

A Novel Time Efficient Approach to Enhance Security of Text Data in CSV file format on Cloud Environment

^[1]Mr. Ankur N. Shah (0000-0001-5150-6202), ^[2]Dr. Vishal Dahiya (0000-0001-8290-385X),
^[3]Dr. Jay A. Dave³ (0000-0003-3274-8702)

^[1] Research Scholar, Indus University, Ahmedabad, India.

^[2] Director and Professor, Sardar Vallabhbhai Global University, Ahmedabad, India.

^[3] Associate Professor, Silver Oak University, Ahmedabad, India.

E-mail: ^[1]ankur11586@gmail.com, ^[2]director_imca@svgu.ac.in,
^[3]jay.dave4u@gmail.com

Abstract. Now a day's huge amount of data is generated via various media like mobile phone, books, internet, satellite data etc. To handle this kind of big data is very difficult task for any organization. To work with cloud computing for handling big data is one of the good solutions. These both technologies big data and cloud computing work together and can give number of benefits to users like providing amenity available on demand, availability over the internet, comfortable management, comfortable analysis of large volume of data, cost savings etc. still we find that many companies are not using these technologies then question may arise in our mind what is the main reason behind it? There are many challenges or disadvantages with these two technologies like the various demanding situations to big records are price, statistics pleasant, fast alternate, professional guy electricity requirement, infrastructure need, shifting facts onto big information platform, want for synchronization across statistics assets and maximum essential is protection of information.. All of these challenges will not be easily solved numerous research projects are being carried out to solve these problems. In this paper we provide information about various works done in the field of security of big data on cloud computing in the form of literature survey and ultimately we provide three different algorithms which secure simple text or csv file data.

***Throughout this paper we have used term base paper algorithm meaning that we are using reference paper [10] algorithm of Mohammad Anwar Hossain and et al.

Keywords: big data, cloud computing, security.

1 Big Data & Cloud Computing

"Big data as its name suggest it is a huge amount of data that cannot be processed and managed by traditional database management tool. These data is also may be structured data, semi-structured data or unstructured data" [13] "Various v's are: volume, variability, velocity, veracity, variety, value, validity etc." [14] "In Cloud Computing, the word Cloud means The Internet, so Cloud Computing means a type of computing in which services are delivered through the Internet." [11] "Cloud computing provides mainly software as a service (SaaS), infrastructure as a service (IaaS), and platform as a service (PaaS)" [12]. The benefits we gain if we combine these two technologies are: "Better Performance, Better Efficiency, Cost Reduction, More Flexibility, More Scalability, Less Management & Reduced Overhead" [18] The challenges with these two technologies are: "Access Control, Honey pot nodes, security" [15] "Distributed Data, Internode Communication, Administrative Rights for Nodes" [16] "Data Storage, Variety of Data, Data Transfer" [17]

2 Literature Survey

In literature survey part we have refer more than 50 papers here we have put 10 best out of that more than 50 papers. E. K. Subramanian, et. al [1], find problem of existing encryption methods access patterns can leak sensitive information and propose system uses an Elliptic curve with Diffie Hellman algorithm for encryption and decryption of data to improve data security in cloud. Uma Narayanan, et. al [2], find problem of big data processing, analytics and storage is secured using cryptography algorithms, which are not appropriate for big data protection over cloud and uses sh3 hashing algorithm for improving privacy and data security in E-healthcare

application. N.Keerthana, et. al [3], find problem of Cloud provider stores the shared data in the large data centers outside the trust domain of the data owner, which may trigger the problem of data confidentiality and propose solution using Secret sharing group key management protocol (SSGK) & RSA which Improved privacy and security of sharing data on cloud and saves 12% storage spaces. S. Ramasamy, et. al [4], find big data security problem in cloud and conventional security systems are used encryption and decryption for providing security and propose Cluster based multilayer user authentication which provide security of data in distributed cloud computing environments. P Geetha Niharika, et. al [5], find problem of Cloud providers store the shared data outside the trust jurisdiction of the data owner in large data centers, which may trigger the issue of data congeniality and propose secret key management group sharing protocol with time constraint which provide more safety to the data. Uma Narayanan, et. al [6], find problem that Hadoop is still developing. There is much powerlessness found in Hadoop, which can scrutinize the security of the sensitive data that undertakings have entrusted on it and propose a solution using salsa20 and ranger key management service which ensures confidentiality, authentication and access control on the hadoop server. Hossein Abroshan [7], find problem that complex cryptographic algorithm is not helpful in cloud environment as computing speed is essential in cloud environment and propose a solution using improve blowfish algorithm and elliptic curve based algorithm which improves throughput, execution time and memory consumption. Nikhil Dwivedi, et. al [8], Finding lack of security due to use of single algorithm like AES, 3DES, RSA etc. and propose a solution using combine AES and RSA algorithm which provide more security to data. Jayachander Surbiryala [9], find problem of security measures provided by the service providers might not be enough to secure the data in the cloud and propose method to divide big data file into small file so hackers even hack the file cannot get full data and use index technique using keyword to join all file and get full data. Mohammad Anwar Hossain, et. al [10], find the problem that need of algorithm which is less time consuming and can encrypt more data at a time and propose solution using own 192 bit symmetric key algorithm.

Here we found that none of the research paper method is working for file encryption and most of the research paper is working on 128 bits key size. Mohammad Anwar Hossain, et. al [10], working on 192 bits of data but it can encrypt only 24 bytes of input text (192 bits). Here we modified base algorithm using bit permutation instead of mix column operation, so algorithm can work more efficiently. And we give name as base paper algorithm modification - 1. Our base paper algorithm modification - 1 is also able to encrypt csv files. Then we modified algorithm with different matrixes generation and changes accordingly which is shown here as base paper algorithm modification - 2. Then we modified algorithm to work with 128 bits of data which is shown here as base paper algorithm modification - 3.

3 Base Paper Algorithm Modification - 1

Following figure-1 shows steps of base paper algorithm modification - 1 for encryption process.

1. Input plain text of any size and key text of size 24 bytes.
2. Generate 4*6 Matrixes.
3. Convert message characters into ASCII equivalent.
4. Apply round function
 - A. Perform shift row operation.
[Here Row1 – 3 bit left circular shift, Row2 – 2 bit left circular shift,
Row3 – 1 bit left circular shift, Row4 – 0 bit left circular shift]
 - B. Perform Permutation – 1.
[Interchange Column C1 by C2 and C2 by C1, Interchange Column C3 by C4 and C4 by C3,
Interchange Column C5 by C6 and C6 by C5, Interchange Row R1 by R3 and R3 by R1,
Interchange Row R2 by R4 and R4 by R2]
 - C. Reverse the whole block of matrix
 - D. Perform Permutation – 2.
[Interchange Column C5 by C6 and C6 by C5, Interchange Row R1 by R4 and R4 by R1,
XOR between Column C1 and C3, XOR between Column C2 and C4]
 - E. Convert into Hexadecimal Values
 - F. Apply S-Box

- G. Convert into decimal values
- H. Perform Bit Permutation
- I. Calculate Key.
[Generate 4*6 block matrix for key text of 24 bytes, convert key characters into ASCII]
- J. Perform XOR between resultant of Bit permutation and calculated key matrix.
- 5. Arrange matrix values as plain text.
- 6. Using base 32/64/128 bits encoder convert plain text into corresponding encoding characters.

Figure-1: Steps of base paper algorithm modification - 1 for encryption process

Firstly this encryption algorithm takes 24 bytes of input text after then it arranges it in 4*6 matrix form. In next step it converts it into ASCII values. In next step it will perform row shifting operations. In next step algorithm performs permutation-1 where row and column interchanging is carried out. In next step of algorithm it reverses the whole block of matrix. In next step of algorithm it will perform permutation-2 where row and column interchanging and XOR operation is performed. In next step of algorithm it converts values into Hexadecimal values. In next step of algorithm it applies S-box replacement. In next step of algorithm it converts it into decimal values. In next step of algorithm it performs bit permutation operation. In next step of algorithm it arranges key text into 4*6 matrixes and then converts it into ASCII values. In next step of algorithm it performs XOR between result of bit permutation operation and 4*6 matrixes of ASCII values of key text. In next step of algorithm it arranges matrixes values as plain text. In final step of algorithm it use base 32/64/128 bits encoder to convert plain text to encoding characters.

Following figure-2 shows steps of base paper algorithm modification - 1 for decryption process.

- 1. Receive cipher text for encryption process.
- 2. Decode cipher text using base 32/64/128 bits decoder.
- 3. Generate 4*6 block matrix for decoded value.
- 4. Calculate Key.
[Generate 4*6 block matrix for key text of 24 bytes, convert key characters into ASCII]
- 5. Apply round function
 - A. Perform XOR between matrix of decoded value and calculated key matrix.
 - B. Perform bit permutation operation.
 - C. Convert into Hexadecimal Values
 - D. Apply reverse S-Box
 - E. Convert into decimal values
 - F. Perform Permutation – 4
[X = XOR between Column C1 and C3, Y = XOR between Column C2 and C4,
Replace C1 by X and C2 by Y, Interchange the Row R1 by R4 and R4 by R1,
Interchange the Column C5 by C6 and C6 by C5]
 - G. Reverse the whole block of matrix.
 - H. Perform Permutation – 3
[Interchange Row R2 by R4 and R4 by R2, Interchange Row R1 by R3 and R3 by R1, Interchange
Column C5 by C6 and C6 by C5, Interchange Column C3 by C4 and C4 by C3,
Interchange Column C1 by C2 and C2 by C1]
 - I. Perform Inverse shift rows.
- 6. Convert into ASCII.
- 7. Arrange ASCII into plaintext

Figure-2: Steps of base paper algorithm modification - 1 for decryption process

Firstly this decryption algorithm receives cipher text from encryption process. In next step of algorithm it use base 32/64/128 bits decoder to decode the cipher text. In next step of algorithm it arranges decoded text in 4*6 matrix form. In next step of algorithm it arranges key text into 4*6 matrixes and then converts it into ASCII values. In next step of algorithm it performs XOR between 4*6 matrix of decoded values and 4*6 matrixes of

ASCII values of key text. In next step of algorithm it performs bit permutation operation. . In next step of algorithm it converts values into Hexadecimal values. In next step of algorithm it applies S-box replacement. In next step of algorithm it converts it into decimal values. In next step of algorithm it will perform permutation-4 where row and column interchanging and XOR operation is performed. In next step of algorithm it reverses the whole block of matrix. . In next step algorithm performs permutation-3 where row and column interchanging is carried out. In next step it will perform inverse row shifting operations. In next step it converts it into ASCII values. In next step of algorithm arrange ASCII to plain text.

4 Base Paper Algorithm Modification - 2

Following figure-3 shows steps of base paper algorithm modification - 2 for encryption process. Here we modified the algorithm to work with 6*4 matrixes rather than 4*6 matrixes.

1. Input plain text of any size and key text of size 24 bytes.
2. Generate 6*4 Matrixes.
3. Convert message characters into ASCII equivalent.
4. Apply round function
 - A. Perform shift row operation.
[Here Row1 – 5 bit left circular shift, Row2 – 4 bit left circular shift,
Row3 – 3 bit left circular shift, Row4 – 2 bit left circular shift,
Row5 – 1 bit left circular shift, Row6 – 0 bit left circular shift]
 - B. Perform Permutation – 1.
[Interchange Column C1 by C2 and C2 by C1, Interchange Column C3 by C4 and C4 by C3,
Interchange Row R1 by R3 and R3 by R1, Interchange Row R2 by R4 and R4 by R2, Interchange
Row R5 by R6 and R6 by R5]
 - C. Reverse the whole block of matrix
 - D. Perform Permutation – 2.
[Interchange Row R5 by R6 and R6 by R5, Interchange Row R1 by R4 and R4 by R1,
XOR between Column C1 and C3, XOR between Column C2 and C4]
 - E. Convert into Hexadecimal Values
 - F. Apply S-Box
 - G. Convert into decimal values
 - H. Perform Bit Permutation
 - I. Calculate Key.
[Generate 6*4 block matrix for key text of 24 bytes, convert key characters into ASCII]
 - J. Perform XOR between resultant of Bit permutation and calculated key matrix.
5. Arrange matrix values as plain text.
6. Using base32/64/128 encoder convert plain text into corresponding encoding characters.

Figure-3: Steps of base paper algorithm modification - 2 for encryption process

Firstly this encryption algorithm takes 24 bytes of input text after then it arranges it in 6*4 matrix form. In next step it converts it into ASCII values. In next step it will perform row shifting operations. In next step algorithm performs permutation-1 where row and column interchanging is carried out. In next step of algorithm it reverses the whole block of matrix. In next step of algorithm it will perform permutation-2 where row and column interchanging and XOR operation is performed. In next step of algorithm it converts values into Hexadecimal values. In next step of algorithm it applies S-box replacement. In next step of algorithm it converts it into decimal values. In next step of algorithm it performs bit permutation operation. In next step of algorithm it arranges key text into 6*4 matrixes and then converts it into ASCII values. In next step of algorithm it performs XOR between result of bit permutation operation and 6*4 matrixes of ASCII values of key text. In next step of algorithm it arranges matrixes values as plain text. In final step of algorithm it use base 32/64/128 bits encoder to convert plain text to encoding characters.

Following figure-4 shows steps of base paper algorithm modification - 2 for decryption process.

1. Receive cipher text for encryption process.
2. Decode cipher text using base32/64/128 decoder.
3. Generate 6*4 block matrix for decoded value.
4. Calculate Key.
[Generate 6*4 block matrix for key text of 24 bytes, convert key characters into ASCII]
5. Apply round function
 - A. Perform XOR between matrix of decoded value and calculated key matrix.
 - B. Perform bit permutation operation.
 - C. Convert into Hexadecimal Values
 - D. Apply reverse S-Box
 - E. Convert into decimal values
 - F. Perform Permutation – 4
[X = XOR between Column C1 and C3, Y = XOR between Column C2 and C4,
Replace C1 by X and C2 by Y, Interchange the Row R5 by R6 and R6 by R5,
Interchange the Row R1 by R4 and R4 by R1]
 - G. Reverse the whole block of matrix.
 - H. Perform Permutation – 3
[Interchange Row R5 by R6 and R6 by R5, Interchange Row R2 by R4 and R4 by R2,
Interchange Row R1 by R3 and R3 by R1, Interchange Column C3 by C4 and C4 by C3,
Interchange Column C1 by C2 and C2 by C1]
 - I. Perform Inverse shift rows.
6. Convert into ASCII.
7. Arrange ASCII into plaintext

Figure-4: Steps of base paper algorithm modification - 2 for decryption process

Firstly this decryption algorithm receives cipher text from encryption process. In next step of algorithm it use base 32/64/128 bits decoder to decode the cipher text. In next step of algorithm it arranges decoded text in 6*4 matrix form. In next step of algorithm it arranges key text into 6*4 matrixes and then converts it into ASCII values. In next step of algorithm it performs XOR between 6*4 matrix of decoded values and 6*4 matrixes of ASCII values of key text. In next step of algorithm it performs bit permutation operation. . In next step of algorithm it converts values into Hexadecimal values. In next step of algorithm it applies S-box replacement. In next step of algorithm it converts it into decimal values. In next step of algorithm it will perform permutation-4 where row and column interchanging and XOR operation is performed. In next step of algorithm it reverses the whole block of matrix. . In next step algorithm performs permutation-3 where row and column interchanging is carried out. In next step it will perform inverse row shifting operations. In next step it converts it into ASCII values. In next step of algorithm arrange ASCII to plain text.

5 Base Paper Algorithm Modification - 3

Following figure-5 shows steps of base paper algorithm modification - 3 for encryption process. Here we modified the algorithm to work with 128 bits of data.

1. Input plain text of any size and key text of size 16 bytes.
2. Generate 4*4 Matrixes.
3. Convert message characters into ASCII equivalent.
4. Apply round function
 - A. Perform shift row operation.
[Here Row1 – 3 bit left circular shift, Row2 – 2 bit left circular shift,
Row3 – 1 bit left circular shift, Row4 – 0 bit left circular shift]
 - B. Perform Permutation – 1.
[Interchange Column C1 by C2 and C2 by C1, Interchange Column C3 by C4 and C4 by C3,
Interchange Row R1 by R3 and R3 by R1, Interchange Row R2 by R4 and R4 by R2]
 - C. Reverse the whole block of matrix

- D. Perform Permutation – 2.
[Interchange Column C1 by C4 and C4 by C1, Interchange Row R1 by R4 and R4 by R1, XOR between Column C1 and C3, XOR between Column C2 and C4]
- E. Convert into Hexadecimal Values
- F. Apply S-Box
- G. Convert into decimal values
- H. Perform Bit Permutation
- I. Calculate Key.
[Generate 4*4 block matrix for key text of 16 bytes, convert key characters into ASCII]
- J. Perform XOR between resultant of Bit permutation and calculated key matrix.
5. Arrange matrix values as plain text.
6. Using base32/64/128 encoder convert plain text into corresponding encoding characters.

Figure-5: Steps of base paper algorithm modification - 3 for encryption process

Firstly this encryption algorithm takes 16 bytes of input text after then it arranges it in 4*4 matrix form. In next step it converts it into ASCII values. In next step it will perform row shifting operations. In next step algorithm performs permutation-1 where row and column interchanging is carried out. In next step of algorithm it reverses the whole block of matrix. In next step of algorithm it will perform permutation-2 where row and column interchanging and XOR operation is performed. In next step of algorithm it converts values into Hexadecimal values. In next step of algorithm it applies S-box replacement. In next step of algorithm it converts it into decimal values. In next step of algorithm it performs bit permutation operation. In next step of algorithm it arranges key text into 4*4 matrixes and then converts it into ASCII values. In next step of algorithm it performs XOR between result of bit permutation operation and 4*4 matrixes of ASCII values of key text. In next step of algorithm it arranges matrixes values as plain text. In final step of algorithm it use base 32/64/128 bits encoder to convert plain text to encoding characters.

Following figure-6 shows steps of base paper algorithm modification - 3 for decryption process.

1. Receive cipher text for encryption process.
2. Decode cipher text using base32/64/128 decoder.
3. Generate 4*4 block matrix for decoded value.
4. Calculate Key.
[Generate 4*4 block matrix for key text of 16 bytes, convert key characters into ASCII]
5. Apply round function
 - A. Perform XOR between matrix of decoded value and calculated key matrix.
 - B. Perform bit permutation operation.
 - C. Convert into Hexadecimal Values
 - D. Apply reverse S-Box
 - E. Convert into decimal values
 - F. Perform Permutation – 4
[X = XOR between Column C1 and C3, Y = XOR between Column C2 and C4, Replace C1 by X and C2 by Y, Interchange the Row R1 by R4 and R4 by R1, Interchange the Column C1 by C4 and C4 by C1]
 - G. Reverse the whole block of matrix.
 - H. Perform Permutation – 3
[Interchange Row R2 by R4 and R4 by R2, Interchange Row R1 by R3 and R3 by R1, Interchange Column C3 by C4 and C4 by C3, Interchange Column C1 by C2 and C2 by C1]
 - I. Perform Inverse shift rows.
6. Convert into ASCII.
7. Arrange ASCII into plaintext

Figure-6: Steps of base paper algorithm modification - 3 for decryption process

Firstly this decryption algorithm receives cipher text from encryption process. In next step of algorithm it use base 32/64/128 bits decoder to decode the cipher text. In next step of algorithm it arranges decoded text in

4*4 matrix form. In next step of algorithm it arranges key text into 4*4 matrixes and then converts it into ASCII values. In next step of algorithm it performs XOR between 4*4 matrix of decoded values and 4*4 matrixes of ASCII values of key text. In next step of algorithm it performs bit permutation operation. . In next step of algorithm it converts values into Hexadecimal values. In next step of algorithm it applies S-box replacement. In next step of algorithm it converts it into decimal values. In next step of algorithm it will perform permutation-4 where row and column interchanging and XOR operation is performed. In next step of algorithm it reverses the whole block of matrix. . In next step algorithm performs permutation-3 where row and column interchanging is carried out. In next step it will perform inverse row shifting operations. In next step it converts it into ASCII values. In next step of algorithm arrange ASCII to plain text.

6 System requirement for implementation of modified algorithms (1 & 2 & 3)

The proposed algorithms are implemented in python language. To implement it successfully we required following.

Operating System = Windows 7 or higher version

Processor = I3 or higher version

RAM = 4 GB or more

Language = Python

Python Modules

numpy = 1.21.6

pandas = 0.25.3

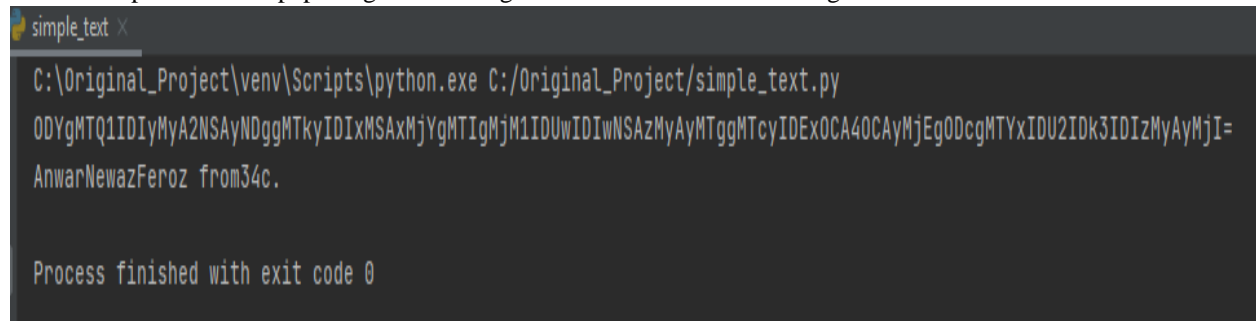
python-dateutil = 2.8.2

pytz = 2022.7.1

six = 1.16.0

7 Results and Discussion for Base Paper Algorithm Modification - 1

First we implement base paper algorithm and got result as shown in below figure – 7.

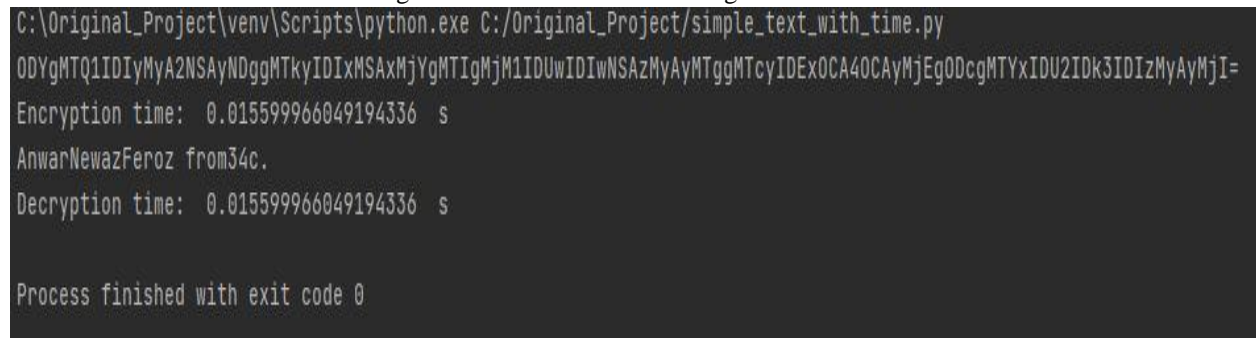


```
simple_text x
C:\Original_Project\venv\Scripts\python.exe C:/Original_Project/simple_text.py
ODYgMTQ1IDIyMyA2NSAyNDggMTkyIDIxMSAxMjYgMTIgMjM1IDUwIDUwNSAzMyAyMTggMTcyIDExOCA4OCAyMjEgODcgMTYxIDU2IDk3IDZlMyAyMjI=
AnwarNewazFeroz from34c.

Process finished with exit code 0
```

Figure-7: Result of base paper.

Then we added time factor to it and got result as shown in below figure – 8.



```
C:\Original_Project\venv\Scripts\python.exe C:/Original_Project/simple_text_with_time.py
ODYgMTQ1IDIyMyA2NSAyNDggMTkyIDIxMSAxMjYgMTIgMjM1IDUwIDUwNSAzMyAyMTggMTcyIDExOCA4OCAyMjEgODcgMTYxIDU2IDk3IDZlMyAyMjI=
Encryption time: 0.015599966049194336 s
AnwarNewazFeroz from34c.
Decryption time: 0.015599966049194336 s

Process finished with exit code 0
```

Figure-8: Result of base paper with time.

Then we execute for another input as shown in below figure – 9.

```
C:\Original_Project\venv\Scripts\python.exe C:/Original_Project/simple_text_with_time.py
MjEzIDg0IDIyMyA0MSAzMCAxOTIgNzUgMTU4IDE0OCAYMzUgNTAgMTkxIDE2NiAyNTIgMTU0IDEyOAAzMyA1NCAYNTQgMTk1IDIXMyAzMSAxMTggMjIy
Encryption time: 0.015599966049194336 s
Ankur Shah from Wadhwan.
Decryption time: 0.015600204467773438 s

Process finished with exit code 0
```

Figure-9: Result for another input to base paper with time.

Then we modified input type to csv file and use same algorithm we got result as shown in below figure – 10 & 11.

File Size	Base Paper Algorithm	
	Encryption (Time In Minutes)	Decryption (Time In Minutes)
154 KB	0.5	0.7
466 KB	0.7	0.7
1.7 MB	2.32	2.62
5.6 MB	12.58	15.25
23.7 MB	43.5	53

Figure-10: Result for csv file with different sizes for base paper.

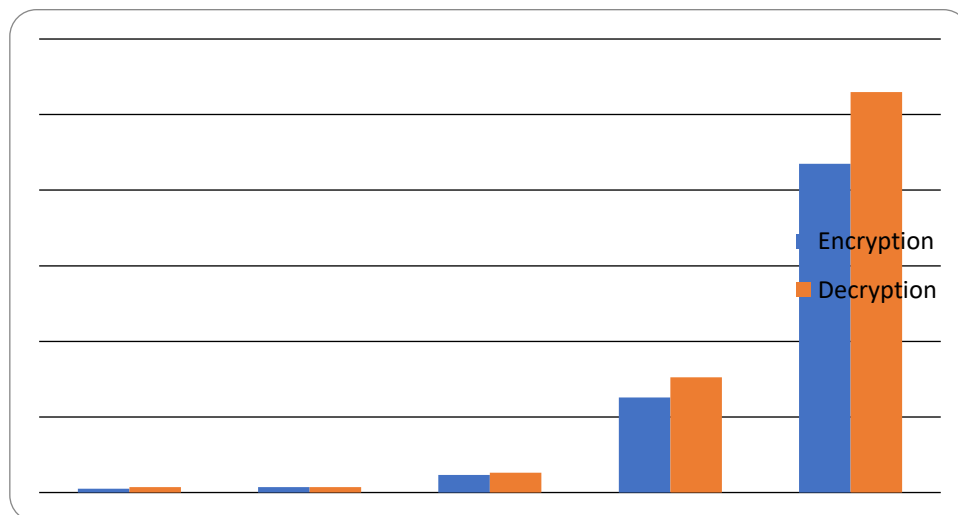


Figure-11: Result for csv file with different sizes for base paper.

Below figure – 12 shows base paper algorithm modification - 1 output for simple text with three different encoder and decoder.

```
C:\Python\python.exe C:/Ankur_modified_work/Main_En_De_modified_simple_text_all.py

Input text: AnwarNewazFeroz from34c.

===== Encryption process =====

>>> Base32
Cipher text: 6EYTIIBRGYYSANJREA3DEIBSGUYCAMJTGIDKNBAGIYTSIBSGI3SAMJRGMDQ00JAGEZTMIBRHE3CAMJS6QQDMNRAGIZTQIBTG4QDCNJVEAYTCNBAGE2D0IBSGEZCAMRZEAZDKMBAGEZTM
===
Encryption Time(s): 0.0

>>> Base64
Cipher text: MTE0IDE2MSA1MSA2MiAyNTAgMTMyIDU0IDIXOSAYMjcGMEZIDc5IDEzNiAxOTYgMTI0IDY2IDIZOCANyAxNTUgMTE0IDE0NyAyMTIgMjkgMjUwIDEzNg==
Encryption Time(s): 0.0

>>> Base128
Cipher text: [b'\x18L&\x01Dl1', b'\x10\r&\x12\x01Xd ', b'\x19\r&\x02\x01f2', b'\x10\r&\x01Hb9', b'\x10\x0cF#9\x00b1', b'\x19H\x06sI\x00b3',
b'\x1b\x08\x06\x13IX@1', b'\x19\r\x04\x031X@2', b'\x19N\x04\x03\x19\@1', b'\x1aM$\x03\t0h ', b'\x18M\x06r\x01Hb2', b'\x10\x0cG\x12\x01Hj0',
b'\x10\x0c&3\x06', [4]]
Encryption Time(s): 0.0

===== Decryption process =====

>>> Base32
Recovered plain text: AnwarNewazFeroz from34c.
Decryption Time(s): 0.0

>>> Base64
Recovered plain text: AnwarNewazFeroz from34c.
Decryption Time(s): 0.0

>>> Base128
Recovered plain text: AnwarNewazFeroz from34c.
Decryption Time(s): 0.0

Process finished with exit code 0
```

Figure-12: Result for simple text with different base of 32/64/128 bits for base paper algorithm modification - 1

Below figure – 13 & 14 shows base paper algorithm modification - 1 output for different sizes csv files with three different encoder and decoder.

File Size	Base Paper Algorithm Modification - 1					
	Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.5	0.5	0.4	0.5	0.5	0.5
466 KB	0.5	0.6	0.5	0.6	0.6	0.6
1.7 MB	1.91	2.27	1.86	2.08	2.12	2.4
5.6 MB	12	15.19	11.58	14.31	13.13	15.35
23.7 MB	38.75	45.69	37.13	43.38	43.37	51.79

Figure-13: Result for various file sizes and different base of 32/64/128 bits for algorithm modification - 1

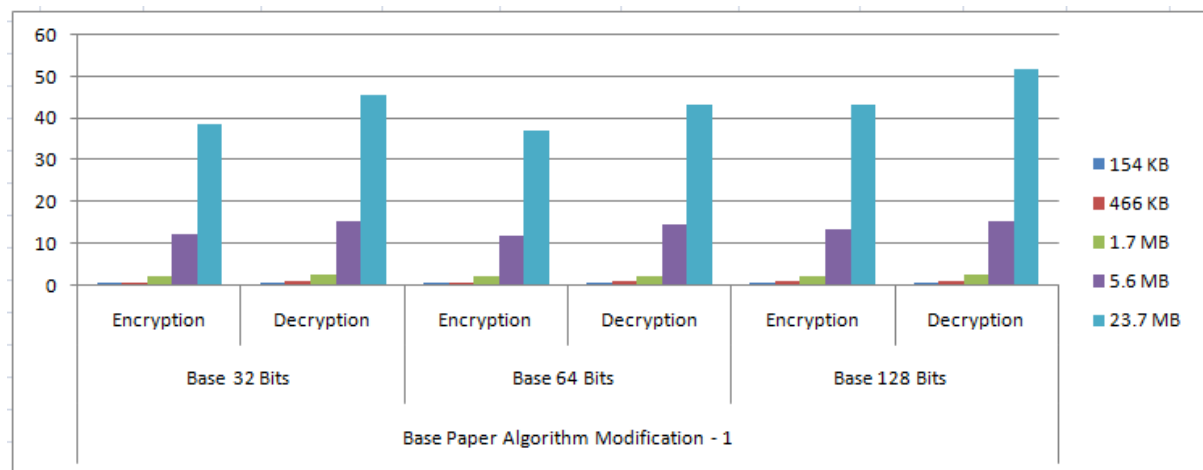


Figure-14: Result for various file sizes and different base of 32/64/128 bits for algorithm modification - 1

From above results it's clearly seen that time requirement of base paper modification algorithm - 1 base 64 bits encryption/decryption is less than base 32 bits and base 128 bits.

Below figure – 15 & 16 shows comparison between base paper algorithm output and base paper algorithm modification - 1 output for different sizes csv files with three different encoders and decoders.

File Size	Base Paper Algorithm		Base Paper Algorithm Modification - 1					
			Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.5	0.7	0.5	0.5	0.4	0.5	0.5	0.5
466 KB	0.7	0.7	0.5	0.6	0.5	0.6	0.6	0.6
1.7 MB	2.32	2.62	1.91	2.27	1.86	2.08	2.12	2.4
5.6 MB	12.58	15.25	12	15.19	11.58	14.31	13.13	15.35
23.7 MB	43.5	53	38.75	45.69	37.13	43.38	43.37	51.79

Figure-15: Comparison with base paper and different base 32/64/128 bits of algorithm modification - 1 for various file sizes

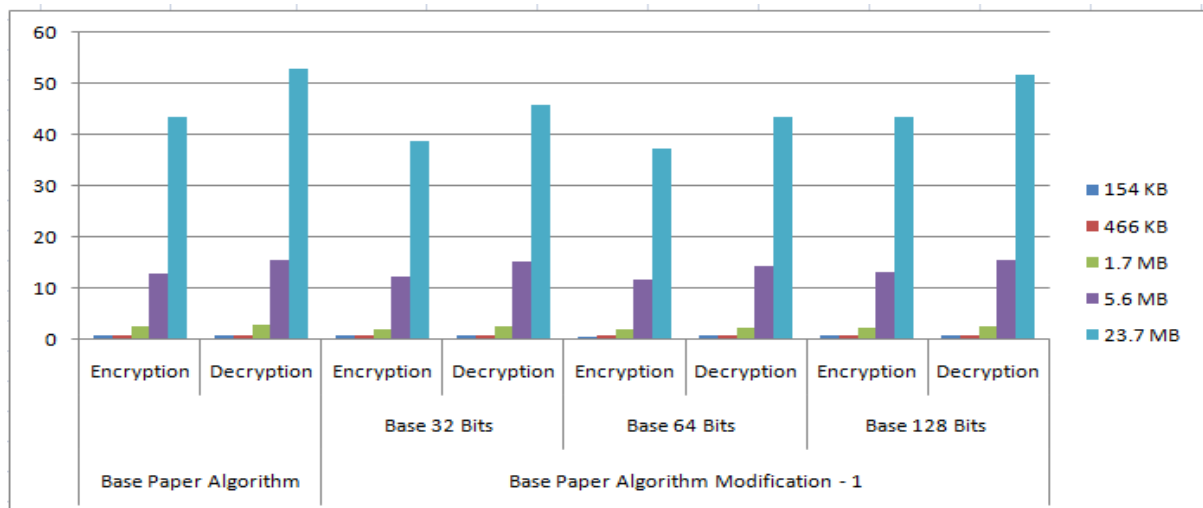


Figure-16: Comparison with base paper and different base 32/64/128 bits of algorithm modification - 1 for various file sizes

From above results it's clearly seen that time requirement of base paper algorithm modification - 1 base 64 bits encryption/decryption is less among all.

8 Results and Discussion for Base Paper Algorithm Modification - 2

First we implemented base paper algorithm modification - 2 for simple text and we got result as shown in figure 17.

```
C:\Python\python.exe C:/Ankur_modified_work/Main_En_De_changed_simple_text_all.py

Input text: AnwarNewazFeroz from34c.

===== Encryption process =====

>>> Base32
Cipher text: 6EYDAIBSGH3CAMJXHEQDC0JWEAZDENBAGE3D0IBUGAQCNJQEAZDENBAGEYDIIBVGEQDCHWEAYTGMJAGEYDOI8UGUQOMNRAGU2CAMRRQQDCNZXEAYTCMZAGIYSANBTEAYTSMJAGIYTG
===
Encryption Time(s): 0.0009999275267519531

>>> Base64
Cipher text: MTAwIDIzN1AxNzkzMk2IDIyNCAxNjc0NDAgMTUwIDIyNCAxMDQgNTEgMTM2IDEzMSAxNDcgNDUgNjYgNTQgMjE0IDE3NyAxMTMgMjEgNDMgMTkxIDIxMw==
Encryption Time(s): 0.0010001659393310547

>>> Base128
Cipher text: [b'\x18L\x06\x02\x01Hf6', b'\x10\x0c6sI\x00b9', b'\x1b\x00\x04#\x11P01', b'\x1b\rd\x03!001', b'\x1aL\x04\x03\x11Hh ', b'\x18L\x060\x01Tb ',
b'\x16Lfb\x010f1', b'\x10\x0c6\x039\x00h5', b'\x10\rFb\x01Th ', b'\x19\x0c6B\x010n7', b'\x10\x0c6\x13\x19\x00d1', b'\x16\r\x062\x010r1',
b'\x10\x0cF\x13\x03', [4]]
Encryption Time(s): 0.0010001659393310547

===== Decryption process =====

>>> Base32
Recovered plain text: AnwarNewazFeroz from34c.
Decryption Time(s): 0.0010001659393310547

>>> Base64
Recovered plain text: AnwarNewazFeroz from34c.
Decryption Time(s): 0.0019998550415039062

>>> Base128
Recovered plain text: AnwarNewazFeroz from34c.
Decryption Time(s): 0.0010001659393310547

Process finished with exit code 0
```

Figure-17: Result for simple text with different base of 32/64/128 bits for algorithm modification – 2

Below figure – 18 & 19 shows encryption & decryption time requirement for base paper algorithm modification - 2 outputs for different sizes csv files with three different encoder and decoder.

File Size	Base Paper Algorithm Modification - 2					
	Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.43	0.54	0.42	0.53	0.48	0.66
466 KB	0.51	0.64	0.5	0.63	0.57	0.71
1.7 MB	2	2.57	1.87	2.34	2.19	2.74
5.6 MB	11.84	14.98	11.33	14.94	12.95	16.45
23.7 MB	38.35	49.84	37.69	47.93	44.33	59.48

Figure-18: Result for various file sizes and different base of 32/64/128 bits for algorithm modification-2

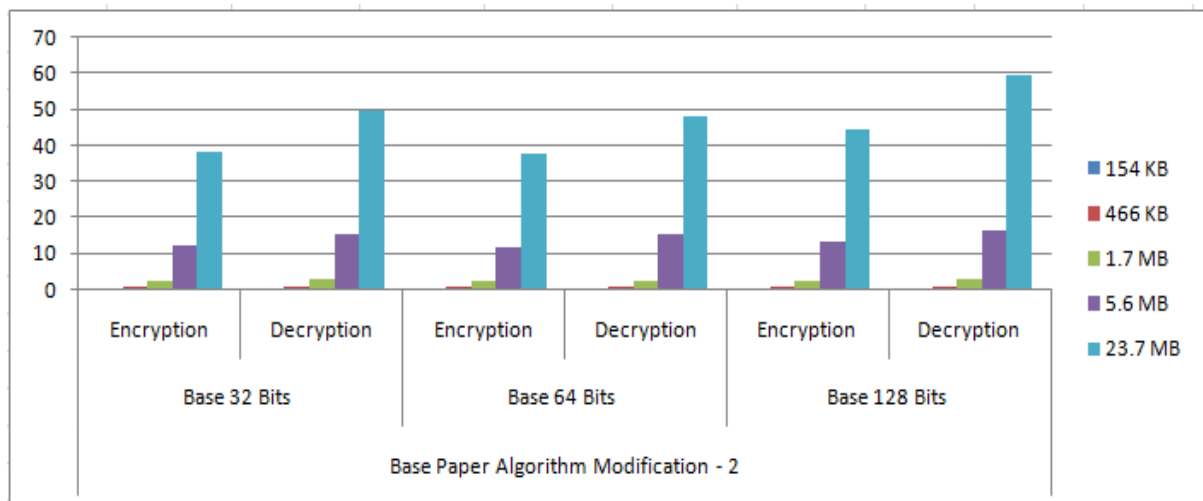


Figure-19: Result for various file sizes and different base of 32/64/128 bits for algorithm modification-2

From above results it's clearly seen that time requirement of base paper algorithm modification - 2 base 64 bits encryption/decryption is less than base 32 bits and base 128 bits.

Below figure – 20 & 21 shows comparison between base paper algorithm output and base paper algorithm modification - 2 outputs for different sizes csv files with three different encoders and decoders.

File Size	Base Paper Algorithm		Base Paper Algorithm Modification - 2					
			Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.5	0.7	0.43	0.54	0.42	0.53	0.48	0.66
466 KB	0.7	0.7	0.51	0.64	0.5	0.63	0.57	0.71
1.7 MB	2.32	2.62	2	2.57	1.87	2.34	2.19	2.74
5.6 MB	12.58	15.25	11.84	14.98	11.33	14.94	12.95	16.45
23.7 MB	43.5	53	38.35	49.84	37.69	47.93	44.33	59.48

Figure-20: Comparison with base paper and different base 32/64/128 bits of algorithm modification - 2 for various file sizes

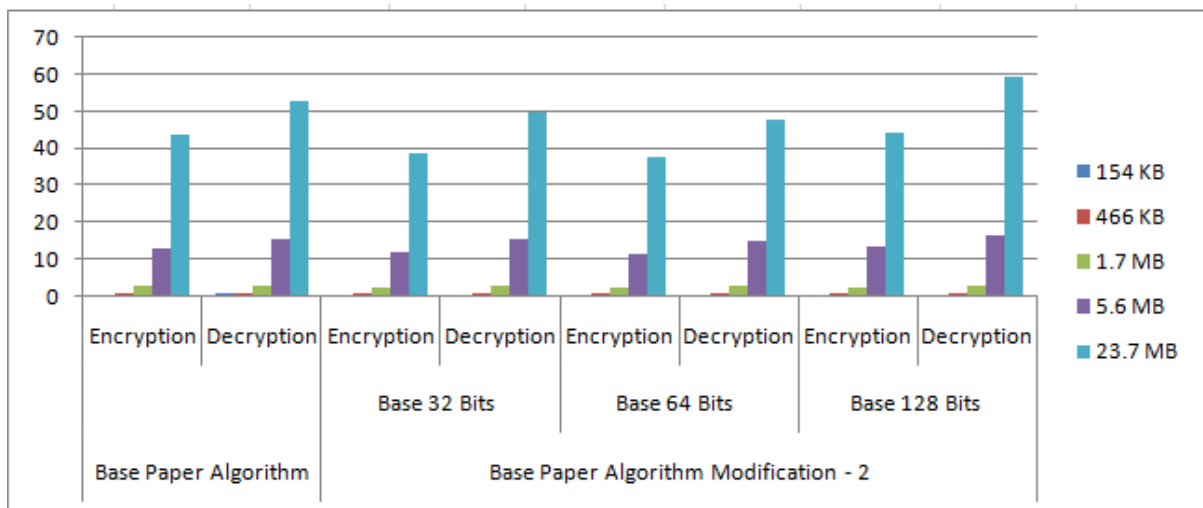


Figure-21: Comparison with base paper and different base 32/64/128 bits of algorithm modification - 2 for various file sizes

From above results it's clearly seen that time requirement of base paper algorithm modification - 2 base 64 bits encryption/decryption is less among all.

9 Results and Discussion for Base Paper Algorithm Modification - 3

First we implemented base paper algorithm modification - 3 for simple text and we got result as shown in below figure 22.

```

C:\Python\python.exe C:/Ankur_modified_work/Main_En_De_modified_128bits_simple_text_all.py

Input text: Ankurfrom Rajkot

===== Encryption process =====

>>> Base32
Cipher text: GIYD0IBR0U4SANJXEA4TNI85GEQDEMRQEAZTII8SGE4SA0JXEA3CAMRUGAQDENJQEA2TEIBRGQYCAMR5GIQCNBQ
Encryption Time(s): 0.0

>>> Base64
Cipher text: MjA3IDE1O0SA1NyA5N1AyMSAyMjAgMzQzMjE5IDw3IDYgMjQwIDII1MCA1M1AxNDAgMjIyIDE0MA==
Encryption Time(s): 0.0

>>> Base128
Cipher text: [b'\x19\x0c\x06r\x01Dj9', b'\x10\r6r\x01d1 ', b'\x19\x0c$\x03\x11H', b'\x19M\x04\x03\x11Dr ', b'\x1cMd\x03I\x00d4', b'\x18\x08\x06#)@5',
b'\x19\x08\x06\x13!@2', b'\x19\x0c0\x03\tp0', [6]]
Encryption Time(s): 0.0

===== Decryption process =====

>>> Base32
Recovered plain text: Ankurfrom Rajkot
Decryption Time(s): 0.015600204467773438

>>> Base64
Recovered plain text: Ankurfrom Rajkot
Decryption Time(s): 0.0

>>> Base128
Recovered plain text: Ankurfrom Rajkot
Decryption Time(s): 0.0

Process finished with exit code 0

```

Figure-22: Result for simple text with different base of 32/64/128 bits for algorithm modification - 3

Below figure – 23 & 24 shows encryption & decryption time requirement for our algorithm modification - 3 outputs for different sizes csv files with three different encoders and decoders.

File Size	Base Paper Algorithm Modification - 3					
	Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.33	0.44	0.32	0.4	0.35	0.43
466 KB	0.4	0.5	0.37	0.46	0.42	0.53
1.7 MB	1.43	1.81	1.35	1.69	1.69	1.94
5.6 MB	9.17	11.62	8.61	10.99	9.95	12.42
23.7 MB	29.36	39.65	27.98	37.71	31.35	39.43

Figure-23: Result for various file sizes and different base of 32/64/128 bits for algorithm modification-3

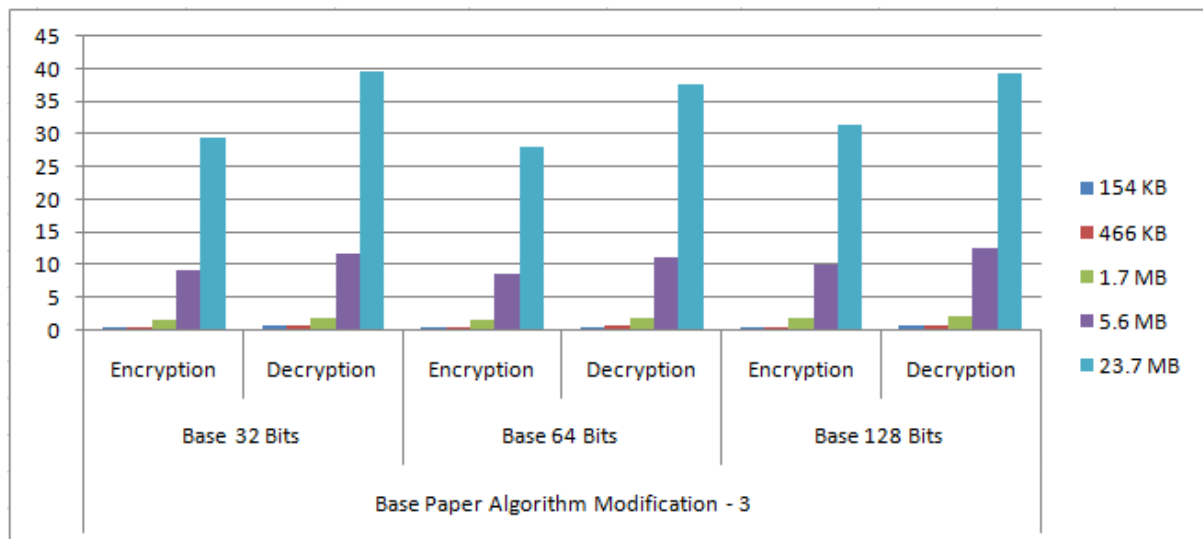


Figure-24: Result for various file sizes and different base of 32/64/128 bits for algorithm modification-3

From above results it's clearly seen that time requirement of algorithm modification - 3 base 64 bits encryption/decryption is less than base 32 bits and base 128 bits.

Below figure – 25 & 26 shows comparison between base paper algorithm output and base paper algorithm modification - 3 outputs for different sizes csv files with three different encoders and decoders.

File Size	Base Paper Algorithm		Base Paper Algorithm Modification - 3					
			Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.5	0.7	0.33	0.44	0.32	0.4	0.35	0.43
466 KB	0.7	0.7	0.4	0.5	0.37	0.46	0.42	0.53
1.7 MB	2.32	2.62	1.43	1.81	1.35	1.69	1.69	1.94
5.6 MB	12.58	15.25	9.17	11.62	8.61	10.99	9.95	12.42
23.7 MB	43.5	53	29.36	39.65	27.98	37.71	31.35	39.43

Figure-25: Comparison with base paper and different base 32/64/128 bits of algorithm modification - 3 for various file sizes

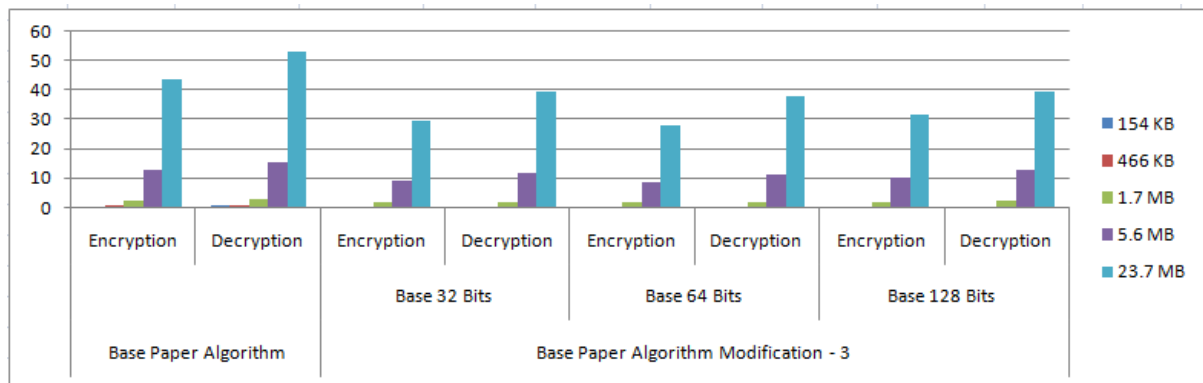


Figure-26: Comparison with base paper and different base 32/64/128 bits of algorithm modification - 3 for various file sizes

From above results it's clearly seen that time requirement of base paper algorithm modification - 3 base 64 bits encryption/decryption is less among all.

10 Results and Discussion

From base paper algorithms modification 1 & 2 & 3 after implementation it is clear that better result can be obtained when we use base 64 bits encryption/decryption.

If we compare algorithm modification - 1 (24 bytes input with 4*6 matrixes) and algorithm modification - 2 (24 bytes of input with 6*4 matrixes) then algorithm modification - 1 with base 64 gives better result in terms of time efficiency. Following figure – 27 and 28 shows the same.

File Size	Base Paper Algorithm Modification-1		Base Paper Algorithm Modification - 2					
	Base 64 Bits		Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.4	0.5	0.43	0.54	0.42	0.53	0.48	0.66
466 KB	0.5	0.6	0.51	0.64	0.5	0.63	0.57	0.71
1.7 MB	1.86	2.08	2	2.57	1.87	2.34	2.19	2.74
5.6 MB	11.58	14.31	11.84	14.98	11.33	14.94	12.95	16.45
23.7 MB	37.13	43.38	38.35	49.84	37.69	47.93	44.33	59.48

Figure-27: Comparison with algorithm modification – 1 and algorithm modification – 2

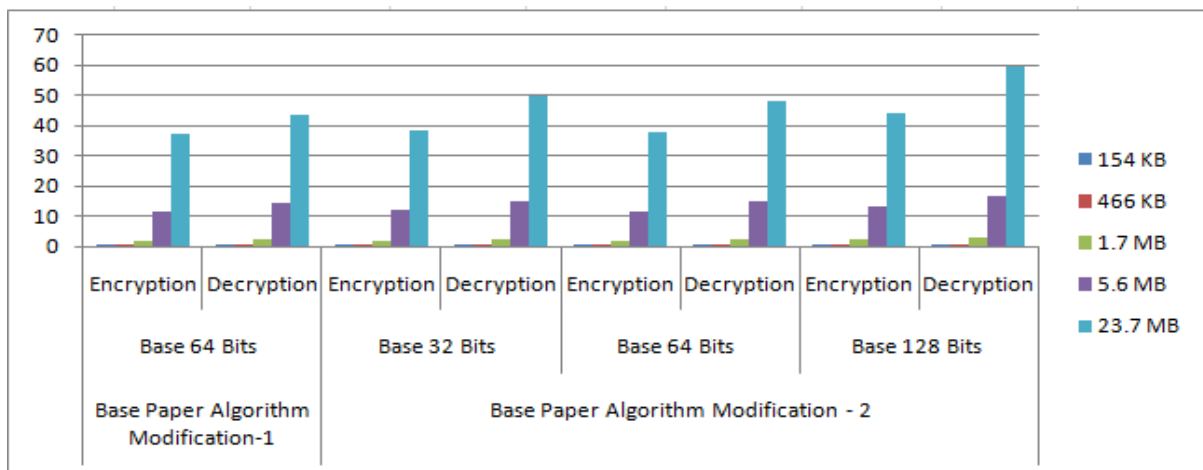


Figure-28: Comparison with algorithm modification – 1 and algorithm modification – 2

If we compare the time efficiency of existing and all modified algorithm then algorithm modification - 3 gives better output i.e. when we use 4*4 matrixes (16 bytes). Following figure – 29 and 30 shows the same.

File Size	Base Paper Algorithm		Base Paper Algorithm Modification - 1		Base Paper Algorithm Modification - 2		Base Paper Algorithm Modification - 3	
	Base 64 Bits		Base 64 Bits		Base 64 Bits		Base 64 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.5	0.7	0.4	0.5	0.42	0.53	0.32	0.4
466 KB	0.7	0.7	0.5	0.6	0.5	0.63	0.37	0.46
1.7 MB	2.32	2.62	1.86	2.08	1.87	2.34	1.35	1.69
5.6 MB	12.58	15.25	11.58	14.31	11.33	14.94	8.61	10.99
23.7 MB	43.5	53	37.13	43.38	37.69	47.93	27.98	37.71

Figure-29: Comparison with algorithm modification – 1, algorithm modification – 2 and algorithm modification – 3

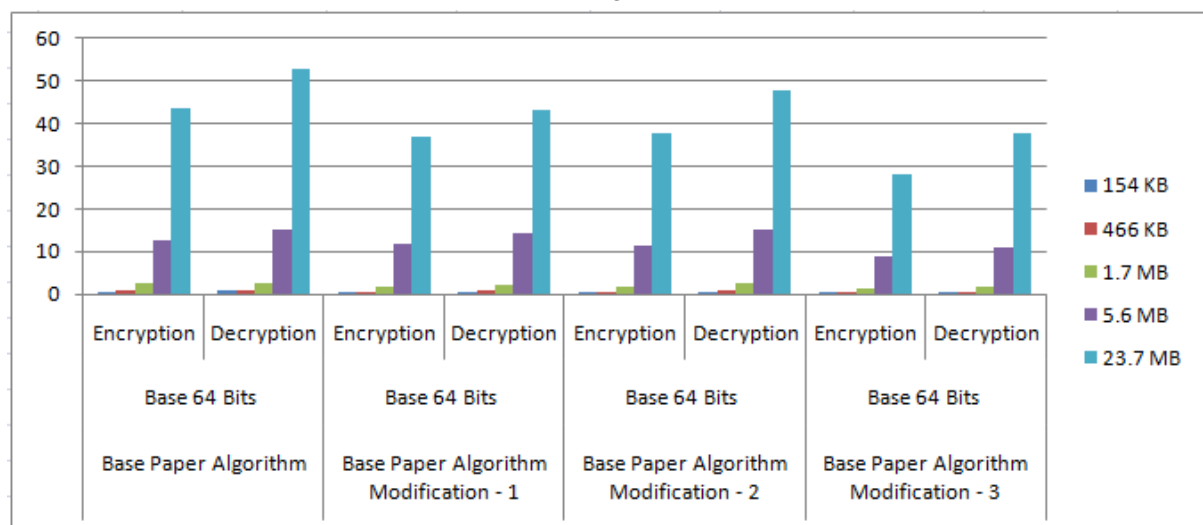


Figure-29: Comparison with algorithm modification – 1, algorithm modification – 2 and algorithm modification – 3

All output we gain for proposed algorithms – 1, 2 & 3 is using system with windows 7, i3 processor with RAM of 4 GB. Better output may obtain in terms of time if we go with higher version of window operating system or higher version of processor or increased RAM size.

11 Conclusion

In today's time, big data and cloud computing are not only necessary but also highly sought after. Numerous studies have already been conducted in order to safeguard significant data on cloud computing, and ongoing research continues. Despite advancements in cloud computing, individuals and institutions are eagerly anticipating a solution that can guarantee the protection of their large-scale data, allowing them to finally rest assured about its security. In this paper, we address the problem by reviewing different works that aim to ensure security for big data and also presenting different modification to base paper algorithm. These all methods works well for text data csv file.

References

- [1] E. K. Subramanian, Latha Tamilselvan, "Elliptic curve Diffie–Hellman cryptosystem in big data cloud security", Springer (2020).
- [2] Uma Narayanan, Varghese Paul, Shelbi Joseph, "A novel system architecture for secure authentication and data sharing in cloud enabled Big Data Environment" Elsevier (2020).

- [3] N.Keerthana, Y.Vinay Kumar, Ch.D.Sunil Kumar, U.Prasanna, P.Chandana Sireesha, "Implementation of ssgk protocol for securely share data in cloud storage", Juni Khyat Vol.10 Issue 5 No. 5 (2020).
- [4] S. Ramasamy, R. K. Gnanamurthy, "Cluster Based Multi Layer User Authentication Data Center Storage Architecture for Big Data Security in Cloud Computing", JIT Vol. 21 (2020).
- [5] P Geetha Niharika, K. Raghu, "Secure sharing of big data information using time constraint in cloud computing", IRJMETs Vol.2 Issue. 9 (2020).
- [6] Uma Narayanan, Varghese Paul, Shelbi Joseph, "A light weight encryption over big data in information stockpiling on cloud", IJEECS Vol.17 No.1 Page No. 389-397 (2020)
- [7] Hossein Abroshan "A Hybrid Encryption Solution to Improve Cloud Computing Security using Symmetric and Asymmetric Cryptography Algorithms", IJACSA Vol.12 No. 6 (2021)
- [8] Nikhil Dwivedi, Arun Malik, "Analysis of Novel Hybrid Encryption Algorithm for Providing Security in Big Data", Springer Page No. 158-169 (2019)
- [9] Jayachander Surbiryala, "PhD Forum: Improving the Security for Storing the Big Data in Cloud Environment", IEEE (2017).
- [10] Mohammad Anwar Hossain, Ahsan Ullah, Newaz Ibrahim Khan, Md Feroz Alam, "Design and Development of a Novel Symmetric Algorithm for Enhancing Data Security in Cloud Computing", JIS Page No. 199-236 (2019).
- [11] Venkata Narasimha Inukollu, Sailaja Arsi and Srinivasa Rao Ravuri, "Security issues associated with big data in cloud computing", IJNSA, Vol. 6 No.3 (2014)
- [12] Jay Dave "An Optimised Robust Model for Big Data Security on the Cloud Environment: The Numerous User-Level Data Compaction", ICIMMI Springer (2019)
- [13] Mr. Ankur N. Shah, Dr. Jay A. Dave "Classification and security enforcement of the semi-structured health records on MongoDB leveraging hybridization of Modified AES algorithm with Apriory algorithm", Juni Khyat, Vol.10 Issue 10 No.1 (2020)
- [14] Mr. Ankur N. Shah, Dr. Jay A. Dave "Design and Development of an Algorithm to Secure Big Data on Cloud Computing", ICTIS Springer(2021)
- [15] Mythreyee S, Poornima Purohit, Apoorva D.R, Harshitha R, Lathashree P.V, "A Study on Use of Big Data in Cloud Computing Environment", IJARIT, Vol.3 Issue 3, (2017)
- [16] Pooja Bavarva, Yash Punjabi, Rushabh Raolji "Security issues associated with Big Data in cloud Computing", IJSRD (2017)
- [17] Nabeel Zanoon, Abdullah Al-Haj, Sufian M Khwaldeh, "Cloud Computing and Big Data is there a Relation between the Two: A Study", IJAER, Vol. 12 (2017)
- [18] Samir A. El-Seoud, Hosam F. El-Sofany, Mohamed Abdelfattah, Reham Mohamed, "Big Data and Cloud Computing: Trends and Challenges", IJIM, Vol.11, Issue. 02 (2017)