Design and Analysis of New Symmetric Block Cipher Algorithm

KALAICHELVI V && MANIMOZHI K
Asst. Professor, SRC- SASTRA University, Kumbakonam, India
E-mail: kalaichelvi2k@yahoo.com

Abstract

Cryptography provides many methods and techniques for secure communication. Currently there are many encryption/decryption algorithms such as DES, AES (Rijndael), Blowfish, etc., are published. However, they are fairly complex and require that one spend a lot of time to comprehend and implement them. This paper introduces simple encryption /decryption algorithm that works fast and fairly secure. In this paper, a new symmetric block cipher algorithm has been developed and is compared with the existing popular algorithms. Simulation results are given to demonstrate the effectiveness of each algorithm. Blowfish and the newly developed algorithms exhibit better performance than other popular existing algorithms.

Keywords: Cryptography, Encryption, decryption & Block Cipher

1.0 Introduction

As of today, computer technology has been highly developed. Different types of applications require specialized security levels. Security is the primary concern of all those people who deal with activities, which involve protection of risk. The branch of science cryptography, which is concerned with the security of information, developed in the hands of military people and it was nurtured by them for quite a long time as their private property. For this reason many algorithms are developed for encryption and decryption which provides high security. All these algorithms are kept open to the public and the secrecy of the algorithm lies entirely in the key. This paper stands different that the development of algorithm addresses the user needs in specific, there by more flexibility.

2.0 Proposed Algorithm

All the existing algorithms are based on the Feistel cipher structure except AES. The proposed algorithm is also based on the Feistel cipher structure. It is a block cipher. It operates on 64-bit plain text blocks. The key is 128 bit long. The same algorithm is used for both encryption and decryption. *Fig 1*. shows an overview of the proposed algorithm. The 64-bit data block is divided into two 32-bit sub blocks: X_L and X_R. These two sub blocks become the input to the first round of the algorithm. There are 16 rounds in total. In each round four 32-bit sub keys are used. The processing of the plain text proceeds in four phases.

At first, the two 32-bit sub blocks of plain text passes through an initial substitution. This is followed by a phase consisting of 16 rounds of the same function; then, the left and right halves of the output are swapped. Finally, the output is passed through a inverse substitution to produce the 64-bit Cipher text.

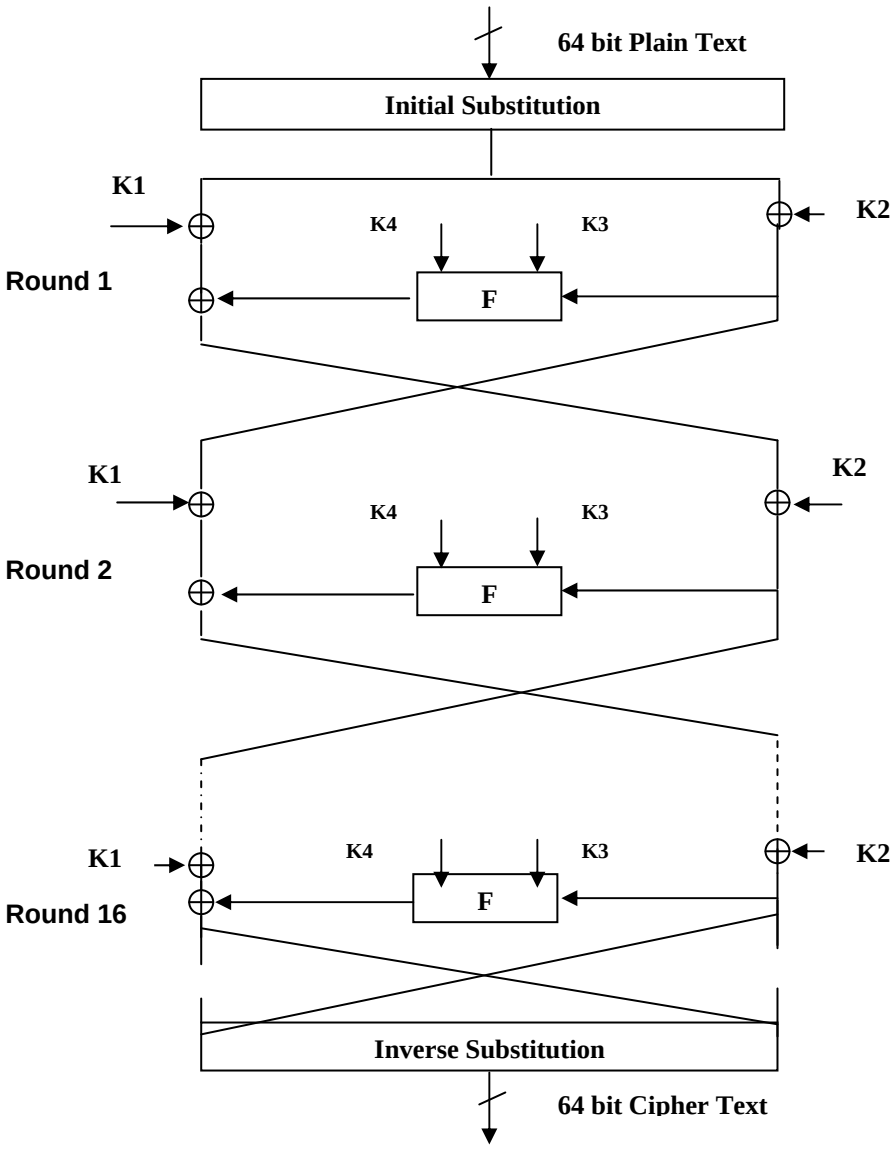


Fig1. Diagrammatic Representation of Proposed Algorithm

2.1 Initial Substitution

This substitution table is constructed from the ASCII table. Initially, based on the type of characters the ASCII table is divided into 33(0 – 32) blocks. To construct the substitution table [*Table 1*]., it uses permutation (P33) value and the procedure is presented already [3].

In this table, the row indicates left digit and the column indicates the right digit. For example, if the plain text is “a”, the corresponding ASCII value is 97, but it is converted into 206 (row 9 and column 7) by using the above substitution table. Similarly, all the plain texts are transformed by using Initial substitution.

Table 1. Substitution Table

	0	1	2	3	4	5	6	7	8	9
0	125	126	127	128	129	130	131	132	248	249
1	250	251	252	253	254	255	117	118	119	120
2	121	122	123	124	240	241	242	243	244	245
3	246	247	109	110	111	112	113	114	115	116
4	232	233	234	235	236	237	238	239	99	100
5	101	102	103	104	105	106	107	108	225	226
6	227	228	229	230	231	92	93	94	95	96
7	97	98	219	220	221	222	223	224	85	86
8	87	88	89	90	91	213	214	215	216	217
9	218	79	80	81	82	83	84	206	207	208
10	209	210	211	212	73	74	75	76	77	78
11	199	200	201	202	203	204	205	67	68	69
12	70	71	72	195	196	197	198	58	59	60
13	61	62	63	64	65	66	186	187	188	189
14	190	191	192	193	194	50	51	52	53	54
15	55	56	57	176	177	178	179	180	181	182
16	183	184	185	40	41	42	43	44	45	46
17	47	48	49	166	167	168	169	170	171	172
18	173	174	175	31	32	33	34	35	36	37
19	38	39	158	159	160	161	162	163	164	165
20	23	24	25	26	27	28	29	30	150	151
21	152	153	154	155	156	157	16	17	18	19
22	20	21	22	141	142	143	144	145	146	147
23	148	149	8	9	10	11	12	13	14	15
24	133	134	135	136	137	138	139	140	0	1
25	2	3	5	6	7					

2.2 Round Function

The subsequent steps (shown in Fig 2) are executed 16 times. This round function uses three functions namely, Mapping function, Folding function and odd-even function.

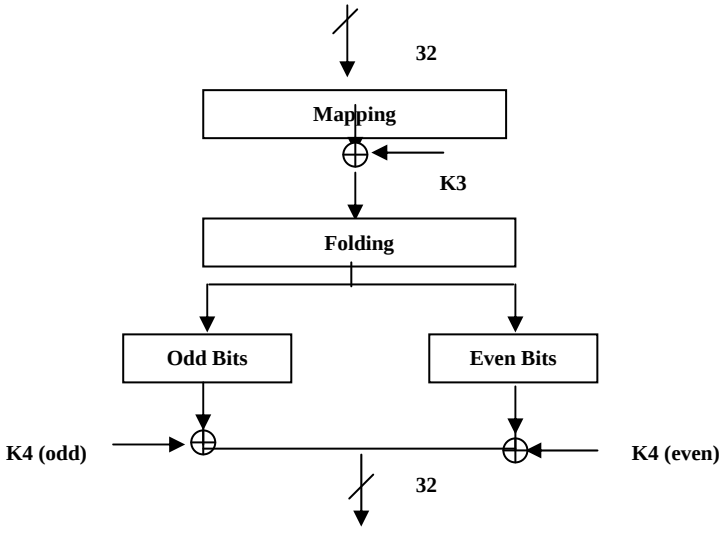


Fig 2. Round Function

2.2.1 Mapping Function:

Let A = [00002, 11112]. The mapping function is a bijection [6] f: A → A | f(x) ≠ x, for every x in A. Since 'f' is a bijection, there exists the inverse function f^{-1}: A → A | x = f^{-1}(f(x)). This means that if we encrypt a 4-bit input b1b2b3b4, using the function 'f', we can always get the original bits back using the inverse function f^{-1}. That is b1b2b3b4=f^{-1}(f(b1b2b3b4)). An example of a bijection function is a 16-element array E is defined as following:

E[0000] = 0001; E[0001] = 0010;

E[0010] = 0011; E[0011] = 0100;

E[0100] = 0101; E[0101] = 0110;

E[0110] = 0111; E[0111] = 1000;

E[1000] = 1001; E[1001] = 1010;

E[1010] = 1011; E[1011] = 1100;

E[1100] = 1101; E[1101] = 1110;

E[1110] = 1111; E[1111] = 0000;

The corresponding inverse function is the 16-element array D is defined as following: D[E[b1b2b3b4]] = b1b2b3b4. It is easily seen that any 4 bits of information b1b2b3b4 encrypted by E[b1b2b3b4] can be decrypted by D[E[b1b2b3b4]].

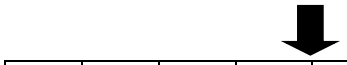
2.2.2 Folding Function

Here, the 32-bit is copied into a two dimensional 4 x 8 array from left to right and from top to bottom [Shown in ***Fig.3***]. The origin of folding is from paper folding [3] nature.

This folding is broadly divided into 2 angles of processing:

1. Horizontal Folding
2. Vertical Folding

CRYPTOGRAPHY 4 SECURE COMMUNICATION



C	**R**	**Y**	**P**	**T**	**O**	**G**	**R**
A	**P**	**H**	**Y**	**4**	**S**	**E**	**C**
U	**R**	**E**	**C**	**O**	**M**	**M**	**U**
N	**I**	**C**	**A**	**T**	**I**	**O**	**N**

N	**I**	**C**	**A**	**T**	**I**	**O**	**N**
U	**R**	**E**	**C**	**O**	**M**	**M**	**U**
A	**P**	**H**	**Y**	**4**	**S**	**E**	**C**
C	**R**	**Y**	**P**	**T**	**O**	**G**	**R**

After Horizontal Folding

N	**O**	**I**	**T**	**A**	**C**	**I**	**N**
U	**M**	**M**	**O**	**C**	**E**	**R**	**U**
C	**E**	**S**	**4**	**Y**	**H**	**P**	**A**
R	**G**	**O**	**T**	**P**	**Y**	**R**	**C**

After Vertical Folding

Fig 3. Types of Folding

4.2.3 Odd-Even Function

The output of the folding function is the key in for this function. The odd number bits of data bits are 'XOR'ed with odd number bits of key (K4) and the even number bits of data bits are 'XOR'ed with even number bits of key (K4). Finally, the above output is 'XOR' ed with the left-hand 32-bit.

2.3 Swapping Function

The left and right halves of the output are swapped. Note that all these operations are performed only on the 32-bit right half portion of the 64-bit. The left half portion is untouched so far. At this juncture, the old right half becomes the new left half and the old left half becomes the new right half.

2.4 Inverse Substitution

From the substitution table, the inverse substitution table [***Table 2***] is constructed.

Table 2. Inverse Substitution Table

	0	1	2	3	4	5	6	7	8	9
0	248	249	250	251	252	253	254	255	232	233
1	234	235	236	237	238	239	216	217	218	219
2	220	221	222	200	201	202	203	204	205	206
3	207	183	184	185	186	187	188	189	190	191
4	163	164	165	166	167	168	169	170	171	171
5	145	146	147	148	149	150	151	152	127	128
6	129	130	131	132	133	134	135	117	118	119
7	120	121	122	104	105	106	107	108	109	91
8	92	93	94	95	96	78	79	80	81	82
9	83	84	65	66	67	68	69	70	71	48
10	49	50	51	52	53	54	55	56	57	32
11	33	34	35	36	37	38	39	16	17	18
12	19	20	21	22	23	0	1	2	3	4
13	5	6	7	240	241	242	243	244	245	246
14	247	223	224	225	226	227	228	229	230	231
15	208	209	210	211	212	213	214	215	192	193
16	194	195	196	197	198	199	173	174	175	176
17	177	178	179	180	181	182	153	154	155	156
18	157	158	159	160	161	162	136	137	138	139
19	140	141	142	143	144	123	124	125	126	110
20	111	112	113	114	115	116	97	98	99	100
21	101	102	103	85	86	87	88	89	90	72
22	73	74	75	76	77	58	59	60	61	62
23	63	64	40	41	42	43	44	45	46	47
24	24	25	26	27	28	29	30	31	8	9
25	10	11	12	13	14	15				

2.5 Sub Key Generation

The algorithm uses 64 sub keys (4 for each of the 16 rounds). First, the 128-bit key is divided into four 32-bit sub keys. These are the four sub keys for the first round. Then, the key is rotated 8-bits to the left and again divided into 4 sub keys and so on until the end of the algorithm.

2.6 Decryption

Decryption process is exactly the same as the encryption process.

2.7 The Strength of Proposed Algorithm

It uses a 128-bit key, which is double than the key size of DES. Thus, to break this algorithm, 2^{128} (ie., 10^{38}) encryption operations would be required.

3.0. Comparison of different Existing Algorithms with the Proposed Algorithm

The results of various popular symmetric key algorithms like DES, 3 DES, Blowfish and AES from [1-2] were analyzed. It is understood that the results of different algorithms were implemented in a uniform language (JAVA), using their standard specifications and it was tested on the same hardware platform ie., Pentium-IV 2.4 GHz machine. In this paper too, the same platform and the same language have been used to compare the performance of proposed algorithm with the existing above algorithms. [*Tables 3-4*].

Table 3.Comparative execution times (in seconds) of encryption algorithms in ECB mode on a P-IV 2.4 GHz machine

InputSize (Bytes)	DES	3DES	AES	Blowfish	Proposed Algorithm
20,527	2	7	4	2	2
36,002	4	13	6	3	4
45,911	5	17	8	4	4
59,852	7	23	11	6	5
69,545	9	26	13	7	7
1,37,325	17	51	26	14	14
1,58,959	20	60	30	16	16
1,66,364	21	62	31	17	17
191,383	24	72	36	19	19
232,398	30	87	44	24	24
Average Time	**14**	**42**	**21**	**11**	**11**
Throughput Bytes/Sec	**8170**	**2681**	**5383**	**10065**	**10065**

Table 4. Algorithm Settings

Algorithm	# of Rounds	Key Size (Bits)	Block Size (Bits)
DES	16	56	64
3DES	16	168	64
AES	10	256	128
Blowfish	16	448	64
Proposed Algorithm	**16**	**128**	**64**

From the results, it is easy to understand that Blowfish and the proposed algorithm have the same throughput [***Figs 4-5***]. AES shows poor performance results as compared to other algorithms since it requires more processing power than DES. Amazingly it also shows that 3DES has almost 1/3 throughput of DES, or in other words, it needs 3 times more than DES to process the same amount of data. Proposed algorithm has a better performance when compared with the other existing algorithms except Blowfish.

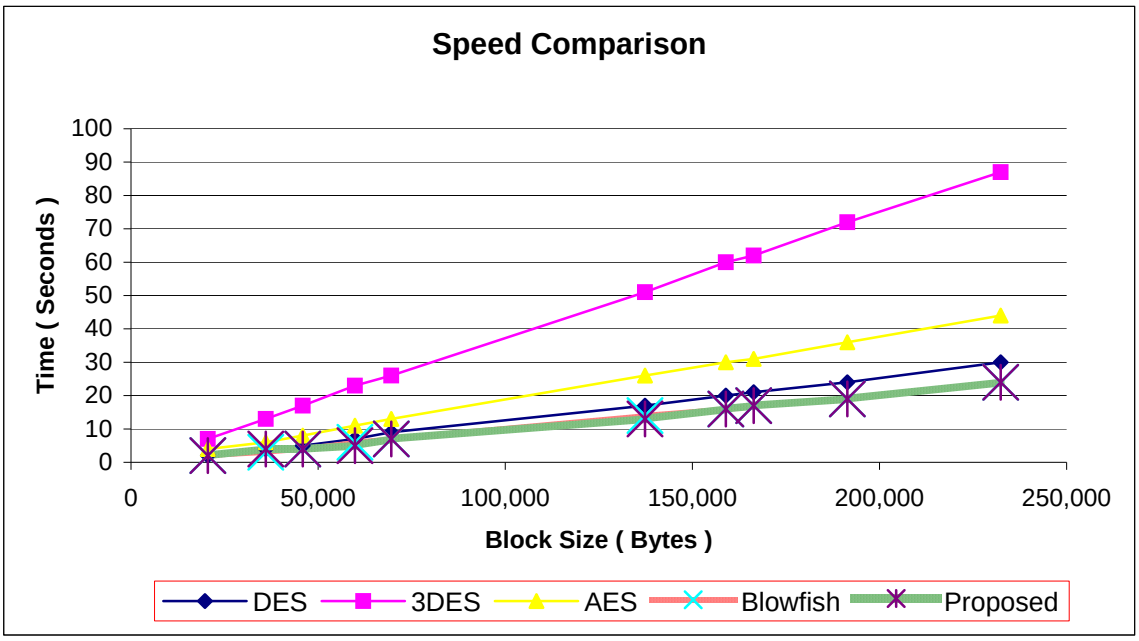


Fig 4. Performance results with ECB mode

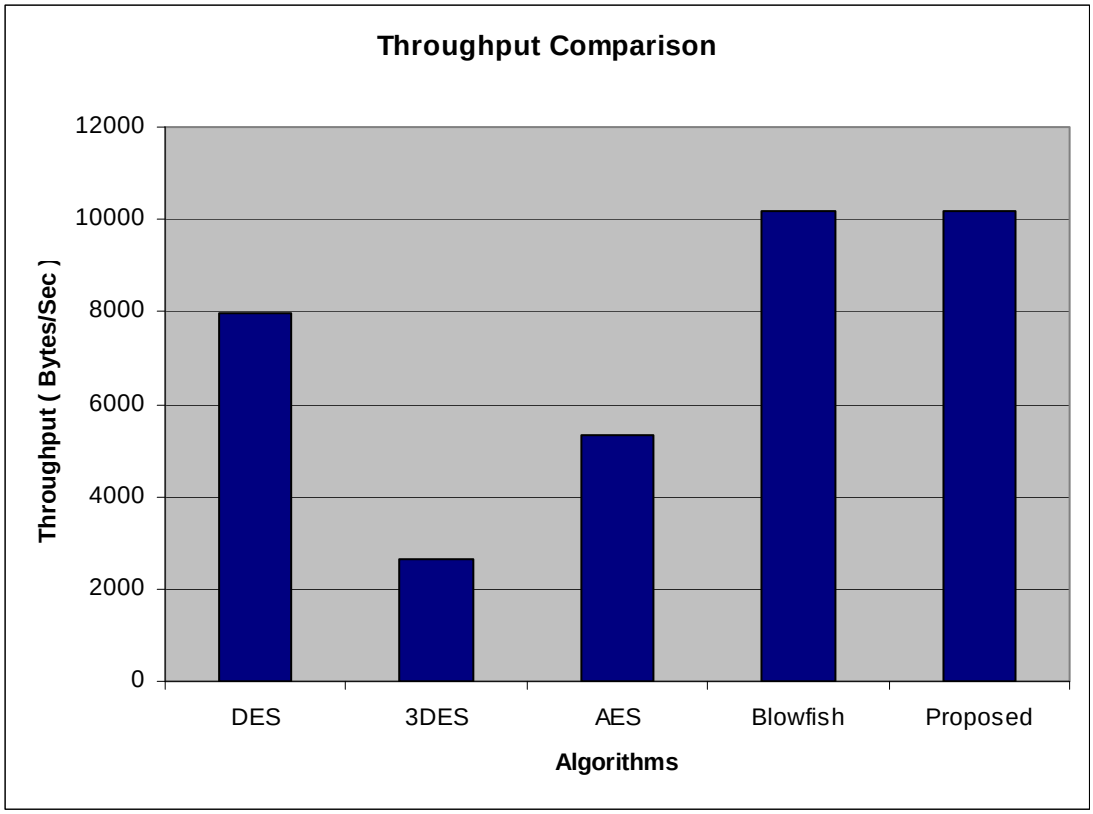


Fig 5. Throughput Comparison

4.0. Experimental Results

By considering different sizes of data blocks (0.5MB to 20MB) the algorithms are evaluated in terms of the execution time required to encrypt and decrypt the data. The Simulation program [***Fig. 6***] accepts two inputs: algorithm and data block. After a successful execution of encryption process, it shows the encrypted data in the form of some combination of special symbols and it also shows that execution time. Similarly, the encrypted data is decrypted and it shows the original message.

4.1 The effect of changing block size for cryptography algorithm on power consumption

Encryption time is used to calculate the throughput of an encryption scheme. It indicates the speed of encryption. The throughput of the encryption scheme is calculated by dividing the total plaintext in Megabytes encrypted on the total encryption time for each algorithm. As the throughput value is increased, the power consumption of this encryption technique is decreased.

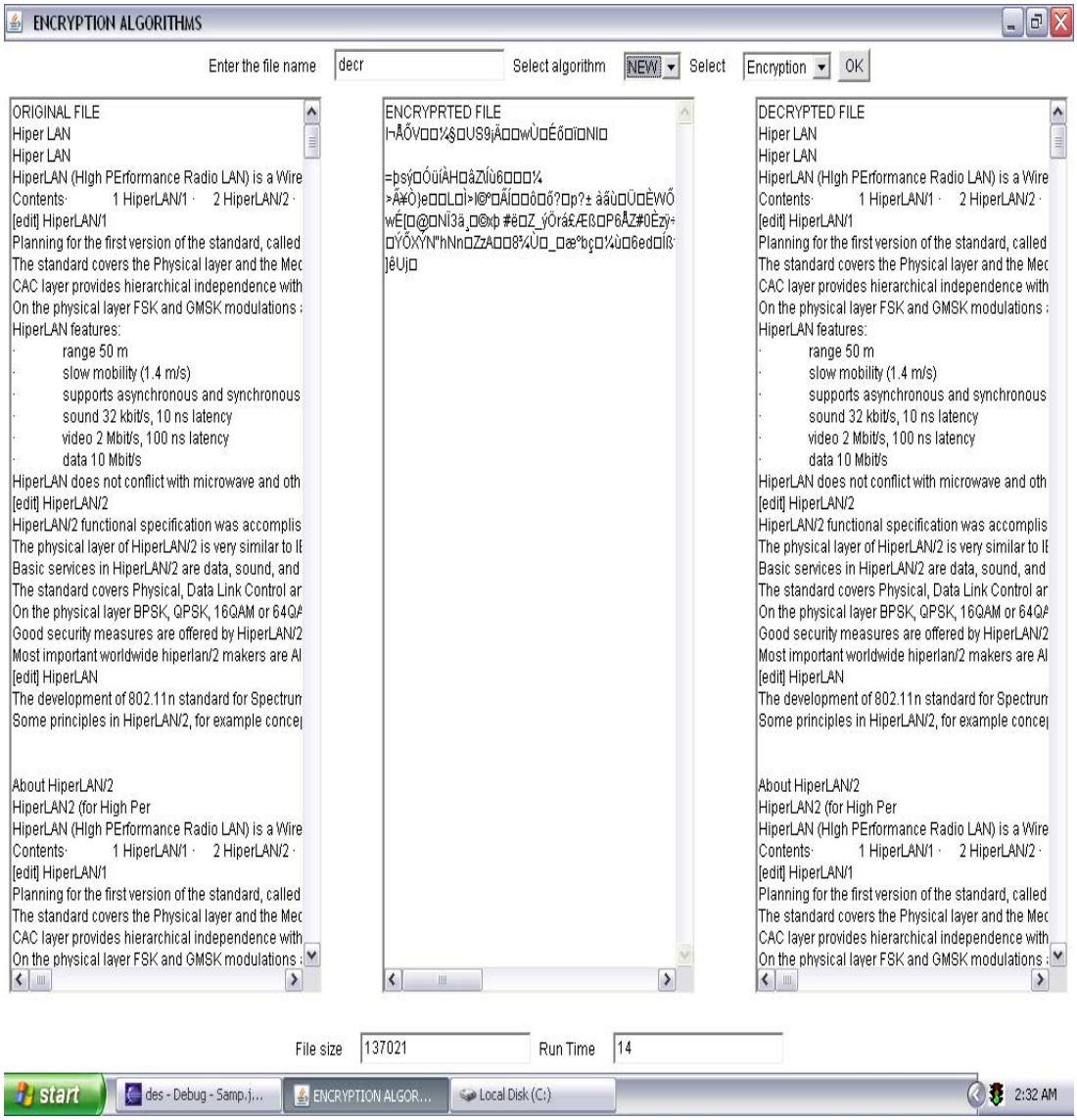


Fig 6. GUI of the Simulation Program

From the results, the following ***inferences*** can be derived:

(i) Blowfish and the proposed algorithm are superior to the other algorithms in terms of the processing time

(ii) DES requires less time than all algorithms except Blowfish and the proposed one.

(iii) AES has an advantage over 3DES.

(iv) 3-DES has low performance in terms of power consumption and throughput, when compared with DES and AES. Always, it requires more time than DES because of its triple phase encryption characteristics.

5.0. Conclusion

This paper presents a new symmetric block cipher algorithm and its performance is compared with the existing popular algorithms like DES, 3DES, Blowfish and AES. Important conclusions are:

(i) There is no significant difference between popular Blowfish algorithm and the proposed algorithm. Because, both Blowfish and the proposed algorithm have the same time complexity.

(ii) In the case of changing packet (block) size, it was confirmed that Blowfish and the proposed algorithm have better performance than the other common encryption algorithms.

(iii) AES showed poor performance results compared to the other algorithms except 3DES since it requires more processing power.

(iv) Blowfish and the newly proposed algorithm consume less amount of processing time.

References

[1] Aamer Nadeem et al, "A Performance Comparison of Data Encryption Algorithms", IEEE 2005
[2] D.S.Abdul, M.M.Hadhoud et al, “Performance Evaluation of Symmetric Encryption Algorithms”, Volume 8, IJCSNS (International Journal of Computer Science and Network security) 2009.
[3] Kalaichelvi. V && RM.Chandrasekaran “FSP Algorithm for Encryption/Decryption”, ISBN: 978-1-4244-3595-1/08 IEEE 2008
[4] Kalaichelvi V, Subha N, Venkatesh M “H_2O algorithm for secure communication” International Journal of mathematical Science 2006,5(2),237-47.
[5] William Stallings "Cryptography and Network Security", Pearson Edn Pvt.Ltd.
[6] Bruce Schneier, "Applied Cryptography" , John Wiley & Sons, Inc 1996
[7] Richard Smith "Internet Cryptography",Pearson Edn Pvt.Ltd
[8] Atul Kahate “Cryptography and Network Security”, Tata Mc.Graw Hill Edn.
[9] [RFC2828],"Internet Security Glossary", http://www.faqs.org/rfcs/rfc2828.html
[10] "Block Cipher", http://en.wikipedia.org/wiki/Block_cipher