Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-022-04379-7

ORIGINAL RESEARCH

DeCrypt: a 3DES inspired optimised cryptographic algorithm

Deepraj Chowdhury[1] · Ajoy Dey[2] · Ritam Garai[3] · Subhrangshu Adhikary[3] · Ashutosh Dhar Dwivedi[4] · Uttam Ghosh[5] · Waleed S. Alnumay[6]

Received: 7 September 2021 / Accepted: 30 July 2022

Abstract
Triple Data Encryption Standard (also known as 3DES) is a symmetric encryption algorithm. European Traffic Management System popularly uses 3DES for authentication and encryption. However, as per a draft published by NIST in 2018, 3DES is officially being retired and not suggested to use for new applications. Several attacks were imposed on 3DES, and the biggest threat to 3DES is meet in the middle attack. Therefore for long term security, it is essential to enhance the security of such algorithms. This paper proposed a new cipher DeCrypt inspired by 3DES, an improved version of the 3DES algorithm, and secured against meet in the middle attack. As per the experiment performed, DeCrypt cipher is 61 per cent faster than 3DES, providing long-term and better security against sweet-32 attacks than other symmetric algorithms. The proposed algorithm is also faster than 3DES due to reduced encryption and decryption time.

Keywords Symmetric cryptography · Feistel cipher · Meet in the middle attack · Sweet-32 attack

1 Introduction

Data encryption standard (DES) was the first internationally accepted cryptographic standard which is now obsolete because of compromised security (Patil et al. 2016). IBM developed DES as a standard cryptosystem, and the U.S. National Bureau of Standards adopted it for 20 years. Further newer versions of the algorithm have emerged, such as 2DES and 3DES, which are more secure than the original DES but are still not as secure and fast as the newly introduced AES algorithm (Mehmood et al. 2019). With the emergence of the Advanced Encryption Standard (AES), the requirement of DES was obsolete as AES is much securer (Chowdhury et al. 2021). As the cryptography techniques hav-e evolved, different types of intrusion techniques have also emerged by which the attackers intend to eavesdrop and understand the encrypted message (see (Amorado et al. 2019; Semenov et al. 2018; Wang et al. 2022; Sanwar Hosen et al. 2022)). DES is mainly vulnerable against several types of attacks such as Brute force, Meet-in-the-Middle (MITM) and Sweet32 attack. Brute force attacks are the easiest and work as a hit and trial method, and random strings are

✉ Ashutosh Dhar Dwivedi
add.digi@cbs.dk; ashudhar7@gmail.com

Deepraj Chowdhury
deepraj19101@iiitnr.edu.in

Ajoy Dey
deyajoy80@gmail.com

Ritam Garai
ritam.garai@spiraldevs.com

Subhrangshu Adhikary
subhrangshu.adhikary@spiraldevs.com

Uttam Ghosh
ughosh@mmc.edu

Waleed S. Alnumay
wnumay@ksu.edu.sa

[1] Department of Electronics and Communication Engineering, IIIT, Naya Raipur, India

[2] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India

[3] Department of Computer Science and Engineering, Dr. B.C. Roy Engineering College, Durgapur, India

[4] Centre for Business Data Analytics, Department of Digitalization, Copenhagen Business School, Frederiksberg 2000, Denmark

[5] Department of CS and DS, School of Applied Computational Sciences (SACS), Meharry Medical College, 1005 Dr DB Todd Jr Blvd, Nashville, TN 37208, USA

[6] Riyadh Community College, Computer Science Department, King Saud University, Riyadh, Saudi Arabia

Published online: 23 September 2022

encrypted and matched with the encoded string until the perfect match is found. This, however, requires a huge amount of time to test different combinations and is therefore often not used in its original form and instead used in combination with other attack techniques (Lin et al. 2021). The meet-in-the-Middle attack is a popular intrusion technique that works by space or time tradeoff to optimize brute force. It generally requires a plaintext block and its corresponding ciphertext (Lin et al. 2014). Within the 3DES algorithm, the MITM attack generally takes place during the intermediate stages of the 3DES encryption algorithm (Siddiqui et al. 2020b). Sweet32 is another attack that targets the cryptographic algorithm's design flaws by recovering some portions of the plaintext from the ciphertext. Sweet32 attack is effective in deciphering the 3DES algorithm. Different approaches have been made to make 2DES and 3DES more securer and faster, including different parallelizing techniques, tweaking the number of blocks and block size, introducing streaming methods while encrypting data, etc. But each of them has some demerits (Ahmed et al. 2021; Dwivedi 2020). For example, parallelizing the algorithm makes the encryption and decryption faster, but as the processes are partly or fully independent, each process provides scope for a new attack (Koo and Qureshi 2021; Singh and Singh 2022; Thakur et al. 2019). Similarly, tweaking block size and the number of blocks often slows down the processing speed. Different algorithms have been introduced, but they still have different kinds of vulnerabilities. Original DES is much weaker than other algorithms, and with present technology, it is vulnerable to several attacks. 2DES and 3DES algorithms are vulnerable to multiple attacks (Vuppala et al. 2020; Chong and Salam 2021; Dwivedi 2021). These all vulnerabilities and optimisation issues for different algorithms require to be addressed within one algorithm to make it an ultimate algorithm that motivates to introduce a novel "DeCrypt" algorithm, which 3DES inspire.

1.1 Our contribution

The limitations discussed above of the existing 3DES algorithm makes it weaker than the AES algorithm for encryption. Also, besides these, 3DES is slower compared to AES. Therefore to overcome these limitations, the article presents a novel cipher called "DeCrypt" inspired by 3DES, which is much securer and faster than 3DES and can be further improved to make it comparable to AES. The main contribution of the paper includes:

- This paper proposed a new cipher DeCrypt inspired by 3DES, but DeCrypt is optimized, secured, and provides long-term security compared to 3DES.
- DeCrypt provides strong security against major cryptanalytic attacks such as Brute Force, Meet-in-the-Middle and Sweet32.
- DeCrypt is faster than 3DES in terms of encryption and decryption.

1.2 Related works

The Data Encryption Standard (DES) attracted the attention of the whole world in 1977 after the National Bureau of Standards (see Morris et al. 1977) specified the cipher. In 1976, before adoption as a standard, several researchers like Pohlig, Diffie, Schroeppel, Washington, Merkle and Hellman raised concern about its small key size. Diffie and Hellman (in Diffie and Hellman 1977) raised concern about the key size of 56 bits. Most of the researchers predicted that hardware exists that can easily crack DES within a day after a few years.

DESX (in Biryukov 2011) is an upgrade to DES, which uses a key whitening technique to make the DES resistant to brute force attacks. Rivest proposed DESX in 1984, but he never revealed its design in any journal or conference at that time. The algorithm was implemented within RSA Data Security products and described in the documentation of the product. However, this had improved the security but was not comparable to AES. Advanced slide attacks have been found to be effective in breaking DESX algorithm.

Blowfish De Cannière (2005) is a similar and relatively secure algorithm that works on 64-bit block size while key length could range from 32 to 448 bits which work on 16 round Feistel cipher with large independent S-boxes. Files larger than 4GB are not recommended to encrypt with Blowfish because of their relatively small block size. Blowfish is vulnerable to Sweet32 attack because of its small block size.

2 Structure of the algorithm

Decrypt is a symmetric cipher, and therefore the same key is used to encrypt and decrypt the data. The algorithm is developed as a better substituent to the existing 3DES cipher. The algorithm is developed taking two important factors into account – *Security* and *Speed*. It uses a key of the effective length of 168 bits which provides sufficient security against Brute force attack and is as par with the NIST standards for key sizes. In combination with the longer key length, a single encryption process is used to ensure protection against the Meet in the Middle Attack. Usage of single encryption also provides a greater speed of the algorithm compared to its other counterparts. The plaintext is used in the form of blocks of 192 bits for the cipher. The basic structure of the algorithm is derived from the existing DES with changes made to the Feistel structure and the size of blocks. The

permutations and internal mechanisms of the Feistel structure incorporate diffusion factors, ensuring better mixing protecting against attack. The usage of 8 S-Boxes implements the confusion factor in the algorithm. The compact design of the S-Boxes protects from linear and differential cryptanalysis. The salient feature of the algorithm lies in its simplicity. It uses basic operations like XOR and permutations, which can be easily implemented over software and hardware.

The algorithm takes plaintext as a block of size 192 bits. These 192 bits are first divided into three equal blocks of length 64 bits and are processed through a 16 round Feistel cipher.

2.1 Encryption process

The encryption process comprises several steps to be followed in a synchronised order. The first step involves splitting the plaintext block (192 bits) into three equal-sized (64 bits) blocks. The next step consists in processing these three blocks through a pre-defined Permutation Box (64 bits) which rearranges the bit pattern of these blocks. In the third step, the 16 round Feistel cipher is used. The last step takes the output from the Feistel cipher and process it through the second Permutation Box (64 bits), and the final ciphertext is obtained (see Fig. 1).

2.2 Initial and final permutation

Both are bitwise permutations and do not increase any security to the cipher. The initial permutation is denoted as IP (see Table 1) while the final permutation is IP^{-1} (see Table 2) and both perform inverse to each other.

2.3 Single round processing

A single-round consists of three Feistel structures (each size with 64 bits input). Each Feistel input is further sub-divided

Table 1 Permutation box IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

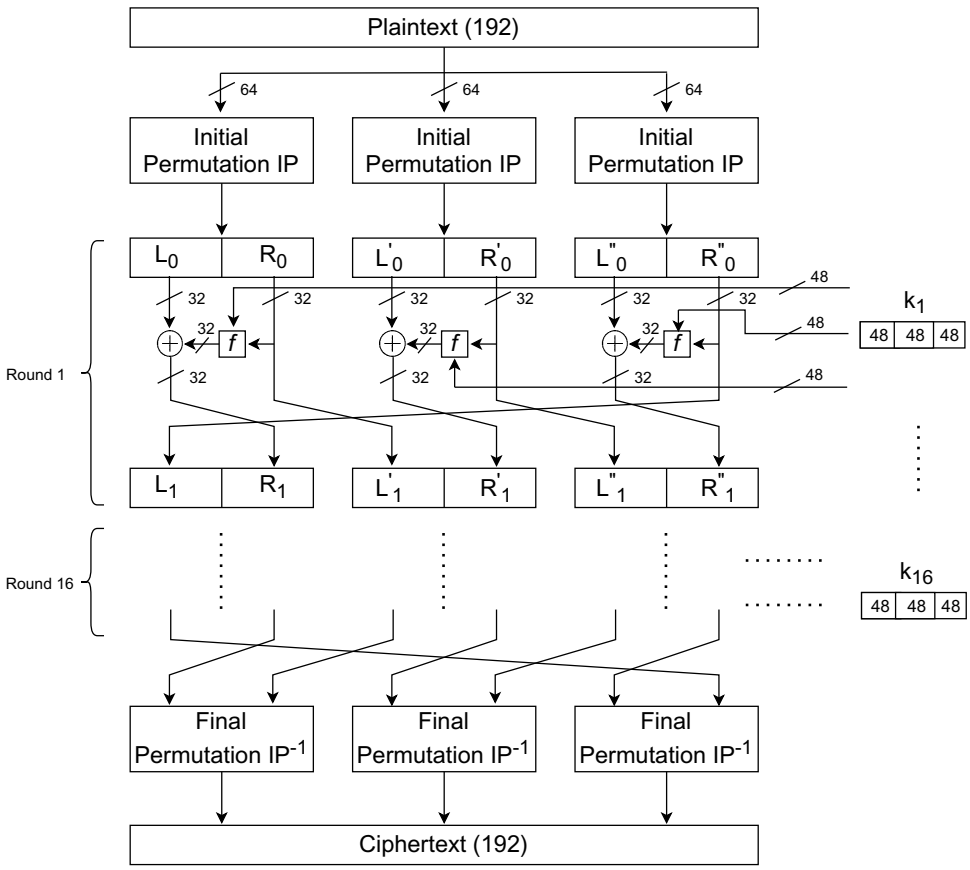


Fig. 1 Feistel structure of decrypt

Table 2 Permutation box IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

into two halves of 32 bits. Left Feistel block is divided into L_i and R_i, middle Feistel block into L_i' and R_i'; and right Feistel block into L_i'' and R_i''. The sub-key K_i is also split into three equal parts (each 48 bits), say K_L (to be used with left block), K_M (to be used with middle block) and K_R (to be used with right block). First, the right half of each Feistel block, i.e. (R_i, R_i', R_i'') and subsequent sub-keys (K_L, K_M, K_R) are processed through the round F function. The round function involves several simple operations like expansion, XORing and further compression using S-boxes. The second step involves XORing left halves with output from the round function. The last step involves swap. The entire process is repeated for 16 rounds.

$$L_i = R_{i-1}'' \tag{1a}$$

$$L_i' = R_{i-1} \tag{1b}$$

$$L_i'' = R_{i-1}' \tag{1c}$$

$$R_i = L_{i-1} \oplus f(K_L, R_{i-1}) \tag{1d}$$

$$R_i' = L_{i-1}' \oplus f(K_M, R_{i-1}') \tag{1e}$$

$$R_i'' = L_{i-1}'' \oplus f(K_R, R_{i-1}'') \tag{1f}$$

2.4 The $\mathcal{F}$ -function

The f-function plays an important role (see Fig. 2). Firstly the right half (32 bits) is expanded into 48 bits using a predefined Expansion Box (E) to match the sub-key length. The next step involves XOR operation with key. In the final step, 8 S-Boxes are used, which takes 48 bits as input and return 32 bits output.

2.4.1 Expansion E

The Expansion Box used in Decrypt is given in Table 3. The Expansion Box takes 32-bit input and returns a 48-bit output. The expansion is obtained by repeating 16 input bits to appear twice in the output. The remaining 16 bits appear only once.

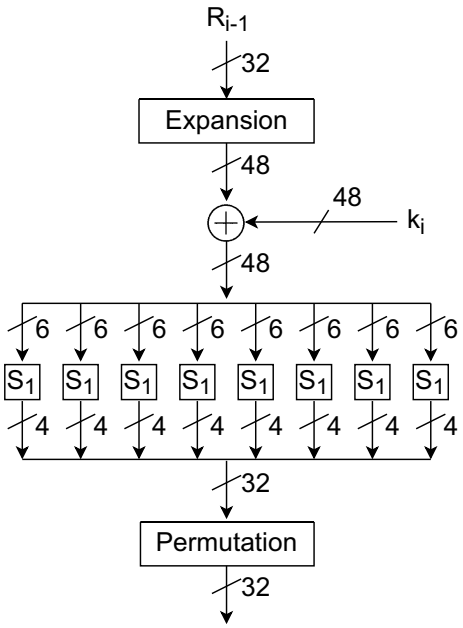


Fig. 2 f-Function

2.4.2 S-boxes

S-boxes incorporate the confusion factor into the algorithm. These are the only non-linear component of the algorithm. The entire security of the algorithm depends on the robustness of these S-boxes. The 8 S-boxes used in DeCrypt are different, but their function is essentially the same. Each S-Box takes 6 bits as input and returns 4 bits. The working of these S-Boxes can be understood well through an example. Let's take a look into S-Boxes used in DeCrypt (Tables 4, 5, 6, 7, 8, 9, 10, 11).

Let us suppose we have a six-bit input to the S-Box-4 as $r_1c_3c_2c_1c_0r_0$. The row number is obtained by taking the decimal equivalent of outermost bits, i.e. r_1r_0, and the column number is obtained by taking the decimal equivalent

Table 3 Expansion box (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Table 4 S-box - 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Table 5 S-box - 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Table 6 S-box - 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Table 7 S-box - 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Table 8 S-box - 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Table 9 S-box - 6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

of the remaining bits, i.e. $c_3c_2c_1c_0$. For example, if the input is 001101, the row number will be the decimal equivalent of 01, i.e. 1, and the column number will be the decimal equivalent of 0110, i.e. 6. From S-Box-4, we can infer the output to be 0, i.e. 0000 (4-bit equivalent).

2.4.3 Permutation within f-function

The function f uses a permutation P (see Table 12). This permutation provides diffusion to the cipher, similar to the initial and final permutation.

Table 10 S-box - 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	8	1	4	10	7	9	5	0	15	14	2	3	12	

Table 11 S-box - 8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 12 Permutation box (P) in f-function

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

2.5 Round key generation

This section describes the entire 16 round sub-key generation process (see Fig. 3). The first step involves removing all the parity check bits from the 192 bits key. For this purpose, the key is passed through a permutation box (see Table 13) (removes all the 8^{th} bits), which gives a 168-bit output. The next step involves splitting this 168-bit key into 3 equal divisions of 56 bits each. The 56-bit division blocks are further divided into two 28 blocks each, which further undergo the right circular bit shift depending upon the round number. The shift in each round is given by a list 2.

$$[1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1] \tag{2}$$

After a bit shifting, the 3 halves are merged. The merged key is passed through another Permutation Box (see Table 14), which reduces the key size from 168 bits to 144 bits. The sub-key for i^{th} round is thus obtained.

2.6 Decryption process

The decryption process is essentially similar to the encryption process, with a slight change in the Feistel structure. The decryption process takes ciphertext as input and is processed through the same procedure with sub-keys used in the reverse order. The sub-key k_i(168 bits) is divided into 3 equal halves K_L,K_M,K_R. The changes made to the Feistel structure are as follows:

– If i^{th} round is a multiple of 3, the operation is as follows:

$$R_i = L_{i-1} \oplus f(K_L, R_{i-1}) \tag{3a}$$

$$R'_i = L'_{i-1} \oplus f(K_M, R'_{i-1}) \tag{3b}$$

$$R''_i = L''_{i-1} \oplus f(K_R, R''_{i-1}) \tag{3c}$$

– If i%3 is equal to 1, the operation is as follows:

$$R_i = L_{i-1} \oplus f(K_R, R_{i-1}) \tag{4a}$$

$$R'_i = L'_{i-1} \oplus f(K_L, R'_{i-1}) \tag{4b}$$

$$R''_i = L''_{i-1} \oplus f(K_M, R''_{i-1}) \tag{4c}$$

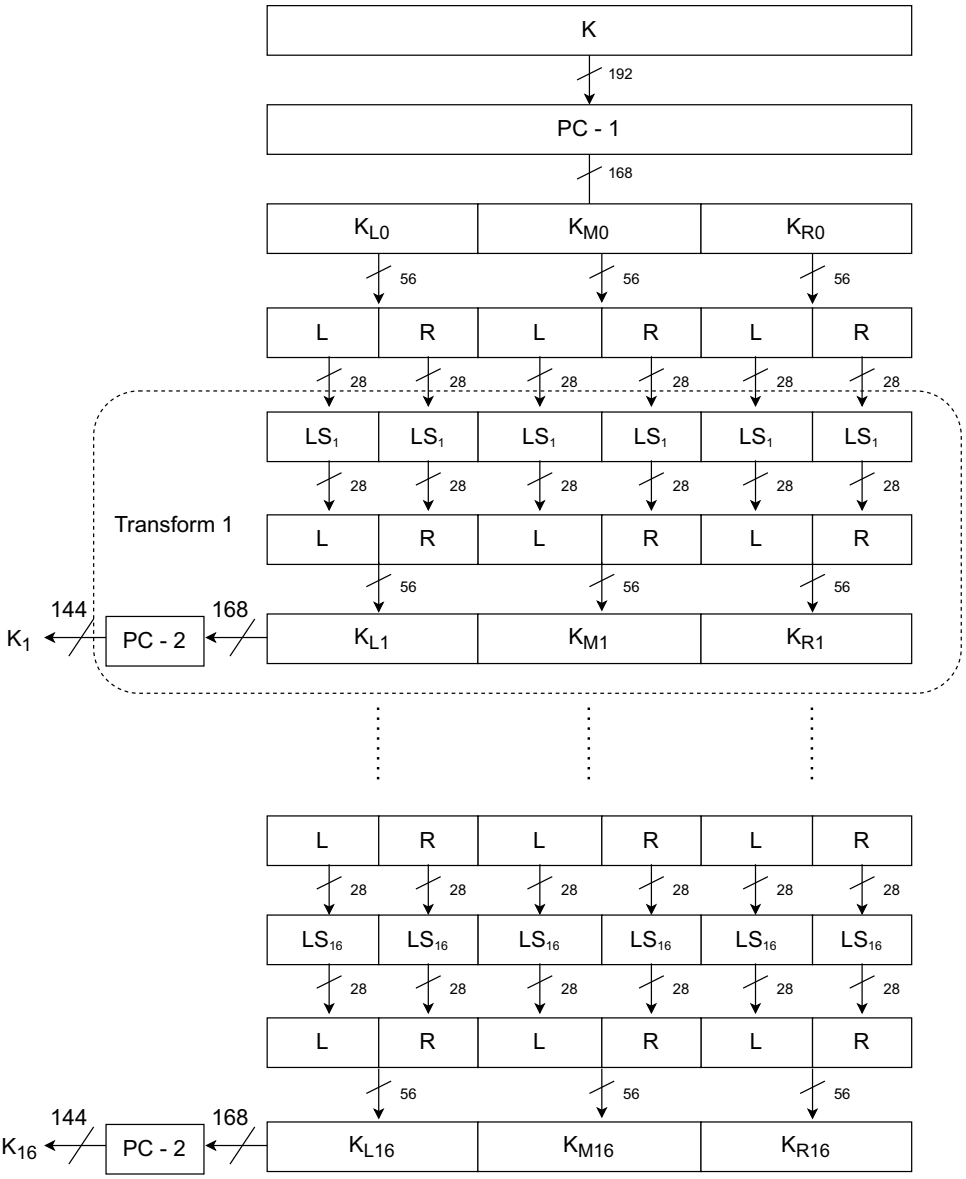
– If i%3 is equal to 2, the operation is as follows:

$$R_i = L_{i-1} \oplus f(K_M, R_{i-1}) \tag{5a}$$

$$R'_i = L'_{i-1} \oplus f(K_R, R'_{i-1}) \tag{5b}$$

$$R''_i = L''_{i-1} \oplus f(K_L, R''_{i-1}) \tag{5c}$$

Fig. 3 Single round processing



3 Results and discussion

Here, in this section, we have provided a comparative study among the existing symmetric ciphers. The comparison is made based on block size, key size, encryption-decryption time, brute force complexity, MITM complexity and protection against sweet 32 attacks (see Table 15).

3.1 Analysis of the proposed algorithm

In this section, the characteristics of the proposed algorithm related to cryptographic strength are listed below:

- **Block length:** The block length of the proposed algorithm should be long enough to deter the statistical analysis. This is valid for our proposed algorithm also. Our algorithm uses 192 bits blocks, which is thrice of the algorithms like DES-X,3-DES and blowfish.
- **Key length:** The key length should be long enough to prevent exhaustive key searches. So in our proposed algorithms, we have used 192 bits, out of which 168 bits of the key are used for encryption, and the rest 24 bits are used for parity checking. So in this area, Our algorithm will be far secure in the near future.
- **Confusion:** The ciphertext in a complicated and associated way should rely on the plaintext and ciphertext, so the multiple swapping among the 6 sub-blocks in all the 16 rounds will be providing a huge confusion for the eve while doing cryptanalysis of the ciphertext and making the reverse engineering tough. Also, the use of S-Box is added more confusion to our algorithm.

Table 13 PC-1

57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4
121	113	105	97	89	81	73	65
122	114	106	98	90	82	74	66
123	115	107	99	91	83	75	67
124	116	108	100	127	119	111	103
95	87	79	71	126	118	110	102
94	86	78	70	125	117	109	101
93	85	77	69	92	84	76	68
185	177	169	161	153	145	137	129
186	178	170	162	154	146	138	130
187	179	171	163	155	147	139	131
188	180	172	164	191	183	175	167
159	151	143	135	190	182	174	166
158	150	142	134	189	181	173	165
157	149	141	133	156	148	140	132

Table 14 PC-2

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	56
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32
70	73	67	80	57	61	59	84
71	62	77	66	79	75	68	60
82	64	72	63	83	76	69	58
97	108	87	93	103	111	86	96
107	101	89	104	100	105	95	112
90	109	102	98	106	92	85	88
126	129	122	136	113	117	115	140
127	118	133	122	135	131	124	116
138	120	128	119	139	132	125	114
153	164	143	149	159	167	142	152
163	157	145	160	156	161	151	168
146	165	158	154	162	148	141	144

- **Diffusion:** Each plaintext bit should affect the each ciphertext bit, and each key bit should effect each ciphertext bit. Avalanche effect is the amount of change in the ciphertext with a bit change in plaintext. More the avalanche effect better the algorithm. In our algorithm, each bit of output depends on each bit of input bit from plaintext and each bit of sub-key in the first round, and the process repeats for further more rounds.

3.2 Security analysis of proposed system

3.2.1 Brute force

A brute force or exhaustive key search attack is the most primitive attack on a cryptographic algorithm. In this attack, an attacker tries to find the key by hit and trails. Eve tries to permute through all the bits of the key to find the correct key.

Table 15 Attack's complexity comparison

Algorithms	Block size	Key length	Brute force	MITM	SWEET-32 storage
DES	64	64	2^{56}	NA	32GB
2-DES	64	128	2^{112}	2^{57}	32GB
3-DES	64	192	2^{168}	2^{112}	32GB
DESX	64	192	2^{184}	2^{120}	32GB
AES-128	128	128	2^{128}	NA	$2.7x2^{11}$ GB
Blowfish	64	32- 448	2^{128}	NA	32 GB
Decrypt	192	192	2^{168}	NA	$1.77X2^{21}$ GB

The brute force complexity is solely dependent on the key size. The best case is when the Eve predicts in 1st attempt, and worst is 2^n, on an average attacker tries for 2^{n-1}.

3.2.2 Meet in the middle attack

Meet in the middle attack is a known-plaintext attack. It is a trade-off between time and space. An attacker can effectively make MITM attacks on cryptographic algorithms that rely on sequentially using multiple encryption operations like 2DES, DESX, 3DES, etc. Consider we have a block cipher that uses two encryption operations with keys k_1 and k_2. Normal brute force would require approximately $2^{k_1+k_2}$ operations.

$$c = ENC_{k2}(ENC_{k1}(p)) \tag{6}$$

Let us suppose the attacker knows certain plaintext-ciphertext pairs; then the cipher can be broken with a MITM attack in less computation time. The operation $DEC_{k2}(c)$ and $ENC_{k1}(p)$ results in same bitstream.

$$DEC_{k2}(c) = DEC_{k2}(ENC_{k2}(ENC_{k1}(p))) \tag{7}$$

$$DEC_{k2}(c) = ENC_{k1}(p) \tag{8}$$

Here the attacker calculates $DEC_{k2}(c)$ and $ENC_{k1}(p)$, stores the values in a hashtable. Once a match is found, keys k_1 and k_2 are checked with other plaintext-ciphertext pairs. The two processes have a computational complexity of 2^{k_1} and 2^{k_2} respectively. Time complexity of MITM attack is thus $2^{k_1} + 2^{k_2}$. Space complexity for MITM is the greater of the two 2^{k_1} and 2^{k_2}. Using MITM attack 2DES can be broken by a mere 2^{57} operations (k_1 and k_2 both equals to 56) and a space complexity of 2^{56}. 3DES cam be broken by 2^{112} operations with a space complexity of 2^{112} using MITM attack. Decrypt doesnot make use of multiple encryption process and therefore is safe from MITM attack.

3.2.3 Sweet-32 attack

This kind of attack is prominent over the algorithms where the algorithm breaks itself into small blocks like DES, DES-X, 3-DES, Blowfish. In these algorithms, the data are split into 64 bits block sizes and then encrypted, so the attack based on one of the popular theories of probability, birthday problems, finds the collision. 64 bits of block merely need 32 GB space for the hacker to perform this attack, making these algorithms very vulnerable. But due to the huge block size, as big as 192 bits of our algorithms, this can be regarded as a much secure alternative than the aforementioned algorithms against this sweet-32 attack.

3.2.4 Encryption decryption time

Speed in cryptography is one of the major concerns. The faster an algorithm with the same level of security, the better it is. Blowfish and AES are preferred over 3DES or DESX, as they are faster and, at the same time, brute force complexity is comparable. Our algorithm uses single encryption, giving it an edge over multiple encryption algorithms like DESX and 3DES in terms of encryption-decryption time. We encrypted the same plaintext with different encryption algorithms and obtained the result shown in Table 16. As seen from the results, our algorithm has a decent edge over 3 DES, DESX and their counterparts in encryption-decryption time (see Figs. 4, 5). Further, our algorithm is not susceptible to MITM attacks which makes breaking the algorithm hard for attackers.

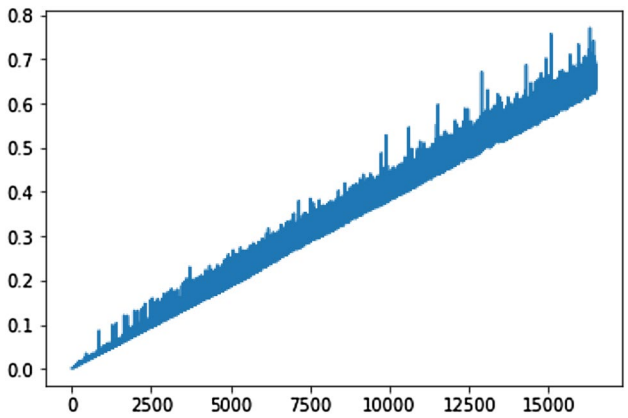


Fig. 4 Encryption time vs word length

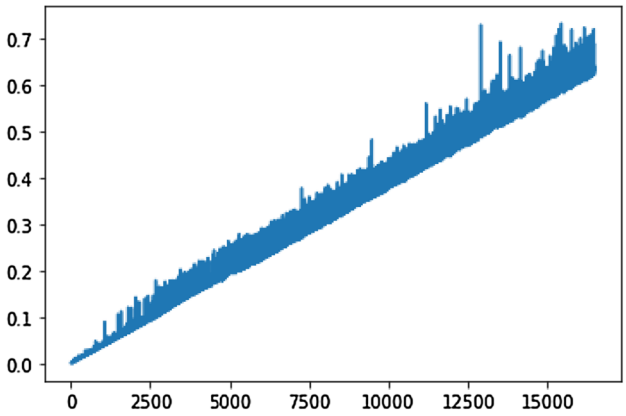


Fig. 5 Decryption time vs word length

Table 16 Encryption and Decryption time comparison

Algorithm	Encryption time	Decryption time	Network
DES	8.45e−3	9.76e−3	Feistel
2-DES	1.66e−2	1.70e−2	Feistel
3-DES	2.51e−2	2.97e−2	Feistel
DESX	1.21e−2	1.26e−2	Feistel
Blowfish	2.38e−4	7.51e−5	Feistel
Our algorithm	9.62e−3	1.21e−2	Feistel

4 Conclusion

This paper proposed a new encryption algorithm DeCrypt, an extended version of 3DES. DeCrypt is an alternative for all algorithms based on the Feistel cipher, where block length is not a constraint. DeCrypt have been designed so that the cipher is protected against meet in the middle and Sweet-32 attack (also known as birthday attack). Any 64-bit block size algorithm like DES, 2-DES, DES-X, 3-DES, Twofish, Blowfish requires around 32 GB of memory to break the algorithm, whereas the proposed algorithms needs 1.77^{21} GB of memory to break the algorithm. Besides these, experimental results show that the proposed algorithm encrypts and decrypts approximately around 61% faster compared to 3-DES. Future works involve further optimising the algorithm, improving the security, and performing cryptanalysis on the other known attack techniques like slide attacks. The proposed algorithm can be compared against popular symmetric algorithms like AES-128, AES-192, etc. The algorithm can be easily deployed over several communication channels for a securer data transfer.The security could be further enhanced by employing deep learning methods (Chong and Salam 2021). Further application can be found in IOT domain (Siddiqui et al. 2020a; Koo and Qureshi 2020)

Funding The work of Ashutosh Dhar Dwivedi is supported by the Independent Research Fund Denmark for Technology and Production under Grant 8022-00348A. Waleed Al-Numay acknowledges financial support from the Researchers Supporting Project number (RSP-2021/250), King Saud University, Riyadh, Saudi Arabia.

Data Availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

References

Ahmed Usman, Lin Jerry Chun-Wei, Srivastava Gautam (2021) Privacy-preserving deep reinforcement learning in vehicle adhoc networks. IEEE Consumer Electronics Magazine 1–1. https://doi.org/10.1109/MCE.2021.3088408

Amorado Ryndel V, Sison Ariel M, Medina Ruji P (2019) Enhanced data encryption standard (des) algorithm based on filtering and striding techniques. In: Proceedings of the 2019 2nd International Conference on Information Science and Systems, ICISS 2019, New York, NY, USA, pp 252–256. Association for Computing Machinery. ISBN 9781450361033. https://doi.org/10.1145/3322645.3322671

Biryukov A (2011) DES-X (or DESX). In: van Tilborg HCA, Jajodia S (eds) Encyclopedia of Cryptography and Security, 2nd edn. Springer, Berlin, p 331. https://doi.org/10.1007/978-1-4419-5906-5_570

De Cannière C (2005) Blowfish. In: van Tilborg HCA (ed) Encyclopedia of Cryptography and Security. Springer, Berlin. https://doi.org/10.1007/0-387-23483-7_34

Chong Bang Yuan, Salam Iftekhar (2021) Investigating deep learning approaches on the security analysis of cryptographic algorithms. Cryptography 5(4). https://doi.org/10.3390/cryptography5040030. ISSN 2410-387X

Chowdhury Deepraj, Dey Ajoy, Anand Harshit, Sengupta Sohang, Chakraborty Sourav (2021) An approach to avoid meet in the middle attack in 2 des. In: Interdisciplinary Research in Technology and Management. CRC Press, pp 602–608

Diffie W, Hellman ME (1977) Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer 10(6):74–84. https://doi.org/10.1109/C-M.1977.217750

Dwivedi AD (2020) Security analysis of lightweight iot cipher: Chaskey. Cryptogr. 4(3):22. https://doi.org/10.3390/cryptography4030022

Dwivedi AD (2021) BRISK: dynamic encryption based cipher for long term security. Sensors 21(17):5744. https://doi.org/10.3390/s21175744

Koo J, Qureshi NM (2020) Fine-grained data processing framework for heterogeneous iot devices in sub-aquatic edge computing environment. Wireless Pers Commun 116(2):1407–1422. https://doi.org/10.1007/s11277-020-07803-3

Koo J, Qureshi NMF (2021) Fine-grained data processing framework for heterogeneous iot devices in sub-aquatic edge computing environment. Wireless Pers Commun 116(2):1407–1422

Lin JC-W, Srivastava G, Zhang Y, Djenouri Y, Aloqaily M (2021) Privacy-preserving multiobjective sanitization model in 6g iot environments. IEEE Internet Things J 8(7):5340–5349. https://doi.org/10.1109/JIOT.2020.3032896

Lin C-W, Hong T-P, Hsu H-C (2014) Reducing side effects of hiding sensitive itemsets in privacy preserving data mining. Sci World J

Mehmood MS, Shahid MR, Jamil A, Ashraf R, Mahmood T, Mehmood A (2019) A comprehensive literature review of data encryption

techniques in cloud computing and iot environment. In: 2019 8th International Conference on Information and Communication Technologies (ICICT), pp 54–59. https://doi.org/10.1109/ICICT47744.2019.9001945

Morris RH Sr., Sloane NJA, Wyner AD (1977) Assessment of the national bureau of standards proposed federal data encryption standard. Cryptologia 1(3):281–291. https://doi.org/10.1080/0161-117791833020

Patil P, Narayankar P, Narayan DC, Meena SM (2016) A comprehensive evaluation of cryptographic algorithms: Des, 3des, aes, rsa and blowfish. Procedia Computer Science 78:617–624. https://doi.org/10.1016/j.procs.2016.02.108 (**1st International Conference on Information Security & Privacy 2015. ISSN 1877-0509**)

Sanwar Hosen ASM, Sharma PK, Ra I-H, Cho GH (2022) SPTM-EC: A security and privacy-preserving task management in edge computing for iiot. IEEE Trans. Ind. Informatics 18(9):6330–6339. https://doi.org/10.1109/TII.2021.3123260

Semenov Alexander, Zaikin Oleg, Otpuschennikov Ilya, Kochemazov Stepan, Ignatiev Alexey (2018) On cryptographic attacks using backdoors for sat. In: Thirty-Second AAAI Conference on Artificial Intelligence

Siddiqui IF, Qureshi NMF, Chowdhry BS, Uqaili MA (2020) Pseudo-cache-based iot small files management framework in hdfs cluster. Wirel Pers Commun 113(3):1495–1522. https://doi.org/10.1007/s11277-020-07312-3 (. **ISSN 0929-6212**)

Siddiqui IF, Qureshi NMF, Chowdhry BS, Uqaili MA (2020) Pseudo-cache-based iot small files management framework in hdfs cluster. Wireless Pers Commun 113(3):1495–1522

Singh KN, Singh AK (2022) Towards integrating image encryption with compression: A survey. ACM Trans. Multim. Comput. Commun. Appl. 18(3):891–8921. https://doi.org/10.1145/3498342

Thakur S, Singh AK, Ghrera SP, Elhoseny M (2019) Multi-layer security of medical data through watermarking and chaotic encryption for tele-health applications. Multim. Tools Appl. 78(3):3457–3470. https://doi.org/10.1007/s11042-018-6263-3

Vuppala A, Roshan RS, Nawaz S, Ravindra JVR (2020) An efficient optimization and secured triple data encryption standard using enhanced key scheduling algorithm. Procedia Computer Science 171:1054–1063. https://doi.org/10.1016/j.procs.2020.04.113 (**ISSN 1877-0509. Third International Conference on Computing and Network Communications (CoCoNet '19)**)

Wang J, Chen J, Ren Y, Sharma PK, Alfarraj O, Tolba A (2022) Data security storage mechanism based on blockchain industrial internet of things. Comput Ind Eng 164:107903. https://doi.org/10.1016/j.cie.2021.107903