. Encryption algorithm plays a vital role in the information security system. Many algorithm techniques are used to provide data confidentiality and privacy by making the information indecipherable which can be only be decoded or decrypted by the party that possesses the associated key. However, concurrently implementing these algorithm techniques consume a significant amount of computing resources such as CPU time, memory, and power. [1] So it is crucial to find out the effect of the algorithm upon these resources. It will help us to determine significant factors that were affecting the performance of the algorithm. This project provides an evaluation of the Python AES module from pycrypto[2] package, cryptographic algorithms by taking different types of files like mp3, text, image, py scripts, pdf, and video files. A comparison has been conducted among various resources using evaluation metrics such as encryption time, decryption time, RAM size, folder size etc. Simulation results are given to demonstrate the effectiveness of each.

Keywords: Algorithm, Encryption, Decryption, AES, Python

INTRODUCTION

Data is the most important thing in this era. The amount of data is increasing day by day with the use of the internet. So it needs to ensure the safety of data as efficiently as possible. Different encryption technique is used to ensure the safety of data or secure the data. Now Advanced Encryption Standard AES encryption is the most popular encryption technique that is used all over the world. [3] In this project, pycrypto, a python package used to analyze the performance of the AES 256bit module. The implementation code is available on GitHub. 2^k factorial design is used for the performance analysis. Using 2^k factorial design, we can find out significant variables which affect the process or not. If k number of variables/factors are studied to determine/screen the important ones, the total number of treatment combinations for a k number of factors can be calculated. Therefore, this screening technique is known as the 2^K design of experiments. [4] Python programming language used to monitor the resource uses for the AES encryption algorithm. AES (Advanced Encryption Standard) is a symmetric block cipher standardized by NIST. It has a fixed data block size of 16 bytes. Its keys can be 128, 192, or 256 bits long. AES is speedy and secure, and it is the standard for symmetric encryption.

METHODOLOGY

Python programming language with the 2^k factorial design is used to identify the significant factors for AES algorithm implementation.

A. ENCRYPTION AND DECRYPTION WITH AES

Encryption is a process that transforms the original information into an unrecognizable form which is known as ciphertext. This new form of the message is entirely different from the original message. That's why a hacker is not able to read the data as senders use an encryption algorithm. Encryption is usually done using key algorithms. 128, 192, or 256 bits cryptographic keys can be used to encrypt and decrypt data in blocks of 128 bits in the AES algorithm.

The Advanced Encryption Standard (AES) is a Federal Information Processing Standard (FIPS) which was declared after the competition for encryption algorithms held by National Standards and Technology in 2011. AES is a very high-security algorithm [1]. Some algorithms were selected as candidates in the top 5 in a row after Rijndael are Serpent, Twofish, RC6, and MARS algorithms [5]. AES is proven immune to conventional attacks (linear and differential are: resistant to known password analysis, flexible to use in various hardware and software, good for hash functions, suitable for devices that require fast key agility, and ideal for stream ciphers.

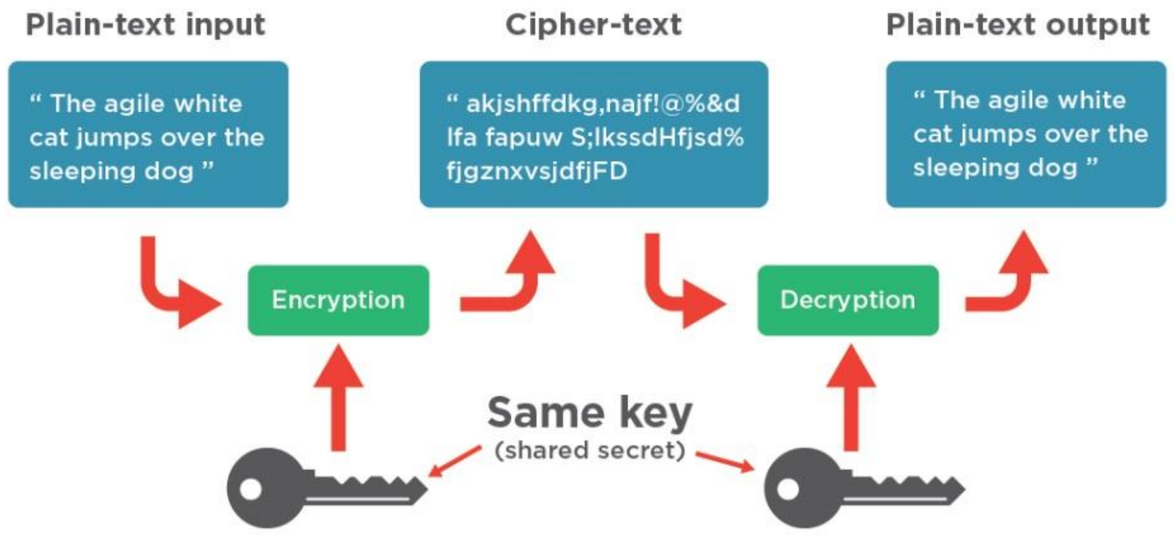

Plain-text input
" The agile white cat jumps over the sleeping dog "
Cipher-text
" akjshffdkg,najf!@%&d lfa fapuw S;lkssdHfjsd% fjgznxvsjdfjFD
Plain-text output
" The agile white cat jumps over the sleeping dog "
Encryption
Decryption
Same key
(shared secret)

Figure 1: Encryption and Decryption Process [6]

Decryption is converting encoded/encrypted data in a readable form and understood by a human or a computer. This method is performed by un-encrypting the text manually or using keys to encrypt the original data.

B. PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Advantages of Python [7]

- Presence of third-party modules
- Extensive support libraries
- Open source and community development
- Versatile, Easy to read, learn and write
- User-friendly data structures
- High-level language
- Portable and Interactive

- Ideal for prototypes – provide more functionality with less coding

C. PYCRYPTO

Python crypto module provides a simple interface to symmetric Gnu Privacy Guard (gpg) encryption and decryption for one or more files on UNIX and Linux platforms. It runs on top of gpg and requires a gpg install on your system. Encryption is performed with the AES256 cipher algorithm. Benchmarks relative to default gpg settings are available for text and binary file mime types.

D. DATA SELECTION

To collect the data, two steps are taken into consideration.

1. Installing essential programs and modules.

The first step is to install python into the machine. To run AES encryption and decryption, Crypto module is needed, and it has installed using the following command:

```
pip install crypto
```

Another module used to compute the system information, which is psutil, available by default.

2. Writing script to compute the effect of total CPU utilization, main memory (ram utilization, the execution time of the AES encryption algorithm.

E. FACTOR IDENTIFICATION

To apply 2^k factorial design, it's essential to identify factors and parameters. CPU utilization, ram utilization, ram usage, execution time, total ram, CPU usage, disk usage, etc are shown in the data set; these are the performance matrices. Some of them are the system information, which has no direct relationship with the program. Also, CPU utilization is 0 percent for the program. That's mean this program is not CPU sensitive. Then there is three-factor that may affect the performance of the AES algorithm.

- **Size of folder**

  As encryption and decryption operation has been performed over different files and folders, it will be a factor.

- **Sizes of RAM**

  Two types of RAM used for this experiment. So it will be another factor.

- **Operation type (Encryption, Decryption)**

  Two types of operation is performed.

As encryption and decryption are performed at bytes level, so the format of the file does not matter and has no impact as a factor. On the other hand, for simplifying calculation, folder sizes are considered rather than each file size.

cpu_utlilization %	memory_ utilization %	memory_ usage MB	directory size MB	time_taken sec	operation_ type	total cpu in use %	total ram GB	ram in use %	disk_usage %
0	0.198752403	16.0234375	22.83258343	0.338886261	encription	19.1	7.867298126	40.3	65.9
0	0.198655427	16.015625	732.4565725	14.90413904	encription	28	7.867298126	38	68.1
0	0.400014879	16.38671875	27.05152798	0.571216106	encription	7.5	3.99956131	40.1	61.2
0	0.391526247	16.04296875	175.2835598	5.858032465	encription	29.5	3.99956131	39.8	61.2
0	0.199867628	16.11328125	22.83254623	0.200044632	decryption	39.2	7.867298126	38.1	65.7
0	0.199188795	16.05859375	732.4565115	9.689086676	decryption	29.2	7.867298126	38.5	68.1
0	0.39047709	16	27.05177689	0.651234627	decryption	13.4	3.99956131	39.8	61.2
0	0.391621624	16.046875	175.2836447	3.669473648	decryption	33.2	3.99956131	39.8	61.2

Table 1: Output and Selected dataset for 2^k factorial design (GitHub). Some properties belong to the total system. So these are ignored during 2^k factorial design

F. DESIGNING WORKLOAD

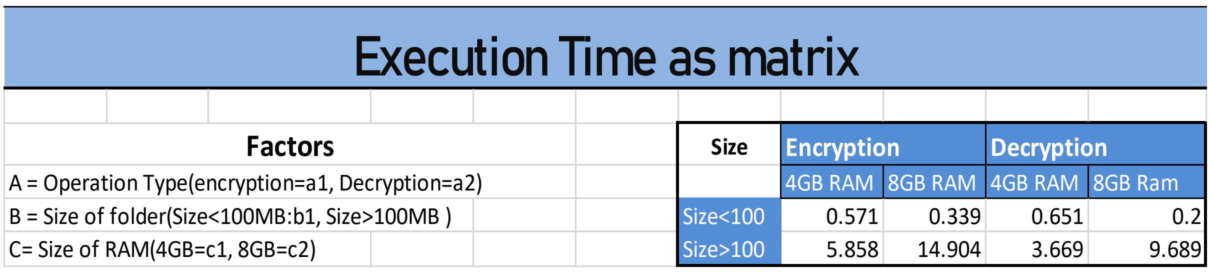
For the analysis, the existing system was used as a real workload. Several requests have been made by running the script.py file. We categorized different folders with different sizes to run the experiment. There are two types of operation- encryption and decryption, two different folder sizes, and two systems with two different RAM sizes (8GB and 4GB).

Folder Size	**Encryption**		**Decryption**	
	4GB RAM	8GB RAM	4GB RAM	8GB Ram
Size<100	Selected matrix	Selected matrix	Selected matrix	Selected matrix
Size>100	Selected matrix	Selected matrix	Selected matrix	Selected matrix

Table 2: Key idea for 2^k factorial design

RESULT & DISCUSSION

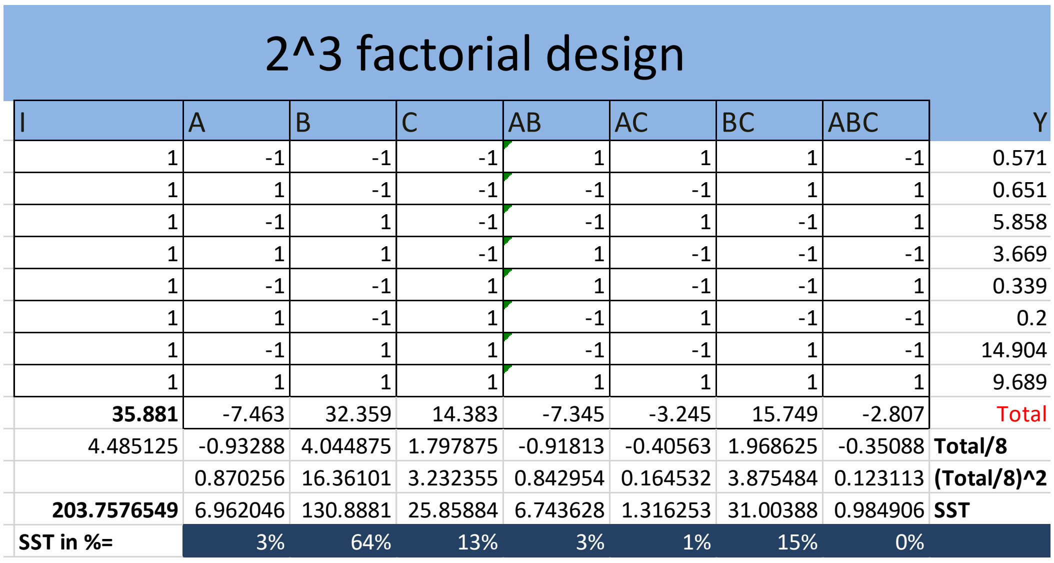
As 2^k factorial design with three factors, 2^3 = 8 experiments are conducted. Four encryption and four decryption are performed in two different systems. Three matrices are taken into consideration as output variables from the dataset in this project work. These are Execution time- Time taken to execute the program, Memory usage, Memory utilization.


Execution Time as matrix

Factors
A = Operation Type(encryption=a1, Decryption=a2)
B = Size of folder(Size<100MB:b1, Size>100MB)
C= Size of RAM(4GB=c1, 8GB=c2)

Size	Encryption		Decryption	
	4GB RAM	8GB RAM	4GB RAM	8GB Ram
Size<100	0.571	0.339	0.651	0.2
Size>100	5.858	14.904	3.669	9.689

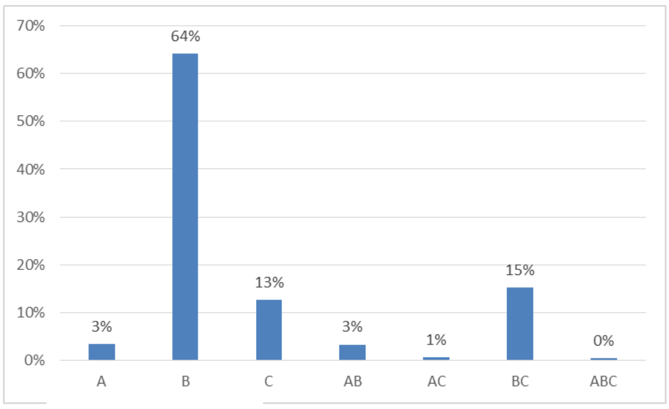
(a)

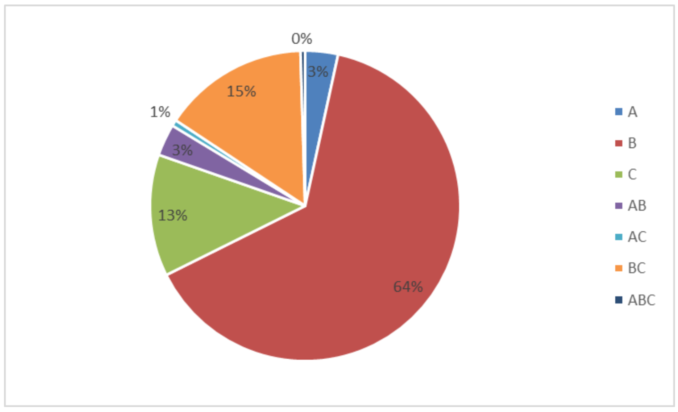

2^3 factorial design

I	A	B	C	AB	AC	BC	ABC	Y
1	-1	-1	-1	1	1	1	-1	0.571
1	1	-1	-1	-1	-1	1	1	0.651
1	-1	1	-1	-1	1	-1	1	5.858
1	1	1	-1	1	-1	-1	-1	3.669
1	-1	-1	1	1	-1	-1	1	0.339
1	1	-1	1	-1	1	-1	-1	0.2
1	-1	1	1	-1	-1	1	-1	14.904
1	1	1	1	1	1	1	1	9.689
35.881	-7.463	32.359	14.383	-7.345	-3.245	15.749	-2.807	Total
4.485125	-0.93288	4.044875	1.797875	-0.91813	-0.40563	1.968625	-0.35088	**Total/8**
	0.870256	16.36101	3.232355	0.842954	0.164532	3.875484	0.123113	**(Total/8)^2**
203.7576549	6.962046	130.8881	25.85884	6.743628	1.316253	31.00388	0.984906	**SST**
SST in %=	3%	64%	13%	3%	1%	15%	0%	

(b)

Table-3(a,b): Shows the factorial design for execution time





(a) (b)

Figure 2(a,b): Graphical representation of 2^k factorial design for execution time

Table 3 and figure 2(github) show us B means folder size is the most important factor affecting the execution time for encryption and decryption. And BC together is the next significant factor, then C means RAM size is the most significant factor, and so on.

Memory usage(MB) as matrix

Factors
A = Operation Type(encryption=a1, Decryption=a2)
B = Size of folder(Size<100MB:b1, Size>100MB)
C= Size of RAM(4GB=c1, 8GB=c2)

Size	Encryption		Decryption	
	4GB RAM	8GB RAM	4GB RAM	8GB Ram
Size<100	16.388	16.023	16	16.113
Size>100	16.043	16.016	16.047	16.059

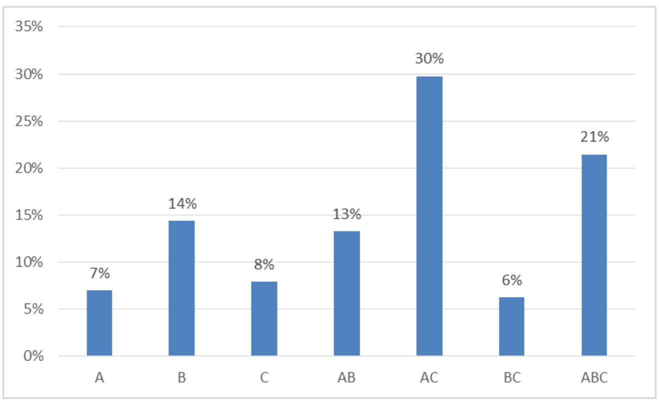
(a)

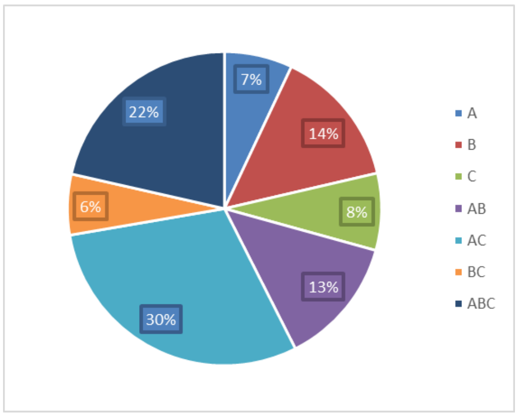
2^3 factorial design

I	A	B	C	AB	AC	BC	ABC	Y
1	-1	-1	-1	1	1	1	-1	16.388
1	1	-1	-1	-1	-1	1	1	16
1	-1	1	-1	-1	1	-1	1	16.043
1	1	1	-1	1	-1	-1	-1	16.047
1	-1	-1	1	1	-1	-1	1	16.023
1	1	-1	1	-1	1	-1	-1	16.113
1	-1	1	1	-1	-1	1	-1	16.016
1	1	1	1	1	1	1	1	16.059
128.689	-0.251	-0.359	-0.267	0.345	0.517	0.237	-0.439	Total
16.08613	-0.03137	-0.04488	-0.03338	0.043125	0.064625	0.029625	-0.05488	**Total/8**
	0.000984	0.002014	0.001114	0.00186	0.004176	0.000878	0.003011	**(Total/8)^2**
0.112297	0.007875	0.01611	0.008911	0.014878	0.033411	0.007021	0.02409	**SST**
SST in %=	7%	14%	8%	13%	30%	6%	21%	

(b)

Table 4(a,b): Shows the factorial design for RAM usage





(a) **(b)**

Figure 3(a,b): Graphical representation of 2^k factorial design for memory usage

Table 4 and figure 3(github) shows us the analytical result 2^k factorial design for RAM usage. When RAM usage has used the combination of A(Operation type) and C(RAM size) shows the highest impact on performance.

Memory utilization as matrix

Factors
A = Operation Type(encryption=a1, Decryption=a2)
B = Size of folder(Size<100MB:b1, Size>100MB)
C= Size of RAM(4GB=c1, 8GB=c2)

Size	Encryption		Decryption	
	4GB RAM	8GB RAM	4GB RAM	8GB Ram
Size<100	0.4	0.199	0.39	0.199
Size>100	0.392	0.199	0.392	0.199

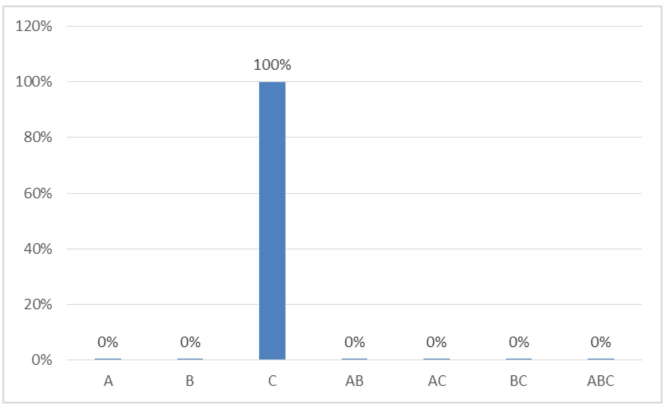
(a)

2^3 factorial design

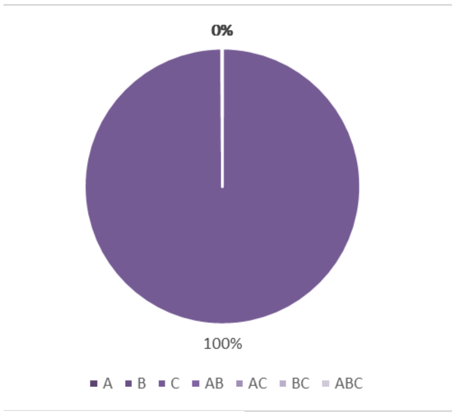
I	A	B	C	AB	AC	BC	ABC	Y
1	-1	-1	-1	1	1	1	-1	0.4
1	1	-1	-1	-1	-1	1	1	0.39
1	-1	1	-1	-1	1	-1	1	0.392
1	1	1	-1	1	-1	-1	-1	0.392
1	-1	-1	1	1	-1	-1	1	0.199
1	1	-1	1	-1	1	-1	-1	0.199
1	-1	1	1	-1	-1	1	-1	0.199
1	1	1	1	1	1	1	1	0.199
2.37	-0.01	-0.006	-0.778	0.01	0.01	0.006	-0.01	Total
0.29625	-0.00125	-0.00075	-0.09725	0.00125	0.00125	0.00075	-0.00125	**Total/8**
	0.0000	0.0000	0.0095	0.0000	0.0000	0.0000	0.0000	**(Total/8)^2**
0.075719	0.0000	0.0000	0.0757	0.0000	0.0000	0.0000	0.0000	**SST**
SST in %=	0%	0%	100%	0%	0%	0%	0%	

(b)

Table 4(a,b): Shows the factorial design for RAM usage



(a)



(b)

Figure 4(a,b): Graphical representation of 2^k factorial design for memory utilization

Table 4 and figure 4 show (github) that when we take memory utilization as matric, it depends on RAM size. The size of the folder, operation type have no impact on the performance of the program. The program is totally RAM-sensitive.

CONCLUSION & FUTURE WORK

This project shows us that folder size and RAM is the most important factor, to measure the performance of the Python AES algorithm implementation. In 2nd analysis, it indicates that RAM usage depends on the operation type and RAM size mostly.

And in the 3rd analysis shows memory utilization totally depends on the memory size. So depending on these three results, we can say that the performance of the PyCrypto package AES (module) algorithm depends on folder size and ram mostly. And, this experiment shows us, it can be extended by using different systems or machines. Because here only 2 type system used. We can continue it for different ram size systems and different CPU configurations. Also, it can be done on other programming languages with appropriate packages. This will help us to understand the impact of algorithm implementation of different languages on different machines. After that, we can pick up the best one for our operational purpose.

REFERENCES

1. M. Panda, "Performance analysis of encryption algorithms for security," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016, pp. 278-284, doi: 10.1109/SCOPES.2016.7955835.
2. pycrypto. PyPI. ((2021, June 30).). https://pypi.org/project/pycrypto/.
3. Kumar P dan Rana S B 2016 Development of modified AES algorithm for data security Opt. - Int. J. Light Electron Opt. **127** 2341–5
4. The Open Educator - 1. What is 2K Design. (n.d.). https://www.theopeneducator.com/doe/2K-Factorial-Design-of-Experiments/What-is-2K-Factorial-Design-of-Experiments.
5. Karale S N, Pendke K dan Dahiwale P 2015 The survey of various techniques & algorithms for SMS security *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (IEEE) hal 1–6
6. Transcend. (n.d.). *AES Encryption*. AES Encryption - Transcend Information, Inc. https://www.transcend-info.com/Embedded/Essay-15.
7. *Python Language advantages and applications*. GeeksforGeeks. (2021, June 30). https://www.geeksforgeeks.org/python-language-advantages-applications/.