An Innovative and Time Efficient Approach to Secure Text Data Specifically on Cloud Environment

Mr. Ankur N. Shah[1*] Dr. Vishal Dahiya[2], Dr. Jay A. Dave[3]

[1*]*Research Scholar,Indus University, Ahmedabad, India.ankur11586@gmail.com*
[2]*Director and Professor, Sardar Patel Global University, Ahmedabad, India.director_imca@svgu.ac.in*
[3]*Associate Professor, Silver Oak University, Ahmedabad, India.jay.dave4u@gmail.com*

Abstract. Data security is highly vital and crucial for every organization, regardless of size of organization—whether it it is a small scale organization, medium scale organization, or large scale organization. This applies to the both inside and outside the organization. The essential prerequisite for anyone's confidence and trust in any organization is data security. In day to day life human generates various data in different formats like text data, graphical data, voice data, video data etc. using various medium like youtube, twitter, facebook, instragram, emails etc. These data may be a structural data, may be a semi structural data or may be an unstructured data. In today's life all the data which are generated is generally huge in size which we termed as big data. To handle this big data is very difficult task. Cloud computing is one of the technique which can handle this very well. But the main fact is that there are various challenges to both of these technologies. One of the biggest challenges is to provide security to this kind of data. There are various research has been done and still going on to address this issue effectively. In this paper we have also tried to address this issue after doing exhaustive literature work then we provide our own AVJ security algorithm which can provide security to text data. The writers' names—A for Mr. Ankur Shah, V for Dr. Vishal Dahiya, and J for Dr. Jay A Dave—led to the creation of the moniker AVJ***Throughout this paper we have used term base paper algorithm meaning that we are using reference paper [5] algorithm of Mohammad Anwar Hossain and et al.

Keywords: Data Security, Cloud Computing, Big Data.

1. DATA SECURITY

"Data security is the process of safeguarding digital information throughout its entire life cycle to protect it from corruption, theft, or unauthorized access." [8] "Data security also ensures data is available to anyone in the organization who has access to it." [10] Typical Data Security Threats are "Human error, Internal Threats, Social Engineering Attacks, Ransomware" [9] etc.

1.1. Cloud Computing

There are various definitions of cloud computing. "Cloud computing is a computing model that provides dynamically scalable virtualized resources to users through the Internet. Users do not need to know how to manage the infrastructure supporting cloud computing. The definition of NIST provided by the National Institute of Standards and Technology is: Cloud is a parallel distributed system composed of a group of interconnected virtualized computers. It is based on the dynamics of service contracts between service providers and consumers." [6] It "offers various services like storage, database, networking, software etc. The provider will take charges as per usage such as we are paying gas pipeline bill at our home whatever we use we are paying for that." [7]

1.2. Background

In this part we have refer various research and review papers about data security and cloud computing. JayachanderSurbiryala [1], Propose a way to split up large data files into smaller ones so that even if they are cracked, they cannot access all of the data. Then, using the keyword indexing methodology, combine all the files to access the whole data. DeepakPuthal, et. al [2], proposes dynamic prime number based security verification (DPBSV). By comparing DPBSV with AES author shows DPBSV giving better result. Nikhil Dwivedi, et. al [3], provides novel hybrid encryption algorithm for security of data by combining AES and RSA algorithm. Mahnaz Nasseri, et. al [4], provides a new method for improving the security and access time to data blocks in big databases. Mohammad Anwar Hossain, et. al [5], 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 [5], working on 192 bits of data but it can encrypt only 24 bytes of input text (192 bits). So we proposed two algorithms which are faster because it use bit permutation instead of mix column operation.Our proposed algorithms are also able to encrypt csv files.

2. PROPOSED AVJ SECURITY ALGORITHM 128 BITS

Following are steps of proposed algorithm for encryption process.

1. Input user name and password for authentication purpose.
2. Input plain text of any size and key text of size 16 bytes.
3. Generate 4*4 Matrixes.
4. Convert message characters into octal equivalent.
5. Perform row shifting operations.

[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]

6. Perform column interchanging operations.

[Interchange Column C1 by C3 and C3 by C1, Interchange Column C2 by C4 and C4 by C2]

7. Perform row interchanging operations.

[Interchange Row R1 by R3 and R3 by R1, Interchange Row R2 by R4 and R4 by R2]

8. Convert to equivalent ASCII.
9. Perform XOR operations between rows.
10. Convert into Hexadecimal Values
11. Replace values using S-Box
12. Convert into decimal values
13. Perform Bit Permutation
14. Calculate Key.

[Generate 4*4 block matrix for key text of 16 bytes, convert key characters into ASCII]

15. Perform XOR between resultant of Bit permutation and calculated key matrix.
16. Arrange matrix values as plain text.
17. Using base 32/64/128 encoders convert plain text into corresponding encoding characters.

Following are steps of proposed algorithm for decryption process.

1. Input user name and password for authentication purpose.

2. Input cipher text
3. Decode cipher text using base 32/64/128 bits decoder and arrange into 4x4matrixes
4. Input key text and arrange into 4x4 matrixes and convert to ASCII
5. Perform XOR between decoded cipher text and key text ASCII matrixes
6. Perform Bit permutations.
7. Convert into Hexadecimal Values
8. Replace values using S-Box
9. Convert into decimal values
10. Perform XOR operations between rows.
11. Perform column interchanging operations.[Interchange Column C1 by C3 and C3 by C1, Interchange Column C2 by C4 and C4 by C2]
12. Perform row interchanging operations.[Interchange Row R1 by R3 and R3 by R1, Interchange Row R2 by R4 and R4 by R2]
13. Perform row shifting operations.[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]
14. Convert ASCII to Plain Text.
15. Checking integrity of data.

3. PROPOSED AVJ SECURITY ALGORITHM 192 BITS

Following are steps of proposed algorithm for encryption process.

1. Input user name and password for authentication purpose.
2. Input plain text of any size and key text of size 24 bytes.
3. Generate 4*6 Matrixes.
4. Convert message characters into octal equivalent.
5. Perform row shifting operations.

   [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]
6. Perform column interchanging operations.

   [Interchange Column C1 by C3 and C3 by C1, Interchange Column C2 by C5 and C5 by C2, Interchange Column C4 by C6 and C6 by C4]
7. Perform row interchanging operations.

   [Interchange Row R1 by R3 and R3 by R1, Interchange Row R2 by R4 and R4 by R2]

8. Convert to equivalent ASCII.
9. Perform XOR operations between rows.
10. Convert into Hexadecimal Values
11. Replace values using S-Box
12. Convert into decimal values
13. Perform Bit Permutation
14. Calculate Key.

[Generate 4*6 block matrix for key text of 24 bytes, convert key characters into ASCII]

15. Perform XOR between resultant of Bit permutation and calculated key matrix.
16. Arrange matrix values as plain text.
17. Using base 32/64/128 encoders convert plain text into corresponding encoding characters.

Following are steps of proposed algorithm for decryption process.

1. Input user name and password for authentication purpose.
2. Input cipher text
3. Decode cipher text using base 32/64/128 bits decoder and arrange into 4x6 matrixes
4. Input key text and arrange into 4x6 matrixes and convert to ASCII
5. Perform XOR between decoded cipher text and key text ASCII matrixes
6. Perform Bit permutations.
7. Convert into Hexadecimal Values
8. Replace values using S-Box
9. Convert into decimal values
10. Perform XOR operations between rows.
11. Perform column interchanging operations.

[Interchange Column C1 by C3 and C3 by C1, Interchange Column C2 by C5 and C5 by C2, Interchange Column C4 by C6 and C6 by C4]

12. Perform row interchanging operations.

[Interchange Row R1 by R3 and R3 by R1, Interchange Row R2 by R4 and R4 by R2]

16. Perform row shifting operations.

[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]

17. Convert ASCII to Plain Text.

18. Checking integrity of data.

4. SYSTEM REQUIREMENT FOR IMPLEMENTATION OF PROPOSED AVJ SECURITY ALGORITHMS

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

5. DETAIL DISCUSSION ABOUT ENCRYPTION PROCESS FOR AVJ SECURITY ALGORITHM 192 BITS

Following is detail about encryption process step wise. To understand it kindly considers plain text value is Ankur Shah from Wadhwan.

Plain text = Ankur Shah from Wadhwan.

	C1	C2	C3	C4	C5	C6
R1	A	N	k	u	r	
R2	S	H	a	h		f
R3	r	O	m		W	a
R4	d	H	w	a	n	.

4*6 Matrix for plaintext

	C1	C2	C3	C4	C5	C6
R1	101	156	153	165	162	40
R2	123	150	141	150	40	146
R3	162	157	155	40	127	141
R4	144	150	167	141	156	56

Octal values for 4*6 Matrix of plaintext

	C1	C2	C3	C4	C5	C6
R1	165	162	40	101	156	153
R2	141	150	40	146	123	150
R3	157	155	40	127	141	162
R4	144	150	167	141	156	56

Shift Rows

	C1	C2	C3	C4	C5	C6
R1	40	162	165	101	156	153
R2	40	150	141	146	123	150
R3	40	155	157	127	141	162
R4	167	150	144	141	156	56

Interchange Column Operation,

Step-1: Change Col C1 by C3 and C3 by C1

	C1	C2	C3	C4	C5	C6
R1	40	156	165	101	162	153
R2	40	123	141	146	150	150
R3	40	141	157	127	155	162
R4	167	156	144	141	150	56

Interchange Column Operation,
Step-2: Change Col C2 by C5 and C5 by C2

	C1	C2	C3	C4	C5	C6
R1	40	156	165	153	162	101
R2	40	123	141	150	150	146
R3	40	141	157	162	155	127
R4	167	156	144	56	150	141

Interchange Column Operation,
Step-3: Change Col C4 by C6 and C6 by C4
(Result of Interchange Column Operations)

	C1	C2	C3	C4	C5	C6
R1	40	141	157	162	155	127
R2	40	123	141	150	150	146
R3	40	156	165	153	162	101
R4	167	156	144	56	150	141

Interchange Row Operations,
Step-1: Change Row R1 by R3 and R3 by R1

	C1	C2	C3	C4	C5	C6
R1	40	141	157	162	155	127
R2	167	156	144	56	150	141
R3	40	156	165	153	162	101
R4	40	123	141	150	150	146

Interchange Row Operations,
Step-2: Change Row R2 by R4 and R4 by R2
(Result of Interchange Row Operations)

	C1	C2	C3	C4	C5	C6
R1	32	97	111	114	109	87
R2	119	110	100	46	104	97
R3	32	110	117	107	114	65
R4	32	83	97	104	104	102

Conversion to ASCII Values

	C1	C2	C3	C4	C5	C6
R1	0	15	26	25	31	22
R2	119	110	100	46	104	97
R3	32	110	117	107	114	65
R4	32	83	97	104	104	102

XOR Operations between Rows R1 & R3

	C1	C2	C3	C4	C5	C6
R1	0	15	26	25	31	22
R2	87	61	5	70	0	7
R3	32	110	117	107	114	65
R4	32	83	97	104	104	102

XOR Operations between Rows R2 & R4
(Result of XOR Operations between Rows)

	C1	C2	C3	C4	C5	C6
R1	0	F	1a	19	1f	16
R2	57	3d	5	46	0	7
R3	20	6e	75	6b	72	41
R4	20	53	61	68	68	66

Hexadecimal Conversion

	C1	C2	C3	C4	C5	C6
R1	63	76	a2	d4	c0	47
R2	5b	27	6b	5a	63	c5
R3	b7	9f	9d	7f	40	83
R4	b7	Ed	ef	45	45	33

Value Replacement Using S-Box

	C1	C2	C3	C4	C5	C6
R1	99	118	162	212	192	71
R2	91	39	107	90	99	197
R3	183	159	157	127	64	131
R4	183	237	239	69	69	51

Decimal Conversion

	C1	C2	C3	C4	C5	C6
R1	183	237	239	69	69	51
R2	183	159	157	127	64	131
R3	91	39	107	90	99	197
R4	99	118	162	212	192	71

Bit Permutation

	C1	C2	C3	C4	C5	C6
R1	S	Y	m	m	e	t
R2	r	l	c		k	e
R3	y		i	s		g
R4	i	V	e	n	.	

4*6 Matrix for key text

	C1	C2	C3	C4	C5	C6
R1	83	121	109	109	101	116
R2	114	105	99	32	107	101
R3	121	32	105	115	32	103
R4	105	118	101	110	46	32

ASCII value of key text

	C1	C2	C3	C4	C5	C6		C1	C2	C3	C4	C5	C6
R1	183	237	239	69	69	51	XOR	83	121	109	109	101	116
R2	183	159	157	127	64	131		114	105	99	32	107	101
R3	91	39	107	90	99	197		121	32	105	115	32	103
R4	99	118	162	212	192	71		105	118	101	110	46	32

XOR between result of Bit Permutation and ASCII value of key text

C1		C1		Result
183	XOR	83	=	228
183		114		197
91		121		34
99		105		10

Step-1: C1 XOR C1

C2		C2		Result
237	XOR	121	=	148
159		105		246
39		32		7
118		118		0

Step-2: C2 XOR C2

C3		C3		Result
239	XOR	109	=	130
157		99		254
107		105		2
162		101		199

Step-3: C3 XOR C3

C4		C4		Result
69	XOR	109	=	40
127		32		95
90		115		41
212		110		186

Step-4: C4 XOR C4

C5		C5		Result
69	XOR	101	=	32
64		107		43
99		32		67
192		46		238

Step-5: C5 XOR C5

C6		C6		Result

51	XOR	116	=	71
131		101		230
197		103		162
71		32		103

Step-6: C6 XOR C6

	C1	C2	C3	C4	C5	C6
R1	228	148	130	40	32	71
R2	197	246	254	95	43	230
R3	34	7	2	41	67	162
R4	10	0	199	186	238	103

Result of XOR between result of Bit Permutation and ASCII value of key text

Here Encoded characters are: 228 148 130 40 32 71 197 246 254 95 43 230 34 7 2 41 67 162 10 0 199 186 238 103

From these encoded characters we got cipher text.

6. DETAIL DISCUSSION ABOUT DECRYPTION PROCESS FOR AVJ SECURITY ALGORITHM 192 BITS

In decryption part first it takes cipher text generated from encryption process. Here considers the cipher text generated by base 64 bits is,

MjI4IDE0OCAxMzAgNDAgMzIgNzEgMTk3IDI0NiAyNTQgOTUgNDMgMjMwIDM0IDcgMiA0MSA2NyAxNjIgM
TAgMCAxOTkgMTg2IDIzOCAxMDM=

	C1	C2	C3	C4	C5	C6
R1	228	148	130	40	32	71
R2	197	246	254	95	43	230
R3	34	7	2	41	67	162
R4	10	0	199	186	238	103

4*6 Matrix for decoded cipher-text

	C1	C2	C3	C4	C5	C6
R1	83	121	109	109	101	116
R2	114	105	99	32	107	101
R3	121	32	105	115	32	103
R4	105	118	101	110	46	32

4*6 Matrix for ASCII of key text

	C1	C2	C3	C4	C5	C6
R1	183	237	239	69	69	51
R2	183	159	157	127	64	131
R3	91	39	107	90	99	197
R4	99	118	162	212	192	71

XOR between decoded cipher text and ASCII of key text

	C1	C2	C3	C4	C5	C6
R1	99	118	162	212	192	71
R2	91	39	107	90	99	197
R3	183	159	157	127	64	131
R4	183	237	239	69	69	51

Bit Permutation

	C1	C2	C3	C4	C5	C6
R1	63	76	a2	d4	c0	47
R2	5b	27	6b	5a	63	c5
R3	b7	9f	9d	7f	40	83
R4	b7	Ed	ef	45	45	33

Hexadecimal Conversion

	C1	C2	C3	C4	C5	C6
R1	0	0f	1a	19	1f	16
R2	57	3d	5	46	0	7
R3	20	6e	75	6b	72	41
R4	20	53	61	68	68	66

S-Box Replacement

	C1	C2	C3	C4	C5	C6
R1	0	15	26	25	31	22
R2	87	61	5	70	0	7
R3	32	110	117	107	114	65
R4	32	83	97	104	104	102

Decimal Conversion

	C1	C2	C3	C4	C5	C6
R1	32	97	111	114	109	87
R2	87	61	5	70	0	7
R3	32	110	117	107	114	65
R4	32	83	97	104	104	102

XOR Operations between Rows R1 & R3

	C1	C2	C3	C4	C5	C6
R1	32	97	111	114	109	87
R2	119	110	100	46	104	97
R3	32	110	117	107	114	65
R4	32	83	97	104	104	102

XOR Operations between Rows R2 & R4
(Result of XOR Operations between Rows)

	C1	C2	C3	C4	C5	C6
R1	111	97	32	114	109	87
R2	100	110	119	46	104	97
R3	117	110	32	107	114	65
R4	97	83	32	104	104	102

Interchange Column Operation,
Step-1: Change Col C1 by C3 and C3 by C1

	C1	C2	C3	C4	C5	C6
R1	111	109	32	114	97	87
R2	100	104	119	46	110	97
R3	117	114	32	107	110	65
R4	97	104	32	104	83	102

Interchange Column Operation,
Step-2: Change Col C2 by C5 and C5 by C2

	C1	C2	C3	C4	C5	C6
R1	111	109	32	87	97	114
R2	100	104	119	97	110	46
R3	117	114	32	65	110	107
R4	97	104	32	102	83	104

Interchange Column Operation,
Step-3: Change Col C4 by C6 and C6 by C4
(Result of Interchange Column Operations)

	C1	C2	C3	C4	C5	C6
R1	117	114	32	65	110	107
R2	100	104	119	97	110	46
R3	111	109	32	87	97	114
R4	97	104	32	102	83	104

Interchange Row Operations,
Step-1: Change Row R1 by R3 and R3 by R1

	C1	C2	C3	C4	C5	C6
R1	117	114	32	65	110	107
R2	97	104	32	102	83	104
R3	111	109	32	87	97	114
R4	100	104	119	97	110	46

Interchange Row Operations,
Step-2: Change Row R2 by R4 and R4 by R2
(Result of Interchange Row Operations)

	C1	C2	C3	C4	C5	C6
R1	65	110	107	117	114	32
R2	83	104	97	104	32	102
R3	114	111	109	32	87	97
R4	100	104	119	97	110	46

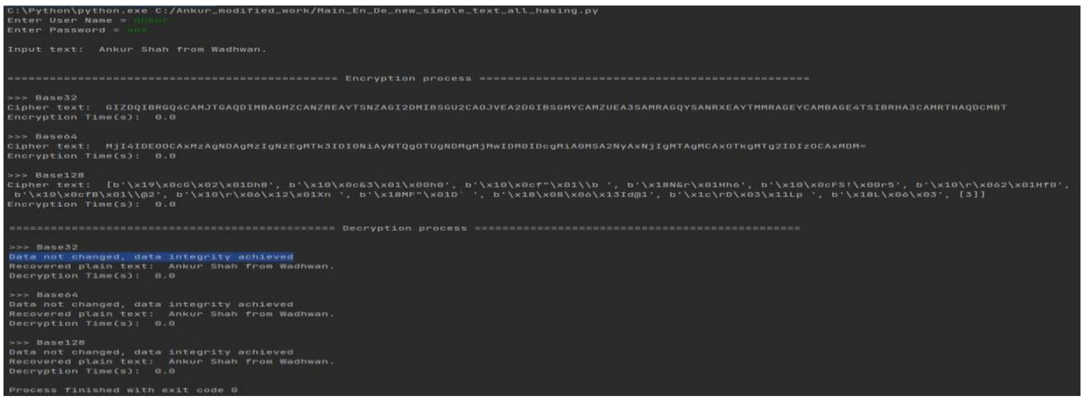
Inverse Shift Rows

	C1	C2	C3	C4	C5	C6
R1	A	n	k	u	r	
R2	S	h	a	h		f
R3	r	o	m		W	a
R4	d	h	w	a	n	.

Plain Text

7. RESULT AND DISCUSSION

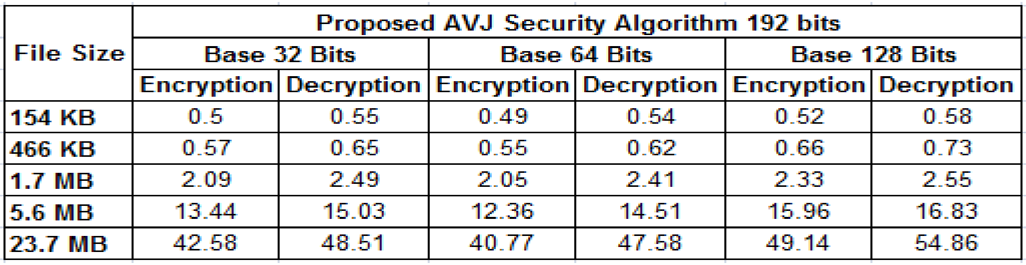
We run algorithm (192bits) for simple text of 24 bytes and using key “Symmetric key is given.” and we got output as shown in below figure-1.



```
C:\Python\python.exe C:/Ankur_modified_work/Main_En_De_new_simple_text_all_hasing.py
Enter User Name = ankur
Enter Password = ans

Input text:  Ankur Shah from Wadhwan.


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

>>> Base32
Cipher text:  GIZDQIBRGQ4CAMJTGAQDIMBAGMZCANZREAYTSNZAGI2DMIBSGU2CA0JVEA2DGIBSGMYCAMZUEA3SAMRAGQYSANRXEAYTMMRAGEYCAMBAGE4TSIBRHA3CAMRTHAQDCMBT
Encryption Time(s):  0.0

>>> Base64
Cipher text:  MjI4IDE0OCAxMzAgNDAgMzIgNzEgMTk3IDI0NiAyNTQg0TUgNDMgMjMwIDM0IDcgMiA0MSA2NyAxNjIgMTAgMCAx0TkgMTg2IDIzOCAxMDM=
Encryption Time(s):  0.0

>>> Base128
Cipher text:  [b'\x19\x0cG\x02\x01Dh8', b'\x10\x0c&3\x01\x00h0', b'\x10\x0cf"\x01\\b ', b'\x18N&r\x01Hh6', b'\x10\x0cFS!\x00r5', b'\x10\r\x062\x01Hf0',
 b'\x10\x0cfB\x01\\@2', b'\x10\r\x06\x12\x01Xn ', b'\x18MF"\x01D` ', b'\x18\x08\x06\x13Id@1', b'\x1c\rD\x03\x11Lp ', b'\x18L\x06\x03', [3]]
Encryption Time(s):  0.0

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

>>> Base32
Data not changed, data integrity achieved
Recovered plain text:  Ankur Shah from Wadhwan.
Decryption Time(s):  0.0

>>> Base64
Data not changed, data integrity achieved
Recovered plain text:  Ankur Shah from Wadhwan.
Decryption Time(s):  0.0

>>> Base128
Data not changed, data integrity achieved
Recovered plain text:  Ankur Shah from Wadhwan.
Decryption Time(s):  0.0

Process finished with exit code 0
```

Fig.1. Result for simple text with different base of 32/64/128 bits for proposed AVJ security algorithm 192 bits

For csv file format data with proposed AVJ security algorithm 192 bits we got the result as shown in below figure – 2 and figure – 3.



File Size	Proposed AVJ Security Algorithm 192 bits					
	Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.5	0.55	0.49	0.54	0.52	0.58
466 KB	0.57	0.65	0.55	0.62	0.66	0.73
1.7 MB	2.09	2.49	2.05	2.41	2.33	2.55
5.6 MB	13.44	15.03	12.36	14.51	15.96	16.83
23.7 MB	42.58	48.51	40.77	47.58	49.14	54.86

Fig.2. Result for various file sizes and different base of 32/64/128 bits for proposed AVJ security algorithm (192 bits)

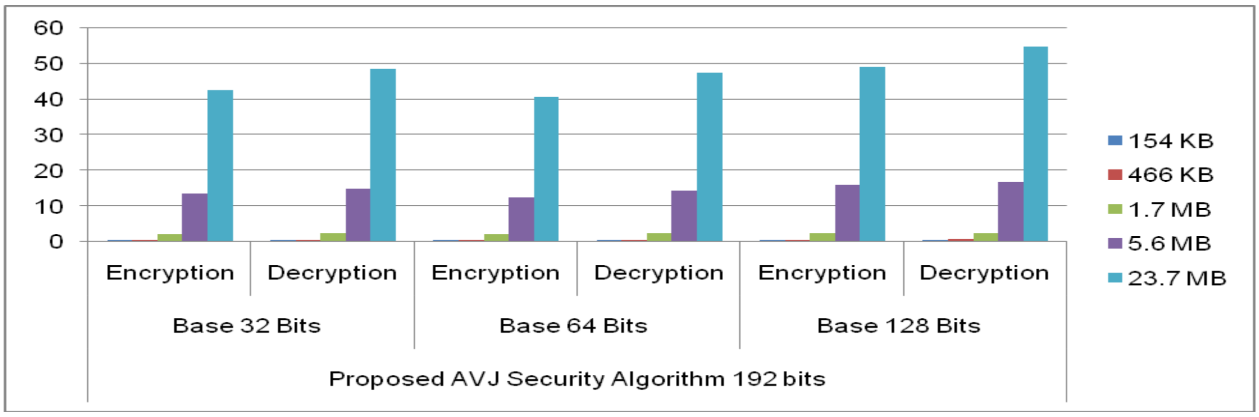
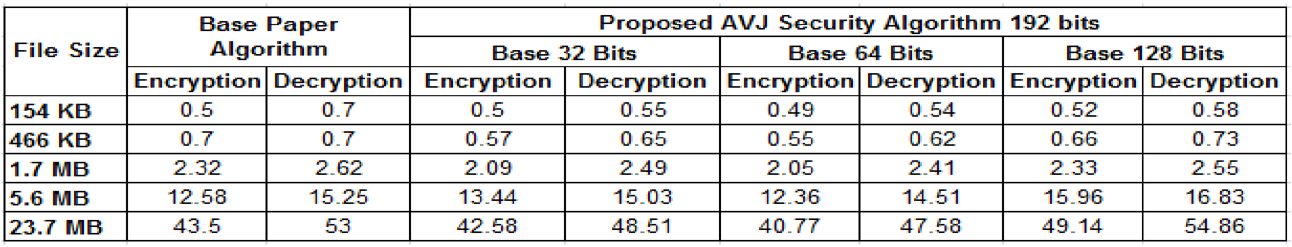


Fig.3. Result for various file sizes and different base of 32/64/128 bits for proposed AVJ security algorithm (192 bits)

As from the above result it's clearly show that base 64 is the best among the three bases in terms of time efficiency of the proposed AVJ security algorithm 192 bits.

Then we have compare our work with base paper algorithm (existing algorithm of reference paper [5]) where we need to modify base paper algorithm to work with csv file and for one round of encryption and decryption after then we have compare with our proposed AVJ security algorithm 192 bits. Following figure 4 and 5 shows this.



File Size	Base Paper Algorithm		Proposed AVJ Security Algorithm 192 bits					
			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	0.55	0.49	0.54	0.52	0.58
466 KB	0.7	0.7	0.57	0.65	0.55	0.62	0.66	0.73
1.7 MB	2.32	2.62	2.09	2.49	2.05	2.41	2.33	2.55
5.6 MB	12.58	15.25	13.44	15.03	12.36	14.51	15.96	16.83
23.7 MB	43.5	53	42.58	48.51	40.77	47.58	49.14	54.86

Fig.4. Comparison with base paper algorithm and different base 32/64/128 bits of proposed AVJ security algorithm (192 bits) for various file sizes

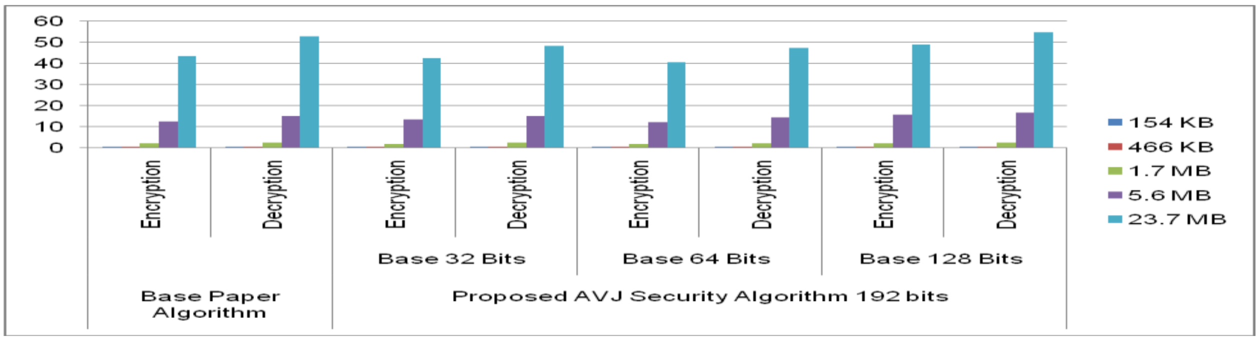
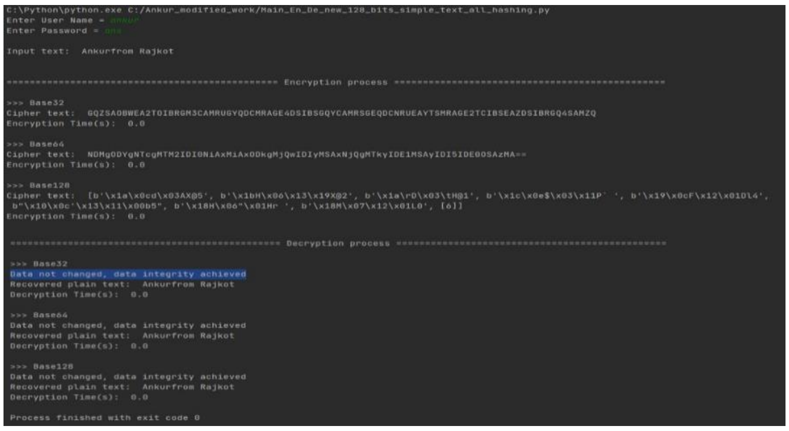


Fig.5. Comparison with base paper algorithm and different base 32/64/128 bits of proposed AVJ security algorithm (192 bits) for various file sizes

From above comparison we have found that our proposed AVJ security algorithm (192 bits) is better than existing one in terms of time efficiency. Our proposed algorithms (128 bits & 192 bits) are also able to encrypt file whereas existing one is only for simple text of 24 bytes.

First we run proposed AVJ security algorithm (128bits) for simple text of 16 bytes and using key "key is AnkurShah" and we got output as shown in below figure-6.We have also check output with different key "key is symmetric" and "KeyisAmam_Krish", for all we got the same result of time efficiency of algorithm.



```
C:\Python\python.exe C:/Ankur_modified_work/Main_En_De_new_128_bits_simple_text_all_hashing.py
Enter User Name = ankur
Enter Password = ans

Input text:  Ankurfrom Rajkot


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

>>> Base32
Cipher text:  GQZSAOBWEA2TOIBRGM3CAMRUGYQDCMRAGE4DSIBSGQYCAMRSGEQDCNRUEAYTSMRAGE2TCIBSEAZDSIBRGQ4SAMZQ
Encryption Time(s):  0.0

>>> Base64
Cipher text:  NDMgODYgNTcgMTM2IDI0NiAxMiAxODkgMjQwIDIyMSAxNjQgMTkyIDE1MSAyIDI5IDE0OSAzMA==
Encryption Time(s):  0.0

>>> Base128
Cipher text:  [b'\x1a\x0cd\x03AX@5', b'\x1bH\x06\x13\x19X@2', b'\x1a\rD\x03\tH@1', b'\x1c\x0e$\x03\x11P ', b'\x19\x0cF\x12\x01Dl4',
 b"\x10\x0c'\x13\x11\x00b5", b'\x18H\x06"\x01Hr ', b'\x18M\x07\x12\x01L0', [6]]
Encryption Time(s):  0.0


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

 >>> Base32
 Data not changed, data integrity achieved
 Recovered plain text:  Ankurfrom Rajkot
 Decryption Time(s):  0.0

 >>> Base64
 Data not changed, data integrity achieved
 Recovered plain text:  Ankurfrom Rajkot
 Decryption Time(s):  0.0

 >>> Base128
 Data not changed, data integrity achieved
 Recovered plain text:  Ankurfrom Rajkot
 Decryption Time(s):  0.0

 Process finished with exit code 0
```

Fig.6. Result for simple text with different base of 32/64/128 bits for proposed AVJ security algorithm 128 bits

For csv file format data with proposed AVJ security algorithm 128bits we got the result as shown in below figure – 7 and figure – 8.

File Size	Proposed AVJ Security Algorithm 128 bits					
	Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.36	0.44	0.35	0.4	0.39	0.46
466 KB	0.45	0.54	0.41	0.48	0.46	0.54
1.7 MB	1.54	1.89	1.52	1.79	1.74	2.02
5.6 MB	10.12	11.89	9.82	11.82	11.04	12.69
23.7 MB	33.99	40.18	31.87	37.46	35.8	42.45

Fig.7. Result for various file sizes and different base of 32/64/128 bits for proposed AVJ security algorithm (128 bits)

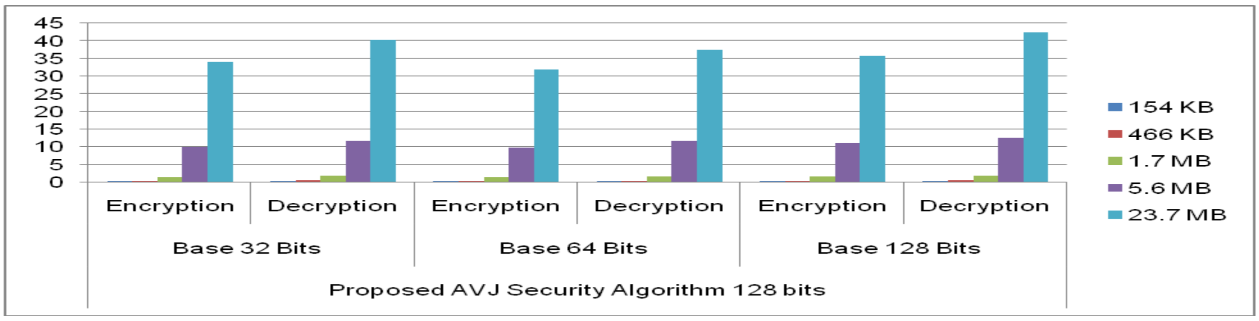


Fig.8. Result for various file sizes and different base of 32/64/128 bits for proposed AVJ security algorithm (128 bits)

As from the above result it's clearly show that base 64 is the best among the three bases in terms of time efficiency of the proposed AVJ security algorithm 128 bits.

Then we have compare our work with base paper algorithm (existing algorithm of reference paper [5]) where we need to modify base paper algorithm to work with csv file and for one round of encryption and decryption after then we have compare with our proposed AVJ security algorithm 128 bits. Following figure 9 and 10 shows this.

File Size	Base Paper Algorithm		Proposed AVJ Security Algorithm 128 bits					
			Base 32 Bits		Base 64 Bits		Base 128 Bits	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
154 KB	0.5	0.7	0.36	0.44	0.35	0.4	0.39	0.46
466 KB	0.7	0.7	0.45	0.54	0.41	0.48	0.46	0.54
1.7 MB	2.32	2.62	1.54	1.89	1.52	1.79	1.74	2.02
5.6 MB	12.58	15.25	10.12	11.89	9.82	11.82	11.04	12.69
23.7 MB	43.5	53	33.99	40.18	31.87	37.46	35.8	42.45

Fig.9. Comparison with base paper algorithm and different base 32/64/128 bits of proposed AVJ security algorithm (128 bits) for various file sizes

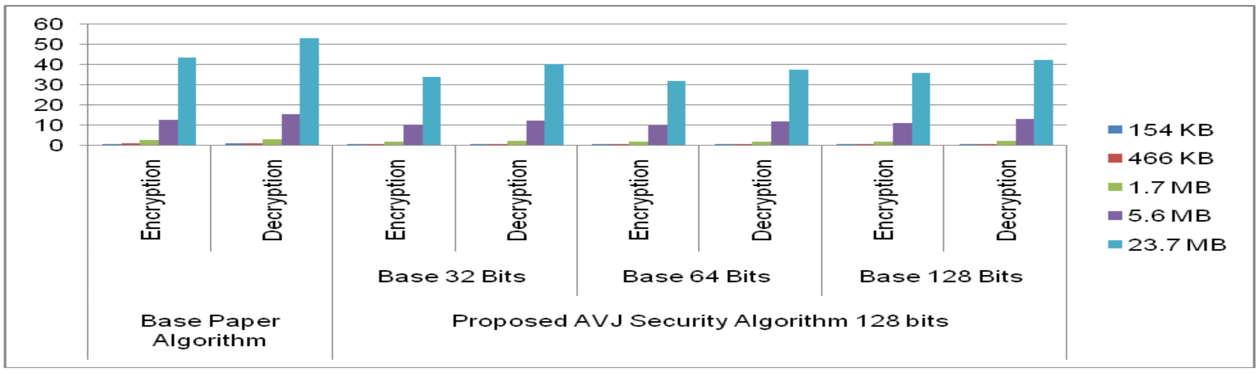
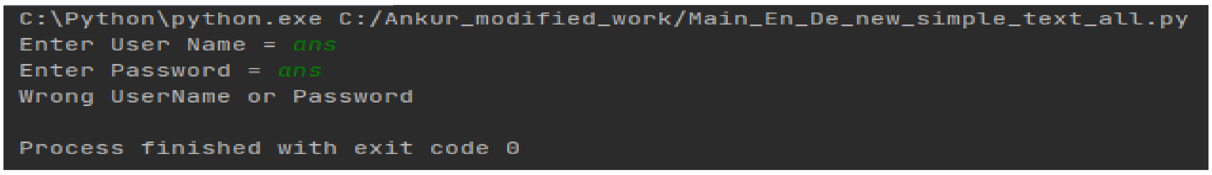


Fig.10. Comparison with base paper algorithm and different base 32/64/128 bits of proposed AVJ security algorithm (128 bits) for various file sizes

From above comparison we have found that our proposed AVJ security algorithm (128 bits) is better than existing one in terms of time efficiency.

Following figure 11 shows our proposed AVJ security algorithms provides data authentication. First our program will ask for username and password if both are correct then only it will run otherwise it will give simple message that wrong username or password as shown in below figure 11.



```
C:\Python\python.exe C:/Ankur_modified_work/Main_En_De_new_simple_text_all.py
Enter User Name = ans
Enter Password = ans
Wrong UserName or Password

Process finished with exit code 0
```

Fig.11. Proposed AVJ security algorithms (128 bits & 192 bits) with username and password to provide data authentication

Following figure 12 shows our proposed model for proposed AVJ security algorithms.

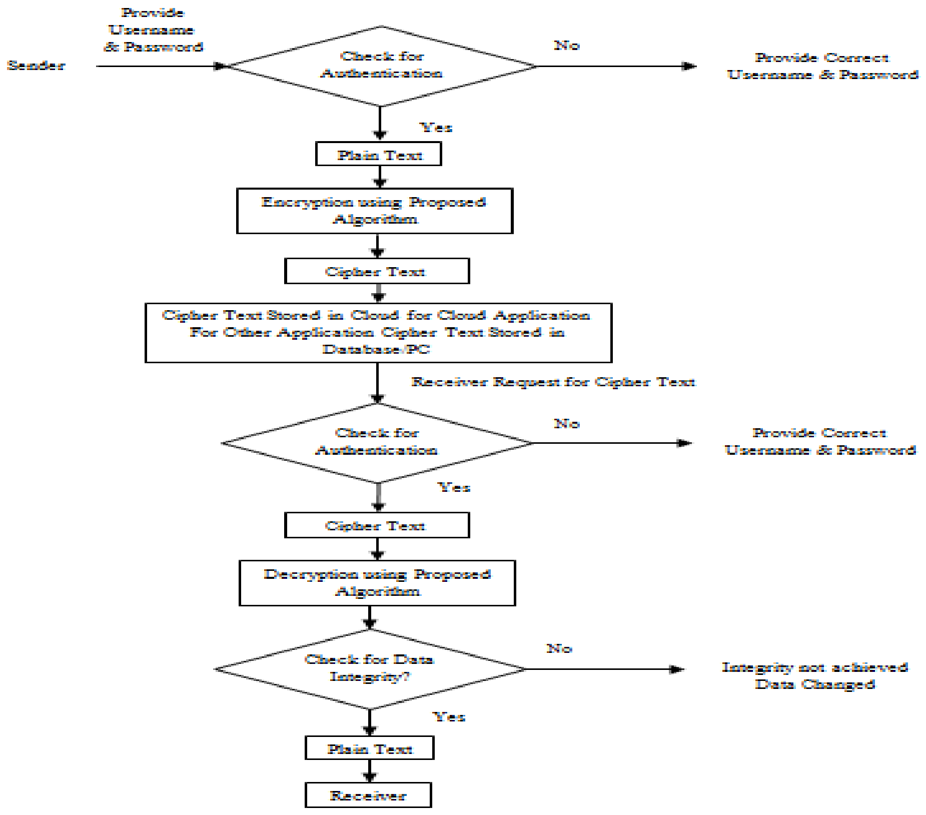

Provide Username & Password
Sender
Check for Authentication
No
Provide Correct Username & Password
Yes
Plain Text
Encryption using Proposed Algorithm
Cipher Text
Cipher Text Stored in Cloud for Cloud Application For Other Application Cipher Text Stored in Database/PC
Receiver Request for Cipher Text
Check for Authentication
No
Provide Correct Username & Password
Yes
Cipher Text
Decryption using Proposed Algorithm
Check for Data Integrity?
No
Integrity not achieved Data Changed
Yes
Plain Text
Receiver

Fig.12. Proposed Model for Proposed AVJ security algorithms

CONCLUSION

Our proposed algorithms aim to resolve security issues of data and provide data confidentiality, data authentication and data integrity. These algorithms provide security to text data of size 128 bits (16 bytes) or 192 bits (24 bytes). These algorithms mainly design for providing security to cloud data but it is applicable to every application where security is needed for text data. We have implemented this algorithm for simple 16 bytes/24 bytes of text data or for csv file format data. Therefore, further work may be done for different types of file or data formats including audio, video, images, etc.

REFERENCES

[1]. JayachanderSurbiryala, “PhD Forum: Improving the Security for Storing the Big Data in Cloud Environment”, IEEE (2017)
[2]. DeepakPuthal, SuryaNepal, RajivRanjan, JinjunChen, "A dynamic prime number based efficient security mechanism for big sensing data streams", Elsevier (2017)
[3]. Nikhil Dwivedi, Arun Malik, "Analysis of Novel Hybrid Encryption Algorithm for Providing Security in Big Data", Springer (2019)
[4]. Mahnaz Nasseri, Mahdi Jameii, Mohsen Mohseni, "A New Method for Improving the Security and Access Time to Data Blocks in Big Databases", IEEE (2018)

[5]. 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).
[6]. Xiaxia Niu, Yan Zhao, "Research on Big Data Platform Security Based on Cloud Computing", Springer (2019)
[7]. Mr. Ankur N. Shah, Dr. Jay A. Dave “Design and Development of an Algorithm to Secure Big Data on Cloud Computing”, ICTIS Springer(2021)
[8]. https://www.fortinet.com/resources/cyberglossary/data-security#:~:text=Data%20security%20is%20the%20process,and%20organizations'%20policies%20and%20procedures,last accessed on 09-may-2023.
[9]. https://www.microfocus.com/en-us/what-is/data-security, last accessed on 09-may-2023.
[10].https://www.imperva.com/learn/data-security/data-security/, last accessed on 09-may-2023.

DOI: https://doi.org/10.15379/ijmst.v10i2.3123